



DISSERTAÇÃO DE MESTRADO

**RETROFITTING DO ROBÔ ASEA IRB6-S2 BASEADO EM
TECNOLOGIAS DE COMANDO NUMÉRICO USANDO LINUXCNC**

JUAN SEBASTIAN TOQUICA ARENAS

Brasília, Dezembro de 2016

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

**UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA**

**RETROFITTING DO ROBÔ ASEA IRB6-S2 BASEADO EM
TECNOLOGIAS DE COMANDO NUMÉRICO USANDO LINUXCNC**

JUAN SEBASTIAN TOQUICA ARENAS

**DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE
ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA
UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS
NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM
SISTEMAS MECATRÔNICOS**

APROVADA POR:

Prof. Dr. Alberto J. Alvares, PPMEC/UnB
Orientador

Prof. Dr. André Murilo de Almeida, Gama/UnB
Membro Interno

Prof. Dr. Renan Bonnard, MENESR/França
Membro Externo

BRASÍLIA/DF, 07 DEZEMBRO DE 2016

Toquica Arenas, Juan Sebastian

Retrofitting do robô ASEA IRB6-S2 baseado em tecnologias de comando numérico usando LinuxCNC / JUAN SEBASTIAN TOQUICA ARENAS. –Brasil, 2016.

170 p.

Orientador: Alberto José Alvares

Dissertação (Mestrado) – Universidade de Brasília – UnB

Faculdade de Tecnologia – FT

Programa de Pós-Graduação em Sistemas Mecatrônicos – PPMEC, 2016.

1. Retrofitting. 2. Robô Asea IRB6-S2. 3. Cinemática. 4. LinuxCNC. 5. Arquitetura aberta de controle. I. Alberto José Alvares, orientador. II. Universidade de Brasília. III. Faculdade de Tecnologia.

Agradecimentos

Em primeiro lugar a Deus, por me permitir a oportunidade de estar com vida e honrar seu nome por meio de minhas ações diárias.

Ao meu filho sendo uma infinita motivação diária, sempre confiante em romper os meus limites imaginários com o intuito de atingir meus objetivos sem receios. "A engenharia é como a vida", digo ao meu filho, "é uma arte especial que permite descobrir caminhos nunca antes percorridos, com convicção de que sempre existe mais alguma coisa por fazer. Sendo assim, querido Juan José sempre estará em capacidade de superar seus próprios limites, suas próprias fronteiras geográficas e culturais, seus sonhos, amo a você na distancia"

Ao meu maravilhoso núcleo familiar: minha mãe Clara Inés, meu Pai Efrain e minha irmã Diana Andrea que sempre estiveram e estão ao meu lado, sem se importarem com as condições externas, me contribuindo continuamente ao fortalecimento, mesmo distante, dos laços familiares. Saúdo também à todos os membros de minha família que me apoiaram em meus sonhos.

Agradeço carinhosamente à eterna professora de português, que sem ela não haveria condições de obter a realização desta etapa de minha vida. Sou muito grato à ela ter sempre acreditado em meu potencial, até quando eu duvidava de mim mesmo. Obrigado senhorita Carvalho Bonifácio por estar neste caminho de aprendizagem constante ao meu lado, sempre me protegendo, me contagiando com essa energia de esperança e felicidade.

Aos colegas e amigos Diego Benavides e Luiz Eduardo, que ao longo desta decisão acadêmica nos tornamos maduros com as experiências de vida. Condições estas que nos são necessárias para conquistarmos qualquer meta e cumprir quaisquer sonhos. Reconhecendo a importância da disciplina e paixão.

A todas as pessoas que de alguma maneira, direta e indiretamente, contribuíram para que este sonho pudesse hoje estar concluído, através de uma palavra, um abraço, uma caminhada noturna. Detalhes que marcam as vidas de pessoas que convivem na mesma realidade.

Por último, mas não menos importante, ao professor Alberto Alvares pela oportunidade de levar este projeto até o final, pela paciência quando precisei recuperar o foco na minha estadia no Brasil. O mais importante foi ter aprendido com sua sabedoria que os objetivos só são atingidos com disciplina, sacrifício e objetividade.

Ao CNPq pela bolsa de mestrado que me permitiu manter ao longo dos 24 meses de dedicação exclusiva. Aportando dinamicamente um grão de areia da literatura relacionada com minha paixão, a robótica. Ao PPMEC e especialmente ao professor Edson por me ter dado a oportunidade de fazer concretizar um ideal de vida.

JUAN SEBASTIAN TOQUICA ARENAS

RESUMO

Este trabalho apresenta uma proposta de *retrofitting* para o controlador do manipulador ASEA IRB6-S2 concebido a partir de uma arquitetura aberta de controle baseada na utilização do LinuxCNC para plataforma PC X86 com Ubuntu/Linux, permitindo assim o aproveitamento da estrutura mecânica do manipulador ASEA IRB6-S2, que está em ótimo estado (juntas, elos e *harmonic drives*, etc), assim como parte do gabinete original para a centralização do sistema geral de controle. Também é apresentada a modernização do controlador trocando os motores, *driver* de potência, *encoder* e fontes de alimentação, além da inclusão do controlador LinuxCNC com o sistema operacional Ubuntu, o que permitirá o controle do manipulador usando linguagem de programação de Comando Numérico baseado na norma RS-274.

O modelo cinemático direto e inverso será concebido usando a notação D-H (Denavit-Hartenberg), incorporando e descrevendo as equações homogêneas diretamente na plataforma LinuxCNC, através da definição da cinemática do Robô ASEA IRB6-S2 com cinco graus de liberdade mediante a programação em linguagem C e compatível com *okernel* em tempo real RTAI, sendo uma contribuição real do trabalho, pois este robô e sua cinemática não estão disponíveis na plataforma LinuxCNC.

É apresentada uma visão geral do trabalho realizado sobre questões como uma proposta metodológica para o *retrofitting* de manipuladores industriais baseada em uma arquitetura aberta de controle, para uma atualização e adaptação de robôs que por diversos motivos foram considerados como obsoletos ou simplesmente não operacionais. Da mesma forma, a adaptação e contribuição do desenvolvimento com sistema LinuxCNC é justificado no processo de conhecer e usar suas capacidades para controlar corretamente manipuladores industriais e máquinas CNC, como a modelagem do robô ASEA IRB6-S2 com o LinuxCNC para inclusão na biblioteca padrão de manipuladores da plataforma, validando assim o conceito de arquitetura aberta que pode ser adaptado para outros robôs com cinemática não-trivial semelhantes.

A modernização deste robô vai possibilitar o desenvolvimento de novos projetos que permitam consolidar o trabalho feito com uma arquitetura aberta que suporta modificações para as novas tecnologias desenvolvidas em robótica, é validada com a atualização de um robô com mais de 40 anos de ser fabricado e que se tornou novamente operacional através da metodologia proposta baseada na técnica *retrofitting*.

ABSTRACT

This work presents a retrofitting proposal for the ASEA IRB6-S2 Robot Controller through an open architecture using LinuxCNC for PC X86 platform with Ubuntu/Linux, taking advantage of a manipulator ASEA IRB6-S2 mechanical structure, that It is in optimal state (joint, axes and the harmonic drive gears), and using part of original control cabinet to centralize the robot control general system. Also It is presented the updating of the original controller allowing the change of servo-motor, driver, encoder and power supplies also including LinuxCNC platform with Ubuntu to enable the manipulator robot control using Numerical Control language based in RS-274 standard.

The robot forward and inverse kinematic model is going to be generated by DH (Denativ-Hartenberg) convention, including and describing the specific ASEA IRB6-S2 homogenous equations directly in the LinuxCNC controller, through C programming language files with properly logical structure compatible with LinuxCNC controller and the real time kernel for Linux (RTAI), being a real contribution of this work, because this robot model and its kinematics are not available in LinuxCNC.

It is presented a detailed view about the work done with relative aspects like a proposal of retrofitting methodology for industrial manipulators based in open architecture controller, updating and adapting obsolete or "out of service" robots. In the same way, adapting and developing a LinuxCNC based system is possible to take advantage of its capabilities to control correctly industrial manipulators and CNC machines, specifically for modeling the ASEA IRB6-S2 robot with LinuxCNC to include it in the example machines library, as well validating the open architecture concept to integrate other non-trivial kinematics robots to LinuxCNC controller.

With the robot modernization is possible to develop new projects to consolidate this work with an open architecture that supports the integration with new technologies in robotics field, and validated with a retrofitting methodology proposal applied to a robot was built more than 40 years ago.

SUMÁRIO

RESUMO	i
ABSTRACT	ii
LISTA DE FIGURAS	vi
LISTA DE TABELAS	x
LISTA DE ABREVIATURAS E ACROGRAMAS	xii
1 Introdução	1
1.1 Contextualização	1
1.2 Definição do Problema.....	2
1.3 Objetivos da Dissertação	3
1.3.1 Objetivo Geral	3
1.3.2 Objetivos Específicos.....	3
1.4 Apresentação do Documento	3
2 Revisão Bibliográfica	6
2.1 Introdução.....	6
2.2 Arquiteturas CNC Abertas: Estado da Arte.....	6
2.3 <i>Retrofitting</i> em Robôs Industrias	7
2.3.1 Análise dos Sensores para o <i>Retrofitting</i> de um Robô Industrial	9
2.3.2 Implementação de <i>Retrofitting</i> de Controladores de Equipamentos Industriais	9
2.3.3 Atualização de <i>Hardware</i> e <i>Software</i> de um Robô Industrial	9
2.3.4 Controlador de Robôs com Arquitetura Aberta Aplicado às Tarefas de Interação	11
2.3.5 Sistema de Usinagem Robótico Controlado como Máquina-Ferramenta.....	11
2.3.6 Arquitetura Aberta para <i>Retrofitting</i> de Robôs	12
2.3.7 Remanufatura de Manipuladores Robóticos com Arquitetura Aberta.....	13
2.3.8 Simulação de Manipulador Serial: Cinemática e Implementação em LinuxCNC.....	13
2.4 Plataformas de Controle para Robôs Avaliadas no Trabalho	15
2.4.1 LinuxCNC	15
2.4.2 Mach3/Matlab.....	15
2.4.3 ROS - Sistema Operacional para Robôs	16

2.4.4	iGETRobot.....	16
2.4.5	Step-NC Machine	16
2.4.6	Adept.....	16
2.4.7	4DIAC - Framework for Distributed Industrial Automation and Control	17
2.4.8	Microsoft Robotics Developer Studio.....	17
2.5	Controlador LinuxCNC.....	17
2.6	Manipuladores Industriais.....	21
2.6.1	Classificação de Manipuladores Industriais.....	21
2.6.2	Cinemática de Manipuladores Industriais	22
2.6.3	Cinemática Manipulador ASEA IRB6-S2	26
2.7	Sistemas de Controle para Manipuladores Industriais	32
2.7.1	Sintonização do Controlador PID	34
3	Proposta de Metodologia para o <i>Retrofitting</i> de Robôs Industriais	35
3.1	Introdução.....	35
3.2	Arquitetura de Controle Proposta	35
3.3	Metodologia Baseada no <i>Retrofitting</i> para Robôs Industriais: Modelagem IDEF0	37
3.4	Desmontagem e Diagnóstico do Robô	40
3.4.1	Diagnóstico da Estrutura Física do Robô.....	40
3.4.2	Sistema de Alimentação do Robô	40
3.4.3	Sistema de Controle do Robô	42
3.4.4	Desmontagem dos Componentes.....	42
3.5	Atualização <i>Hardware</i>	42
3.6	Atualização <i>Software</i>	43
3.6.1	Comparação Plataforma de Controle para Robô.....	46
3.6.2	Arquitetura de Controle Baseada em LinuxCNC.....	48
3.7	Montagem dos Componentes	50
3.8	Validação <i>Retrofitting</i> : Ensaios e Testes	52
4	Retrofitting do Robô ASEA IRB6-S2	55
4.1	Introdução.....	55
4.2	Desmontagem e Diagnóstico do Manipulador	55
4.2.1	Diagnóstico do Estado Original do Robô	56
4.2.2	Desmontagem dos Componentes Originais do Manipulador.....	60
4.3	Atualização <i>Hardware</i>	69
4.3.1	Especificação de Componentes para o <i>Retrofitting</i>	69
4.3.2	Projeção e Fabricação de Novos Componentes para o <i>Retrofitting</i> do Robô ASEA IRB6-S2	71
4.4	Atualização <i>Software</i>	75
4.4.1	Características Mínimas do <i>Hardware</i> : LinuxCNC	75
4.4.2	Incluir Cinemática não-trivial no LinuxCNC.....	76
4.4.3	Gerar e Configurar Novo Projeto de Manipulador em LinuxCNC	79
4.4.4	Configuração Arquivo INI	86
4.4.5	Configuração Arquivo HAL	88

4.4.6	Modelo Virtual do Robô ASEA IRB6-S2 para Simulação em LinuxCNC.....	90
4.5	Montagem dos Novos Componentes no Manipulador ASEA IRB6-S2.....	92
4.5.1	Montagem dos Conjuntos de Junta	92
4.5.2	Montagem do Gabinete de Controle.....	93
4.6	Testes Iniciais após o <i>Retrofitting</i>	96
4.6.1	Banco de Testes para Operação Inicial com LinuxCNC.....	96
4.6.2	Simulação da Execução Programa NC.....	97
4.6.3	Calibração da Posição de Cada Junta.....	99
4.6.4	Estratégia de Sintonia do PID	100
4.6.5	Configuração Posicionamento <i>home</i> do Manipulador.....	102
5	Estudo de Casos e Análise de Capabilidade	106
5.1	Introdução.....	106
5.2	Validação <i>Retrofitting</i> Baseado em Programação por <i>teach-in</i>	107
5.3	Validação <i>Retrofitting</i> com Programa NC Gerado por CAD/CAM	112
5.4	Estudo de Repetibilidade com Peça de Geometria Complexa em 2D Gerada por CAD/- CAM	114
5.5	Ensaio Geométrico com Peça Padrão Moldura: Erro de Retilidade e Erro de Per- pendicularidade	121
5.6	Análise dos Resultados Obtidos	127
6	Conclusões	129
6.1	Conclusões do Trabalho	129
6.2	Sugestões para Trabalhos Futuros.....	131
	REFERÊNCIAS BIBLIOGRÁFICAS.....	133
	APÊNDICES	139
A	Programa NC - Teach-in	140
B	Programa NC - Repetibilidade	141
C	Desenho Mecânico das Luvas	143
D	Desenho Mecânico Flange: Junta Dois.....	144
E	Desenho Mecânico Flange: Junta Três.....	145
F	Ferramenta de extração de pinos	146
G	Uso da ferramenta de extração de pinos	147
H	Desenho Técnico Placa Capacitores.....	148
I	Diagrama Elétrico nas Conexões Placas/Motores/Drives/Encoders/Fonte/Computador	149

J	Arquivos com a Cinemática do Manipulador.....	150
K	Arquivo do Modelo Virtual.....	158
L	Modulo Teach-in	160
M	Arquivo de Configuração INI	162
N	Arquivo de Configuração HAL.....	168

LISTA DE FIGURAS

2.1	Arquitetura de controle para ABB IRB2000.....	10
2.2	Arquitetura de controle baseada em OSI para REIS-Rv15	11
2.3	Arquitetura de controle para LOLA 50	12
2.4	Arquitetura de controle para	13
2.5	Arquitetura controle para ASEA IRB6-S2	14
2.6	Robô virtual em LinuxCNC	14
2.7	Arquitetura Modular do Controlador LinuxCNC.....	18
2.8	Modulo EMCMOT.....	19
2.9	Definição básica camada HAL	20
2.10	Modulo EMCMOT.....	20
2.11	Tipos de robôs industriais Fonte: (BARRIENTOS, 2007)	22
2.12	Sistemas de origem segundo a convenção DH	23
2.13	Posição e orientação de um elemento físico a) Sistema de coordenadas fixo; b) Sistema coordenadas móvel.....	24
2.14	Definição do sistema de coordenadas segundo DH	25
2.15	Sistemas de origem segundo a convenção DH para o robô ASEA IRB6-S2.....	27
2.16	Configuração controle de manipulador em malha fechada	32
2.17	Resposta dos controladores P-I-D a um degrau	33
3.1	Arquitetura geral por camadas do <i>retrofitting</i>	36
3.2	Diagrama IDEF0: Retrofitting nível A0	38
3.3	Diagrama IDEF0: Retrofitting nível A0	39
3.4	Diagrama IDEF0: Atividades desmontagem e o diagnóstico do robô.....	41
3.5	Diagrama IDEF0: Atividades especificação atualização Hardware	44
3.6	Diagrama IDEF0: Atividades especificação atualização software.....	45
3.7	Diagrama IDEF0: Atividades comparação de plataformas.....	47
3.8	Diagrama IDEF0: Atividades configuração LinuxCNC	49
3.9	Diagrama IDEF0: Atividades da montagem.....	51
3.10	Diagrama IDEF0: Validação <i>retrofitting</i> : ensaios e testes	53
4.1	Manipulador inativo antes do <i>retrofitting</i>	57
4.2	Gabinete de controle original do manipulador	58
4.3	Placas de controle do gabinete original do manipulador.....	59
4.4	Definição da configuração do manipulador ASEA IRB6-S2	60

4.5	Conjunto de juntas quatro e cinco	61
4.6	Conjunto da junta cinco acoplado ao manipulador	61
4.7	Detalhe do conjunto das juntas quatro e cinco a) Vista lateral; b) Vista frontal	62
4.8	<i>Harmonic-Drive</i> da junta cinco a) <i>Flexspline</i> , b) <i>Wavegenerator</i>	62
4.9	Flange das juntas quatro e cinco	63
4.10	Desenho dos conjuntos das juntas dois e três	63
4.11	Conjunto de juntas dois e três no manipulador ASEA IRB6-S2	64
4.12	Elementos do conjunto das juntas 2 e 3	64
4.13	Desenho do conjunto da junta um	65
4.14	Guincho hidráulico usado para a desmontagem.....	66
4.15	Desmontagem da estrutura mecânica últimas quatro juntas a) Segurar a estrutura; b) Separar da base do manipulador	66
4.16	Desenho em detalhe do conjunto da junta um	67
4.17	Conjunto da junta um na base do manipulador ASEA IRB6-S2	68
4.18	Elementos do conjunto da junta um fora da base do robô	68
4.19	Luvas projetadas e fabricadas	71
4.20	Motor com o <i>Harmonic drive</i> a) Servomotor com Luvas em ABS; b) Servomotor antes de Montagem	71
4.21	Flange fabricada e projetada para a junta dois.....	72
4.22	Flange fabricada e projetada para a junta três	72
4.23	Processo de fabricação da PCB para o banco de capacitores	74
4.24	Banco de capacitores projetado para o manipulador	75
4.25	Arquivos disponíveis na pasta do pacote fonte	77
4.26	Informação geral do pacote fonte da plataforma.....	77
4.27	Instruções de compilação e instalação da plataforma.....	77
4.28	Arquivos disponíveis no diretório de compilação	78
4.29	Edição arquivo " <i>Makefile</i> " para a nova configuração	78
4.30	Configuração do arquivo com a cinemática do manipulador	79
4.31	Comandos iniciais para compilação da plataforma	79
4.32	Compilar e instalar a plataforma com a nova configuração do manipulador.....	80
4.33	Tela inicial " <i>Stepconf</i> "	81
4.34	Escolha da configuração do projeto	81
4.35	Informação básica da máquina ou projeto.....	82
4.36	Teste de latência no LinuxCNC	83
4.37	Configuração de entradas e saídas.....	84
4.38	Configuração de entradas e saídas.....	84
4.39	Projeto no diretório das configurações padrões	85
4.40	Valores da seção [EMC] do arquivo INI	86
4.41	Valores da seção [HAL] do arquivo INI	87
4.42	Valores da seção [TRAJ] do arquivo INI	87
4.43	Valores da seção [AXES] do arquivo INI.....	88
4.44	Módulos RT do arquivo HAL	88

4.45	Configuração saídas da placa paralela no arquivo HAL.....	89
4.46	Parâmetros para geração de sinais dos eixos via HAL	89
4.47	Definição das variáveis HAL	90
4.48	Modelo virtual do manipulador ASEA IRB6-S2	91
4.49	Instruções no arquivo HAL para o controle da simulação	91
4.50	Montagem juntas quatro e cinco	92
4.51	Montagem juntas dois e três	93
4.52	Distribuição interna do gabinete de controle.....	93
4.53	Fonte de alimentação no gabinete de controle.....	94
4.54	Disposição da interface paralela e os <i>drives</i> de controle	94
4.55	Parte frontal do gabinete de controle	95
4.56	Manipulador ASEA IRB6-S2 após <i>retrofitting</i>	95
4.57	Banco de testes dos atuadores do manipulador.....	96
4.58	Componentes necessários para o controle das juntas	97
4.59	Movimentação das juntas desde a interface de usuário do LinuxCNC	98
4.60	Simulação da operação do manipulador em LinuxCNC	98
4.61	Resultado da simulação do manipulador em LinuxCNC	99
4.62	Assistente de calibração na interface de usuário do LinuxCNC	100
4.63	Variável SCALE para relacionar rotação servomotores/juntas	101
4.64	Parâmetros do PID ajustáveis no <i>drive</i> empregado.....	101
4.65	Configuração sequência <i>home</i> no LinuxCNC	103
4.66	Configuração juntas um e dois arquivo HAL	104
4.67	Configuração juntas três, quatro e cinco arquivo HAL.....	104
4.68	Configuração <i>home</i> robô ASEA IRB6-S2.....	105
5.1	Processo de obtenção de pontos com o módulo <i>teach-in</i> em <i>LinuxCNC</i>	107
5.2	GUI inicial <i>teach-in</i> com <i>LinuxCNC</i>	108
5.3	GUI do módulo <i>teach-in</i>	108
5.4	Pendant P4S compatível com <i>LinuxCNC</i> Fonte: VistaCNC (2016, p. 1).....	109
5.5	Arquivos gerados em <i>LinuxCNC</i> com a opção <i>teach-in</i> a) Arquivo pontos adquiridos; b) Arquivo código G	109
5.6	Estudo de caso pick and place.....	110
5.7	Trajétoria desejada via <i>teach-in</i> programa NC número 1	110
5.8	Simulação posicionamento peça <i>LEGO</i> Programa NC número 1	111
5.9	Trajétoria desejada via <i>teach-in</i> programa NC número 2.....	111
5.10	Simulação posicionamento peça <i>LEGO</i> programa NC número 2.....	112
5.11	Trajétoria a ser atingida pelo manipulador	112
5.12	Simulação da trajetória gerada pelo programa NC	113
5.13	Robô em operação executando o programa NC.....	114
5.14	Resultado do traço feito pela caneta como efetuator do robô	114
5.15	Simulação inicial da trajetória do efetuator	115
5.16	Registro digital da repetibilidade	115

5.17	Resultado final da repetibilidade do programa NC	116
5.18	Resultado gráfico para medição das espessuras	116
5.19	Projetor de perfil <i>Mitutoyo PJ-A3000</i>	117
5.20	Opção " <i>line to point</i> " da interface de usuário do projetor de perfil.....	118
5.21	Curva normal das espessuras medidas do traço único	119
5.22	Curva normal das espessuras medidas com repetição	120
5.23	Marcação da caneta na área de trabalho	122
5.24	Paquímetro digital usado para as medições	122
5.25	Pontos de referências para as medições	123
5.26	Regressão lineal do erro de retilidade em X	124
5.27	Regressão lineal do erro de retilidade em Y	125
5.28	Regressão linear do erro em Y com eixos trocados	126

LISTA DE TABELAS

2.1	Comparação plataformas de tecnologias de comando numérico abertas (<i>Open CNC</i>)	8
2.2	Tabela Parâmetros DH para o Robô IRB6-S2	27
2.3	Distancias da configuração mecânica do manipulador ASEA IRB6-S2	31
2.4	Valores das reduções do manipulador ASEA IRB6-S2	32
3.1	Comparação plataformas de controle para robôs	48
3.2	Comparação entre as plataforma de controle para robô	50
4.1	Especificações do transformador gabinete de controle	56
4.2	Sinais de controle e potência para o robô	58
4.3	Especificação dos componentes originais: Conjunto de junta	59
4.4	Especificação técnica dos componentes presentes no efetuador	59
4.5	Especificação dos componentes originais: Conjunto de junta	69
4.6	Disposição dos componentes por cada subconjunto do sistema	69
4.7	Definição componentes dos subconjuntos do robô	70
4.8	Especificação dos componentes para aquisição	70
4.9	Componentes do <i>retrofitting</i> para fabricação	70
4.10	Especificação do motor DC - KL23-130-60	73
4.11	Especificação do banco de capacitores	74
4.12	Especificação e características do hardware para LinuxCNC	75
4.13	Especificação do computador para LinuxCNC	83
4.14	Valores de calibração das juntas	100
4.15	Valores das variáveis de velocidade para a sequência <i>home</i>	104
5.1	Medição das espessuras do traço único	117
5.2	Medição das espessuras do traço com 10 repetições	120
5.3	Medição do comprimento ao longo dos eixos X e Y	124
5.4	Síntese dos resultados obtidos	127

LISTA DE ABREVIATURAS E ACROGRAMAS

ABNT	Associação Brasileira de Normas Técnicas
AM	An architectural Model
ATF	Automatic Transmission Fluid
ASEA	General Swedish Electric Company
ABB	Asea Brown Boveri
API	Application Programming Interfaces
BSD	Berkeley Software Distribution
CAD	Computer-aided design
CAD/CAM	Integração entre CAD e CAM
CAM	Computer-aided manufacturing
CAN-Bus	Barramento de dados
CLP	Controlador Lógico Programável
CNC	Controle Numérico Computadorizado
DOF	Degrees of Freedom
DH	Denavit-Hartenberg
DLL	Dynamic-link library
DSC	Digital Signal Controller
EMC	The Enhanced Machine Controller
EMCMOT	Modulo em LinuxCNC para o controle dos eixos.
EMCIO	Modulo em LinuxCNC para variáveis tipo I/O
FIFO	First In, First Out
GUI	Graphical User Interface

GRACO	Grupo de Automação e Controle
GNU/GPL	General Public License version
HAL	Hardware Abstraction Layer
HOAM-CNC	Hierarchical Open Architecture Multi-processor system
HTTP	Hyper Text Transfer Protocol
IDEF0	Integration DEFinition language 0
INI	Arquivo de configuração em LinuxCNC
ISO	Internacional Organization for Standarization
MATLAB	Matrix Laboratory
MDI	Manual Data Input
NC	Comando Numérico
NIST	National Institute of Standards and Technology
NML	Neutral Message Language
OAC	Open Architecture Controller
OPC	Object Linking and Embedding for Process Control
OpenGL	Open Graphics Library
OSEC	Open System Environment for Manufacturing Consortium
OSI	Open Systems Interconnection model
PID	Controlador Proporcional, Integral e Derivativo
PUMA	Programmable Universal Machine for Assembly
RS-274	Linguagem de programação baseado Controle Numérico
RTAI	Real Time Application Interface
SCARA	Selective Compliant Articulated Robot for Assembly
STEP	Standard Exchange of Product Model Data
USB	Universal Serial Bus
XML	eXtensible Markup Language

Capítulo 1

Introdução

1.1 Contextualização

A transformação tecnológica tem se caracterizado pelas mudanças entre os atuais sistemas de produção, e também pelos sistemas que serviram como base para as diferentes evoluções tecnológicas mundiais desde o final do século XVIII, bem como a integração dos insumos da produção industrial que tem sido aperfeiçoada através do tempo, segundo as demandas da sociedade. Estas características obedecem conceitos como: a qualidade dos produtos em um ambiente de produção flexível que amplia vinculação de consumidores e aliados de negócios; a articulação da produção e alta qualidade de serviços para produtos híbridos dentre outros (ELMARAGHY; MONOSTORI, 2014). Estes conceitos geram desafios para continuar no ciclo natural da evolução tecnológica, como integrar e adaptar as tecnologias antigas aos sistemas de produção modernos.

Uma adaptação tecnológica tem que ser uma solução baseada em uma metodologia que possibilite: diagnosticar o estado dos sistemas; necessidades da aplicação final; tecnologias compatíveis e desenvolvidas atualmente; arquitetura adequada que possibilite a integração eficaz entre tecnologias; uma etapa de validação da implementação da solução; entre outras etapas segundo os requisitos finais do sistema de produção a ser atualizado (MOCTEZUMA et al., 2012).

Os fornecedores de máquinas industriais desenvolvem arquiteturas de controle fechadas como estratégia de mercado, deixando o equipamento dependente dos pacotes de serviços adicionais dos fabricantes como: manutenção; atualização e/ou venda e substituição de componentes. A multinacional sueca ABB desenvolvedora de tecnologias específicas para a automatização industrial, anteriormente chamada ASEA, no final dos anos 80 houve altos investimentos na área de robótica com uma perspectiva de crescimento a longo prazo. O primeiro modelo de robô industrial do mundo pela empresa ao final da segunda revolução industrial, denominado ASEA IRB6-S2, composto por cinco DOF. Os primeiros robôs desse tipo foram comprados e empregados para a solda automatizada de tubos (MORTIMER; ROOKS, 2013).

No Brasil, a montadora de carros FIAT fez a doação do manipulador ASEA IRB6-S2 para algumas universidades federais, devido a uma estratégia de mercado para aumentar os indicadores de produção, através do aumento no desempenho dos processos industriais mediante a aquisição de novos equipamentos, que atingiriam as demandas do período tecnológico iniciado com a terceira revolução industrial. Dessa forma,

foi possível fortalecer a sinergia característica dos países industrializados entre a **indústria** e **academia** no intercâmbio de equipamentos especializados, como um insumo para a pesquisa no setor de manufatura industrial.

Com a quarta revolução industrial surgiu uma série de oportunidades para o setor produtivo mundial, o que permitiu diminuir custos de operação e melhorias no produto final. Alguns dos esforços para se integrar com a revolução tecnológica atual são: a manutenção preventiva; serviços de manufatura inteligente; gerencia de processos de qualidade; assim como o uso da técnica **retrofitting em máquinas industriais** (WEINBERGER; BILGERI; FLEISCH, 2016).

A quarta revolução industrial também conhecida como Indústria 4.0 é associada com os conceitos da Internet das Coisas (IoT) e os serviços gerados através da internet, integrados em um ambiente de manufatura (GILCHRIST, 2016). Desta forma a Indústria 4.0 esta enfocada no estabelecimento e na integração de produtos inteligentes e processos de produção, permitindo obter um novo conceito para as fabricas do futuro conhecido como *smart factories*, facilitando a comunicação entre humanos, maquinas e produtos através de sistemas físicos-cibernéticos (CPS) (BRETTEL et al., 2014).

Por conseguinte, o *retrofitting* para manipuladores industriais faz parte dos esforços concebidos a partir da revolução tecnológica em andamento. Essa técnica permite obter o diagnóstico dos componentes originais do manipulador, para depois substituir os que estão obsoletos ou danificados, especialmente nos sub-sistemas de potência e controle (Lima II et al., 2004). O *retrofitting* de manipuladores industriais não garante drasticamente o aumento da produtividade numa linha de produção, mas o manipulador após a implementação desse processo terá condições de promover melhorias contínuas no seu desempenho, o que permitirá a progressiva redução do tempo global de um processo produtivo (HUNT, 2012).

1.2 Definição do Problema

Atualmente os processos industriais sofrem transformações com a quarta revolução industrial, onde a integração das tecnologias da informação junto com o gerenciamento de sistemas interconectados têm obrigado as empresas a implementar metodologias baseadas nas tecnologias modernas (LEE; BAGHERI; KAO, 2015). É necessário que todos os equipamentos envolvidos em um modelo de produção das empresas sejam compatíveis com sistemas digitais e com a capacidade de receber, processar, enviar e tomar decisões em tempo real para melhorar os indicadores gerais de produtividade.

Existem empresas que atualmente têm em seu chão de fabrica ou simplesmente estão planejando a aquisição de equipamentos com tecnologias análogas e obsoletas, como parte da estratégia para baixar os custos de produção, mas que atualmente esses equipamentos não possuem compatibilidade com as tecnologias modernas, especificamente manipuladores industriais com mais de quatro décadas de fabricação. Esses equipamentos contam com uma arquitetura de controle totalmente fechada, onde apenas o fornecedor podê fazer algum tipo de modificação na estrutura física ou lógica do robô (TOQUICA; ALVARES, 2015).

Novas aplicações e melhorias complementares para os manipuladores industriais ficam restritas aos serviços adicionais ofertados pelo fornecedores combinados durante a venda do equipamento. Adicio-

nalmente, a falta de arquiteturas abertas para o controle de manipuladores industriais e a forte presença de protocolos de comunicação proprietários, diminuem as opções das empresas que almejam maior competitividade (BECERRA et al., 2004).

Em consequência, é necessário o desenvolvimento de abordagens alternativas a fim de consolidar uma arquitetura totalmente aberta de controle para robôs manipuladores, possibilitando aproveitar os recursos desse tipo de equipamento para o setor industrial. Junto a isso, uma arquitetura de controle aberta com suporte a integração com outras soluções tecnológicas modernas, a fim de garantir melhorias nos processos industriais, aumentando a competitividade das empresas através de soluções flexíveis e de baixo custo.

1.3 Objetivos da Dissertação

1.3.1 Objetivo Geral

O objetivo geral do projeto é implementar uma proposta metodológica baseada em tecnologia de controle numérico para o manipulador ASEA IRB6 usando LinuxCNC através da técnica *retrofitting*, caracterizada por uma arquitetura aberta de controle que possibilite uma posterior integração do manipulador a uma célula de manufatura flexível.

1.3.2 Objetivos Específicos

Para atingir o objetivo geral no presente trabalho, foram estabelecidos os objetivos específicos a seguir:

- Desenvolver uma metodologia genérica para aplicar a técnica *retrofitting* em manipuladores industriais, e implementá-la no robô ASEA IRB6-S2 para que seja possível validá-la.
- Diagnosticar o estado do sistema original do manipulador ASEA IRB6-S2 permitindo especificar os componentes necessários para implementar a metodologia proposta.
- Gerar o projeto técnico a nível mecânico, elétrico-eletrônico e de controle, segundo a implementação da metodologia proposta para garantir o *retrofitting* do manipulador ASEA IRB6-S2.
- Integrar a cinemática direta e inversa do manipulador ASEA IRB6-S2 à plataforma LinuxCNC para gerar trajetórias a partir de programas NC com a informação das coordenadas finais do efetuador.
- Realizar ensaios e testes de validação do *retrofitting* implantado, incluindo o estudo de capacidade do manipulador ASEA IRB6-S2 e publicar os resultados obtidos.

1.4 Apresentação do Documento

O presente trabalho está composto por seis capítulos e catorze apêndices. Inicialmente é apresentada uma revisão de literatura com alguns trabalhos feitos usando a *retrofitting* baseados em arquiteturas abertas de controle, assim como plataformas propostas de controle para robôs. Também são apresentadas as

características básicas do controlador LinuxCNC, além da definição de conceitos básicos acerca dos manipuladores industriais e seu modelamento cinemático. Ao final do Capítulo 2, apresenta-se a definição dos controladores empregados em manipuladores industriais e as técnicas de sintonização.

No Capítulo 3 será apresentada uma proposta metodológica genérica baseada no modelo IDEF0, com as atividades necessárias para implementar a técnica *retrofitting* em manipuladores industriais, como a desmontagem e diagnóstico do manipulador, a atualização da parte do *software* e *hardware*, a montagem de componentes adquiridos e fabricados segundo o diagnóstico feito. Concluindo o Capítulo 3 valida-se a metodologia proposta mediante ensaios e testes.

O Capítulo 4 faz referência a implementação da metodologia proposta no manipulador ASEA IRB6-S2, e que esta dividida em cinco partes: A primeira parte tem como detalhe a desmontagem e o diagnóstico do estado atual dos sub-sistemas do robô, em seguida, apresenta-se a especificação dos componentes que devem ser adquiridos e fabricados, segundo o resultado do diagnóstico feito. Igualmente é detalhada na terceira parte do capítulo da implementação a atualização da plataforma *software* do manipulador, com a integração e configuração do controlador LinuxCNC.

A montagem dos componentes projetados, fabricados e adquiridos no manipulador e no gabinete de controle é apresentado no capítulo de implementação. Posteriormente, serão exibidos testes iniciais após implementar a técnica *retrofitting* com a metodologia proposta, juntamente com uma simulação do manipulador com a execução de um programa NC. Finalizando o quarto capítulo é apresentada uma proposta para a calibração das juntas do manipulador desde a plataforma LinuxCNC, assim como a sintonização dos controladores PID presentes nos *drives* de cada uma das juntas e da configuração do posicionamento *home* do manipulador.

No Capítulo 5 são apresentados quatro propostas de estudo de caso para validar a implementação da metodologia proposta no manipulador ASEA IRB6-S2. As duas primeiras validações se basearam em programas NC, o primeiro gerado através de programação via *teach-in* e o segundo programa feito mediante CAD/CAM. Posteriormente, realizará um estudo de repetibilidade em uma peça 2D, assim como ensaios geométricos para estimar os erros de retilidade e perpendicularidade. Finalizando o quinto capítulo são apresentados as análises dos resultados obtidos da implementação do *retrofitting* no manipulador ASEA IRB6-S2. Por último, no Capítulo 6 são apresentadas as conclusões e sugestões para trabalhos futuros.

No Apêndice A é apresentado o programa NC para a programação via *teach-in*, e no Apêndice B o programa NC usado para o estudo de repetibilidade do manipulador. No Apêndice C é detalhado o desenho mecânico das luvas usadas nos motores adquiridos. Enquanto no Apêndice D e Apêndice E são apresentados os desenhos mecânicos das flanges associadas as juntas dois e três. Posteriormente é apresentado no Apêndice F e no Apêndice G o desenho técnico, assim como o uso da ferramenta para extração de pinos. A seguir é apresentado no Apêndice H o desenho esquemático da placa de capacitores, e no Apêndice I se apresenta o desenho elétrico do sistema de controle projetado e implementado no *retrofitting*.

No Apêndice J são apresentadas as equações cinemáticas e valores dos parâmetros DH, incluídas em dois arquivos em linguagem C. No Apêndice K é apresentado o código do arquivo em *Python* para gerar o modelo virtual do manipulador ASEA IRB6-S2, permitindo simular a movimentação do robô desde LinuxCNC. No Apêndice L é apresentado o arquivo em *Python* do módulo *teach-in*. No Apêndice M

são apresentadas a configurações específicas do manipulador ASEA IRB6-S2 no ambiente da plataforma LinuxCNC, assim como no Apêndice N é apresentado a configuração da camada HAL compatível com LinuxCNC, para controlar as sinais de controle do manipulador ASEA IRB6-S2.

Capítulo 2

Revisão Bibliográfica

2.1 Introdução

No presente capítulo é apresentada a fundamentação teórica associada à implementação da técnica *retrofitting* em manipuladores industriais com a descrição específica das equações homogêneas da cinemática direta e inversa do manipulador ASEA IRB6-S2. São apresentados trabalhos associados à aplicação *retrofitting* em robôs industriais, generalidades dos manipuladores, modelagem matemática da estrutura física de um manipulador industrial, assim como os controladores usados especificamente em robôs antropomórficos e as características da plataforma de controle LinuxCNC.

2.2 Arquiteturas CNC Abertas: Estado da Arte

Segundo o comitê técnico da IEEE, um sistema aberto pode-se definir como um sistema no qual é possível implementar aplicações para ser integradas e executadas em várias plataformas e para vários fornecedores. Esses sistemas devem ser integrados com outras aplicações *software* e *hardware* já existentes, além de proporcionar interfaces de usuário consistentes (IEEE, 1995). Para definir formalmente um sistema aberto são apresentadas a seguir quatro definições (HASCOET; RAUCH, 2016):

- Portabilidade: é a habilidade modular de uma aplicação a ser transferida e usada em diferentes plataformas sem modificações enquanto conserva a integridade das capacidades próprias do módulo. Esses tipos de módulos são feitos em blocos, com informação básica para executar tarefas específicas. São usadas por outras funcionalidades de programas complexos.
- Interação: faz referência a capacidade de incluir qualquer módulo de uma aplicação a um sistema, evitando diminuir suas capacidades ou a geração de conflitos entre o sistema e módulo.
- Interoperabilidade: é a característica que representa a interação dos módulos de uma aplicação no sistema. É necessário garantir o intercâmbio de informação em um padrão definido.
- Escalabilidade: é a habilidade para interpretar os requerimentos do usuário e a implementação do

sistema. O desempenho dos módulos da aplicação segundo as características do *hardware* podem ser adaptados conforme a complexidade do sistema.

A característica de interoperabilidade é considerada indispensável em sistemas de arquitetura aberta baseadas em tecnologia de controle numérico, por causa de um trabalho eficiente e um intercambio de dados com uma lógica definida entre os componentes de um sistema CNC (YU et al., 2009).

A implementação de arquiteturas abertas de controle baseadas em CNC esta ganhando relevância como tecnologia para a automação industrial, devido ao que permite a integração de máquinas, interfaces pre-definidas para a configuração e aprimoramento na comunicação da máquina. Muitos tipos de arquiteturas abertas estão sendo desenvolvidas nos Estados Unidos, Europa e Ásia, usando um computador pessoal para o controle do sistema (HASCOET; RAUCH, 2016). São descritas a seguir duas OAC que junto com LinuxCNC proveem ambientes flexíveis de desenvolvimento para novos projetos baseados em tecnologias de comando numérico.

A arquitetura **OSEC** foi criada em 1994 para estabelecer uma arquitetura aberta para automação industrial no Japão. O objetivo principal da arquitetura é o controle dos equipamentos de manufatura enquanto o desempenho incrementa e a manutenção é simples. A principal vantagem desta plataforma é que o sistema não é definido nem limitado, o que possibilita a implementação de varias opções na configuração compatíveis com *driver*, comunicação entre os processos, bibliotecas estáticas, placas de controladores etc. Desta forma a plataforma permite ser adaptada a qualquer requerimento do usuário final (FUJITA; YOSHIDA, 1996).

A OAC chamada **HOAM-CNC** foi proposta como uma arquitetura para a implementação de algoritmos de maquinas. É focado na integração *hardware* em um projeto modular (ASATO et al., 2002). Desta forma para a parte *hardware* da maquina utiliza dois barramentos: um para o controle adaptativo da maquina; e o segundo permite o controle em tempo real dos eixos através de *drives*. Enquanto a parte *software* é dividida em duas tarefas para controlar a trajetória da ferramenta e para monitorar os processos ativos da maquina (HASCOET; RAUCH, 2016).

Na Tabela 2.1 é apresentada uma comparação das características internas das OAC baseadas em tecnologias CNC, onde a plataforma LinuxCNC (EMC2) é escolhida como a melhor opção pelas facilidades de integração e a modularidade disponíveis para controlar maquinas NC, além de ter uma licença GNU que reduz os custos da implementação.

2.3 *Retrofitting* em Robôs Industriais

Os robôs começaram a fazer parte da cadeia produtiva das empresas na década dos anos 70, sendo normal ou habitual a troca por novos robôs quando a tecnologia deles estivesse desatualizada, no momento que o serviço de manutenção dos fornecedores terminasse ou simplesmente como estratégia para incrementar os indicadores de produtividades. Para as empresas de maior porte é factível, pela facilidade econômica para melhorar os processos produtivos, porém, para pequenas e medias empresas é um pouco mais difícil, fazendo com que o *retrofitting* para robôs industriais tivesse prioridade antes de se pensar em adquirir ou trocar um manipulador.

Tabela 2.1: Comparação plataformas de tecnologias de comando numérico abertas (*Open CNC*)

Critério	OSEC	JOP	HOAM-CNC	OSACA	OMAC	FSA	EMC2	FSMC
Sistema operacional	Windows	Windows	Windows e outros	Windows e outros	Windows e outros	Windows e outros	Linux	Windows e Linux
Tipo de arquitetura	Baseada em componentes	Wrapper	Cliente Servidor	Cliente Servidor	Baseada em componentes	Baseada em componentes e Cliente Servidor	Hierárquica e Baseada em componentes	Cliente Servidor e Baseada em componentes
Distribuída	Não	Não, paralela	Não, paralela	Sim	Sim	Sim	Sim	Sim
Infraestrutura definida	Não	Não	Não	Sim	Não	Sim	Sim	Sim
Nível de modularidade	Baixo	Médio	Alto	Médio	Médio	Muito alto	Alto	Alto
Arquitetura AM	Modular	Modular	Modular	Blocos modulares	Blocos modulares	Modular	Modular, Caixa-preta	Modular, Caixa-preta
Módulos identificados	7	4	6	9	14	7	5	11
Descrição modular	Não definida	Não definida	Parcialmente definida	Não definida	Definida com FSM	Definida	Definida	Definidas parcialmente
Tipo de API	Wrapper	Wrapper	Wrapper	Funções	Orientado a objetos	Orientado a objetos	Orientado a objetos, Wrapper	Orientado a objetos
Vantagens API	Fácil de entender	Fornecedor neutral	Nível de modularidade	Nível de modularidade	Agentes proxy para implementar	Alta modularidade e interconectividade	HAL efetivo	Alta modularidade e interconectividade
Desvantagens	Não orientado a objetos, muito simples	Sem desenvolvimento	Parcialmente definido	Variáveis ao invés funções	API complexa	Alta demanda de latência	Modelos de caixas pretas para AM	Módulos não definidos
Linguagem de programação	C/C++	C/C++	C/C++	C/C++	C/C++,Java	C/C++	C/C++	C/C++/Delphi
Plataforma de comunicação	DLL específicos	Não definido	Não definido	Hierárquica	Não definida	OPC	NML	NML, DLL e OPC
Interesse atual	Muito baixo	Muito baixo	Médio	Baixo	Alto	Alto	Alto	Muito alto

Fonte: (HASCOET; RAUCH, 2016)

Desde os anos 90 a atualização tecnológica desses tipo de equipamento industrial foi evoluindo de sistemas analógicos a digitais, aproveitando que a estrutura mecânica dos robôs em mais de quarenta anos de evolução tecnológica não mudou significativamente, as melhoras deste tipo de manipuladores industriais estão ligadas especificamente aos sensores das juntas para melhorar o posicionamento final do efetuador (LAGES; BRACARENSE, 2003).

O uso de sensores de dados associados a processos o maquinas físicas como uma representação virtual em tempo real contribui a ter sistemas de produção robustos, resistentes a perturbações externas, assim

como supervisão em tempo real para predição e correção de falhas (CHUKWUEKWE et al., 2016). Deste modo, sistemas de manufatura modernos estão integrados por interfaces de comunicação compatíveis para o envio de dados através de padrões específicos, no entanto para máquinas e equipamentos obsoletos esta opção é limitada ou inexistente. Desta forma o *retrofitting* para esse tipo de sistemas antigos deve permitir a integração de interfaces de comunicação compatíveis com tecnologias associadas a Indústria 4.0 como conceitos tais como o Internet das Coisas (IoT), Computação em Nuvem ou serviços Web (HORN; KRÜGER, 2016).

Um dos principais gestores da implementação do *retrofitting* para robôs industriais está na Academia, onde os projetos ligados com as linhas de pesquisa dos departamentos vinculados ao setor da automação industrial estão apresentando e validando as vantagens e facilidades, para que empresas pudessem ter interesse na compra de manipuladores industriais usados, novos, ou simplesmente em obter controle total das características dos sistemas robóticos sem depender das arquiteturas de controle fechadas ou padrões proprietários desenvolvidos pelos fornecedores (TOQUICA; ÁLVARES, 2016).

São apresentados a seguir trabalhos associados ao uso da técnica *retrofitting*, especificamente baseados em arquitetura abertas de controle que garantem a integração de novos requerimentos tecnológicos ou incorporam componentes com padrões de comunicação proprietários.

2.3.1 Análise dos Sensores para o *Retrofitting* de um Robô Industrial

Um software foi projetado com Java e C++ para ser integrado com uma arquitetura aberta, em um computador com sistema operacional Linux e uma variante para o processamento em tempo real com o *kernel* RTAI. As etapas para aplicar a técnica *retrofitting* no projeto foram baseadas no diagnóstico e verificação do sistema mecânico do manipulador, além da atualização do sistema eletrônico/controle. Adicionalmente, o *retrofitting* aplicado gerou como resultado adicional a comparação dos sensores de velocidade e posição originais do manipulador em relação à tecnologia digital, concluindo que os *encoder*, digitais garantem uma maior precisão e fácil integração com a arquitetura de controle proposta (Lima II et al., 2004).

2.3.2 Implementação de *Retrofitting* de Controladores de Equipamentos Industriais

A técnica *retrofitting* permite melhorar a produtividade dos processos em uma linha de produção industrial em máquinas suscetíveis de serem atualizadas tecnologicamente, através da otimização do relacionamento das variáveis do sistemas de controle, supervisão, HMI, ou simplesmente melhorando a programação dos equipamentos que gerenciam as atividades executadas pelas máquinas. Desta forma, a técnica *retrofitting* foi aplicada em duas máquinas industriais, aumentando os indicadores de produção da empresa. Uma das máquinas tinha a responsabilidade da conta de jornais e a segunda responsável por arrumar garrafas de vidro em caixas plásticas (RIBEIRO et al., 2007).

2.3.3 Atualização de *Hardware* e *Software* de um Robô Industrial

Um dos projetos baseados no *retrofitting* no setor acadêmico brasileiro foi aplicado em um robô ABB-IRB2000 composto de seis graus de liberdade (DOF), e com o desenvolvimento de um software com

capacidade de gerenciar uma rede distribuída de transmissão de dados ligados ao controle do robô. O projeto propõe uma arquitetura de controle aberta e flexível onde o gerenciamento dos atuadores das juntas é conectado a uma rede CAN-Bus e ligada a um computador, permitindo dessa forma centralizar a informação dos componentes do robô. Mediante uma interface de usuário feita em Simulink/Matlab.

Na camada superior da arquitetura está disponível o sistema de controle cinemático do robô desenvolvido no Matlab, permitindo calcular os parâmetros das variáveis cinemáticas, assim como a transmissão de dados na rede distribuída para os atuadores (LIMA et al., 2010). A parte software do projeto propõe módulos de controle personalizados para supervisão dos parâmetros do robô, tal como a região de trabalho, o controle da velocidade em cada uma das juntas do robô, parâmetros cinemáticos, dentre outros.

Como característica adicional, foi integrado um dispositivo Wiimote (Controle de nintendo-wii) para comandar o punho do manipulador. O projeto possui uma arquitetura flexível, o que possibilita a integração de outros robôs ou dispositivos compatíveis com a rede distribuída proposta como parte da aplicação do *retrofitting*. Na Figura 2.1 se apresenta arquitetura de controle mencionada anteriormente.

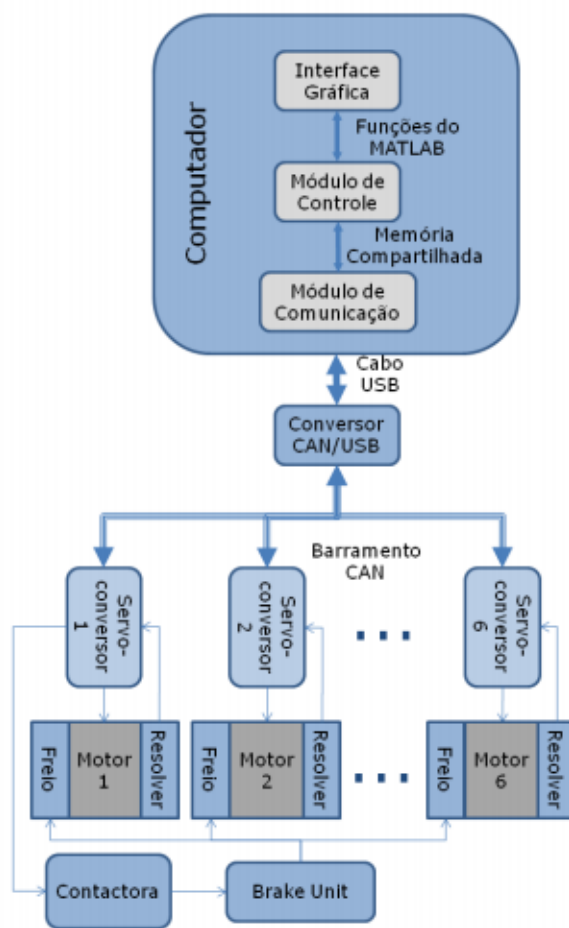
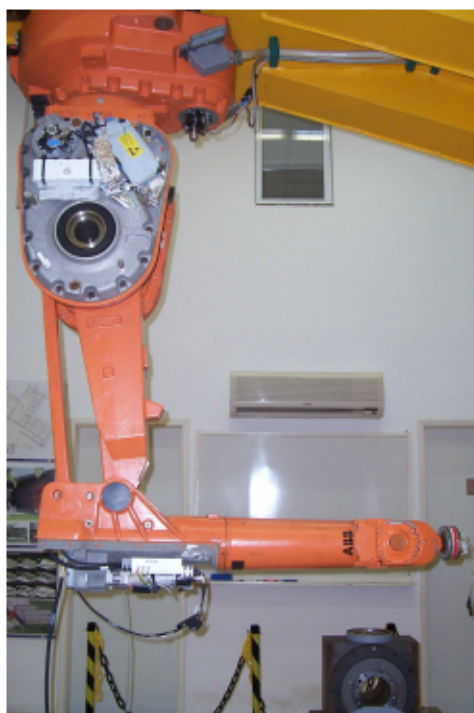


Figura 2.1: Arquitetura de controle para ABB IRB2000
 Fonte: (LIMA et al., 2010)

2.3.4 Controlador de Robôs com Arquitetura Aberta Aplicado às Tarefas de Interação

Um modelo de arquitetura aberta de controle para robôs foi baseado na norma ISO 7498-1 (camadas OSI) implementado em sistemas embarcados, onde a programação das tarefas de controle fora feita em linguagem alto nível e posteriormente compiladas para processadores DSC. Desta forma, foi possível controlar a movimentação dos seis graus de liberdade do manipulador. A capacidade modular desta proposta possibilita a integração e execução de tarefas de cooperação com outros robôs, devido a definição de cinco camadas tipo OSI. Com a proposta da arquitetura de controle modular é possível a integração de novas funcionalidades e melhoras na estrutura de cada uma das camadas propostas (OLIVEIRA A.; MORENO, 2010). Na Figura 2.2 é apresentada a arquitetura de controle.

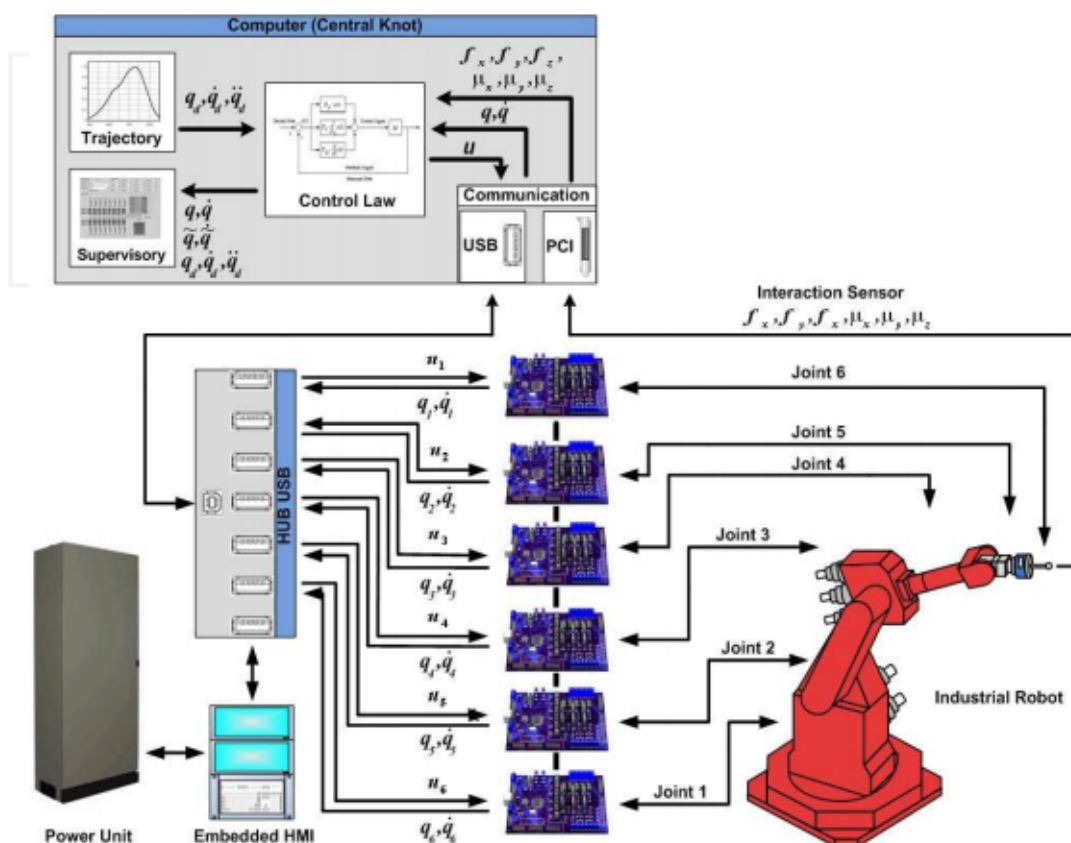


Figura 2.2: Arquitetura de controle baseada em OSI para REIS-Rv15
Fonte: (OLIVEIRA A.; MORENO, 2010)

2.3.5 Sistema de Usinagem Robótica Controlado como Máquina-Ferramenta

Através de uma arquitetura aberta de controle baseada na plataforma LinuxCNC foi aplicada a técnica *retrofitting* na parte do controlador *software*, devido a estrutura mecânica do robô estar em boas condições não foi necessário trocar nenhum componente *hardware* do manipulador LOLA-50. O projeto desenvolvido permite a simulação do robô enquanto é executado o processo de usinagem de peças em material de baixa densidade (MILUTINOVIC, 2011).

O trabalho apresenta a integração da cinemática direta e inversa do manipulador de cinco graus de liberdade com o sistema CAD/CAM por meio do controle com LinuxCNC, substituindo o *software* original fornecido com a compra do manipulador. Na Figura 2.3 se apresenta a configuração física do robô controlado com uma arquitetura tipo *Open Source* compatível com máquinas de comando numérico.



Figura 2.3: Arquitetura de controle para LOLA 50
Fonte: (MILUTINOVIC, 2011)

2.3.6 Arquitetura Aberta para *Retrofitting* de Robôs

Uma aplicação adicional na implementação da técnica *retrofitting* foi desenvolvida com a motivação de procurar alternativas aos altos custos da compra de novos robôs, além dos protocolos e interfaces de comunicação com lógicas fechadas pelos fornecedores como uma estratégia de comercialização, dificultando a integração dos manipuladores com padrões diferentes aos fabricados pelas corporações de robótica industrial (LAGES; BAYAN; Q., 2012). Desta forma, foi apresentada uma arquitetura aberta para o controle do manipulador ASEA IRB6-S2 desenvolvida inicialmente para um robô antropomórfico.

A arquitetura propõe módulos de processamento distribuído para o manipulador, onde o sistema é integrado por dois barramentos de campo para o controle em tempo real dos parâmetros cinemáticos do manipulador e para a aquisição de dados de supervisão, com barramentos tipo CAN e Ethernet respectivamente. O *software* de controle foi desenvolvido com o *kernel* em tempo real do Linux chamado RTAI, permitindo através de módulos desenvolvidos em C++ e Java o controle do manipulador. Na Figura 2.4 se apresenta a arquitetura mencionada anteriormente.

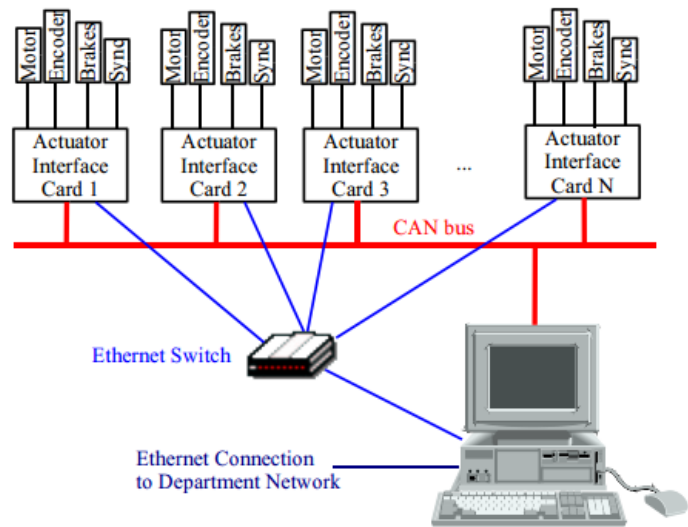


Figura 2.4: Arquitetura de controle para
 Fonte: (LAGES; BAYAN; Q., 2012)

2.3.7 Remanufatura de Manipuladores Robóticos com Arquitetura Aberta

Os custos da inversão para a implementação do *retrofitting* são aproximadamente 14% do valor que atualmente pode ter um robô manipulador com características semelhantes, no mercado brasileiro. Foi possível aproveitar a estrutura mecânica do manipulador ASEA IRB6-S2 trocando os componentes obsoletos de controle e alimentação com a tecnologia moderna, sendo que alguns desses componentes estavam danificados no momento do diagnóstico do sistema geral do manipulador (BOMFIM et al., 2014).

Nesta proposta com o uso da técnica *retrofitting* foram integrados em uma mesma solução, dois *software* compatíveis com o sistema operacional Windows. O *software* Matlab foi usado para calcular os valores das variáveis cinemáticas do manipulador, enquanto o *software* Mach3 foi usado para a geração dos sinais de controle enviadas aos *drives* dos motores da cada junta. A seleção destes *software* licenciados foi motivada pela facilidade na configuração da arquitetura proposta e a flexibilidade da interface de usuário dos *software* anteriormente mencionados. Na Figura 2.5 é apresentada a arquitetura de controle mencionada.

2.3.8 Simulação de Manipulador Serial: Cinemática e Implementação em LinuxCNC

O modelamento cinemático direto e inverso, e a simulação de um manipulador antropomórfico com seis graus de liberdade foi feita com a plataforma de controle baseada em licença GNU, LinuxCNC. Com o resultado do projeto foi gerada a simulação da trajetória percorrida pelo efetuador do manipulador virtual, modelada no simulador Vismach, baseada na API livre OpenGL. Um arquivo em código G permite enviar através do interpretador da norma RS-274 da plataforma de controle as coordenadas da posição final do efetuador (PREEZ, 2014).

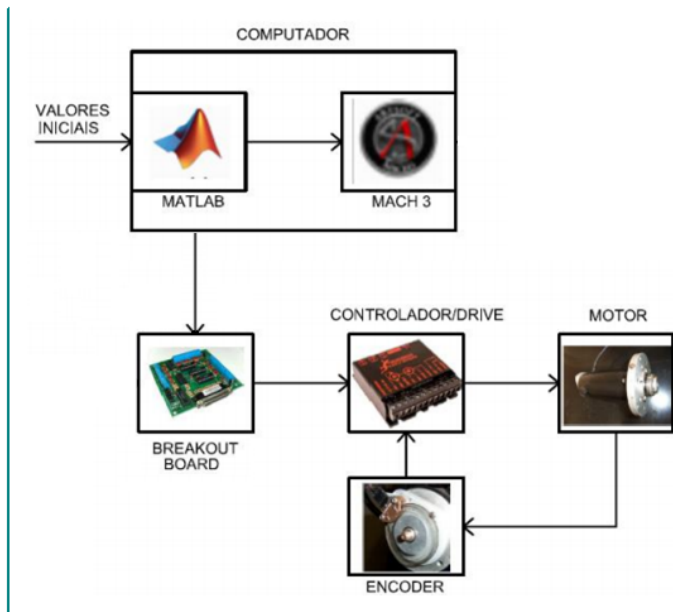


Figura 2.5: Arquitetura controle para ASEA IRB6-S2
 Fonte: (BOMFIM et al., 2014)

Embora o projeto feito não tenha especificado o uso físico de um manipulador, LinuxCNC está sob a normas de código aberto, o que permite tipificar este trabalho como *retrofitting* do sistema *software* de um manipulador serial. Na Figura 2.6 é apresentado o modelo do robô na plataforma LinuxCNC.

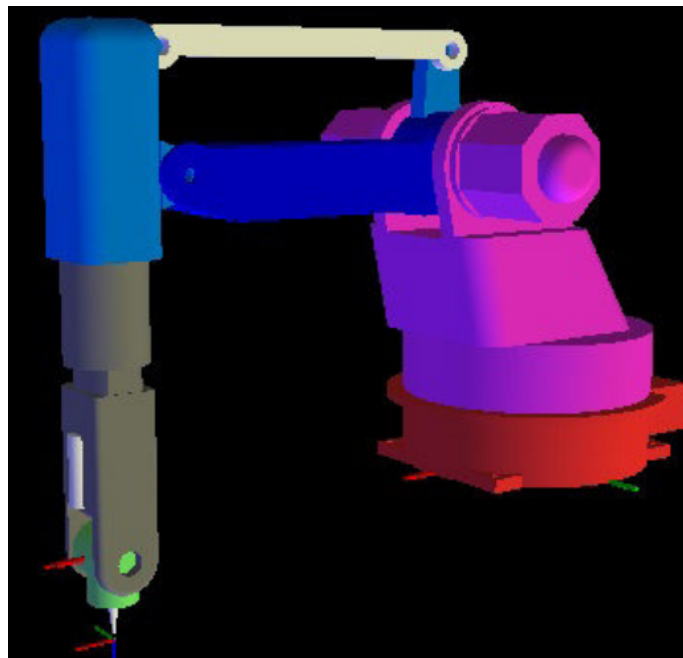


Figura 2.6: Robô virtual em LinuxCNC
 Fonte: (PREEZ, 2014)

2.4 Plataformas de Controle para Robôs Avaliadas no Trabalho

Atualmente existem diversas plataformas para o controle de robôs proprietários e licenças abertas ao público. Para o escopo do presente trabalho foram propostas algumas plataformas com características específicas como: o controle de manipuladores industriais; compatibilidade com sistemas CAD/CAM, assim com o atributo de arquitetura aberta para a integração com outros sistemas *hardware* e *software*. A seguir se apresenta de forma geral as características mais relevantes das plataformas propostas, permitindo ter uma perspectiva integral no momento escolher a plataforma adequada para o *retrofitting* do manipulador ASEA IRB6-S2.

2.4.1 LinuxCNC

Esta plataforma foi concebida inicialmente para controlar máquinas CNC. Depois do desenvolvimento de novas características através de uma comunidade dedicada ao suporte da plataforma, foi possível ao LinuxCNC contar com a capacidade de controlar fresadoras, tornos, máquinas para impressão 3D, robôs móveis e manipuladores etc (MILUTINOVIC, 2011).

A plataforma possui uma arquitetura aberta baseada em Linux, e a instalação é através de modo *Live*, interpretador integrado de programas NC, e compatibilidade com linguagens de programação sob termos GNU, tem a capacidade de controlar até nove eixos através de várias interface de comunicação. Compatibilidade com as distribuições Ubuntu e Debian.

A documentação encontra-se em desenvolvimento pela comunidade. Existe pouca informação para a integração com manipuladores industriais ou outro tipo de robôs com cinemáticas não triviais.

2.4.2 Mach3/Matlab

Mach3 é um programa com interface intuitiva, projetado para controlar máquinas CNC como fresas, tornos e cortadores. Sua função é interpretar o programa em código G e gerar pulsos para os *drives* dos atuadores. **Matlab** é um software de simulação matemática que realiza operações matriciais, construção de algoritmos baseados em bibliotecas tipo *toolbox*, processamento de sinais por meio de uma interface de usuário depurada ao longo de mais de trinta anos no mercado.(BOMFIM et al., 2014)

Integração em uma solução para o cálculo dos valores das variáveis articulares do manipulador e a geração de sinais de controle para cada uma das juntas, *softwares* compatíveis com algumas versões do sistema operacional Windows, gerando alta aceitação pela comunidade relacionada com sistema CNC e robótica (BOMFIM, 2013).

A plataforma **Mach3** é compatível exclusivamente com Windows, o que limita a integração com outros sistemas operacionais. Contudo a solução deve ter licença do sistema operacional e dos *softwares* **Mach3/Matlab**, aumentando o custo do *retrofitting* para manipuladores. Esta solução não trabalha em tempo real o que gera atrasos no cálculo da cinemática e na transmissão de sinais de controle.

2.4.3 ROS - Sistema Operacional para Robôs

O ROS foi desenvolvido em Linux, o que garante uma arquitetura aberta para a integração com dispositivos físicos proprietários. É um *framework* flexível para desenvolver software de controle para robôs. Tem uma coleção própria e com licenciamento BSD de ferramentas, bibliotecas e convenções para desenvolver tarefas complexas em robótica.

Compatibilidade com a maior parte de linguagens de programação para o controle e modelo virtual dos robôs. Concepção modular que permite integração com componentes tipo *software* e *hardware*. Possui uma lógica cliente/servidor que permite saber o estado dos robôs que são controlados.

Obriga ao usuário a aprender uma série de comandos específicos para usar a plataforma corretamente. Não é recomendável usar ROS se a aplicação final do robô precisa de sincronização para comunicação em tempo real (MARTINEZ; FERNÁNDEZ, 2013).

2.4.4 iGETRobot

É uma plataforma integrada com Mach3, na qual existe pouca informação das características da solução para o controle de manipuladores, porque está sendo desenvolvida para posteriormente ser comercializada. A solução integra o interpretador da norma RS-274 para a leitura do arquivo NC. A simulação do modelo virtual do manipulador é executado em paralelo com o controle do robô. Possibilita a integração de cinemáticas não triviais (HARI-Team, 2016).

Essa integração com Mach3 não está em capacidade de trabalhar com aplicações que requeiram resposta em tempo real. Aumentará os custos do *retrofitting* por causa do licenciamento do sistema operacional que é Windows, Mach3 e iGETRobot quando inicie a comercialização.

2.4.5 Step-NC Machine

O Step-NC Machine é uma plataforma para gerar programas de usinagem baseados em modelos com dados de fabricação sob a norma STEP-NC. (ŽIVANOVIĆ; GLAVONJIĆ, 2014). Permite modificar e simular programas STEP-NC em um mesmo ambiente de trabalho, através de máquinas virtuais disponíveis no *software* como Fanuc, Siemens, Okuma, Haas etc.

Trabalha com o sistema operacional Windows o que aumenta o custo do *retrofitting*, além de não trabalhar em tempo real se a aplicação precisasse deste tipo de característica.

2.4.6 Adept

A plataforma possui soluções a nível *software* e *hardware* para o controle de robôs através de um sistema operativo proprietário intitulado V+, focado no controle da movimentação. Tem uma arquitetura lógica de controle para a integração de qualquer configuração de robôs, compatível com a maioria dos padrões de comunicação. O sub-sistema *software* de controle esta em capacidade de trabalhar em tempo real com multi-tarefa (GOTO, 2011).

É uma plataforma proprietária que aumentará o custo do *retrofitting*. O conceito de arquitetura aberta é relativo devido ao controlador *hardware*, o *software* e a linguagem de programação serem licenciados, o que torna fechada a arquitetura de controle.

2.4.7 4DIAC - Framework for Distributed Industrial Automation and Control

A plataforma propõe um estilo de programação baseado na Norma IEC 61499, a qual define uma arquitetura para sistemas distribuídos. Desta forma, o escopo da plataforma é ter uma estrutura de controle flexível para a integração com padrões de comunicação industriais (STRASSER; ROOKER; EBENHOFER, 2008). Está integrada de duas partes: a primeira é responsável por controlar plataformas *hardware* em tempo real como algumas marcas de PLC, assim como Raspberry Pi e LEGO Mindstorms NXT; A outra parte é usada como ambiente de programação por blocos funcionais e pode ser associada com outros *software* de controle.

Possui limitações no uso de algumas plataformas *hardware* de controle. Projeto em desenvolvimento que precisa investimento de tempo para o entendimento da plataforma com documentos ainda em desenvolvimento (EBENHOFER et al., 2013).

2.4.8 Microsoft Robotics Developer Studio

Ambiente de desenvolvimento para o controle de robôs, dispõe de protótipos e ambientes predefinidos. É compatível com plataformas proprietárias tais como MindStorm NXT, LEGO, KUKA, AIBO etc. A lógica de programação para a geração dos algoritmos de controle é feita com blocos funcionais por meio de uma interface de usuário integrada com a plataforma, o que facilita a interação com a solução (KANG et al., 2016).

É uma plataforma proprietária da corporação Microsoft, em consequência deve trabalhar com o sistema operacional Windows o que aumenta o custo do *retrofitting*, além de ter a compatibilidade limitada com alguns padrões de comunicação.

2.5 Controlador LinuxCNC

O controlador foi inicialmente conhecido como EMC2 (Enhanced Machine Controller) e atualmente é distribuído como LinuxCNC. Foi desenvolvido pela NIST (National Institute of Standards and Technology) nos Estados Unidos e posteriormente foi disponibilizado ao público em 2002. Com acesso livre para modificar o código fonte do LinuxCNC, nasceu uma comunidade de desenvolvedores que tornou o projeto com uma licença GNU, e ainda hoje, a comunidade continua melhorando o controlador para máquinas baseadas em cinemáticas triviais e não triviais (PREEZ, 2014).

A estrutura lógica do controlador em alto nível é apresentada na Figura 2.7, integrada por quatro módulos com tarefas independentes. Uma novidade nesta plataforma é a camada HAL (em inglês: Hardware Abstraction Layer), simplificando o controle via *software* da interligação da leitura e processamento de sinais dos componentes *hardware* de um sistema.

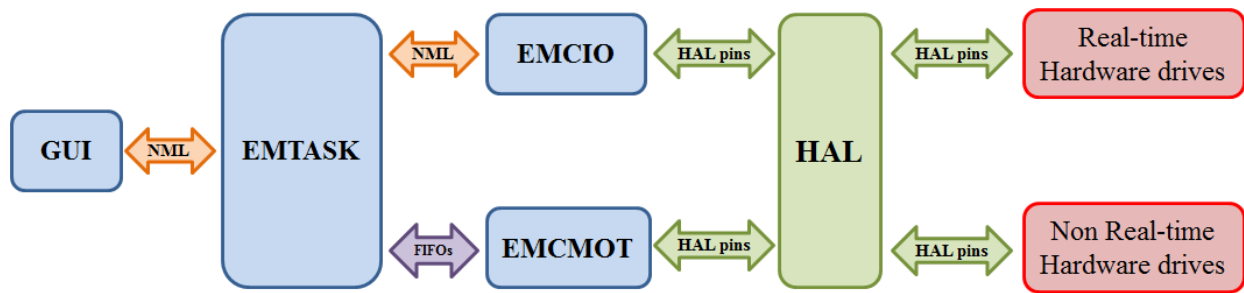


Figura 2.7: Arquitetura Modular do Controlador LinuxCNC
 Fonte: (TOQUICA; ÁLVARES, 2016)

O primeiro módulo chamado "*GUI*" que em termos gerais representa a interface gráfica com o operário de LinuxCNC para possibilitar a supervisão e controle da máquina. O seguinte módulo é o **EMTASK** responsável por coordenar as atividades entre variáveis discretas (**EMCIO**), e o de controle cinemático (**EMCMOT**) para os módulos correspondentes. O interpretador da norma RS-274 dos arquivos NC faz parte do módulo **EMTASK**. O módulo **EMCIO** é responsável por gerenciar as funções diferentes ao controle dos eixos da máquina, por exemplo, variáveis como a troca de ferramenta, lubrificante, parada de emergência etc. (STAROVESKI et al., 2009).

O módulo **EMCMOT** é o responsável pelo controle em tempo real da movimentação dos eixos da máquina, ou seja, com o cálculo das variáveis cinemáticas da máquina. É executado de forma contínua com o uso do *kernel* em Linux baseado em respostas em tempo real, além de definir as equações cinemáticas através de linguagem C, é possível com este modulo definir as características das juntas como parâmetros dinâmicos associados a velocidade e aceleração, as unidades ou a relação entre os dispositivos para o feedback da posição (STAROVESKI et al., 2009). Na Figura 2.8 se apresenta a estrutura interna do módulo mencionado anteriormente.

Uma das camadas fundamentais na estrutura lógica do controlador LinuxCNC é a **HAL**, está integrada em blocos funcionais que podem ser interligados para consolidar um sistema complexo. As conexões são feitas via *software* para facilitar o controle das tarefas realizadas nos módulos **EMCMOT** e **EMCIO**. Os componentes que podem ser interligados sob a camada **HAL** são *drives* de controle, motores, *encoders*, *pendants*, sensor fim de curso etc (LINUXCNC.ORG, 2015b).

Um dos maiores benefícios da camada HAL é a habilidade de controlar os módulos como caixas pretas para permitir o intercâmbio, modificação e troca sem ter que alterar o sistema de controle. Desta forma é possível simular e testar respostas do sistema com resultados aproximados a realidade, o que possibilita detectar e corrigir falhas que danifiquem os componentes do sistema *hardware* (HASCOET; RAUCH, 2016).

Na Figura 2.9 é apresentado o conceito básico do gerenciamento dos sinais na camada HAL, onde existe uma fonte do sinal a ser controlado que é armazenado em uma variável o sinal tipo *software* gerada dentro da camada, para posteriormente direcionar os dados a um ou múltiplos destinos. É necessário esclarecer que a fonte e destino do sinal podem ser elementos tipo *software* ou *hardware* com dados de controle para o sistema (LINUXCNC.ORG, 2015b).

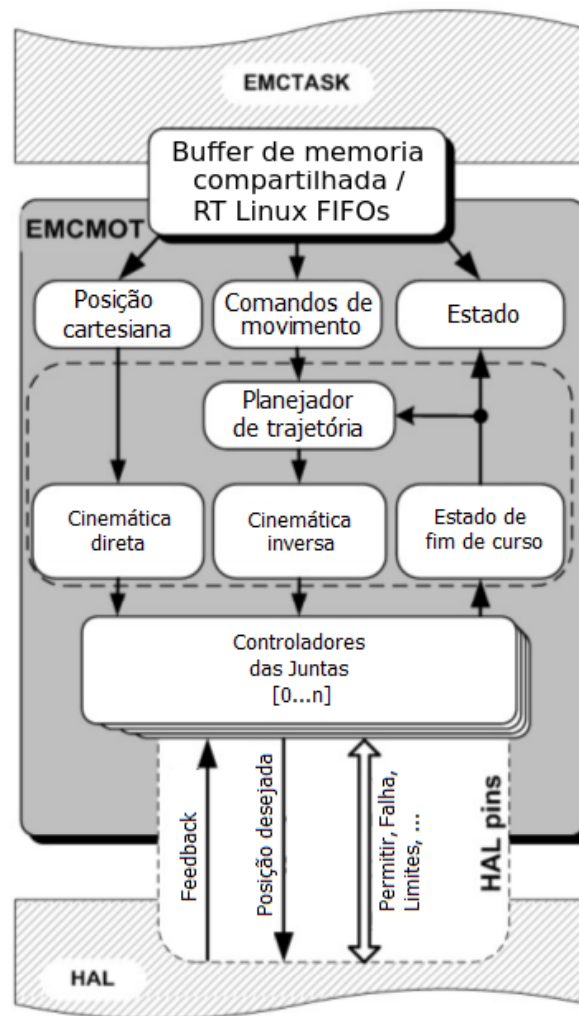


Figura 2.8: Módulo EMCMOT
 Fonte: (HASCOET; RAUCH, 2016)

No módulo **EMCTASK** está integrado o **intérprete NIST RS274NGC**, definido como um sistema software feito em C++ para ler um arquivo baseado em linguagem de controle numérico NC sob a norma RS274 para gerar funções específicas para o controle de máquinas. A saída do intérprete é usada para controlar sistemas de até seis eixos (KRAMER; PROCTOR; MESSINA, 2000), no caso do LinuxCNC está propriedade foi melhorada o que permite controlar ate nove eixos. Para o escopo do presente trabalho foi usado o interprete RS274 para interpretar arquivos gerados em CAD/CAM com as trajetórias a serem atingidas pelo efetuador do manipulador ASEA IRB6-S2.

O método de comunicação entre os módulos da plataforma LinuxCNC é através de NML (em inglês: Neutral Message Language), também conhecido como Linguagem Neutro de Manufatura. O sistema NML foi criado pela NIST para sincronizar sistemas robóticos feitos em C++. NML é uma API para a transmissão de funções, ocultando os detalhes de protocolos específicos para aos programadores (SHACKLEFORD; PROCTOR; MICHALOSKI, 2000). A unidade básica é o *buffer* de memória compartilhada onde é armazenada a informação (GOWDY, 2000). Na Figura 2.10 é apresentada a arquitetura básica NML para a transmissão de dados através da memória compartilhada.

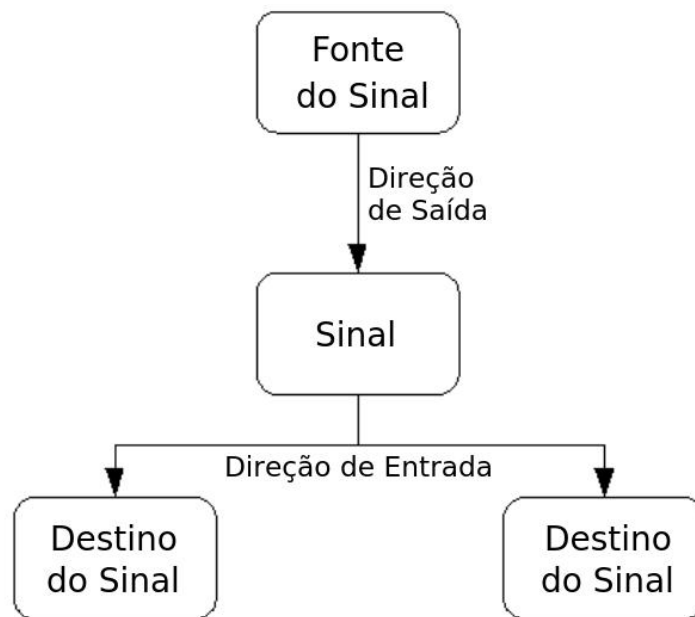


Figura 2.9: Definição básica camada HAL
 Fonte: (LINUXCNC.ORG, 2015b)

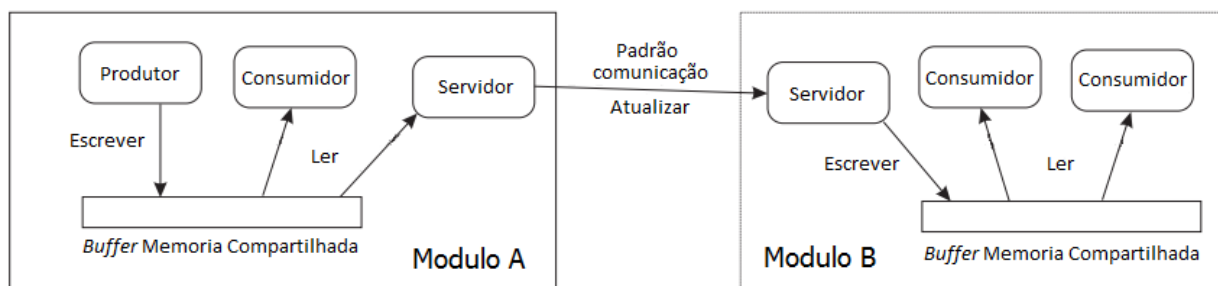


Figura 2.10: Módulo EMCOT
 Fonte: (SHACKLEFORD; PROCTOR; MICHALOSKI, 2000)

LinuxCNC dentro das arquiteturas abertas de controle não proprietárias tem se convertido em uma solução emergente para ser integrada como controlador de máquinas baseadas em comando numérico, devido a outros controladores semelhantes ainda estarem em desenvolvimento e a descrição das funcionalidades é limitada, ou existe dificuldade na integração com as funções próprias das máquinas. Desta forma LinuxCNC é uma opção factível como uma plataforma de controle baseada em arquitetura aberta com benefícios tais como o baixo custo do *hardware* compatível, aumento das funcionalidades da máquina, fácil configuração, integração com outros sistemas de controle e alto desempenho computacional (HUO; HONG; POO, 2015).

2.6 Manipuladores Industriais

Segundo a ISO 8373 o manipulador industrial deve ser controlado automaticamente, reprogramável e multitarefa, podendo ser programado em três ou mais eixos, sendo fixado em um local ou móvel, usado para aplicações industriais (International Federation of Robotics, 2005). No aspecto mecânico e estrutural do manipulador está composto de elementos chamados **elos** ou *links*, conectados através de **juntas** ou *joints*, identificando a primeira junta como a base do robô, e o extremo final denominado como **efetuador** (ROMANO, 2002).

As juntas do manipulador industrial comumente são classificadas em prismáticas e rotativas, o primeiro tipo associa um movimento linear entre dois elos, enquanto o segundo tipo de junta permite ter uma relação de rotação entre dois elos. No contexto de um robô manipulador é possível identificar matematicamente as juntas como variáveis independentes, o que permite conhecer a localização específica do mecanismo, comumente conhecido como grau de liberdade (em inglês: DOF - Degrees of freedom) (SPONG; HUTCHINSON; VIDYASAGAR, 2005). Os graus de liberdade são propriedades distribuídas ao longo da estrutura mecânica do manipulador, com um número suficiente para desenvolver uma tarefa determinada (SICILIANO et al., 2008).

A inserção dos robôs industriais no setor industrial global em 2014 foi um dos mais importantes na história da robótica, como resultado da venda de mais de duzentas mil unidades, principalmente em mercados como China, Japão, Estados Unidos, Coreá do Sul e Alemanha. Em contrapartida a Ásia é o maior fabricante de robôs do mundo, com a venda de 60 % no 2014. O setor mais importante no consumo de robôs industriais é o automotivo, com aproximadamente cem mil novas unidades instaladas, o que significa um aumento de 43% em comparação com o ano 2013 (CANGELOSI; SCHLESINGER; SMITH, 2015). Desta forma, a robótica industrial está tendo um crescimento exponencial com base no investimento para o setor produtivo, por parte dos países industrialmente desenvolvidos.

2.6.1 Classificação de Manipuladores Industriais

A estrutura de um manipulador industrial além das juntas interligadas entre si é caracterizada por um punho (*wrist*), dando habilidades ao manipulador. O efetuador é responsável por melhorar o desempenho do manipulador nas tarefas determinadas. Para estabelecer a posição e orientação do efetuador final do manipulador é necessário ter presente a sequência entre os graus de liberdade que compõem a configuração do robô, permitindo a classificação segundo as três primeiras juntas (DOF) do manipulador. As configurações comumente usadas são:

Cartesianos compostos de três juntas prismáticas, sendo perpendiculares entre si.

Cilíndricos a primeira junta é rotacional, enquanto as duas seguintes são prismáticas.

Esfericos as duas primeiras juntas desta configuração são rotacionais e a última é prismática.

SCARA esta configuração é semelhante a esférica, mas a diferença está em que os eixos são paralelos entre si.

Antropomórficos está integrado por três juntas de revolução, onde a primeira é perpendicular às outras duas seguintes. Estas duas últimas devem ser paralelas entre si.

Na Figura 2.11 são apresentadas as configurações básicas dos robôs industriais já mencionadas.

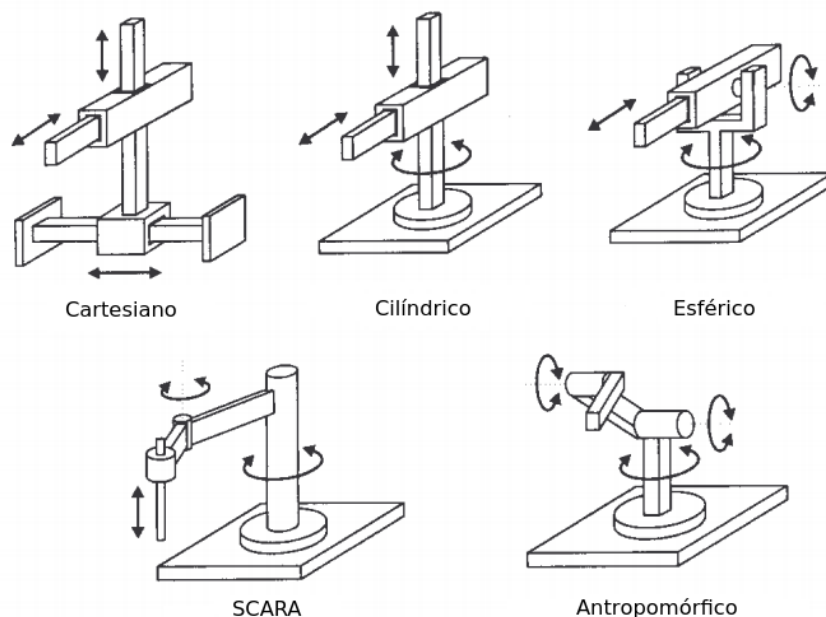


Figura 2.11: Tipos de robôs industriais
Fonte: (BARRIENTOS, 2007)

É importante mencionar que as configurações de manipuladores precisam do punho para garantir a orientação do efetuador num espaço de três dimensões, o punho do manipulador deve estar integrado no mínimo por três graus de liberdade, previstas de juntas rotacionais (SICILIANO et al., 2008).

2.6.2 Cinemática de Manipuladores Industriais

A cinemática é o estudo do movimento sem considerar as forças envolvidas, segundo a configuração da máquina é possível identificar dos tipos de cinemáticas: a primeira chamada **trivial** agrupando máquinas simples onde cada junta é localizada ao longo de um eixo cartesiano, facilitando o modelamento matemático. O segundo tipo é as máquinas com cinemática **não trivial** como robôs, onde cada junta não corresponde com as coordenadas cartesianas, desta forma é necessário usar funções matemáticas para estabelecer a relação (GUPTA; YU, 2001). Por conseguinte, o estudo da cinemática (não trivial) num manipulador industrial é a análise matemática da movimentação espacial do robô em relação a um sistema de referência, especificamente a relação entre a posição e orientação do efetuador do manipulador. A movimentação da estrutura do robô é o produto de movimentos elementares de cada **elo** em respeito ao anterior (SICILIANO et al., 2008).

A estrutura geral de um manipulador industrial é serial ou possui uma *cinemática aberta*, termo usado quando existe uma sequência de **elos** ligados formando uma cadeia com dois extremos. Enquanto a *ci-*

nemática fechada é composta de uma sequência de **elos** formando um ciclo ou laço (CRAIG, 2005). No contexto industrial, o manipulador deve ser analisado em um espaço de três dimensões com a posição e a orientação, de modo que uma cinemática aberta para um manipulador tenha normalmente seis graus de liberdade. Com menos de seis DOF o robô não terá a capacidade de atingir todos os pontos em seu ambiente de trabalho, porém, se existem mais de seis graus de liberdade, aumentará a dificuldade de controlar o manipulador (SPONG; HUTCHINSON; VIDYASAGAR, 2005).

Para o estudo da cinemática de um manipulador industrial existem duas perspectivas, a primeira é a **cinemática direta**, responsável por especificar a posição e orientação do efetuador, em respeito a um sistema de coordenadas de referência, conhecendo os valores articulares do manipulador. Enquanto a segunda perspectiva é a **cinemática inversa**, responsável por determinar os valores articulares das juntas conhecendo a posição e orientação do efetuador (BARRIENTOS, 2007). Na Figura 2.12 é apresentada a relação do estudo cinemático para manipuladores.

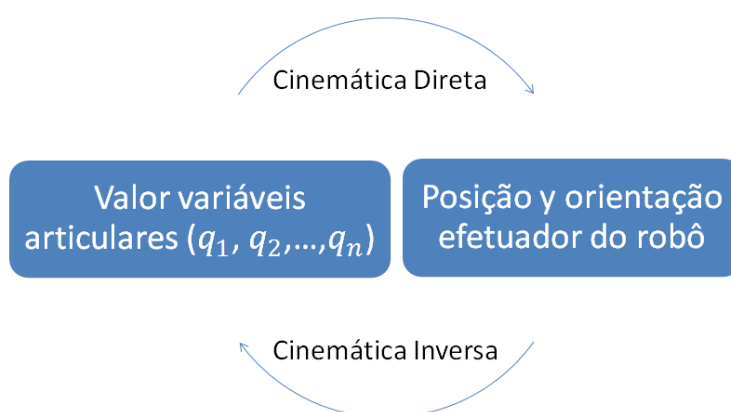


Figura 2.12: Sistemas de origem segundo a convenção DH
Fonte: (CRAIG, 2005)

2.6.2.1 Cinemática Direita

Um robô é integrado de elos ligados entre si através de juntas, determinando uma cadeia cinemática entre um sistema de referência vinculado à base do manipulador, e a localização dos elos em respeito ao sistema de referência da base. Em consequência a análise da cinemática direta do manipulador é descrita matematicamente achando uma matriz de transformação homogênea T , a qual relaciona a posição e orientação do efetuador segundo o sistema de referência inerente à base do robô (BARRIENTOS, 2007).

A posição de um corpo rígido no espaço é expressado fazendo referência a um sistema de coordenadas fixo em relação a um ponto específico do corpo rígido. Enquanto a orientação do mesmo é expressada em termos dos vetores unitários do sistema de coordenadas ligado ao corpo rígido com a referência do sistema de coordenadas fixa. Para descrever matematicamente um elemento do manipulador em termos de posição e orientação é necessário associar um sistema de coordenadas, permitindo a relação matemática com um sistema de coordenadas fixa (CRAIG, 2005). Na Figura 2.13 são apresentados sob a convenção do sistema cartesiano, a descrição matemática da orientação e posição de um elemento do manipulador associado a um sistema fixo de referência (*frame*).

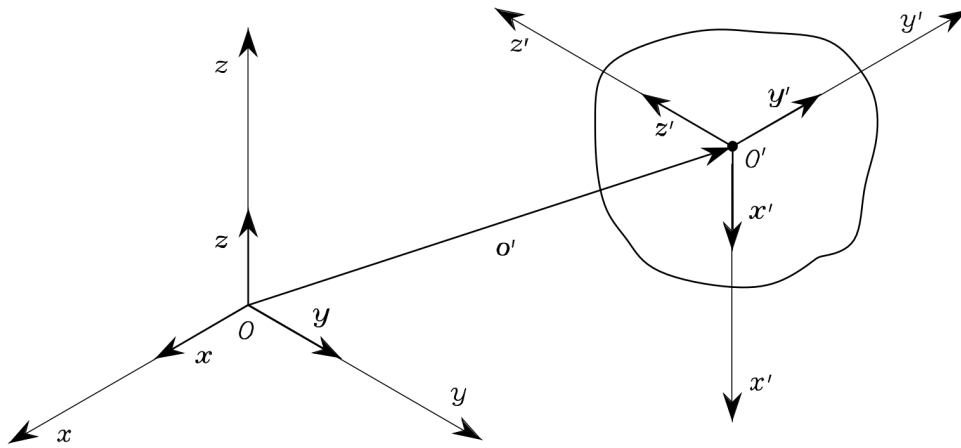


Figura 2.13: Posição e orientação de um elemento físico
a) Sistema de coordenadas fixo; b) Sistema coordenadas móvel
Fonte: (SICILIANO et al., 2008)

Para facilitar a relação entre dois sistemas de coordenadas que relaciona a posição e a orientação de um objeto no espaço é usado o conceito de **matriz de transformação homogênea** T (SICILIANO et al., 2008). Em robótica estas representações são usadas para relacionar dois elos consecutivos, e a representação geral desta matriz é apresentada na Equação 2.1.

$$T = \left[\begin{array}{c|c} \text{Rotação} & \text{Translação} \\ \hline \text{Perspectiva} & \text{Escala} \end{array} \right] \quad (2.1)$$

A representação de uma matriz de transformação homogênea para relacionar dois ou mais elos de um manipulador robótico, considerando a posição e orientação em referência a um sistema de coordenadas fixas, é apresentado na Equação 2.2.

$$H_1^0 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

Onde os vetores n , s e a representam a orientação do sistema de coordenadas móvel x_1, y_1, z_1 , em referência a um sistema fixo $o_0x_0y_0z_0$. O vetor p representa a posição do sistema móvel em relação ao sistema fixo de referência. Para facilitar a análise matemática da cinemática direta é necessário relacionar uma variável para cada uma das juntas do manipulador (SPONG; HUTCHINSON; VIDYASAGAR, 2005). Desta forma, para cada junta i_n é associado a variável q_i dependendo do tipo de junta, como é apresentado a seguir:

$$q_i = \begin{cases} \theta_i, & \text{se a junta } i \text{ é de revolução} \\ d_i, & \text{se a junta } i \text{ é prismática} \end{cases}$$

Para deduzir a cinemática inversa de um manipulador industrial é necessário desenvolver as **equações da configuração cinemática**. No caso do escopo do presente documento, é considerada a configuração de robôs antropomórficos onde as variáveis de junta (q_i) entre os elos são ângulos de rotação. É necessário um procedimento sistemático para a análise matemática da configuração cinemática do manipulador. A convenção comumente usada para a seleção de sistemas de referência em aplicações robóticas é chamada Denavit-Hartenberg, ou notação DH. Neste procedimento cada matriz de transformação homogênea A_i é representada como resultado de quatro transformações básicas apresentadas na Equação 2.3 (SPONG; HUTCHINSON; VIDYASAGAR, 2005).

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$A_i = \begin{bmatrix} \cos(\theta_i) & -\text{sen}(\theta_i)\cos(\alpha_i) & \text{sen}(\theta_i)\text{sen}(\alpha_i) & a_i\cos(\theta_i) \\ \text{sen}(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\text{sen}(\alpha_i) & a_i\text{sen}(\theta_i) \\ 0 & \text{sen}(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

Onde as variáveis a_i , α_i , d_i , θ_i são parâmetros da junta e do elo i . Geralmente são intitulados como: longitude do elo; torção do elo; compensação do elo e ângulo da junta, respectivamente (SPONG; HUTCHINSON; VIDYASAGAR, 2005). É necessário identificar os valores destas variáveis para gerar as matrizes A_i de cada um dos graus de liberdade do manipulador. Na Figura 2.14 é apresentada a distribuição das variáveis DH dos elos associados a junta i .

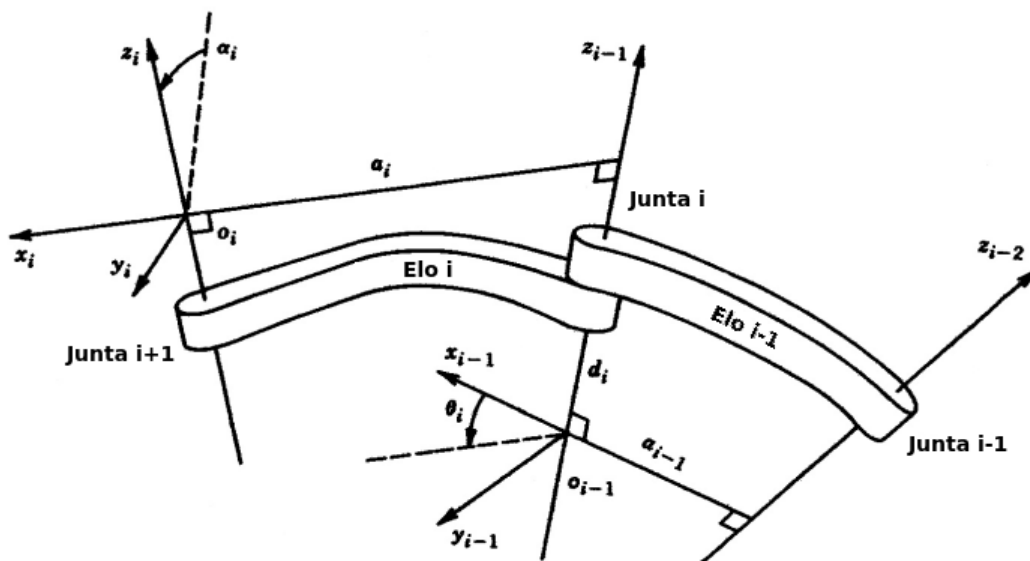


Figura 2.14: Definição do sistema de coordenadas segundo DH
Fonte: (CRAIG, 2005)

É apresentada a seguir a definição de cada uma das variáveis DH, facilitando a dedução dos valores (CRAIG, 2005):

- a_i , distância entre \hat{Z}_i e Z_{i+1} , ao longo do \hat{X}_i
- α_i , ângulo entre \hat{Z}_i e Z_{i+1} , ao longo do \hat{X}_i
- d_i , distância entre X_{i-1} e \hat{X}_i , ao longo do \hat{Z}_i
- θ_i , ângulo entre X_{i-1} e \hat{X}_i , ao longo do \hat{X}_i

O procedimento apresentado anteriormente permite deduzir os valores da matriz de transformação homogênea A_n^0 , fazendo uma associação ao sistema de coordenadas do efetuador em relação ao sistema de referencia de um manipulador com n graus de liberdade.

2.6.2.2 Cinemática Inversa

Na Subsubseção 2.6.2.1 foi considerado o problema para deduzir matematicamente a posição e orientação do efetuador de um manipulador industrial em relação a sua base. Nesta seção é descrito o procedimento para obter o valor das variáveis cinemáticas do manipulador (q_i) com base nas coordenadas de posição e orientação do efetuador. Existem dois métodos semelhantes para obter a solução da cinemática inversa do manipulador, o **algebráico** e **geométrico**. Os métodos diferem só da perspectiva de análises, segundo a configuração cinemática do manipulador (CRAIG, 2005).

Para o escopo do presente trabalho é apresentada a definição do método algebráico para obter as variáveis cinemáticas do manipulador, onde é possível encontrar termos analíticos com expressões polinômicas de quarto grau ou menores. A perspectiva algebráica consiste em comparar as equações da representação geral da matriz de transformação homogênea (Equação 2.2), e a matriz de transformação homogênea resultante da configuração cinemática do manipulador com a notação DH. Desta forma é possível ter um máximo de doze equações para seis incógnitas ou variáveis articulares, três variáveis referentes à posição e as restantes três associadas à orientação do efetuador.

2.6.3 Cinemática Manipulador ASEA IRB6-S2

Para ter o controle da posição e orientação do efetuador do manipulador ASEA IRB6-S2 em qualquer processo de manufatura é necessário definir um modelo matemático da estrutura física do robô. Esta definição matemática é possível através de equações associadas a cinemática direta e inversa do robô (SICILIANO et al., 2008). Permitindo desta forma estabelecer um modelo matemático adequado para identificar a posição das juntas do manipulador ASEA IRB6-S2 em referência a um sistema de coordenadas fixo. Para facilitar a definição matemática do manipulador é necessário definir com a notação DH os sistemas de coordenadas de cada elo do manipulador, tal como é apresentado na Figura 2.15. Desta forma é possível identificar os valores para as variáveis DH associadas ao manipulador ASEA IRB6-S2.

Com a definição dos sistemas de coordenadas associadas aos cinco graus de liberdade do manipulador segundo a convenção DH, é possível determinar os valores DH para o manipulador ASEA IRB6-S2. É necessário esclarecer que os valores das variáveis a_i e d_i associadas as distâncias entre os sistemas de coordenadas das juntas são fornecidos pelo fabricante do manipulador (SZKODNY, 1995).

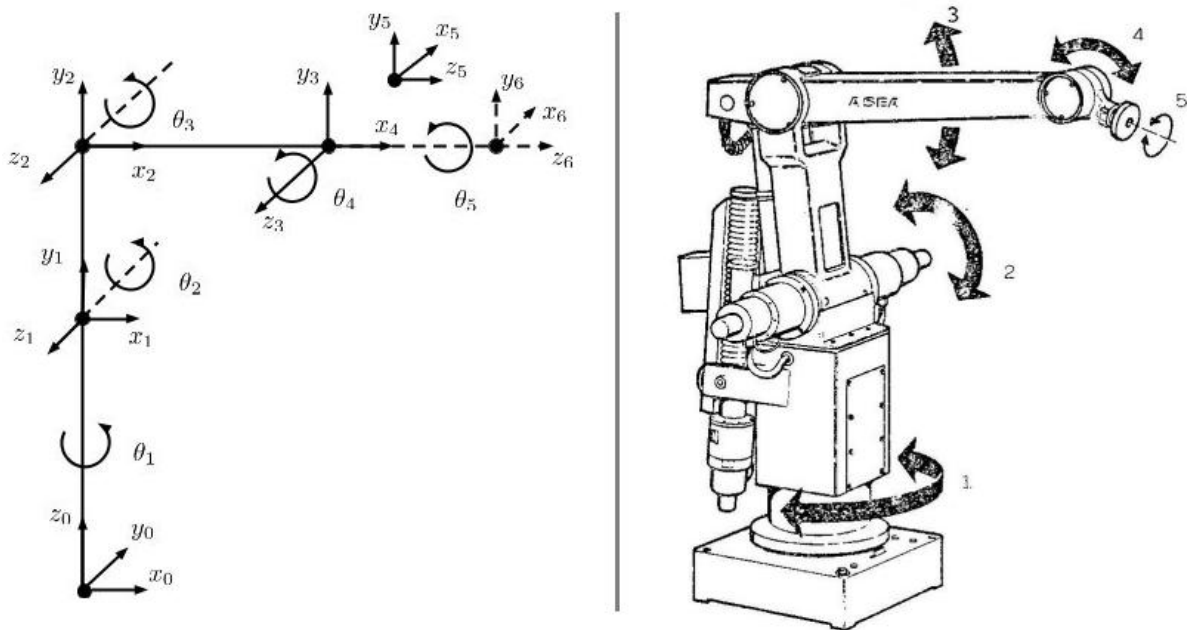


Figura 2.15: Sistemas de origem segundo a convenção DH para o robô ASEA IRB6-S2

Os sistemas de coordenadas referentes às juntas quatro e cinco coexistem na mesma origem de coordenadas, os quais constituem o punho do robô. Os valores dos parâmetros DH próprios do manipulador ASEA IRB6-S2 são apresentados na Tabela 2.2. As variáveis articulares das cinco juntas identificadas como θ_i representam os valores angulares para estabelecer o posicionamento final do manipulador, após a definição das coordenadas do efetuador. São apresentados os valores máximos para cada uma das juntas evitando desta forma as singularidades da cinemática específica do robô.

Tabela 2.2: Tabela Parâmetros DH para o Robô IRB6-S2

Junta	α_i	$a_i[m]$	$d_i[m]$	θ_i	Limite
1	90	0	$\lambda_1 = 0.70$	θ_1	340°
2	0	$l_2 = 0.45$	0	θ_2	$\pm 0^\circ$
3	0	$l_3 = 0.67$	0	θ_3	$\pm 25^\circ$
4	90	0	0	θ_4	$\pm 90^\circ$
5	0	0	$\lambda_5 = 0.095$	θ_5	$\pm 180^\circ$

Após a dedução dos valores para os parâmetros *DH*, foram definidas as equações presentes na matriz de transformação homogênea a qual determina a posição e orientação do efetuador em relação ao sistema de coordenadas da base do manipulador (SZKODNY, 1995). A matriz que relaciona o sistema de coordenadas da base com o efetuador do manipulador é resultado das matrizes que relacionam o elo i e elo $i - 1$. Na Eq. 2.4 é apresentada a obtenção da matriz final dos cinco graus de liberdade do manipulador ASEA IRB6-S2.

$$T_5^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 \quad (2.4)$$

Para simplificar as anotações matemáticas usadas nas matrizes de transformação homogênea de cada uma das juntas do manipulador é necessário usar a designação apresentada da Eq. 2.5.

$$\text{Sen}(\theta_i) = S_i \quad (2.5a)$$

$$\text{Cos}(\theta_i) = C_i \quad (2.5b)$$

$$\text{Sen}(\theta_i + \theta_j) = S_{ij} \quad (2.5c)$$

$$\text{Cos}(\theta_i + \theta_j) = C_{ij} \quad (2.5d)$$

São apresentadas na Eq. 2.6 as matrizes de transformação homogênea associadas aos valores DH para cada uma das cinco juntas do manipulador ASEA IRB6-S2.

$$\begin{aligned} T_1^0 &= \begin{bmatrix} -S_1 & 0 & C_1 & 0 \\ C_1 & 0 & S_1 & 0 \\ 0 & 1 & 0 & \lambda_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} & T_2^1 &= \begin{bmatrix} -S_2 & -C_2 & 0 & -l_2 S_2 \\ C_2 & -S_2 & 0 & l_2 C_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ T_3^2 &= \begin{bmatrix} S_3 & C_3 & 0 & l_3 S_3 \\ -C_3 & S_3 & 0 & -l_3 C_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & T_4^3 &= \begin{bmatrix} -S_4 & 0 & C_4 & 0 \\ C_4 & 0 & S_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ T_5^4 &= \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & \lambda_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (2.6)$$

Para apresentar de forma simplificada as variáveis próprias da matriz de transformação homogênea resultante da T_5^0 , foi aplicado o conceito de ângulos compostos (NICOLAIDES, 2007), expressado na Eq. 2.7. Desta forma é facilitado a integração computacional diminuindo o tempo através de operações matemáticas elementais.

$$S_{23} = (C_2 \cdot S_3) + (S_2 \cdot C_3) \quad (2.7a)$$

$$C_{23} = (C_2 \cdot C_3) - (S_2 \cdot S_3) \quad (2.7b)$$

$$S_{234} = (S_2 \cdot C_3 \cdot C_4) + (C_2 \cdot S_3 \cdot C_4) + (C_2 \cdot C_3 \cdot S_4) - (S_2 \cdot S_3 \cdot S_4) \quad (2.7c)$$

$$C_{234} = (C_2 \cdot C_3 \cdot C_4) - (S_2 \cdot S_3 \cdot C_4) - (S_2 \cdot C_3 \cdot S_4) - (C_2 \cdot S_3 \cdot S_4) \quad (2.7d)$$

Como já mencionado anteriormente, é necessário deduzir as equações características da cinemática direta e associar a matriz de transformação homogênea apresentada na Eq. 2.8, junto com as equações da matriz resultante dos parâmetros DH identificados na Eq. 2.4.

$$T_5^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.8)$$

Desta forma apresenta-se a descrição matemática na Eq. 2.9 da cinemática direta do manipulador ASEA IRB6-S2, através dos elementos da matriz de transformação homogênea com os componentes de posição e orientação (SZKODNY, 1995).

$$n_x = (S_1 \cdot S_{234} \cdot C_5) + (C_1 \cdot S_5) \quad (2.9a)$$

$$n_y = (-C_1 \cdot S_{234} \cdot C_5) + (S_1 \cdot S_5) \quad (2.9b)$$

$$n_z = C_{234} \cdot C_5 \quad (2.9c)$$

$$o_x = (-S_1 \cdot S_{234} \cdot S_5) + (C_1 \cdot C_5) \quad (2.9d)$$

$$o_y = (C_1 \cdot S_{234} \cdot S_5) + (S_1 \cdot C_5) \quad (2.9e)$$

$$o_z = -C_{234} \cdot S_5 \quad (2.9f)$$

$$a_x = -S_1 \cdot C_{234} \quad (2.9g)$$

$$a_y = -C_1 \cdot C_{234} \quad (2.9h)$$

$$a_z = S_{234} \quad (2.9i)$$

$$p_x = (l_2 \cdot S_1 \cdot S_2) - (l_3 \cdot S_1 \cdot C_{23}) - (\lambda_5 \cdot S_1 \cdot C_{234}) \quad (2.9j)$$

$$p_y = -(l_2 \cdot C_1 \cdot S_2) + (l_3 \cdot C_1 \cdot C_{23}) + (\lambda_5 \cdot C_1 \cdot C_{234}) \quad (2.9k)$$

$$p_z = \lambda_1 + (l_2 \cdot C_2) + (l_3 \cdot S_{23}) + (\lambda_5 \cdot S_{234}) \quad (2.9l)$$

Com a cinemática direta do manipulador definida foram geradas doze equações, as quais são usadas para identificar as cinco variáveis articulares do manipulador (θ_i), associadas a cada uma dos graus de liberdade. Em consequência é formado um sistema de doze equações e cinco incógnitas para o problema da cinemática inversa específica do manipulador ASEA IRB6-S2. Através de simplificações matemáticas e com a configuração mecânica do manipulador foi possível reduzir os sistemas a cinco equações com o mesmo número de incógnitas (BOMFIM, 2013). A relação dos ângulos de cada uma das juntas do robô ASEA IRB6-S2 são apresentadas na Eq. 2.10 associada a cinemática inversa do manipulador.

$$\theta_1 = \tan^{-1} \frac{-p_x}{p_y} \quad (2.10a)$$

$$\theta_3 = \tan^{-1} \frac{S_3}{C_3} \quad (2.10b)$$

$$S_3 = \frac{w_1^2 + w_2^2 - (l_2^2 + l_3^2)}{2l_2l_3} \quad (2.10c)$$

$$C_3 = \sqrt{1 - S_3^2} \quad (2.10d)$$

$$\theta_2 = \tan^{-1} \frac{S_2}{C_2} \quad (2.10e)$$

$$S_2 = \frac{w_2 l_3 C_3 - w_1 (l_3 S_3 + l_2)}{l_3^2 C_3^2 + (l_3 S_3 + l_2)^2} \quad (2.10f)$$

$$C_2 = \frac{w_1 l_3 C_3 + w_2 (l_3 S_3 + l_2)}{l_3^2 C_3^2 + (l_3 S_3 + l_2)^2} \quad (2.10g)$$

$$w_1 = -S_1 p_x + C_1 p_y + \lambda_5 S_1 a_x - \lambda_5 C_1 a_y \quad (2.10h)$$

$$w_2 = p_z - \lambda_1 - \lambda_5 a_z \quad (2.10i)$$

$$\theta_{34} = \tan^{-1} \frac{\lambda_5 S_{34}}{\lambda_5 C_{34}} \quad (2.10j)$$

$$\lambda_5 S_{34} = S_1 S_2 p_x - C_1 S_2 p_y + C_2 p_z - \lambda_1 C_2 - l_2 - l_3 C_3 \quad (2.10k)$$

$$\lambda_5 C_{34} = -S_1 C_2 p_x + C_1 C_2 p_y + S_2 p_z - \lambda_1 S_2 - l_3 C_3 \quad (2.10l)$$

$$\theta_4 = \theta_{34} - \theta_3 \quad (2.10m)$$

$$\theta_5 = \frac{S_5}{C_5} \quad (2.10n)$$

$$S_5 = C_1 n_x + S_1 n_y, C_5 = C_1 o_x + S_1 o_y \quad (2.10o)$$

As equações apresentadas anteriormente expressam a relação existente entre os ângulos de cada junta, assim como as dimensões do manipulador. Com a definição destas variáveis articulares é possível gerar a movimentação requerida para cada motor associado aos cinco graus de liberdade, a partir da posição e orientação conhecidas do efetuador.

Comumente nos robôs manipuladores é possível associar os atuadores diretamente a cada junta, permitindo ter uma relação direta entre o ângulo de rotação do motor e o a junta (ASADA; KANADE; TAKEYAMA, 1983). É possível conhecer esta relação com a leitura do sensor localizado no eixo do motor como exemplo os *encoders*, já no manipulador ASEA IRB6-S2 essa medição era feita originalmente com *resolvers*.

Os motores desse modelo de manipulador geram a movimentação das juntas através de quatro tipos de sistemas de transmissão de movimento como: *Harmonic-Drive*; parafuso de esferas; por hastes e pinhão/coroa. Devido a que era prioridade posicionar os motores próximos do eixo Z da base do robô. Desta forma, a inércia associada aos pesos dos motores se reduzia o que simplifica a análise cinemático da estrutura do robô, no entanto a relação matemática entre os motores e as juntas é mais complexa (BOMFIM, 2013).

A estrutura mecânica do robô ASEA IRB6-S2 se caracteriza porque as juntas são dependentes entre si, ou seja, é um robô antropomórfico com restrições mecânicas em relação aos graus de liberdade (MAIA et al., 2003). A única junta independente é da base, enquanto a junta dois é acionada através de um fuso linear, assim como a junta três. De acordo a estrutura projetada para o manipulador a junta três é dependente da movimentação que possa ter a junta dois. Do mesmo modo a junta quatro é dependente da das juntas dois e três, assim como do movimento gerado pelo motor respectivo através de um sistema de hastes. Finalmente a junta cinco também é acionado por um sistema de hastes, além de ter um sistema de engrenagens cônicas, e tem dependência da junta quatro (SZKODNY, 1995).

Devido a configuração mecânica do manipulador ASEA IRB6-S2 a transmissão do movimento entre os atuadores e as juntas é através dos sistemas de transmissão de movimento descritos anteriormente. Por conseguinte é necessário associar matematicamente através de equações, a relação existente entre as variáveis de junta (θ_i) em função do movimentação gerada pelos atuadores (II, 2006). Existem variáveis associadas as distancias específicas do manipulador ASEA IRB6-S2 e são apresentadas na Tabela 2.3.

Tabela 2.3: Distancias da configuração mecânica do manipulador ASEA IRB6-S2

Variável	BA	BC	BD	BJ	DJ	FE	FG	FH
Comprimento (mm)	190	240	140	450	430	190	240	140

Fonte: (BOMFIM, 2013)

Na Eq. 2.11 é apresentada a relação matemática existente entre a movimentação angular dos atuadores (A_i) e as variáveis cinemáticas associadas com as juntas do manipulador ASEA IRB6-S2, onde as variáveis λ_i são valores de *offset* associadas a configuração mecânica do robô (BOMFIM, 2013).

$$\theta_1 = k_1 A_1 + \lambda_1 \quad (2.11a)$$

$$\theta_2 = \cos^{-1} \left(\frac{BC^2 + BD^2 - (k_2 A_2 + \lambda_2)^2}{2 BC BD} \right) + \cos^{-1} \left(\frac{BA}{BC} \right) + \cos^{-1} \left(\frac{BJ^2 + BD^2 - DJ^2}{2 BJ BD} \right) - \pi \quad (2.11b)$$

$$\theta_3 = \cos^{-1} \left(\frac{FG^2 + FH^2 - (k_3 A_3 + \lambda_3)^2}{2 FG FH} \right) + \cos^{-1} \left(\frac{FE}{FG} \right) - \theta_2 - \frac{\pi}{2} \quad (2.11c)$$

$$\theta_4 = -k_4 A_4 - \theta_2 - \theta_3 + \lambda_4 \quad (2.11d)$$

$$\theta_5 = (k_5 A_5 - \theta_4) k_6 + \lambda_5 \quad (2.11e)$$

São apresentadas na Eq. 2.12 as relações dos valores para os atuadores do manipulador (A_i) em função das variáveis articulares dos cinco graus de liberdade (θ_i).

$$A_1 = \frac{\theta_1 - \lambda_1}{k_1} \quad (2.12a)$$

$$A_2 = \frac{\sqrt{BC^2 + BD^2 - 2 BC BD \cos \alpha} - \lambda_2}{k_2} \quad (2.12b)$$

$$\alpha = \theta_2 - \cos^{-1} \frac{BA}{BC} - \cos^{-1} \left(\frac{BJ^2 + BD^2 - DJ^2}{2 BJ BD} \right) + \pi \quad (2.12c)$$

$$A_3 = \frac{1}{k_3} \left(\sqrt{FG^2 + FH^2 - 2 FG FH \cos \left(\theta_3 + \theta_2 - \cos^{-1} \left(\frac{FE}{FG} \right) + \frac{\pi}{2} \right)} - \lambda_3 \right) \quad (2.12d)$$

$$A_4 = -\frac{\theta_2 + \theta_3 + \theta_4 - \lambda_4}{k_4} \quad (2.12e)$$

$$A_5 = \frac{1}{k_5} \left(\frac{\theta_5 - \lambda_5}{k_6} + \theta_4 \right) \quad (2.12f)$$

É necessário mencionar que as variáveis k_i descritas anteriormente estão associadas as reduções dos diferentes sistemas de transmissão de movimento existentes em cada um dos graus de liberdade do manipulador ASEA IRB6-S2. Na Tabela 2.4 são apresentados os valores das reduções e os tipos de sistemas de transmissão (ASEA, 2003).

Tabela 2.4: Valores das reduções do manipulador ASEA IRB6-S2

Junta	Redução (k_i)	Tipo de transmissão
Um	1/158	<i>Harmonic-Drive</i>
Dois e Três	$5/2\pi$	Fuso linear de esferas
Quatro e Cinco	1/128	<i>Harmonic-Drive</i> e hastes
Efetuator	3/2	Pinhão/coroa

2.7 Sistemas de Controle para Manipuladores Industriais

Uma arquitetura de controle para um manipulador está composta geralmente por uma malha fechada para cada uma das juntas, integrada por acionadores (motores/servomotores) e sensores de posição (*encoder/resolver*). Os sensores estão localizados no eixo de cada motor e representam o *feedback* do sistema de controle. Na Figura 2.16 é apresentada a malha de controle para um manipulador industrial.

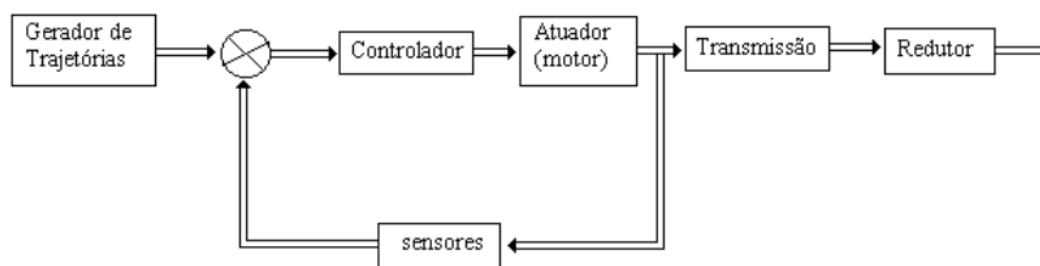


Figura 2.16: Configuração controle de manipulador em malha fechada
Fonte: (ROMANO, 2002)

O controlador comumente usado para aplicações com robôs manipuladores é a sintonia de um PID para cada uma das juntas (NOF, 1999). Onde o objetivo final é garantir as condições de funcionamento do manipulador sem importar as perturbações externas como a fricção de Coulomb (BARRIENTOS, 2007). Esses controladores são usados amplamente nos manipuladores industriais, especialmente pelo componente integral, o que conduz o erro de posição para zero.

A sintonização de controladores, nesse caso os PID, é simples quando existem mecanismos de redução como engrenagens ou correias de transmissão entre os motores e elos das juntas do manipulador, devido a que o comportamento do sistema em malha fechada é modelado através de equações diferenciais lineares. Além desse fato existe um impacto positivo ao sistema, devido ao incremento do torque ou força gerada pelos atuadores (KELLY; DAVILA; PEREZ, 2006). A seguir são descritas as ações de controle deste controlador:

- **Ação proporcional - P**, é um algoritmo simples de controle caracterizado por uma constante que relaciona a entrada e a saída do controlador. O parâmetro ajustável é chamado de ganho proporcional. Este controlador só tem resposta no presente, não tem consideração de erro no passado, nem possíveis erros no futuro (OGATA, 2010).
- **Ação integral - I**, é usualmente chamado de erro em regime estacionário, pois a carga do efetuator muda e estabelece a variável ao *set point* e elimina o *offset*, o que não é feito pelo controlador proporcional. A condição do modo integral é incluída para eliminar o *offset*, gerado por não considerar o histórico do erro. O *offset* é o erro residual que não pode ser eliminado por o controle proporcional (OGATA, 2010).
- **Ação derivativa - D**, é complemento aos anteriores modos de controle por causa da capacidade de antecipar os valores de erros futuros permitindo acelerar a resposta transitória. O propósito desta ação é prever o erro do processo e gerar ações corretivas (OGATA, 2010).

Na Figura 2.17 são apresentadas as respostas de uma variável controlada através dos controladores em função do tempo, onde o controlador PID integra as características das ações de controle permitindo um tempo de estabelecimento mais rápido em comparação aos outros. Assim o PID é o controlador geralmente usado para manipuladores industriais porque gera uma resposta instantânea (LIPTAK, 2005).

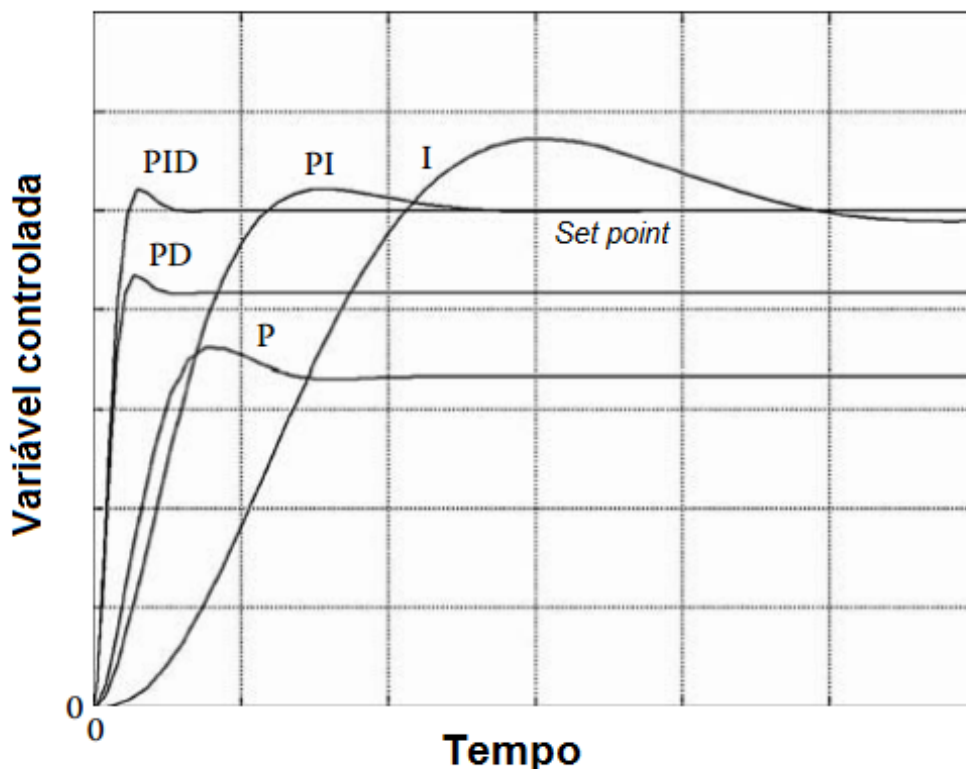


Figura 2.17: Resposta dos controladores P-I-D a um degrau
Fonte: (LIPTAK, 2005)

2.7.1 Sintonização do Controlador PID

Se o modelo do sistema a ser controlado é conhecido, existem várias técnicas para determinar os parâmetros do controlador em um sistema de malha fechada, porém o modelo matemático do sistema é de difícil obtenção, mas no caso de manipuladores industriais é conhecido. Porém, devido às perturbações externas ao processo, a maioria das vezes é necessário optar por técnicas de sintonização experimentais para sintonizar o controlador PID (OGATA, 2010).

Para manipuladores industriais na atualidade é usado um controle PID para cada uma das juntas, dividindo o controle geral do sistema em subsistemas tipo SISO (uma-única-entrada/uma-única-saída) para cada grau de liberdade do manipulador, ajustando os parâmetros do controlador PID diretamente no *drive* usado para cada motor (SICILIANO; KHATIB, 2008). Desta forma, cada fornecedor de fabricante tem uma técnica de sintonização específica, onde o ajuste dos parâmetros do controlador PID (K_p , K_i e K_d) é feito na operação de cada uma das juntas do robô (LIPTAK, 2005).

Existem atualmente controladores avançados para aplicações em robótica industrial com a responsabilidade de ajustar os parâmetros do PID, como algoritmos genéticos, métodos com ferramentas software, baseados em lógica Fuzzy ou com aplicação de redes neurais artificiais. A escolha do tipo de sintonização depende do resultado específico do projeto, como por exemplo o desempenho do controle mesmo ou a evolução e minimização das perturbações que agem diretamente no sistema (EMMANUEL; INYIAMA, 2015).

Capítulo 3

Proposta de Metodologia para o *Retrofitting* de Robôs Industriais

3.1 Introdução

Este capítulo apresenta as atividades necessárias para aplicar uma metodologia baseada na técnica *retrofitting* para robôs industriais que são desconsiderados para operação na indústria ou por não se encontrar em condições para executar alguma tarefa no âmbito da manufatura, como por exemplo a embalagem de peças, usinagem, supervisão etc.

As atividades apresentadas neste capítulo constituem uma síntese de trabalhos correlatos a técnica *retrofitting*, assim como a experiência com o robô ASEA IRB6-S2 para que voltasse a ser operacional. As atividades são: diagnóstico e desmontagem do robô antes da aplicação da técnica, atualizações de *hardware* e *software* que possibilite o robô integrar-se com o sistema de produção de nível acadêmico ou setor produtivo. Após conhecer e adquirir os componentes necessários para a montagem e consolidação do sistema é necessário validar a técnica do *retrofitting* por meio de métodos que permitam avaliar o desempenho do robô, o que dependerá da aplicação final do manipulador após aplicar a metodologia proposta baseada na técnica *retrofitting* para manipuladores industriais.

A metodologia proposta apresentada neste capítulo é uma contribuição relevante na literatura, devido a falta de uma metodologia específica para o *retrofitting* de manipuladores industriais com o detalhamento e a concepção desde o diagnóstico até propostas de validação do trabalho realizado. Desta forma é possível modificar as atividades apresentadas permitindo uma consolidação dinâmica de uma metodologia aberta e robusta especificamente para o aplicar a técnica *retrofitting* em manipuladores industriais.

3.2 Arquitetura de Controle Proposta

Existe uma estrutura funcional no sistema de controle para aplicar a técnica *retrofitting* em um robô industrial, onde os componentes/camadas interagem logicamente em uma comunicação de duas vias, desde o processamento de cálculos matemáticos das variáveis cinemáticas até o acionamento de cada uma das

juntas do manipulador. O modelo proposto de arquitetura de controle para o *retrofitting* do manipulador ASEA IRB6-S2 é apresentado na Figura 3.1 a nível de camadas funcionais, permitindo um controle em tempo real do manipulador, além de estabelecer uma característica modular no modelo proposto permitindo a integração de novos componentes tipo *software* e *hardware* como parte do conceito de arquitetura aberta de controle para manipuladores industriais.

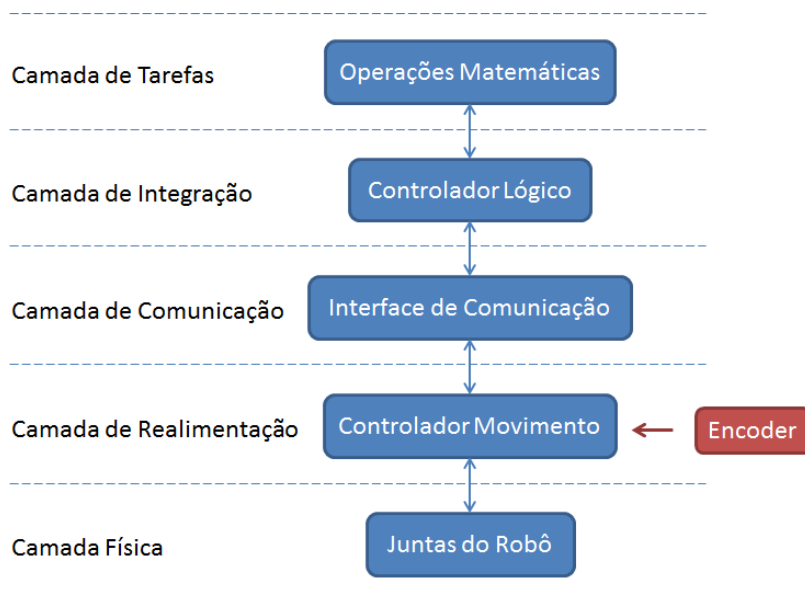


Figura 3.1: Arquitetura geral por camadas do *retrofitting*

A *camada de tarefas* é responsável pelas operações matemáticas correspondentes as equações homogêneas próprias da configuração de cada manipulador industrial, gerando os valores das variáveis articulares para a cinemática direita e inversa segundo as atividades designadas para o robô. Esta informação vai ser transmitida para os motores com a realimentação de sensores por meio das outras camadas funcionais da arquitetura apresentada.

A *camada de integração* é responsável por concentrar e organizar a informação de cada um dos componentes de controle para enviá-la diretamente à camada superior. Há um controle via software para cada um dos sistemas de potência no momento da ativação ou desativação de forma manual ou automática. Estas duas camadas superiores fisicamente coexistem em um ordenador central com um componente GUI para garantir a interação humano-máquina, permitindo a configuração e operação manual do sistema geral do manipulador.

A *camada de comunicação* faz o controle do tráfego de dados entre as camadas superiores e os componentes próprios de cada uma das juntas através de padrões de comunicação abertos ou proprietários. Estes padrões devem ter a capacidade de suportar a transmissão em tempo real, reduzindo as perdas de dados e os atrasos na transmissão.

O controle da rotação de cada um dos motores ligados aos graus de liberdade do manipulador está sob a responsabilidade da *camada de realimentação*, fisicamente conhecidos como controladores de movimento, os quais recebem um sinal de comando a partir do sistema de controle, neste caso das camadas superiores e faz a transmissão da corrente elétrica para os atuadores que geram um movimento proporcional dos sinais

de comando gerados em termos de velocidade e/ou posição. Um sensor ligado ao motor envia uma leitura ao controlador do estado atual do atuador. Estes dados são comparados com o sinal de comando gerados pelas camadas superiores.

Finalmente na *camada física* estão presentes os motores responsáveis pela geração de movimento para cada uma das juntas segundo a configuração específica de cada manipulador.

3.3 Metodologia Baseada no *Retrofitting* para Robôs Industriais: Modelagem IDEF0

Para que um robô manipulador fora de operação se torne funcional por meio da técnica *retrofitting*, teve que ser modelado com a metodologia IDEF0 permitindo descrever as atividades indispensáveis. Na figura 3.2 representa o nível A0 da metodologia IDEF0 com as entradas, saídas, mecanismos, controles e funcionalidades propostas com a metodologia para desenvolver uma arquitetura aberta de controle para manipuladores baseada na técnica *retrofitting*.

As atividades principais da metodologia para o *retrofitting* do robô são apresentadas na Figura 3.3, levando em conta o diagnóstico do estado atual dos diferentes sistemas próprios na operação do robô, até garantir a movimentação do robô validada com algumas variáveis de desempenho propostas.

As atividades principais segundo o diagrama IDEF0 apresentado anteriormente são:

1. Desmontagem e diagnóstico do robô.
2. Especificação atualização *Hardware*.
3. Especificação atualização *Software*.
4. Montagem.
5. Validação *retrofitting*: Ensaios e testes.

A atividade Desmontagem e diagnóstico do robô visa identificar os componentes que podem ser aproveitados segundo sua função no *retrofitting* do robô, assim como aqueles componentes que por seu estado atual não deveriam fazer parte da metodologia para que o robô possa ser operacional. As entradas de dados da atividade descrita anteriormente são os manuais disponíveis do fornecedor do robô, além dos componentes que são responsáveis por controlar o sistema.

A atividade para atualização de *hardware* tem como entradas os componentes que foram identificados para serem reaproveitados e os que foram descartados; assim como o material bruto que será ser usado na fabricação de componentes indispensáveis na metodologia baseada no *retrofitting*. Além dos componentes fabricados, estão como saídas destas atividades a especificação, o projeto e a compra dos componentes necessários para o *retrofitting*.

Na seguinte atividade de Especificação Atualização *Software* é apresentado como entradas o programa NC e o modelo cinemático com a convenção DH, assim como os componentes que foram aproveitados e a

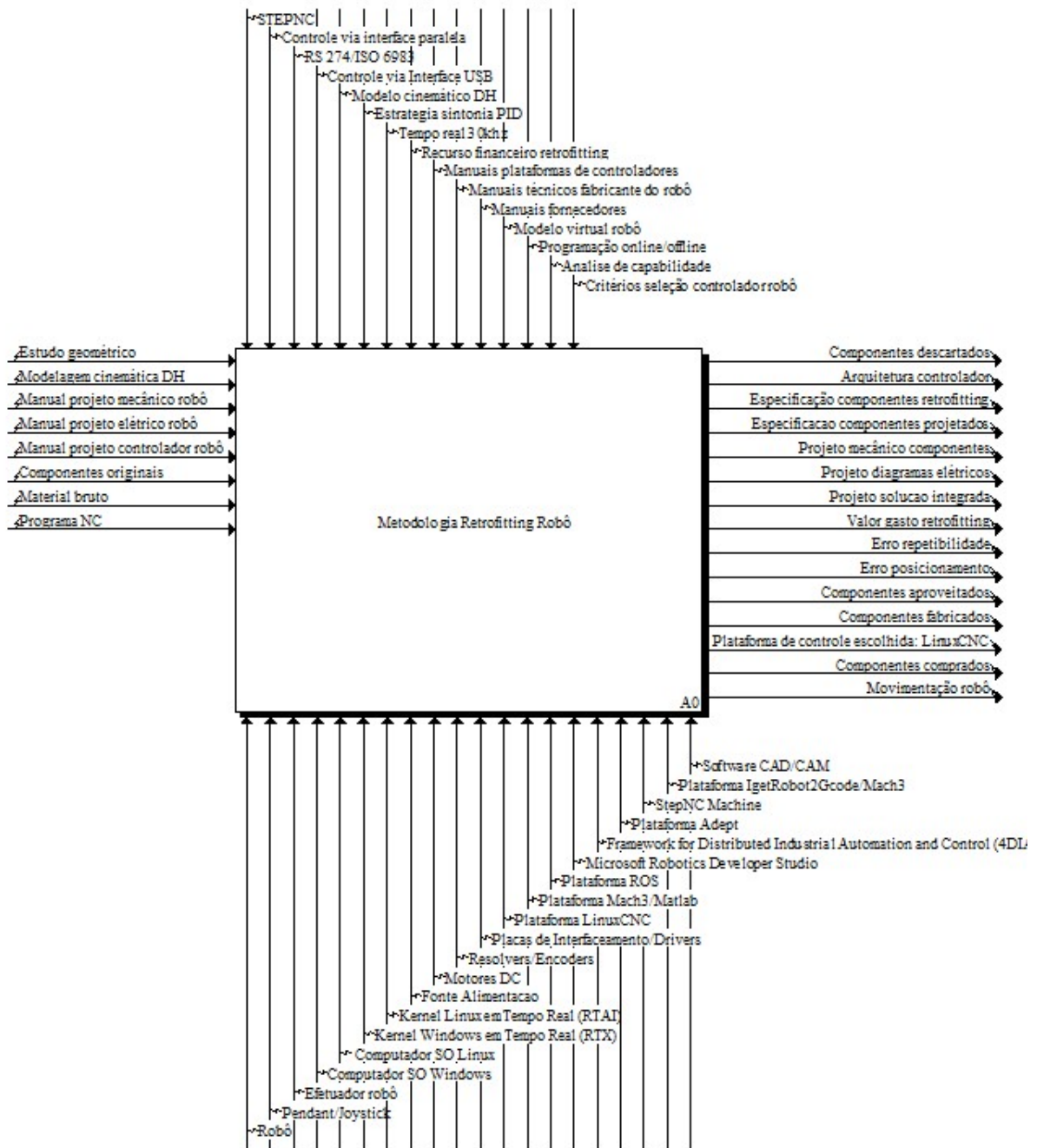


Figura 3.2: Diagrama IDEF0: Retrofitting nível A0

especificação dos que foram adquiridos. Para a atividade de Montagem as entradas serão os componentes fabricados, adquiridos e a plataforma de controle para os manipuladores selecionados. Essa atividade terá como saídas a solução integrada para a movimentação do robô e a documentação técnica equivalente.

Finalizando as atividades principais da metodologia para o uso da técnica *retrofitting* de manipuladores industriais fora de operação, está a validação do trabalho feito através de ensaios e testes, com entradas como a solução integrada baseada na plataforma de controle escolhida, assim como o programa NC que

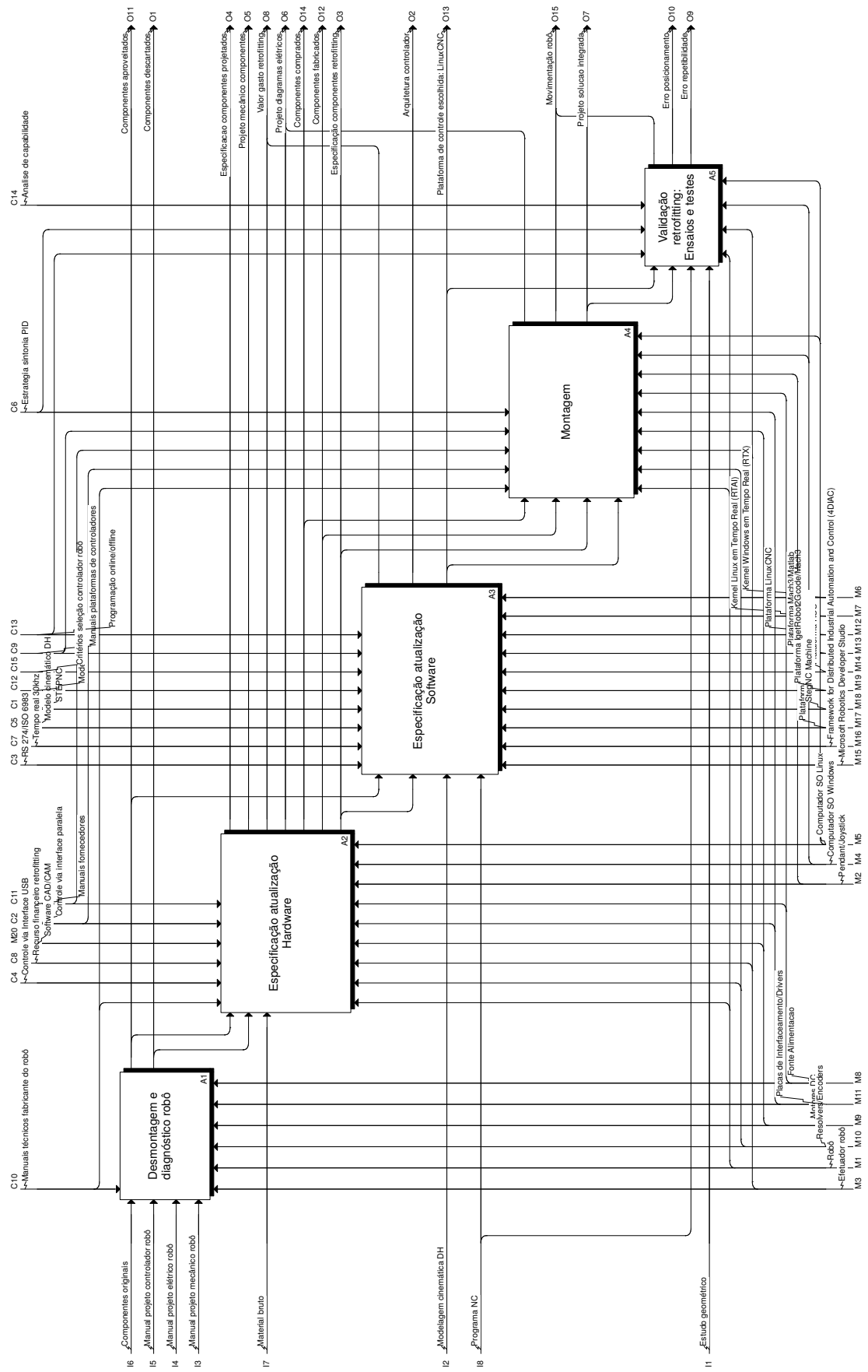


Figura 3.3: Diagrama IDEF0: Retrofitting nível A0

vai permitir avaliar o desempenho do robô com índices propostos nesta metodologia como o erro de posicionamento e repetibilidade na saída da atividade descrita.

A seguir, são apresentados diferentes diagramas IDEF0 associados às atividades descritas anteriormente, próprias da metodologia proposta baseada na técnica *retrofitting*.

3.4 Desmontagem e Diagnóstico do Robô

Como parte da metodologia proposta é necessário verificar todos os componentes originais do sistema do robô, permitindo conhecer o estado atual de cada um deles, assim como aproveitar componentes que após o diagnóstico sejam úteis para que o robô volte a ser operacional. É apresentada na Figura 3.4 a descrição através de quatro atividades no modelo IDEF0 segundo a atividade Desmontagem e o Diagnóstico do robô.

A seguir se descrevem os passos para fazer o diagnóstico do robô e seus subsistemas, permitindo conhecer quais são os componentes que devem ser descartados e posteriormente desmontados, ou os subsistemas do robô. Outra informação relevante neste processo de diagnóstico inicial é ter certeza dos componentes adicionais que devem ser comprados, projetados e/ou fabricados, garantindo a integração do robô com as tecnologias modernas. As atividades associadas a desmontagem e o diagnóstico do robô e sua função específica na metodologia são:

3.4.1 Diagnóstico da Estrutura Física do Robô

Para o sucesso do *retrofitting* é necessário que a estrutura mecânica do robô esteja em boas condições para garantir a operação durante as diferentes aplicações industriais ou acadêmicas. É importante validar os componentes mecânicos da estrutura, tais como os sistemas de transmissão, assim como a correta lubrificação para cada um dos conjuntos de atuadores das juntas. Cada grau de liberdade tem um módulo ou conjunto atuador, composto por um motor que gera o movimento através de um sistema de transmissão para a movimentação da junta. Estes conjuntos de atuadores têm que ser desmontados como parte da identificação e verificação das peças próprias do sistema mecânico e para a montagem correta dos novos conjuntos de atuadores.

3.4.2 Sistema de Alimentação do Robô

O sistema de alimentação do robô é a parte geral do sistema do robô que permite a distribuição eficiente de energia elétrica para os requerimentos de cada um dos subsistemas próprios de um robô manipulador, descritos na seção 3.2. Dependendo das tensões usadas pelos subsistemas deve-se verificar com um instrumento de medição as tensões reais em comparação com as tensões projetadas pelo fornecedor.

Inúmeras causas afetam o sistema de alimentação, especialmente ao subsistema de controle onde a falta de regulação da tensão pode danificar os componentes eletrônicos projetados com uma faixa de operação específica. Desta forma, a medição do sistema de alimentação para subsistemas característicos deste tipo

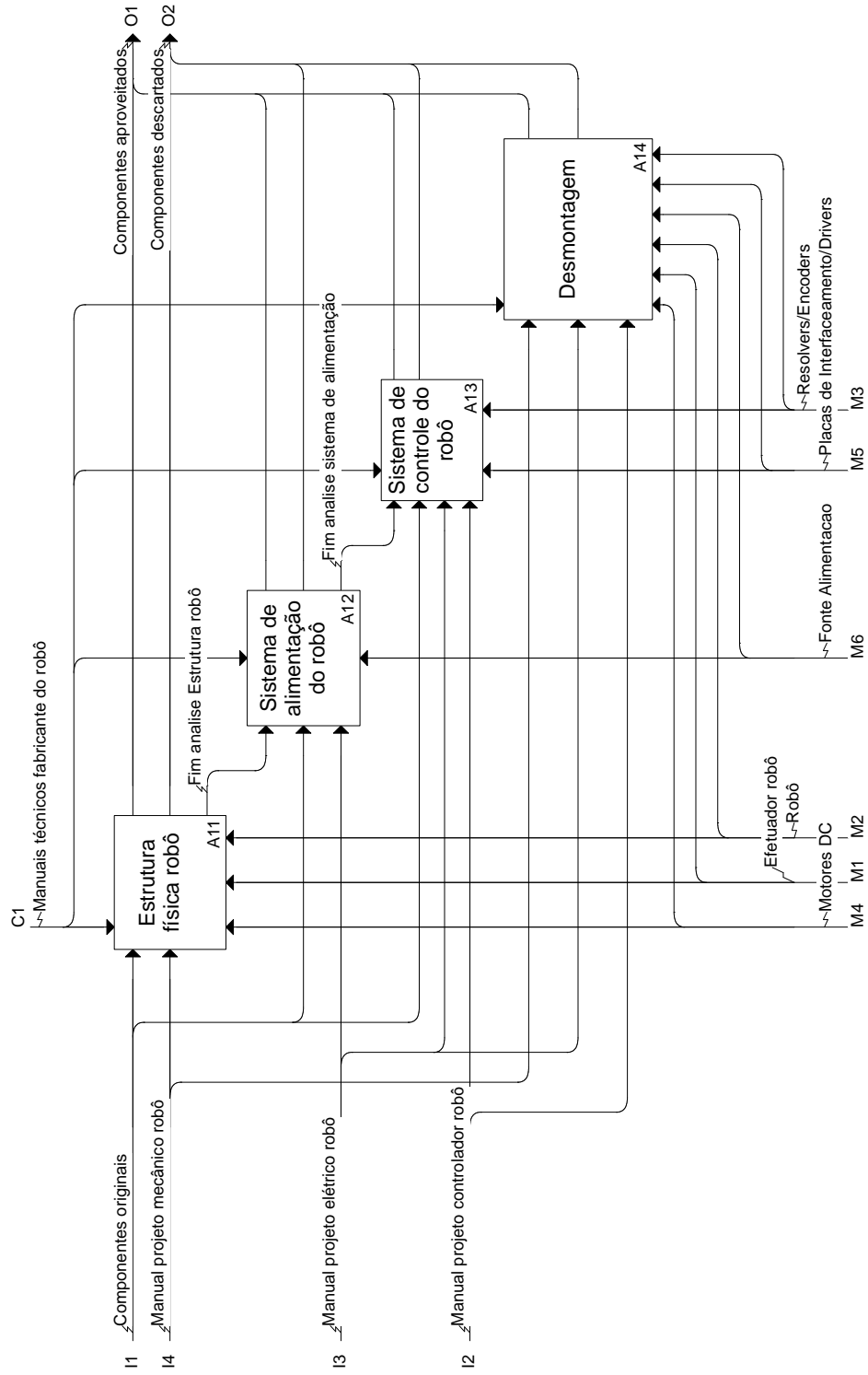


Figura 3.4: Diagrama IDEF0: Atividades desmontagem e o diagnóstico do robô

de sistemas robóticos como os cálculos da cinemática do manipulador, as interfaces de comunicação, o sistema de feedback dos avanços de cada um dos atuadores, dentre outros.

Em sistemas distribuídos de energia, o controle da regulação de tensão deve ser uma das prioridades, principalmente quando os equipamentos que consomem energia elétrica precisam de fluxo de energia constante sem variações significativas. Os momentos onde existe maior diferença de tensão consumida/fornecida é no horário de pico, quando muitos equipamentos estão ligados na mesma fonte de transmissão de energia elétrica. Os sistemas robóticos precisam de arquiteturas de geração de tensão robustas, para evitar falhas não desejadas nos subsistemas distribuídos que possibilitam a movimentação do manipulador (KULKARNI; TAI; ABRAHAM, 2015).

3.4.3 Sistema de Controle do Robô

Os sistemas de controle dos robôs em geral são sistemas distribuídos com varias camadas, e cada uma delas com tarefas específicas como foi apresentada na seção 3.2. Em robôs antigos com tecnologia analógica, é difícil obter componentes compatíveis com a tecnologia digital moderna o que possibilita trocar completamente o sistema de controle, suprimindo o tempo de testes dos diferentes elementos de controle para planejar um subsistema de controle específico para o manipulador segundo a aplicação final. Além disso, a falta de uma arquitetura aberta de controle de robôs limita a capacidade do robô para desenvolver tarefas diferentes devido à pouca flexibilidade das arquiteturas de controle proprietárias (OLIVEIRA, 2007).

3.4.4 Desmontagem dos Componentes

Na presente metodologia não é viável aproveitar os componentes antigos ou originais do manipulador, por causa do desgaste e que alguns destes são obsoletos. Gerando alto consumo de energia e diminuindo a confiabilidade para as aplicações futuras. Por tanto, é necessário trocar os componentes elétricos e eletrônicos, usando só a estrutura mecânica do robô (BOMFIM et al., 2014), garantindo uma operação eficiente e durável das funcionalidades do manipulador.

3.5 Atualização *Hardware*

Após identificar quais são os componentes que podem ser aproveitados para que o manipulador possa ser novamente operacional, é necessário especificar que outros componentes devem ser adquiridos ou fabricados sendo compatíveis com as características técnicas do robô. A atividade Atualização *Hardware* é constituída por quatro partes.

1. Especificar componentes: segundo os componentes aproveitados nesta atividade a saída é a especificação dos componentes que devem ser adquiridas e fabricadas substituindo os componentes não aproveitados e permitindo atualizar a tecnologia original do manipulador com tecnologia moderna.
2. Comprar componentes: nesta atividade é possível conhecer quais são os componentes e devem ser comprados por meio dos fornecedores de tecnologia relacionada a *retrofitting* do manipulador. As saídas são componentes que devem ser instalados no robô incluindo também o custo que representam.

3. Projetar novos componentes: existem componentes que devem ser projetados segundo a especificação de cada manipulador com uma integração entre a tecnologia moderna e a época que fora fabricado o sistema do manipulador. Desta forma, o resultado desta atividade será o projeto e a especificação dos componentes que devem ser fabricados.
4. Fabricar novos componentes: com a especificação e projeto dos componentes, nesta atividade é possível obter peças, acessórios, acoplamentos etc, para adaptar os componentes comprados com a estrutura do robô, assim como os componentes reaproveitados. Uma das entradas dessa atividade é o material dos componentes projetados na atividade anterior.

Na fig. 3.5 é apresentado o modelo IDEF0 associado a atividade de especificação atualização de Hardware.

Finalizada a atividade associada à especificação Atualização *Hardware*, os componentes necessários para aplicar a técnica *retrofitting* na presente metodologia estarão disponíveis para a montagem segundo suas funções no robô.

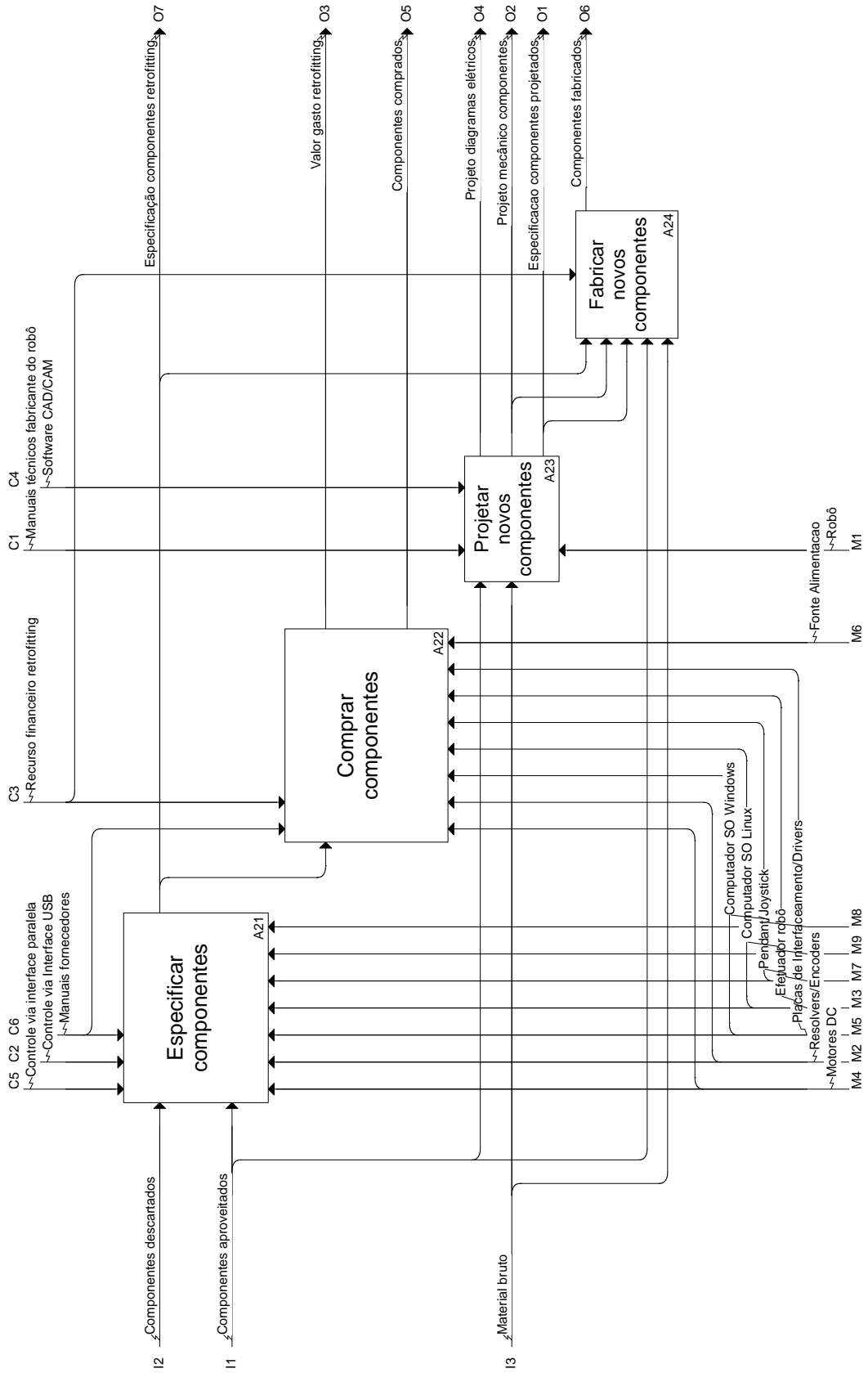
3.6 Atualização Software

Com a especificação dos componentes *hardware* já finalizada como atividade indispensável na metodologia proposta, para que o robô volte novamente a ser operacional, é necessário escolher a melhor solução a nível de plataformas *software* para o controle do sistema robótico. Na Figura 3.6 são apresentadas com o modelo IDEF0, as atividades associadas para a especificação na etapa de atualização *software* para o controle de robôs.

Para escolher uma plataforma de controle para manipuladores industriais adequada, são descritas três atividades associadas com a especificação e atualização do *software*.

1. Especificar requisitos: tem como entradas a especificação dos componentes adquiridos, fabricados e reaproveitados, assim como a modelagem matemática para manipuladores baseado na notação DH. As saídas desta atividade será a arquitetura do controlador ou as especificações mínimas para o controle do robô.
2. Comparar plataformas para o controle de robôs: possibilita segundo os critérios para a seleção da plataforma, comparar algumas das plataformas para manipuladores atualmente disponíveis no mercado, permitindo obter como saída, uma plataforma de controle em conformidade com a aplicação que vai ter o manipulador.
3. Configurar o controlador escolhido: tendo a plataforma de controle para robôs manipuladores selecionada, é necessário configura-la segundo a informação disponível do fornecedor e diretamente com o sistema físico do robô.

Para a geração de movimento de cada uma das juntas do robô é necessária a geração de sinais específicos correspondentes aos valores das variáveis articulares do sistema, obtidos das equações homogêneas



Page 1

Figura 3.5: Diagrama IDEF0: Atividades especificação atualização Hardware

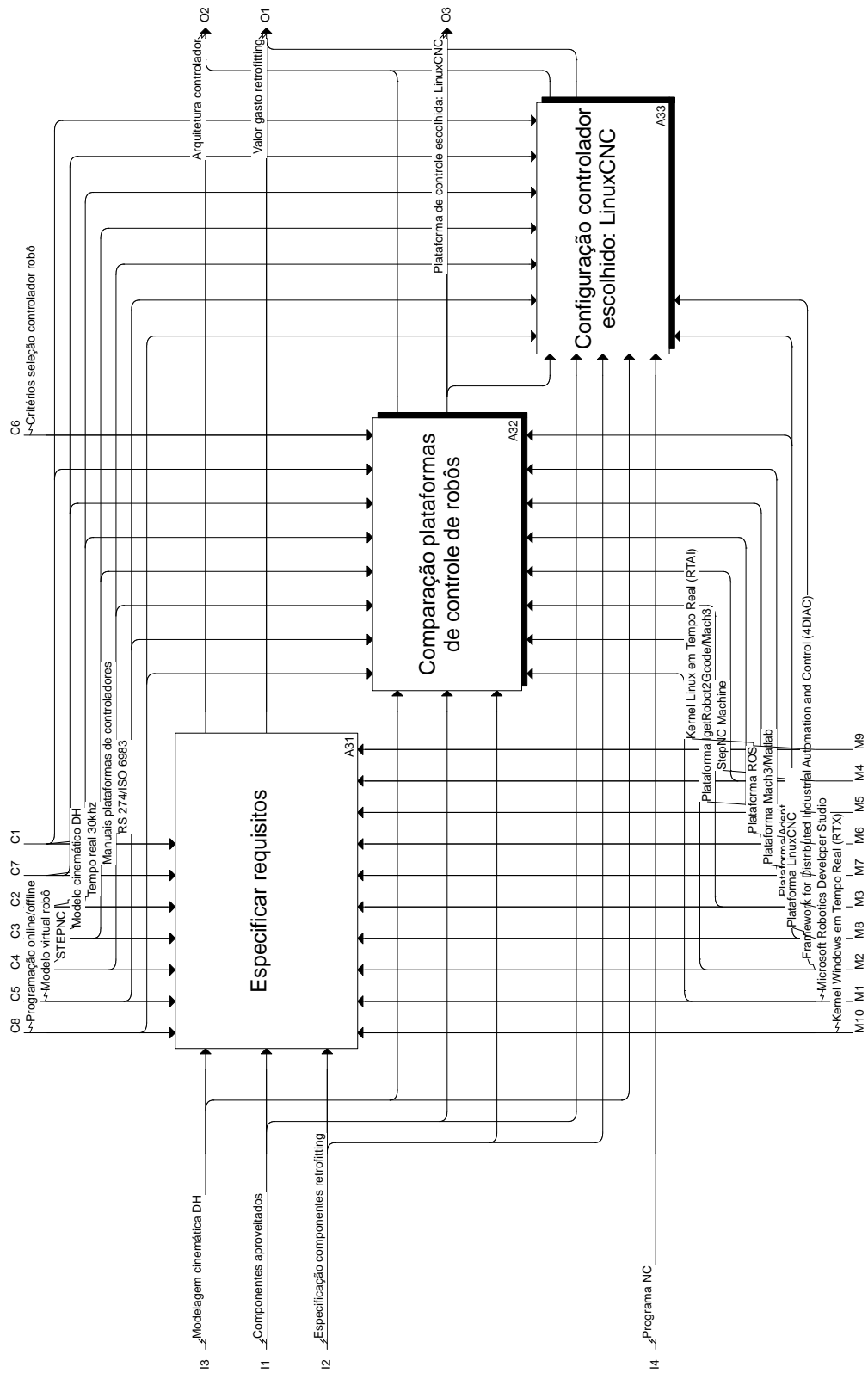


Figura 3.6: Diagrama IDEF0: Atividades especificação atualização software

próprias da cinemática do manipulador. Deste modo, os diferentes módulos de controle segundo as aplicações finais do sistema deveram fazer parte de um *software* de controle.

Originalmente as estruturas lógicas são proprietárias e comercializadas com o manipulador, além de ter uma estrutura fechada para sua modificação ou atualização, limitando assim as aplicações adicionais que possam desenvolver-se e que o manipulador esteja em capacidade de executar (OLIVEIRA A.; MORENO, 2010). Desta forma, o *software* com arquitetura aberta de controle deve ser desenvolvido ou usado para garantir o uso eficiente de todas as características de um manipulador industrial.

3.6.1 Comparação Plataforma de Controle para Robô

Para a escolha de uma arquitetura software adequada segundo os critérios específicos da aplicação final do manipulador, devem ser consideradas algumas plataformas para o controle de robôs usadas atualmente no contexto da robótica. Na Figura 3.7 é apresentado o modelo IDEF0 com as atividades necessárias para caracterizar cada uma das plataformas propostas, e desta forma ter critérios semelhantes segundo a função final que deverá executar o manipulador. Desta forma é viável comparar e escolher uma plataforma de controle adequada como parte fundamental da metodologia proposta no presente trabalho.

As plataformas analisadas segundo as atividades da metodologia proposta que foram indicadas na Figura 3.7, como parte do uso da técnica *retrofitting* para o robô ASEA IRB6-S2 são listadas a continuação:

A321 LinuxCNC	A325 StepNC Machine
A322 Mach3/Matlab	A326 ADEPT
A323 ROS (Robot Operating System)	A327 4DIAC
A324 IgetRobot2Gcode/Mach3	A328 Microsoft Robotics Developer Studio

Para facilitar a seleção da plataforma de controle são apresentados na Tabela 3.1 alguns critérios de seleção baseados em arquiteturas abertas não proprietárias (HAMILTON; HASCOËT; RAUCH, 2011), permitindo a comparação das características internas de cada plataforma e a relevância para o projeto.

Foi possível incluir o critério da compatibilidade com o padrão MTConnect na comparação de plataformas de controle para robôs. O MTConnect é definido como uma linguagem e estrutura comum de comunicação para monitorar dados próprios de máquinas da manufatura. A arquitetura do MTConnect está integrada por um **dispositivo** que é a máquina ferramenta; um **agente** responsável por coletar dados do dispositivo e enviar-los às aplicações; Uma *aplicação* tipo cliente, onde uma aplicação de usuário usa os dados do **dispositivo** e um **servidor** baseado em um protocolo leve para o acesso a diretórios, o qual traduz os nomes do **dispositivo** para o **agente**. O uso deste padrão de comunicação aberto fomenta a interoperabilidade da máquina ferramenta e aplicações para publicar os dados sob protocolos baseados na Internet como XML e HTTP (SHIN et al., 2015).

Para garantir uma escolha adequada da plataforma de controle e como complemento a comparação feita na Tabela 3.1, são avaliados numericamente critérios propostos na Tabela 4.1 associados a integração com sistemas de controle ou padrões de comunicação. Cada critério avaliado tem uma nota mínima de

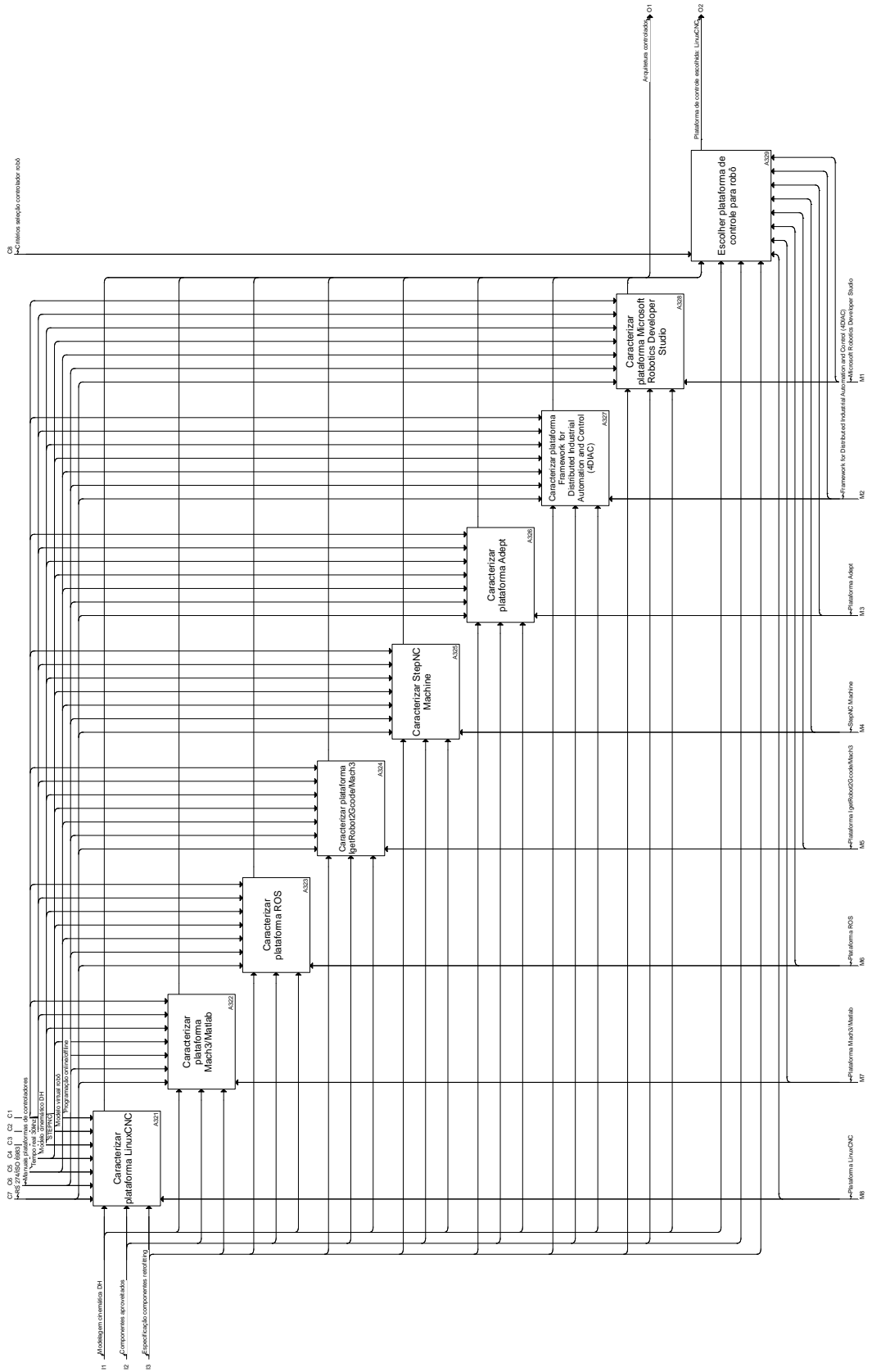


Figura 3.7: Diagrama IDEF0: Atividades comparação de plataformas

Tabela 3.1: Comparação plataformas de controle para robôs

Critério	A321	A322	A323	A324	A325	A326	A327	A328
Sistema operacional	Linux	Windows	Linux	Windows	Windows	Próprio	Linux	Windows
Infraestrutura definida	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim
Nível de modularidade	Alto	Médio	Alto	Baixo	Médio	Alto	Alto	Médio
Linguagens de programação	C, C++ Python	Script Matlab	C++, Python	–	–	V+	C++	C#, NET
Licença <i>open-source</i>	Sim	Não	Sim	Não	Não	Não	Sim	Não
<i>Kernel</i> em tempo real	Sim	Não	Não	–	Não	Sim	Sim	Não
Compatibilidade com RS-274	Sim	Sim	Não	Sim	Sim	Não	Não	Não
Compatibilidade com MTConnect	Sim	Sim	Sim	–	Sim	Não	Não	Não
Desenvolvimento da documentação	Média	Média	Média	–	Alta	Alta	Média	Alta
Prioridade na dissertação	Alta	Baixa	Média	Baixa	Baixa	Baixa	Baixa	Baixa

zero e máxima de cinco pontos. Quando a plataforma marca zero é porque não suporta os requerimentos do critério, ou precisa de uma integração com outro tipo de sistema. Três pontos se existe uma característica igual ao critério avaliado, mas deve ter um tipo de configuração específica dentro do ambiente da plataforma. Então uma nota de cinco pontos, é porque o *software* de controle suporta totalmente o critério que está sendo validado segundo as especificações da plataforma.

A comparação gerou como resultado a escolha de LinuxCNC como plataforma de controle para o *re-trofitting* do manipulador ASEA IRB6-S2, pois ele permite o gerenciamento das variáveis articulares do manipulador em tempo real, além de ter a característica da licença GNU o que diminui o custo financeiro em comparação com as plataformas proprietárias. Embora seja um controlador em desenvolvimento contínuo por meio de uma comunidade ativa, garante a aplicação da metodologia proposta de uma arquitetura aberta em virtude da modularidade presente no LinuxCNC.

3.6.2 Arquitetura de Controle Baseada em LinuxCNC

Após escolher a plataforma de controle para manipuladores é necessário realizar as configurações específicas segundo as particularidades de cada uma. Na fig. 3.8 é apresentada no modelo IDEF0 as atividades de configuração para a plataforma selecionada.

Neste caso, como a plataforma de controle para robôs selecionada foi o LinuxCNC, descrevem-se as atividades básicas para configurar o sistema de controle segundo o modelo IDEF0.

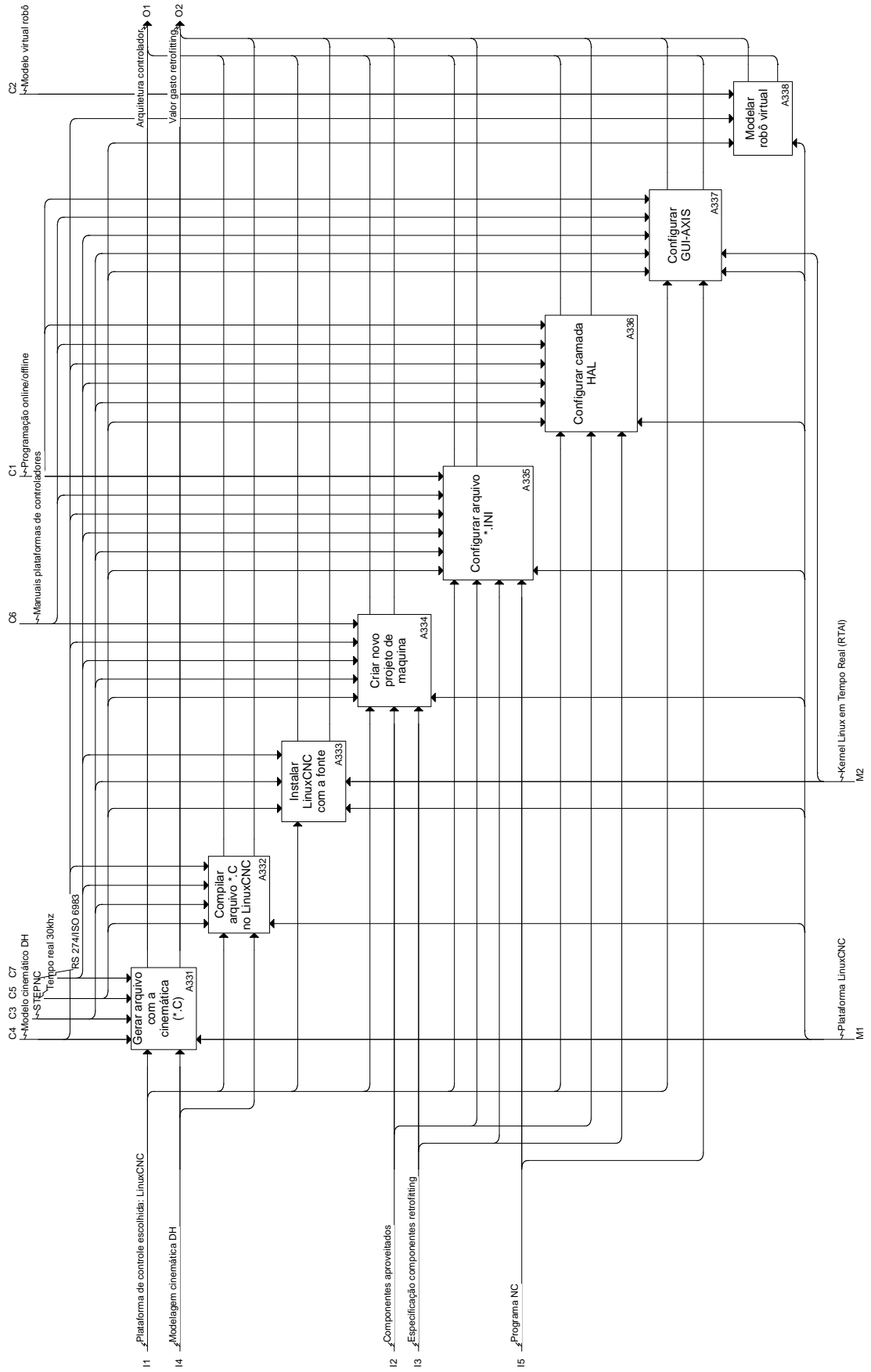


Figura 3.8: Diagrama IDEF0: Atividades configuração LinuxCNC

Tabela 3.2: Comparação entre as plataforma de controle para robô

Critério/Atividade	A321	A322	A323	A324	A325	A326	A327	A328
Cinemática Integrada à Plataforma	5	3	3	5	5	5	3	3
Código G e M Padrão RS 274	5	5	0	5	5	0	3	0
Programação <i>Offline</i>	5	0	0	5	5	3	3	0
Programação Teach-in	5	0	3	5	0	5	3	5
Hardware Baixo Custo	5	5	5	5	3	3	5	5
Software Opensource	5	3	5	3	5	0	5	3
Operação: MDI, Manual com Pendant e Automático	5	5	3	5	3	5	3	5
Integração via TCP/IP	5	5	5	5	0	5	5	5
Acesso Remoto	5	5	5	5	0	5	5	5
SO em Tempo Real	5	0	0	0	0	5	3	0
Integração	5	3	5	3	3	3	3	3
Manutenibilidade	5	5	5	3	5	5	5	5
Modelo virtual robô	5	3	3	5	5	3	0	5
Documentação disponível	3	3	3	3	3	5	3	5
Total Pontos	68	45	45	57	42	52	49	49

3.7 Montagem dos Componentes

A seguinte atividade proposta na metodologia IDEF0 é apresenta na fig. 3.9, constituída por sete atividades que viabilizaram a movimentação do robô por meio da plataforma selecionada, usando os componentes adquiridos e fabricados para que sejam integrados ao robô.

1. Montar estrutura motor-junta, cada uns dos graus de verdade do manipulador está composto por um conjunto de elementos que integrados geram a movimentação desejada de cada atuador. Nesta atividade estão os componentes comprados, fabricados e a especificação de cada um deles para que em conjunto possam ser montados na estrutura mecânica do robô.
2. Montar fontes de alimentação, nesta atividade a saída deverá ser um dos subsistemas que permitem o funcionamento do manipulador, mais especificamente a geração de tensão adequada segundo as especificações dos componentes necessários para a metodologia apresentada no presente capítulo.
3. Montar *drives* de controle, estes componentes especificados e adquiridos tem a responsabilidade de coordenar a resposta dos atuadores segundo os sinais gerados. Para cada grau de verdade é proposto um drive de controle. Este tipo de componente está equipado com um controlador PID que afeta o tempo de amortecimento do atuador para atingir a posição seguindo o sinal comandado.
4. Montar interface de comunicação, baseada na plataforma de controle para robôs manipuladores escolhida e a especificação dos componentes adquiridos, o sistema de comunicação para permitir a troca de sinais deve ser montada em concordância com as especificações de *Software* e *Hardware* definidas.

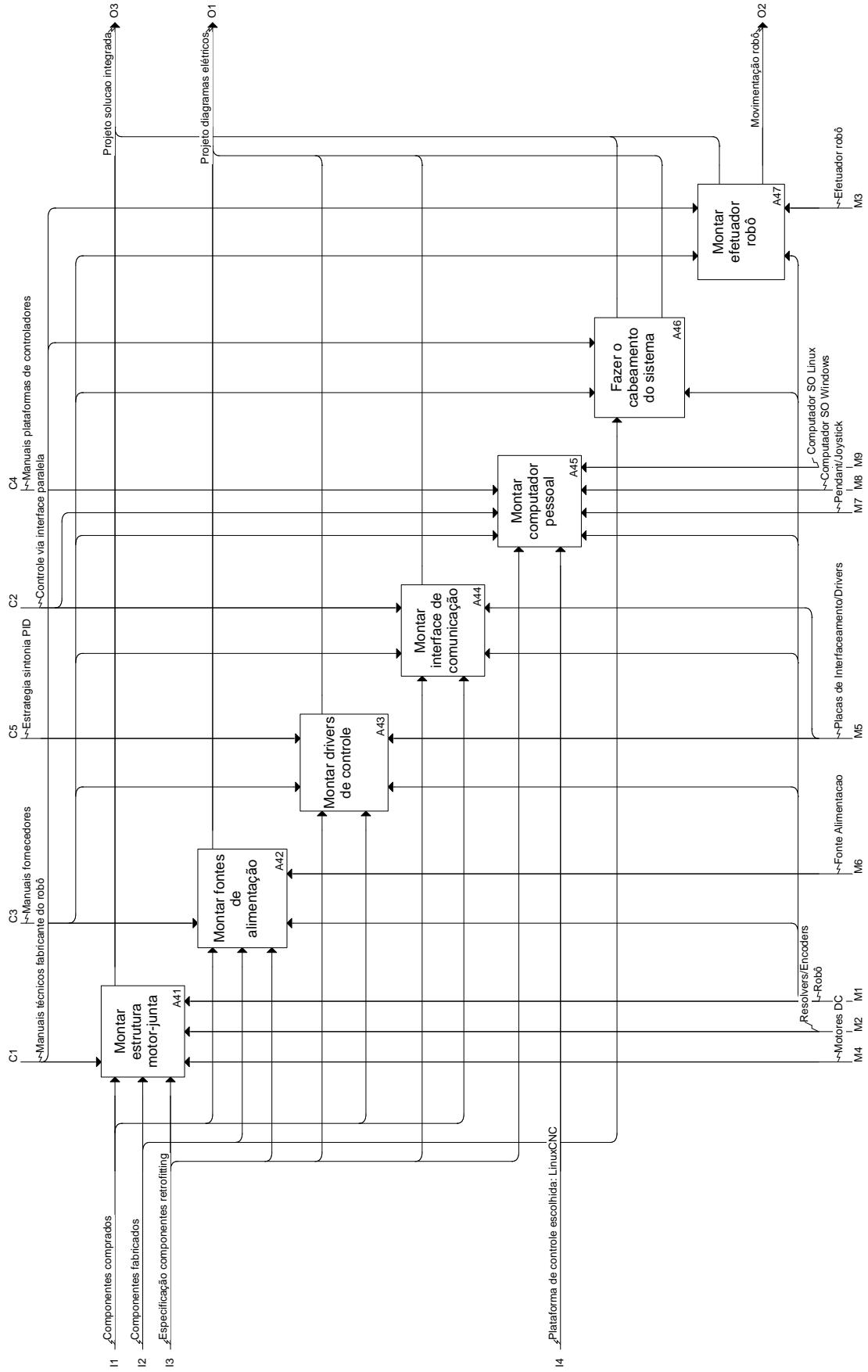


Figura 3.9: Diagrama IDEF0: Atividades da montagem

5. Montar computador pessoal, para instalar a plataforma de controle para robôs manipuladores um dos equipamentos propostos na presente metodologia é o uso de um computador pessoal com as especificações técnicas requeridas pelo software de controle anteriormente selecionado, deve ser configurado segundo os requerimentos da aplicação e montado para que seja compatível com os subsistemas que permitam o controle do robô.
6. Fazer o cabeamento do sistema, após o desdobramento dos diferentes componentes para permitir a movimentação do robô, é necessário fazer o cabeamento entre os diferentes subsistemas que compõem o controle do manipulador como sensores, drives, fonte de alimentação, atuadores etc.
7. Montar efetuador robô, como último elemento da operação que dependerá da aplicação final do manipulador, deve ser montado na última junta e levar em conta as especificações para que possa ser integrado com a plataforma de controle selecionada.

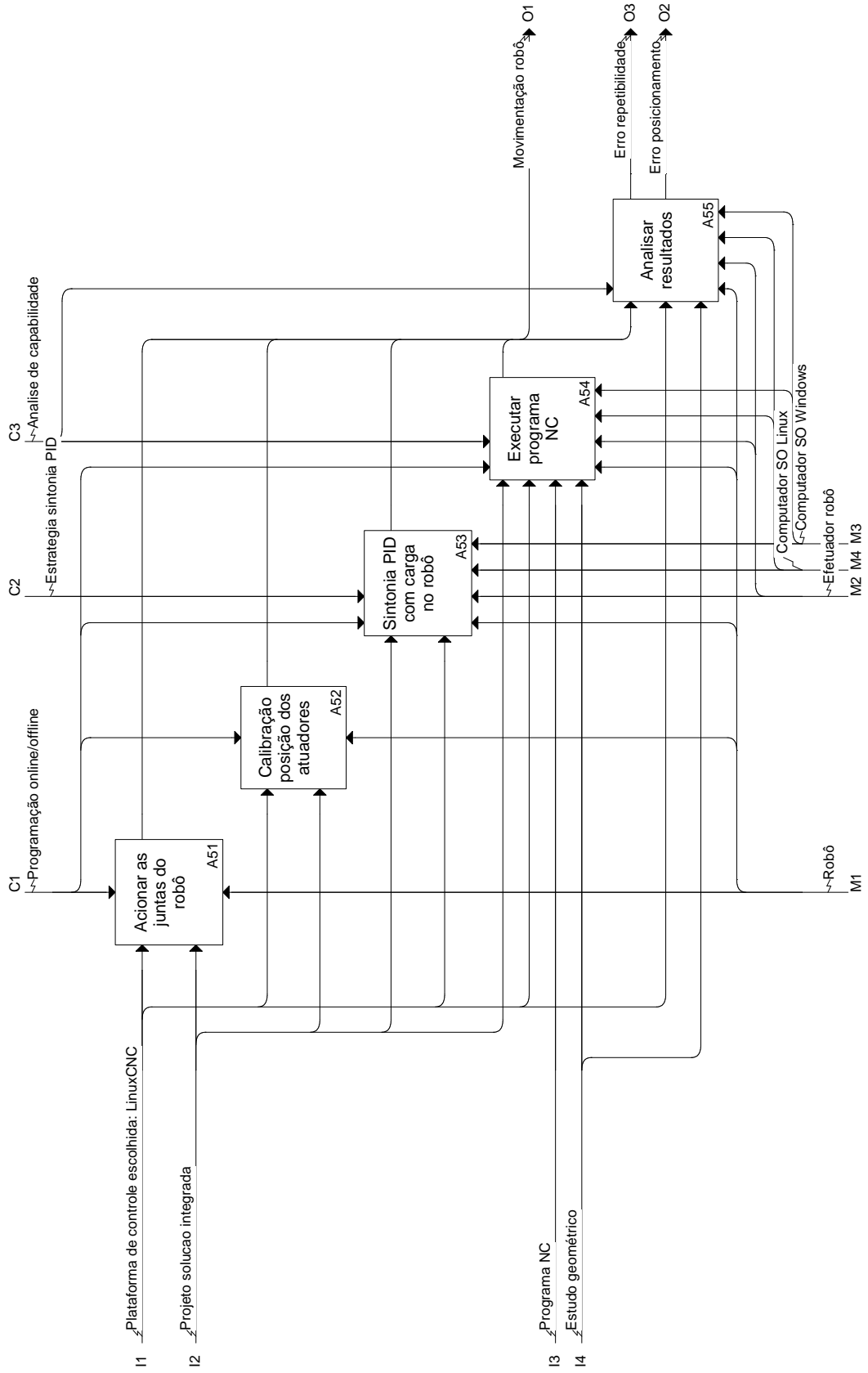
Ao finalizar a atividade associada com a Montagem dos componentes, deve existir uma movimentação inicial de cada um dos graus de liberdade do manipulador, garantindo desta forma que a comunicação entre os componentes e subsistemas montados ao longo da metodologia proposta.

3.8 Validação *Retrofitting*: Ensaios e Testes

Com o robô em operação após a montagem dos diferentes subsistemas necessários para o controle da solução integrada, é possível validar o resultado da implementação da metodologia proposta por meio da técnica *retrofitting*. Na fig. 3.10 são apresentadas por intermédio do modelo IDEF0, as atividades relacionadas com a atividade designada como validação *retrofitting*, por meio de ensaios e testes.

A validação para a montagem feita no robô está composta de cinco atividades que permitirá avaliar o desempenho da solução integrada com a metodologia descrita no presente capítulo.

1. Acionar individualmente as juntas do robô, a partir da plataforma software escolhida e também para verificar a troca e sinais enviados e recebidos nesta atividade é necessário movimentar cada uma das juntas, o que vai possibilitar corrigir possíveis falhas na comunicação ou nas conexões físicas feitas.
2. Calibração posição dos atuadores, para esta atividade deve-se ter uma referência na posição inicial de cada junta e uma estratégia para conhecer qual é o valor da rotação de cada uma, e desta forma, na plataforma de controle escolhida, configurar os parâmetros que permitem o deslocamento desejado de acordo com a aplicação viabilizada para o manipulador.
3. Sintonia PID com carga no robô, o controle de movimento desejado tem a opção de fazer o ajuste dos parâmetros deste controlador via *software* ou *hardware*. Esta sintonia deverá ser feita com carga máxima projetada com a qual o robô trabalhará.
4. Executar programa NC, o arquivo baseado na norma RS-274 deve existir como entrada para que o manipulador possa movimentar o efetuador até a posição especificada no programa NC.



Page 1

Figura 3.10: Diagrama IDEF0: Validação *retrofitting*: ensaios e testes

5. Analisar resultados, como parte da validação do desempenho do manipulador, foi proposto um estudo geométrico para conhecer o erro de repetibilidade e posicionamento do robô, permitindo saber desta forma se existe alguma possibilidade de melhora na configuração na plataforma de controle ou na montagem dos componentes instalados no robô.

Capítulo 4

Retrofitting do Robô ASEA IRB6-S2

4.1 Introdução

Este capítulo descreve a implementação da técnica *retrofitting* baseada na metodologia apresentada no Capítulo 3, com passos necessários para a desmontagem e para o diagnóstico da estrutura mecânica e dos componentes originais do robô, a especificação para atualizar o software e hardware para a integração do robô com tecnologia atual e com a montagem dos componentes, garantindo a movimentação desejada do robô a partir da plataforma software escolhida. Contudo será apresentado no presente capítulo os testes iniciais feitos para verificar a montagem dos componentes novos e fabricados no robô ASEA IRB6-S2.

4.2 Desmontagem e Diagnóstico do Manipulador

O robô fez parte do processo de pintura da empresa FIAT - fornecedora de carros no Brasil -, chegaram manipuladores da mesma referência por meio de uma doação para as universidades federais do País. Foi possível obter essa tecnologia para pesquisas segundo o critério de cada departamento acadêmico. O robô foi entregue à UnB, especificamente no Laboratório GRACO, onde foi usado por alguns anos na FMC para manipular peças brutas e terminadas. Mas de forma inesperada o sistema de controle deixou de funcionar, por falhas no sistema de fornecimento de energia.

O robô foi afetado pela falta de regulagem na tensão da cidade de Brasília, gerando assim um efeito "dominó", como a diferença considerável na tensão de saída para alimentar o sistema de controle. A não operação do robô gerou a oportunidade de aplicar a metodologia descrita na presente dissertação para o manipulador ser novamente operacional.

A seguir são descritas as atividades associadas na Figura 3.4 da metodologia apresentada no Capítulo 3, permitindo detalhar o processo para implementar a técnica *retrofitting* no manipulador ASEA IRB6-S2.

4.2.1 Diagnóstico do Estado Original do Robô

Foram identificados três subsistemas no diagnóstico do manipulador, para facilitar segundo o manual do fabricante conhecer quais são os componentes que deverão ser aproveitados e descartados. Os subsistemas são:

1. Estrutura física do robô.
2. Subsistema de alimentação.
3. Subsistema de controle.

A **estrutura mecânica** do robô estava em boas condições, por isso foi possível movimentar cada uma das juntas com os motores originais do robô sem ter problemas no momento da movimentação de cada junta. No momento da desmontagem dos componentes fixados ao robô foi confirmada as boas condições da estrutura mecânica para movimentar as juntas com a nova arquitetura de controle. A Figura 4.1 apresenta o estado original do robô antes de aplicar a técnica *retrofitting* com a metodologia proposta.

Além da estrutura mecânica, o robô foi operado desde o gabinete de controle projetado pelo fornecedor original do sistema geral, a Empresa chamada ASEA. A Figura 4.2 apresenta o gabinete antes do diagnóstico.

O gabinete contém os componentes principais dos **subsistemas de alimentação e controle**, o que facilitará o diagnóstico do manipulador. Os subsistemas apresentavam tecnologia da época que fora fabricado o robô, em meados dos anos 70.

Na Tabela 4.1 são apresentadas as especificações técnicas originais da fonte de alimentação quando foi projetada, associada ao subsistema de alimentação. Detalhando as entradas da bobina primária e saídas da bobina secundária do transformador.

Tabela 4.1: Especificações do transformador gabinete de controle

Tipo/Variável	Tensão (VAC)	Total
Entradas	220	3
Saídas	22, 24, 36, 52, 220	5

Fonte: ASEA (2003)

As saídas de tensão apresentadas anteriormente eram as responsáveis em fornecer energia a todo sistema elétrico no qual possibilitava a movimentação do manipulador. Posteriormente eram tratadas e retificadas para permitir sua regulação e garantir uma alimentação eficiente para o sistema. Foram identificadas as tensões de saída projetadas pelo fornecedor, desta forma realizou-se a medição de cada uma. Foi possível concluir que as saídas de tensão tinham alterações segundo a faixa projetada de operação em relação ao manual do fornecedor para o robô (ASEA, 2003). Desta forma foi possível inferir que uma das falhas na operação do subsistema de controle estava relacionada à falta de regulação no subsistema de alimentação.

O subsistema de controle original do robô foi projetado baseado em placas com circuitos elétricos/eletrônicos do ano 1975. Cada uma destas placas tinha uma responsabilidade para a operação do sistema tal



Figura 4.1: Manipulador inativo antes do *retrofitting*

como a regulação de tensão para acionar cada uma das juntas, como o módulo para calcular a movimentação do manipulador segundo sua cinemática, a leitura dos sinais lógicos enviados até os acionamentos do manipulador, dentre outros. Na Figura 4.3 são apresentadas algumas das placas que integravam o gabinete do manipulador.

Inicialmente, fora pensado aproveitar o maior número de componentes para aplicar a técnica de *retrofitting*, assim realizou-se um mapeamento dos sinais de controle e potência processadas pelo gabinete de controle. Foram identificados sinais que interagem diretamente no manipulador. Na Tabela 4.2 se apresentam o mapeamento feito:

Cada grau de liberdade do manipulador está integrado por cinco elementos representados na Tabela 4.2, o que será denominado como **conjunto de junta**. Cada conjunto de junta é composto por um motor, o qual



Figura 4.2: Gabinete de controle original do manipulador

Tabela 4.2: Sinais de controle e potência para o robô

Componente/Junta	Junta 1	Junta 2	Junta 3	Junta 4	Junta 5
Motor	2	2	2	2	2
Tacômetro	2	2	2	2	2
Resolver	6	6	6	6	6
Home sensor	2	2	2	2	2
Total	60 sinais				

Fonte: ASEA (2003)

transmite o movimento ao sistema de redução de cada junta. O tacômetro é responsável por adquirir a velocidade do motor, enquanto o *resolver* tem a capacidade de obter e transmitir para o gabinete de controle a posição atual do motor. Por último, existe um sensor de sincronização em cada conjunto de junta para garantir uma posição inicial conhecida para o sistema de controle, permitindo executar uma tarefa associada com o manipulador (ASEA, 2003). Na Tabela 4.3 são apresentadas as especificações dos componentes originais que compõem os cinco conjuntos de junta.

Foi adequado um efetuator para que o manipulador conseguisse movimentar peças no laboratório Graco da Universidade de Brasília. Uma garra eletro-neumática composta por uma bobina solenoide e uma garra neumática. As características se apresentam na Tabela 4.4.

Os manuais técnicos da bobina solenoide e a garra neumática podem ser consultados em (FESTO, 2014) e (SCHUNK, 2014), respectivamente.

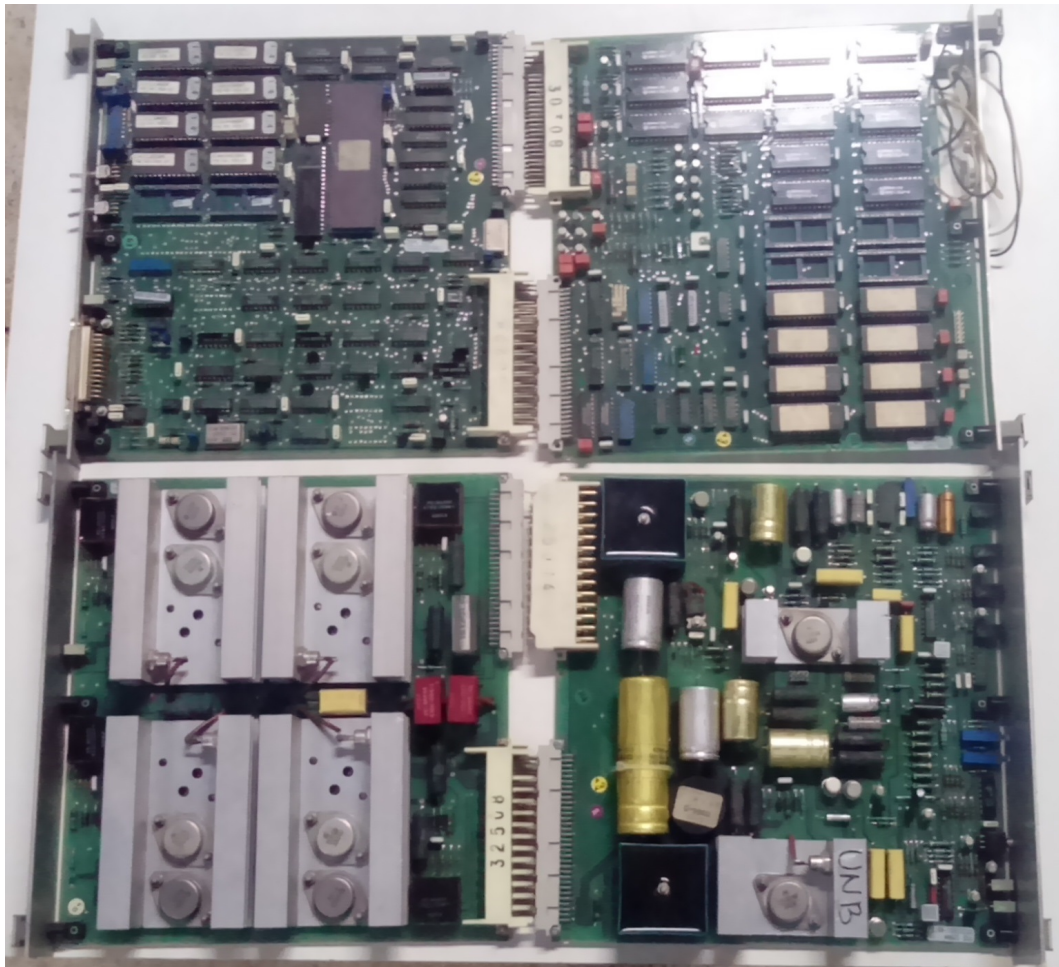


Figura 4.3: Placas de controle do gabinete original do manipulador

Tabela 4.3: Especificação dos componentes originais: Conjunto de junta

Componente/Variável	Tensão max. (VDC)	Corrente max. (A)
Motor	52	10
Tacômetro	6	NA
Resolver	15	NA
Home sensor	24	6

Fonte: ASEA (2003)

Tabela 4.4: Especificação técnica dos componentes presentes no efetuador

Item	Marca	Referencia
Bobina Solenoide	FESTO	MSFG-24/42-50/60
Garra neumatica	SCHUNK	PGN160/1

4.2.2 Desmontagem dos Componentes Originais do Manipulador

Com os subsistemas identificados (alimentação, controle e mecânico), e depois de observar a lógica do projeto feito pelo fornecedor para o robô ASEA IRB6-S2, é necessário desmontar os componentes dos subsistemas de alimentação localizados no gabinete de controle. O subsistema de controle formado pelas placas eletrônicas está localizado no gabinete original do manipulador. Assim como os conjuntos de junta, para os cinco graus de liberdade que fazem parte da estrutura mecânica do robô.

Foram desmontados os conjuntos das juntas associados aos graus de liberdade quatro e cinco do manipulador, posteriormente os conjuntos dois e três. A desmontagem foi finalizada com o primeiro grau de liberdade do manipulador.

O desenho feito pelo fornecedor da estrutura do manipulador com a configuração dos seus cinco graus de liberdade é apresentada na Figura 4.4.

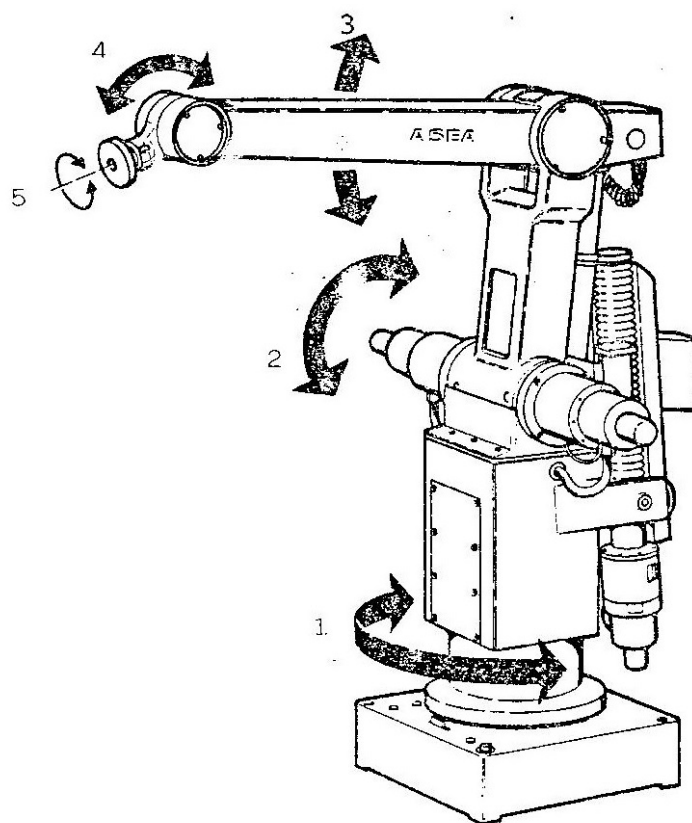


Figura 4.4: Definição da configuração do manipulador ASEA IRB6-S2
Fonte: ASEA (2003)

Os conjuntos das juntas foram desmontados individualmente, permitindo assim observar as características internas da estrutura mecânica do manipulador. Os conjuntos associados às juntas quatro e cinco estão localizados no robô segundo a Figura 4.5.

A Figura 4.6 apresenta a fixação do conjunto da junta cinco diretamente na estrutura mecânica do robô. É necessário esclarecer que os componentes associados aos graus de liberdade quatro e cinco são similares.

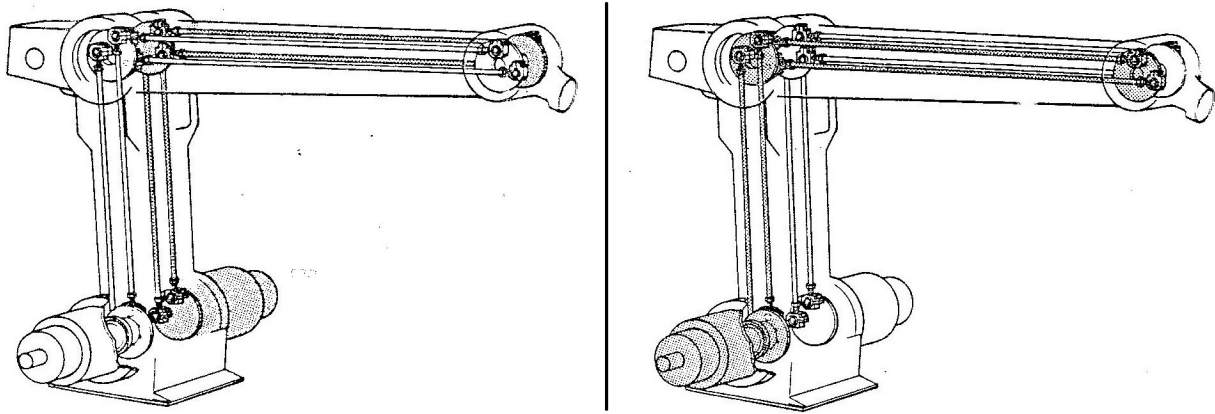


Figura 4.5: Conjunto de juntas quatro e cinco
Fonte: ASEA (2003)

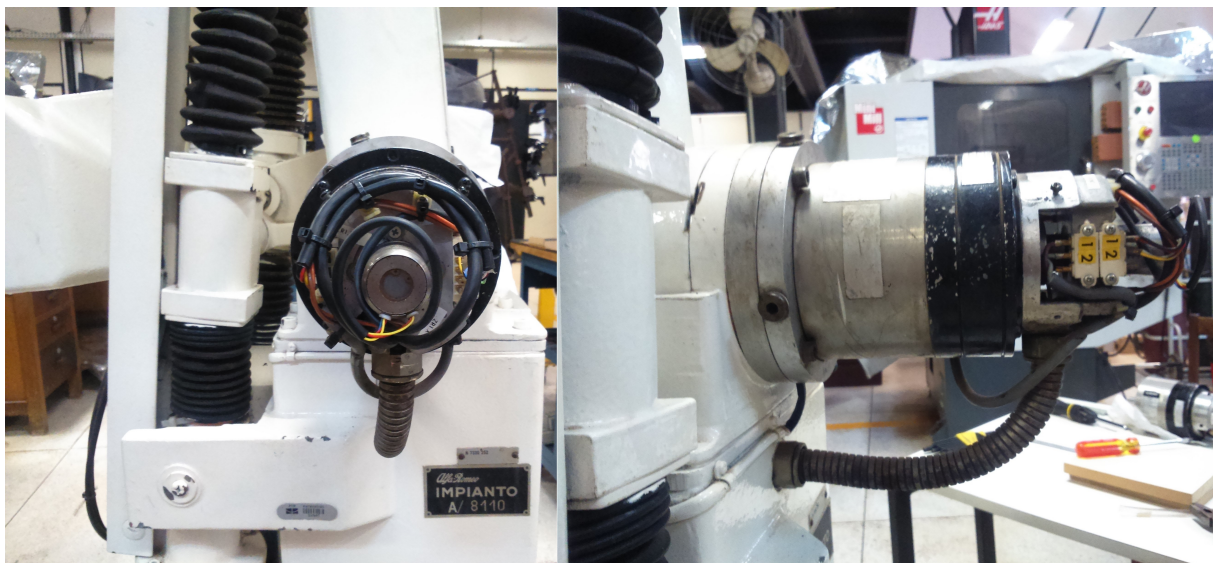


Figura 4.6: Conjunto da junta cinco acoplado ao manipulador

Depois de desmontar o conjunto da junta foi possível identificar o tipo de acoplamento presente no manipulador. A Figura 4.7 apresenta como é integrado o conjunto das juntas quatro e cinco.

Foi possível identificar os componentes que conformam o conjunto das juntas quatro e cinco, após a desmontagem dos motores que transmitem a movimentação para os últimos dois graus de liberdade do manipulador. Incluindo mais dois elementos, o acoplamento fixado ao motor e a flange responsável pela fixação do motor à estrutura mecânica do manipulador. O conjunto das juntas quatro e cinco está composto por:

1. Motor.
2. Tacômetro e Resolver.
3. Acoplamento: *Harmonic-drive*.
4. Flange de fixação.

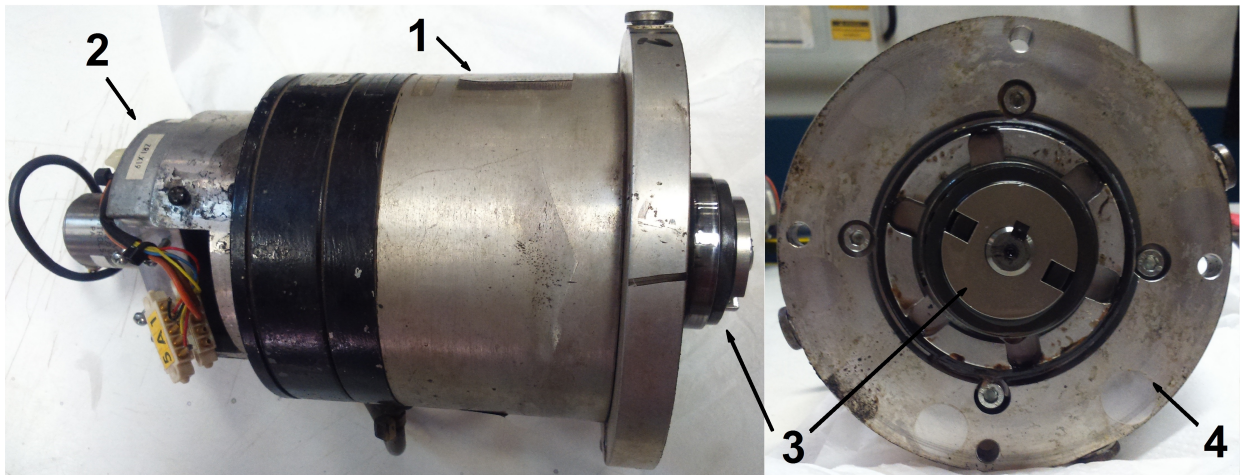


Figura 4.7: Detalhe do conjunto das juntas quatro e cinco
a) Vista lateral; b) Vista frontal

O acoplamento que faz parte das juntas quatro e cinco é denominado *Harmonic Drive*, o mesmo está dividido em dois elementos: um deles fixado ao eixo do motor e o outro à estrutura mecânica do manipulador. O elemento fixado ao motor é conhecido como gerador de onda, ou *wavegenerator*, enquanto o elemento que faz parte da estrutura mecânica do manipulador é denominado como *Flexspline* (LLC Harmonic Drive, 2016). Na Figura 4.8 é apresentado o elemento do *Harmonic Drive* localizado na estrutura do manipulador e o *wavegenerator* que estava acoplado ao eixo motor.

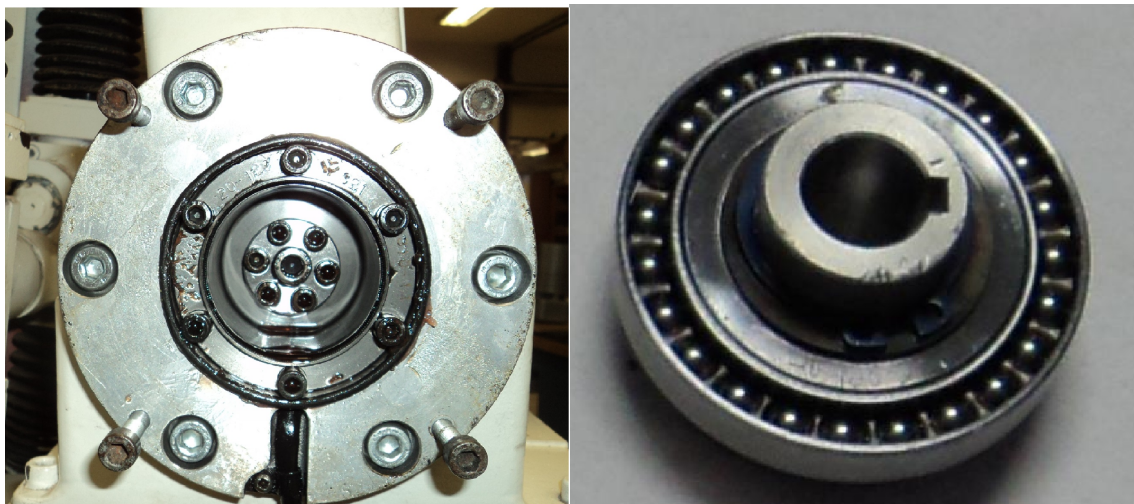


Figura 4.8: *Harmonic-Drive* da junta cinco
a) *Flexspline*, b) *Wavegenerator*
Fonte: Toquica e Álvares (2016)

Na Figura 4.9 é apresentada o flange de fixação separada do motor, elemento integral do conjunto das juntas quatro e cinco.

Este tipo de acoplamentos deve ter 30 ml de óleo tipo ATF para transmissões automáticas automotivas para a lubrificação do *Harmonic-Drive*, em cada uma das juntas quatro e cinco.



Figura 4.9: Flange das juntas quatro e cinco

Após identificar os componentes que formam as juntas quatro e cinco, foram desmontados os conjuntos das juntas dois e três, localizados na estrutura do manipulador tal como é apresentado no desenho feito pelo fornecedor na Figura 4.10.

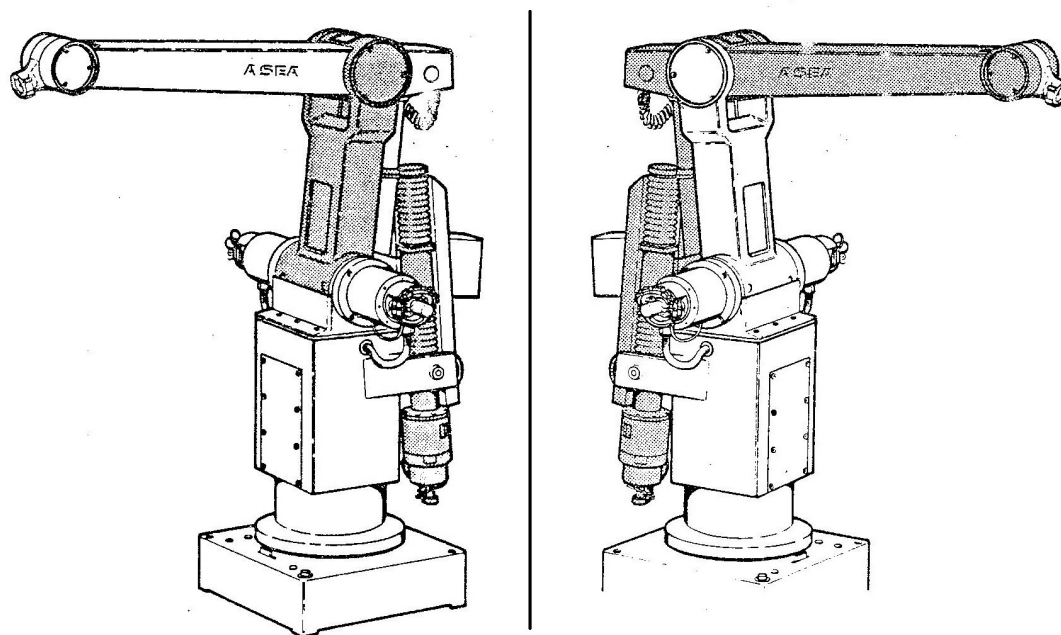


Figura 4.10: Desenho dos conjuntos das juntas dois e três
Fonte: ASEA (2003)

Os conjuntos das juntas dois e três instalados na estrutura do robô ASEA IRB6-S2 é apresentado na Figura 4.11.



Figura 4.11: Conjunto de juntas dois e três no manipulador ASEA IRB6-S2

Na Figura 4.12 são apresentados os elementos desmontados da estrutura mecânica do manipulador. A diferença com os elementos das juntas quatro e cinco é a inexistência das flanges de fixação entre o motor e a estrutura mecânica do robô, ou seja, os motores são fixados diretamente na estrutura mecânica do manipulador.

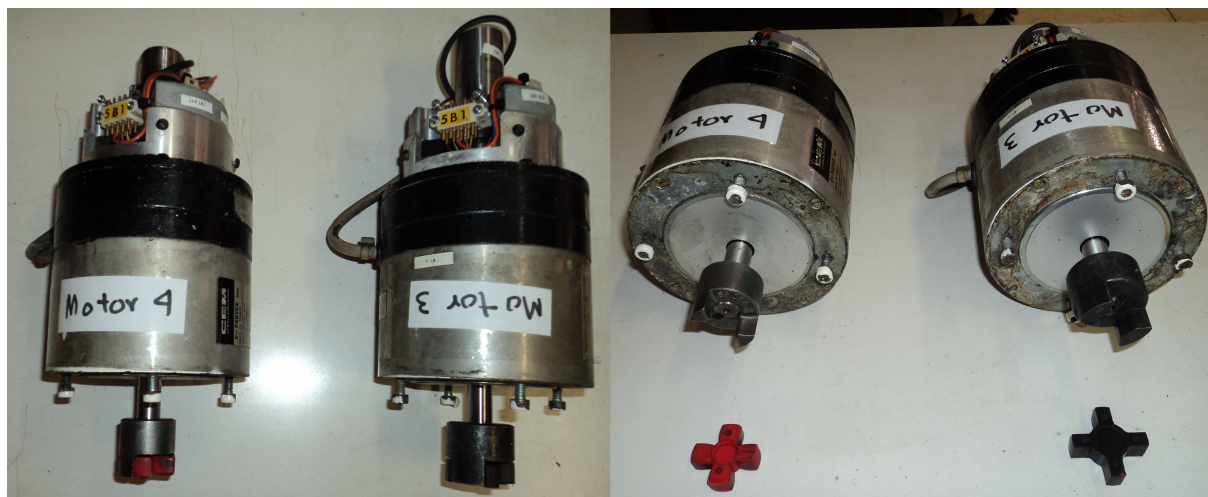


Figura 4.12: Elementos do conjunto das juntas 2 e 3

Outra diferença existente é o acoplamento, neste caso existe uma conexão direta do motor com um parafuso de esferas transformando o movimento rotacional dos motores em movimentação linear. Os conjuntos de junta dois e três têm os seguintes elementos:

- Motor
- Tacômetro e Resolver
- Acoplamento: Direito a um parafuso de esferas

O conjunto de junta um foi o último a ser desmontado, e é apresentado na Figura 4.13.

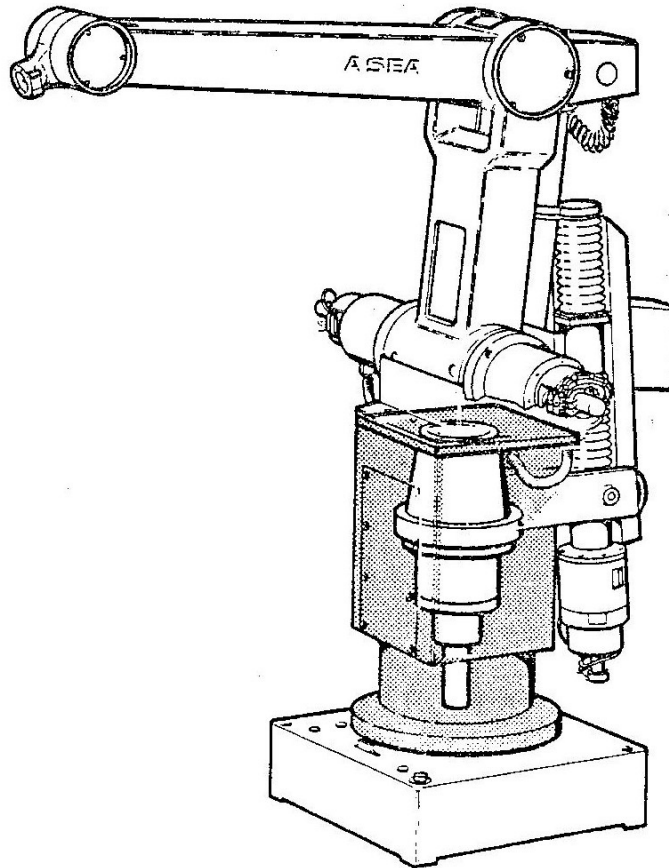


Figura 4.13: Desenho do conjunto da junta um
Fonte: (ASEA, 2003)

Para permitir a desmontagem foi necessário tirar a estrutura mecânica dos conjuntos das juntas anteriores que estão apoiadas na base do manipulador ou no primeiro grau de liberdade. Foi usado um guincho hidráulico para suportar o peso da estrutura mecânica das quatro últimas juntas, enquanto os componentes da junta um eram extraídos. Pode-se observar na Figura 4.14 o guincho usado.

Na Figura 4.15 é apresentado o procedimento gráfico para remover - segundo o manual do fornecedor - o conjunto de componentes da junta um. O primeiro passo foi segurar a estrutura mecânica dos últimos quatro graus de liberdade ao guincho, para depois afastá-la da base do manipulador. O peso da estrutura afastada é de 150 kg (ASEA, 2003). Foi fabricada uma ferramenta de extração de pinos que fixam a base com a estrutura afastada. Pode ser consultado o desenho técnico no Apêndice F, assim como o detalhe para usar a ferramenta é apresentado no Apêndice G.



Figura 4.14: Guincho hidráulico usado para a desmontagem

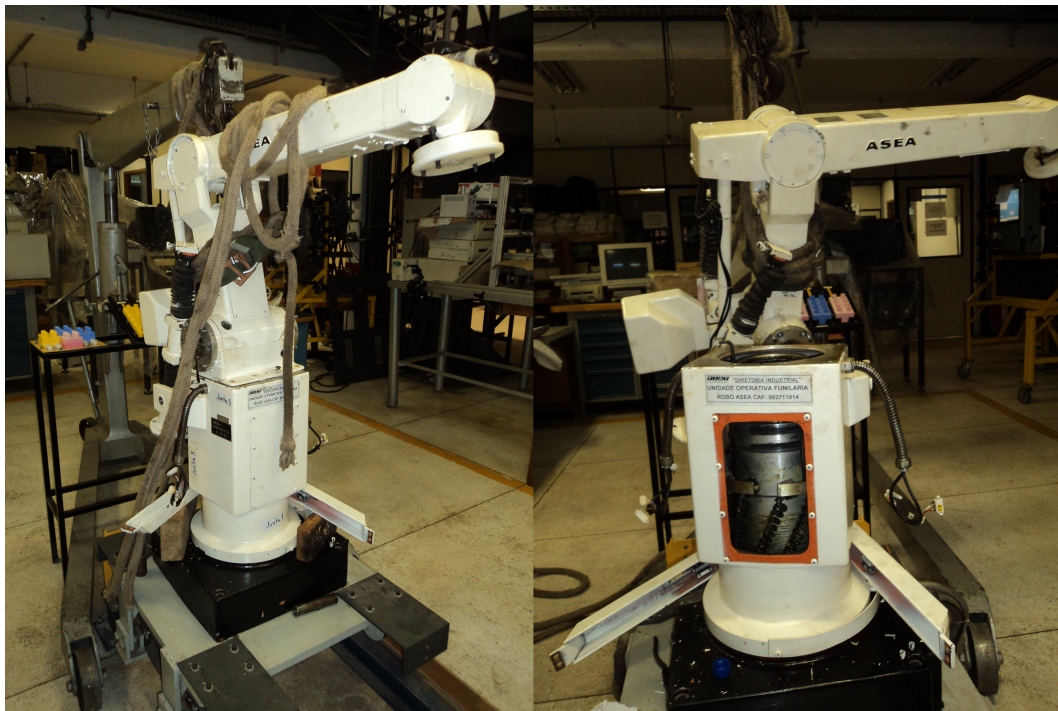


Figura 4.15: Desmontagem da estrutura mecânica últimas quatro juntas
a) Segurar a estrutura; b) Separar da base do manipulador

Foi possível definir para a junta da base os componentes necessários para uma operação adequada. É viável identificar que existe uma placa intermediária para fixar a estrutura mecânica da base com a armação afastada das quatro últimas juntas. A diferença das juntas quatro e cinco pode ser observada na Figura 4.16.

Foi identificado o tipo de acoplamento existente na junta um após retirar a placa intermediária, sendo

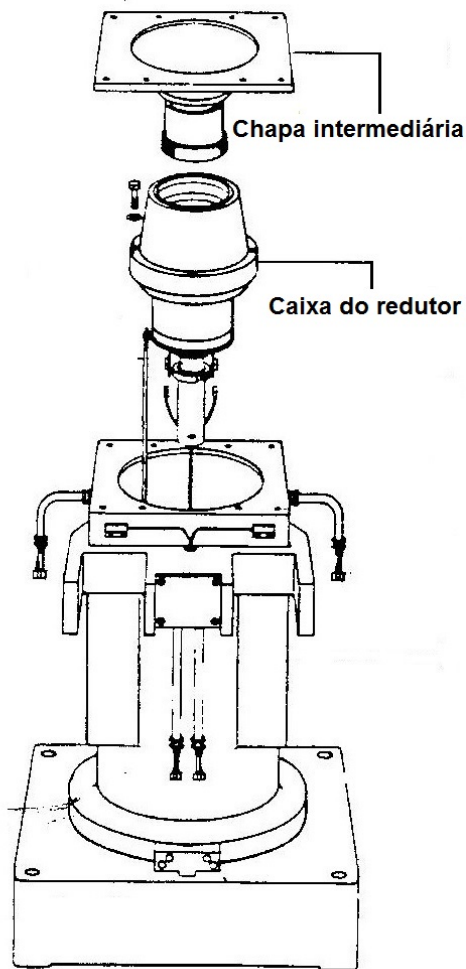


Figura 4.16: Desenho em detalhe do conjunto da junta um
Fonte: (ASEA, 2003)

igual as juntas quatro e cinco, ou seja, *Harmonic Drive*. A Figura 4.17 apresenta o conjunto da junta um posicionado na base do manipulador.

O *Harmonic-Drive* fixado ao eixo do motor é apresentado na Figura 4.18. A junta um deve ter 260 ml de óleo tipo ATF, característico das transmissões automáticas automotivas, garantindo a lubrificação deste tipo de acoplamentos (ASEA, 2003).

Foi possível redefinir os componentes que integram o conjunto da junta um em comparação aos anteriores, formado por:

1. Motor
2. Tacômetro e Resolver
3. Acoplamento: *Harmonic-drive*
4. Chapa intermediária

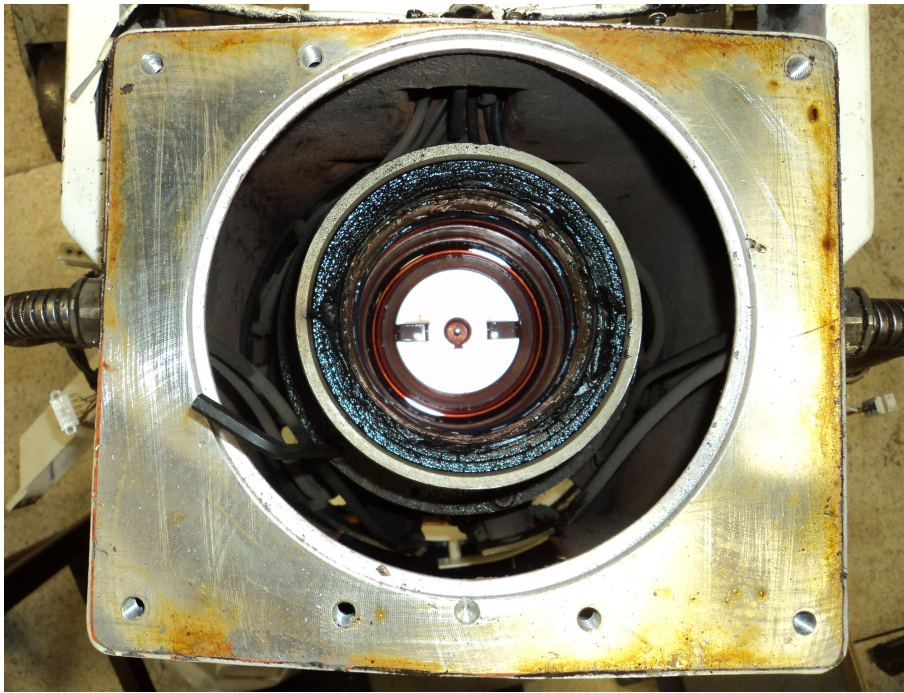


Figura 4.17: Conjunto da junta um na base do manipulador ASEA IRB6-S2

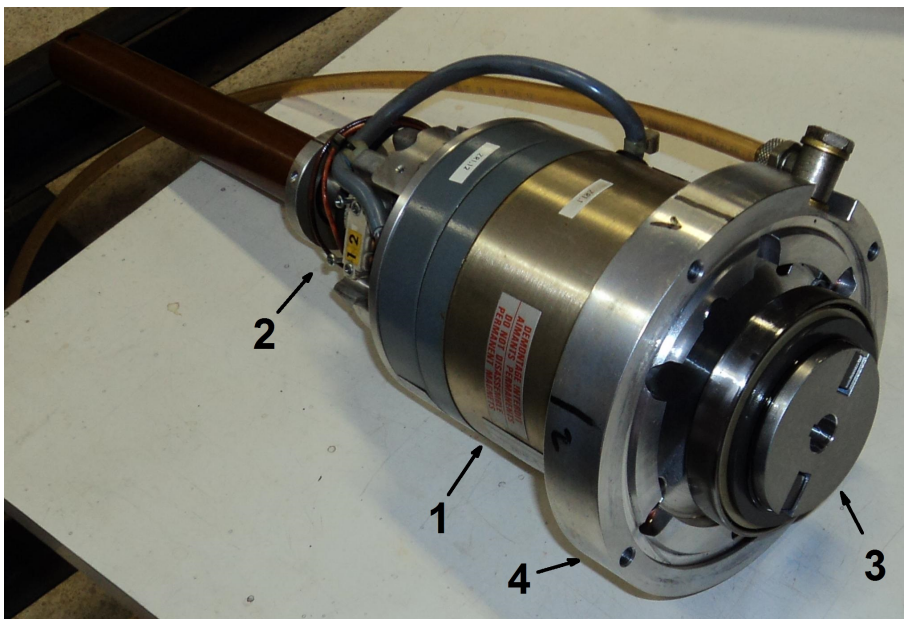


Figura 4.18: Elementos do conjunto da junta um fora da base do robô

Após a desmontagem dos conjuntos das juntas que constituem os cinco graus de liberdade do manipulador é possível conhecer quais são os componentes aproveitados e descartados. A seleção priorizou os componentes originais associados à estrutura mecânica do manipulador e deixou de lado os elementos do subsistema de controle das juntas, como os motores e resolver. Na Tabela 4.5 é apresentada a relação do conjunto das juntas.

Tabela 4.5: Especificação dos componentes originais: Conjunto de junta

Componentes	Junta 1, 4 e 5	Junta 2 e 3
Harmonic-Drive	Aproveitado	-
Acoplamento	-	Aproveitado
Motores	Descartado	Descartado
Tacômetro	Descartado	Descartado
Resolver	Descartado	Descartado
Flange/Chapa Intermediária	Aproveitado	-

Do gabinete do controle original do manipulador aproveitou-se a estrutura para colocar os novos componentes dos subsistemas de alimentação e controle. Além disso, aproveitou-se o cabo de controle entre o gabinete e o robô, para a parte de tensão dos motores novos. No entanto os sinais de controle como o feedback do avanço dos motores, pulsos comandados a partir da plataforma software de controle etc. Foi descartado o cabo original pois estava desgastado por causa do tempo, o que geraria ruído, e assim falhas na transmissão dos dados.

4.3 Atualização *Hardware*

É apresentado a seguir todo o processo relacionado à atualização dos componentes *hardware* para a operação do manipulador ASEA IRB6-S2, em referência a Figura 3.5 da metodologia apresentada no Capítulo 3.

Com a informação disponível das características técnicas dos componentes reaproveitados na etapa de diagnóstico é necessário especificar quais devem ser os novos elementos que deverão ser projetados, fabricados e adquiridos para o *retrofitting* do manipulador.

4.3.1 Especificação de Componentes para o *Retrofitting*

Para os subconjuntos de operação do manipulador identificados na Seção 4.2, serão especificadas as características técnicas para cada subconjunto. Na Tabela 4.6 são apresentados os componentes que devem ter cada um dos subconjuntos para a operação do manipulador.

Tabela 4.6: Disposição dos componentes por cada subconjunto do sistema

Subconjunto	Juntas para 5 DOF
Alimentação	Gabinete
Controle	Gabinete e no Robô
Elementos locomoção	Robô

Com a distribuição física proposta para o *retrofitting* dos subconjuntos que interagem o controle do sistema do manipulador é necessário definir os componentes básicos para cada um dos subsistemas, o qual pode ser observado com mais detalhe na Tabela 4.7. Ainda não foram especificados os detalhes técnicos.

Tabela 4.7: Definição componentes dos subconjuntos do robô

Subconjunto	Junta 1	Junta 4 e 5	Junta 2 e 3	Efetuator
Alimentação	Fonte de tensão para os motores			Fonte tensão e pneumática
Controle	Interface comunicação, Driver, dispositivo feedback			Interface comunicação, dispositivo feedback
Elementos locomoção	Motor, <i>harmonic drive</i>	Motor, redutor original		<i>Fingers</i>

Para a atualização tecnológica do segmento *hardware* para a operação do manipulador ASEA IRB6-S2, é necessário substituir os componentes que foram descartados na 4.2 em cada um dos subsistemas do manipulador. Estes elementos devem ser compatíveis para manipuladores industriais aptos a torna-se operacionais novamente em um espaço acadêmico ou industrial. Na Tabela 4.8 são apresentados os componentes a serem adquiridos (BOMFIM et al., 2014) e atualizados segundo novas referências. Da mesma forma se apresenta o custo dos componentes para o *retrofitting*, o que corresponde ao valor total aproximado investido no projeto, devido a que é usada uma plataforma *open source* na parte *software*.

Tabela 4.8: Especificação dos componentes para aquisição

Item	Componente	Marca	Referencia	Qnt	Custo (USD)
1	Interface paralela	CNC4PC	C11 breakout board	1	100,0
2	Driver digital	Gecko Drive	G320X	5	605,0
3	Motor tipo servo	Keling Technology, Inc	KL23-130-60	5	435,0
4	Encoder digital	CUI Inc	AMT102	5	205,0
5	Fonte de alimentação	Automation Technology	KL-6020	1	130,0
6	Computador pessoal	-	Interface LPT1	1	200,0
Custo total					1675,0

Com os componentes especificados é necessário projetar os que deverão ser fabricados para vincular os elementos originais da parte mecânica da estrutura do manipulador, especificamente no sistema de redução dos conjuntos de junta. Assim como uma placa de capacitores responsável por minimizar a corrente máxima/pico quando os motores iniciarem a operação (BOSTICK, 2013). A Tabela 4.9 apresenta os componentes que devem ser projetados e fabricados.

Tabela 4.9: Componentes do *retrofitting* para fabricação

Conjunto	Relação entre	Componente	Qnt
Junta 1	Motor e <i>harmonic drive</i>	Luva	1
Junta 2 e 3	Motor e parafuso bolas	Luva	2
	Motor e estrutura mecânica robô	Flange	2
Junta 4 e 5	Motor e <i>harmonic drive</i>	Luva	2
Alimentação	Fonte e subsistema controle	Placa de capacitores	1

4.3.2 Projeção e Fabricação de Novos Componentes para o *Retrofitting* do Robô ASEA IRB6-S2

Para a integração dos componentes originais do manipulador e os novos elementos adquiridos, é necessário projetar e fabricar as luvas, flanges e o banco de capacitores indicados na Subseção 4.3.1.

4.3.2.1 Projeto e Fabricação das Luvas

É necessário projetar por meio do software CAD/CAM as peças que possibilitem o acoplamento dos novos motores para que seja compatível com os *harmonic drives* originais do robô. As luvas têm que ser projetadas pois os antigos eixos dos motores tem um diâmetro maior do que os novos. Na Figura 4.19 é apresentado o projeto final em software tipo CAD.

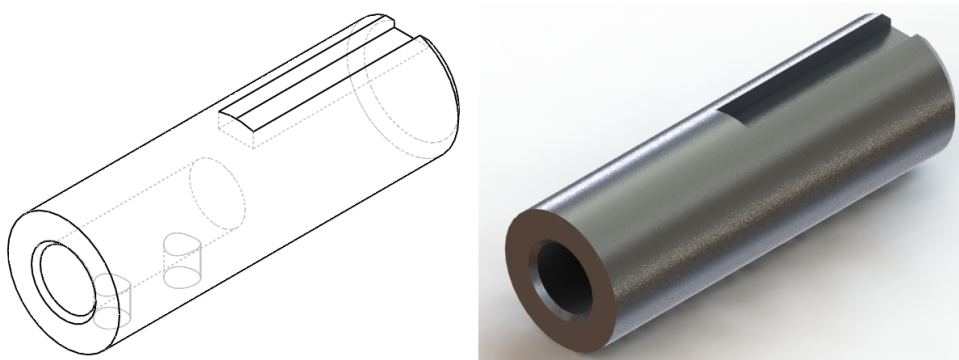


Figura 4.19: Luvas projetadas e fabricadas

Após o planejamento das luvas, as quais foram fabricadas em uma impressora 3D com material ABS e posteriormente foram fixadas aos novos motores como é apresentado na Figura 4.20.

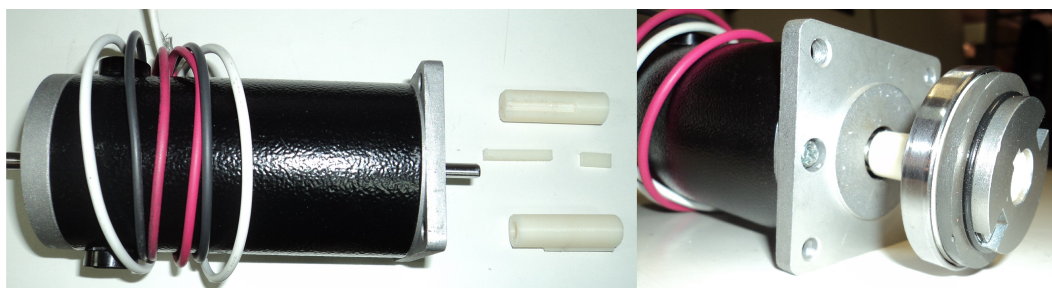


Figura 4.20: Motor com o *Harmonic drive*
a) Servomotor com Luvas em ABS; b) Servomotor antes de Montagem

Após os testes de ajuste entre o eixo do motor e o *wave-generator*, a luva foi fabricada em alumínio tipo 2024-T3. Como os eixos dos motores originais do manipulador eram iguais em suas dimensões, foi possível fazer um único projeto para as cinco luvas dos cinco graus de liberdade do manipulador ASEA IRB6-S2, para adaptar o *wave-generator* do *harmonic drive* aos eixos dos motores selecionados. O projeto detalhado das luvas pode ser consultado no Apêndice C.

4.3.2.2 Projeto e Fabricação das Flanges: Juntas Dois e Três

É necessário projetar e fabricar duas flanges para fixar o motor do conjunto das juntas dois e três aos parafusos de esferas associados à estrutura mecânica do manipulador. Originalmente o fornecedor do robô não projetou as flanges, mas segundo as dimensões dos novos motores é vital a fabricação das peças para o adequado funcionamento da redução tipo parafuso de esferas. Na Figura 4.21 é apresentada a flange projetada e que foi fabricada em alumínio 2024-T3.

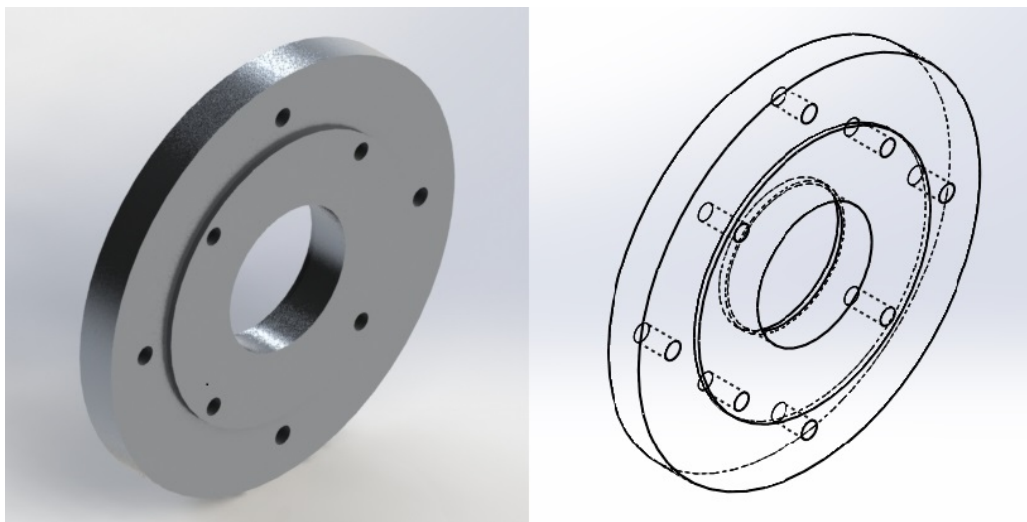


Figura 4.21: Flange fabricada e projetada para a junta dois

Foi projetada a flange para unir o motor da junta três a estrutura mecânica do manipulador. Posteriormente a que foi fabricada no mesmo material (2024-T3) que a flange do conjunto da junta três. A diferença com as dimensões da flange associada à junta dois foi no diâmetro do furo central da flange, sendo proporcional ao diâmetro do acoplamento entre o motor e o parafuso das esferas da estrutura mecânica do manipulador.

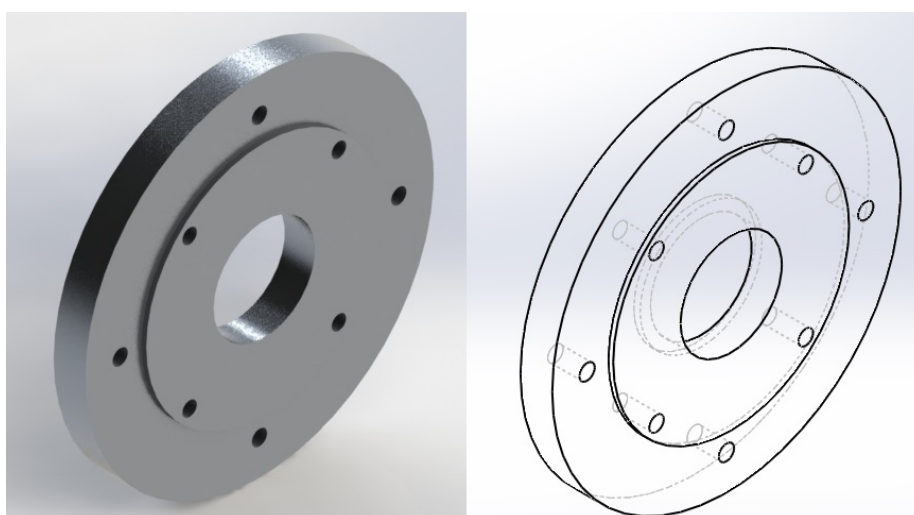


Figura 4.22: Flange fabricada e projetada para a junta três

Os planos técnicos do projeto com maior detalhe para cada uma das flanges projetadas e fabricadas pode ser consultado no Apêndice D e Apêndice E.

4.3.2.3 Projeto e Fabricação do Banco de Capacitores

O manipulador vai ser operado de forma contínua para tarefas próprias de uma célula de manufatura flexível (FMC) presente no Laboratório GRACO da Universidade de Brasília. Além disso, esses robôs industriais foram projetados para funcionarem sem interrupção na indústria. Os motores selecionados na Subseção 4.3.1, segundo suas características técnicas disponíveis no (Keling Technology Inc, 2016), têm uma corrente de pico e uma corrente de operação. A corrente de pico é associada ao torque máximo necessário para que o motor inicie a operação saindo do estado de inércia (BOSTICK, 2013). Na Tabela 4.10 as especificações técnicas do motor necessárias para estabelecer as características do banco de capacitores.

Tabela 4.10: Especificação do motor DC - KL23-130-60

Variavel	Valor nominal	Valor máximo
Corrente pico (I_p)	3.5 A	20 A
Tensão (V_m)	60 VDC	60 VDC
Frequência rede (f_r)	60 Hz	60 Hz
Tempo corrente pico (T_p)	-	90 ms

Fonte: (Keling Technology Inc, 2016)

Foi necessário projetar e fabricar uma placa de capacitores com a capacidade de suportar o tempo requerido até que os motores associados às juntas do manipulador vençam a inércia. Os capacitores usados foram eletrolíticos, recomendados para aplicações com um período de tempo curto, onde existe a fase do torque máximo gerado pelo motor (NATARAJAN, 2005). Nas Eq. 4.1 apresentam-se as operações matemáticas para se obter o valor da capacitância do banco de capacitores para a operação contínua e sem falhas dos cinco motores do manipulador ASEA IRB6-S2 (BOMFIM, 2013).

$$Q_{capacitor} = T_p \cdot I_p = 1.8C \quad (4.1a)$$

$$c = \frac{Q_{capacitor}}{V_m} = 0.03 \text{ F} = 3 \times 10^4 \mu\text{F} \quad (4.1b)$$

$$c_{projetada} = c + 20\% = 3.6 \times 10^4 \mu\text{F} \quad (4.1c)$$

$$V_{projetado} = V_m + 20\% = 72 \text{ V} \quad (4.1d)$$

Para a operação dos cinco motores com a corrente de pico, é necessário que o banco de capacitores tenha $3.6 \times 10^4 \mu\text{F}$. Foi projetada a fonte com quatro capacitores, cada um de $1 \times 10^4 \mu\text{F}$, para uma capacitância projetada total do banco de capacitores de $4 \times 10^4 \mu\text{F}$, além da tensão projetada para cada um dos capacitores de 72 V. Se apresenta na Tabela 4.11 o detalhamento das características do banco projetado com os valores de capacitores comerciais.

A PCB foi projetada para fixar os quatro capacitores, a mesma foi desenvolvida através de uma fresa

Tabela 4.11: Especificação do banco de capacitores

Elemento	Tipo	Capacitância	Tensão	Qnt
Capacitor	Electrolítico	$1 \times 10^4 \mu\text{F}$	80 V	4

CNC na Universidade de Brasília, adaptada para fazer placas de circuito impresso simples. O desenho técnico para a fabricação da placa pode ser consultado no Apêndice H. Na Figura 4.23 é apresentada a máquina no momento da fabricação da PCB do banco de capacitores.

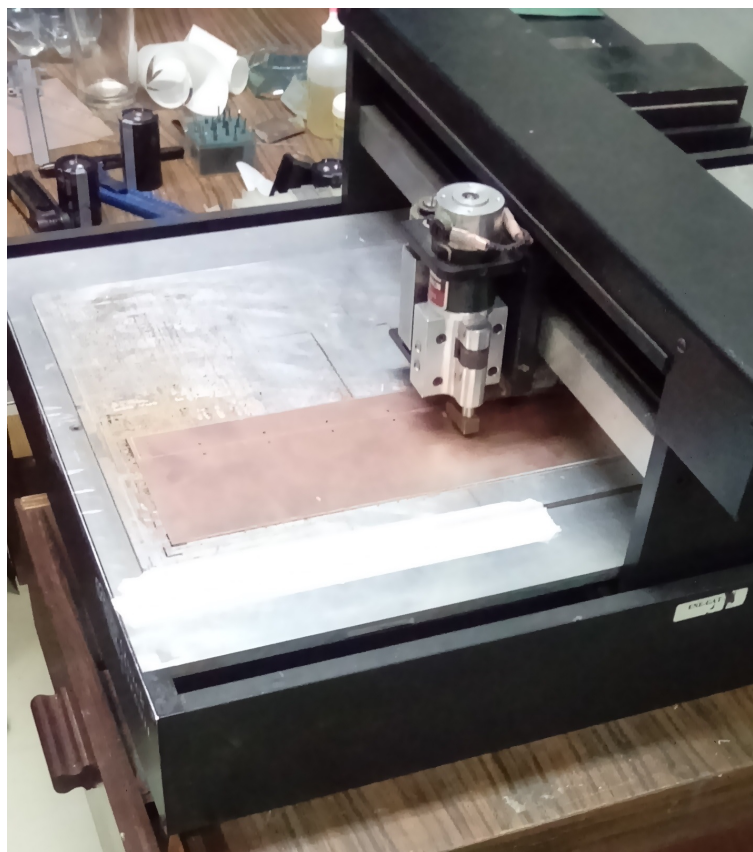


Figura 4.23: Processo de fabricação da PCB para o banco de capacitores

Na Figura 4.24 é apresentado o banco de capacitores projetado com os componentes instalados na placa fabricada. O banco é parte do subsistema de alimentação do manipulador, responsável por fornecer a energia requerida para o controle do robô.

Foram projetados e fabricados os componentes dos subsistemas identificados para o controle do manipulador, além de ter as especificações dos componentes adquiridos para continuar com a metodologia proposta baseada na técnica *retrofitting* para que o robô ASEA IRB6-S2 se torne funcional. O desenho electro/eletrônico do sistema de controle é apresentado no Apêndice I, com a interligação dos componentes escolhidos e adquiridos para o *retrofitting*.



Figura 4.24: Banco de capacitores projetado para o manipulador

4.4 Atualização Software

A seleção de uma plataforma de controle é indispensável na metodologia apresentada no Capítulo 3, integrando a geração de tarefas específicas e os cálculos da cinemática do robô, além da compatibilidade dos elementos escolhidos na Seção 4.3. São detalhados os passos associados à Figura 3.6, com as características e a configuração da plataforma LinuxCNC, permitindo enviar e receber instruções específicas ao manipulador.

4.4.1 Características Mínimas do Hardware: LinuxCNC

A plataforma de controle para manipuladores, neste caso para o ASEA IRB6-S2, deverá ter características recomendadas de acordo com suas capacidades para o desempenho esperado do subsistema de controle. Uma das características necessárias para esse tipo de plataforma é a latência, ou retardo máximo que pode ter o computador no momento em que o controlador esteja fazendo cálculos para movimentar o manipulador. Desta forma é necessário um sistema operacional com resposta em tempo real, garantido baixa latência do computador e continuidade na execução das tarefas feitas pela plataforma. São apresentados na Tabela 4.12 os requerimentos recomendados a nível de *hardware* da plataforma LinuxCNC na versão 2.5.4, com o sistema operacional Ubuntu 10.04:

Tabela 4.12: Especificação e características do hardware para LinuxCNC

Característica	Valor recomendado
Processador	1.2 GHz
RAM	1 GB
Disco rígido	8 GB
Resolução placa gráfica	1024x1078

Fonte: LinuxCNC.org (2015a)

4.4.2 Incluir Cinemática não-trivial no LinuxCNC

Com as equações cinemáticas de um manipulador é possível incluir na plataforma LinuxCNC o modelo matemático para uma nova configuração não existente. A descrição matemática do modelo cinemático do manipulador foi apresentado na Subseção 2.6.3. São apresentados os passos necessários para incluir na plataforma LinuxCNC as equações associadas ao manipulador ASEA IRB6-S2, em relação as atividades indicadas na Figura 3.8 da metodologia apresentada no Capítulo 3.

A configuração do sistema é feita com o respaldo da comunidade de desenvolvedores de LinuxCNC por meio da documentação existente, fórum online, exemplos disponíveis na biblioteca da plataforma e a interação do usuário com as opções do controlador.

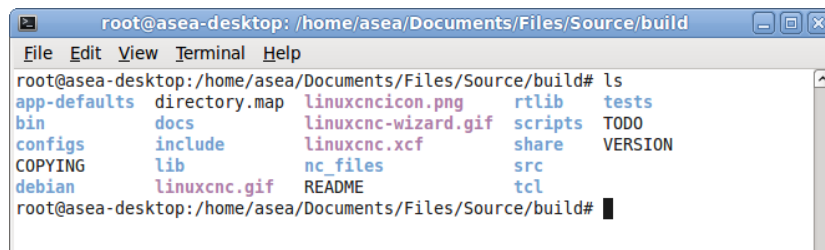
4.4.2.1 Instalação do Controlador LinuxCNC

Foi instalado em um computador pessoal os requisitos de hardware recomendados para a operação de LinuxCNC. Foi usada a versão 2.5.4 do controlador, junto com o sistema operacional Ubuntu. A partir do site da comunidade de desenvolvedores do controlador foi baixado o CD de instalação do sistema operacional, assim como LinuxCNC. O site da comunidade com as novidades, desenvolvimentos e informações do controlador podem ser consultadas em <http://www.linuxcnc.org/>.

4.4.2.2 Compilar o Arquivo com as Equações Cinemáticas do Manipulador

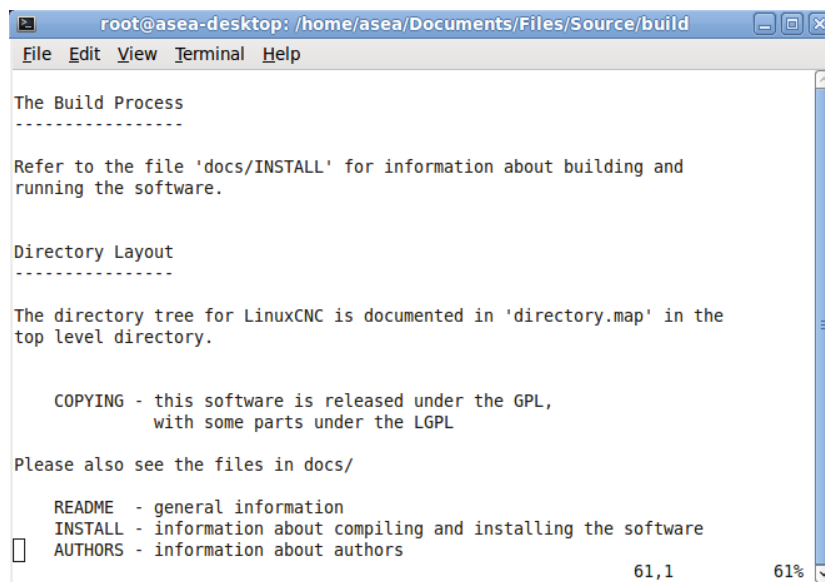
Foi necessário gerar um arquivo em linguagem C para que o compilador do controlador LinuxCNC conseguisse criar um arquivo executável exclusivo com as equações cinemáticas do manipulador ASEA IRB6-S2 ou qualquer outra configuração de cinemáticas não triviais. Os arquivos compilados com a cinemática direta e inversa estão disponíveis no Apêndice J. A seguir será descrito o roteiro para a inclusão dos arquivos na fonte do controlador a partir da interface de usuário do sistema operacional:

1. Baixar a fonte da distribuição LinuxCNC: é necessário identificar a versão do controlador e o tipo de sistema operacional Linux compatível com capacidade em tempo real. Para o projeto foi usada a versão 2.5.4 do LinuxCNC recomendada para trabalhar com o sistema operacional Ubuntu. As fontes das diferentes versões do controlador para ser baixadas estão disponíveis em <http://linuxcnc.org/dists/precise/linuxcnc2.5/source/>.
2. Verificar o conteúdo do arquivo: com a pasta da fonte da versão do LinuxCNC é necessário identificar o arquivo com as instruções para usar o pacote fonte, chamado "README". Na Figura 4.25 é apresentado o conteúdo do pacote fonte baixado.
3. O arquivo aberto tem a informação geral da configuração e o conteúdo do pacote fonte. Neste caso foi usada a parte que corresponde ao processo de instalação da plataforma no momento de modificar o conteúdo. É apresentada na Figura 4.26 o diretório com a informação associada a instalação da plataforma.



```
root@asea-desktop: /home/asea/Documents/Files/Source/build
File Edit View Terminal Help
root@asea-desktop: /home/asea/Documents/Files/Source/build# ls
app-defaults  directory.map  linuxcncicon.png  rtlib  tests
bin           docs          linuxcnc-wizard.gif  scripts  TODO
configs      include      linuxcnc.xcf      share  VERSION
COPYING      lib         nc_files          src
debian       linuxcnc.gif  README           tcl
root@asea-desktop: /home/asea/Documents/Files/Source/build#
```

Figura 4.25: Arquivos disponíveis na pasta do pacote fonte



```
root@asea-desktop: /home/asea/Documents/Files/Source/build
File Edit View Terminal Help

The Build Process
-----

Refer to the file 'docs/INSTALL' for information about building and
running the software.

Directory Layout
-----

The directory tree for LinuxCNC is documented in 'directory.map' in the
top level directory.

COPYING - this software is released under the GPL,
         with some parts under the LGPL

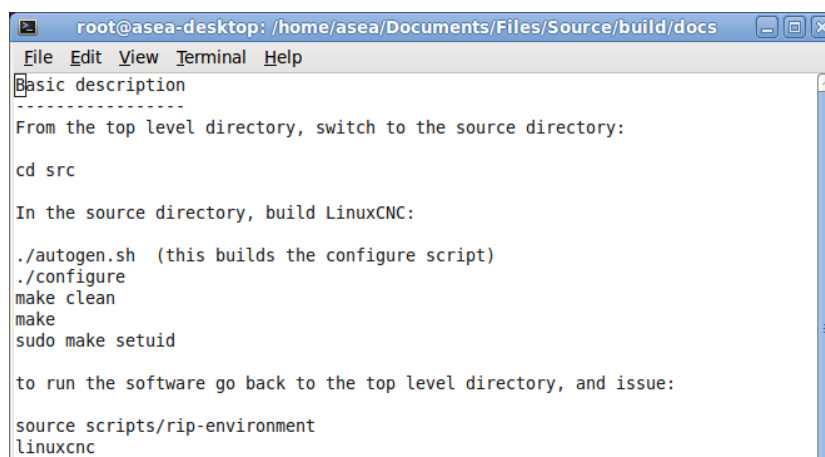
Please also see the files in docs/

README - general information
INSTALL - information about compiling and installing the software
AUTHORS - information about authors

61,1 61%
```

Figura 4.26: Informação geral do pacote fonte da plataforma

4. Após ler o arquivo é necessário navegar pelo terminal Linux para abrir o arquivo com o detalhe da instalação do pacote fonte com o nome de "INSTALL". Na Figura 4.27 são apresentadas as instruções para compilar e instalar a plataforma.



```
root@asea-desktop: /home/asea/Documents/Files/Source/build/docs
File Edit View Terminal Help
Basic description
-----
From the top level directory, switch to the source directory:

cd src

In the source directory, build LinuxCNC:

./autogen.sh (this builds the configure script)
./configure
make clean
make
sudo make setuid

to run the software go back to the top level directory, and issue:

source scripts/rip-environment
linuxcnc
```

Figura 4.27: Instruções de compilação e instalação da plataforma

- Identificar o diretório de compilação: na localização do pacote fonte existe uma pasta onde é possível incluir e compilar a versão do controlador com uma nova configuração da cinemática. Na Figura 4.28 é apresentado os arquivos do diretório principal para a inclusão da nova cinemática.

```

root@asea-desktop: /home/asea/Documents/Files/Source/build/src# ls
autogen.sh      configure.in    Makefile.inc.in  objects
autom4te.cache  depends        Makefile.modinc  po
CodingStyle     doxconfig      Makefile.modinc.in  rtapi
config.h        emc            modsilent.py      Submakefile.skel
config.h.in     hal            module_helper     tests
config.log      libnml         modules.order
config.status   Makefile       Module.symvers
configure       Makefile.inc   move-if-change

```

Figura 4.28: Arquivos disponíveis no diretório de compilação

- Modificar o arquivo "Makefile": possibilita incluir o nome do arquivo principal e associados da nova configuração do manipulador, permitindo que o compilador gere o executável usado na aplicação final para calcular os valores cinemáticos do manipulador. O nome definido para o robô ASEA IRB6-S2 é "irb", logo no arquivo "Makefile" será incluído o nome nas partes definidas pelo controlador. Na Figura 4.29 é apresentado como foi colocado o nome da configuração para o robô ASEA IRB6-S2, no arquivo da compilação.

```

# Edição primeira parte do arquivo "Makefile"
HEADERS := \
    config.h \
    emc/ini/emcIniFile.hh \
    emc/ini/iniaxis.hh \
    emc/ini/initool.hh \
    emc/ini/initraj.hh \
    emc/kinematics/cubic.h \
    emc/kinematics/kinematics.h \
    emc/kinematics/genhexkins.h \
    emc/kinematics/genserkins.h \
    emc/kinematics/pumakins.h \
    emc/kinematics/irb.h \

# Edição segunda parte do arquivo "Makefile"

bj-m += pumakins.o
pumakins-objs := emc/kinematics/pumakins.o
pumakins-objs += libnml/posemath/_posemath.o
pumakins-objs += libnml/posemath/sincos.o $(MATHSTUB)

obj-m += irb.o
irb-objs := emc/kinematics/irb.o
irb-objs += libnml/posemath/_posemath.o
irb-objs += libnml/posemath/sincos.o $(MATHSTUB)

# Edição segunda parte do arquivo "Makefile"
# Rules to make .o (object) files

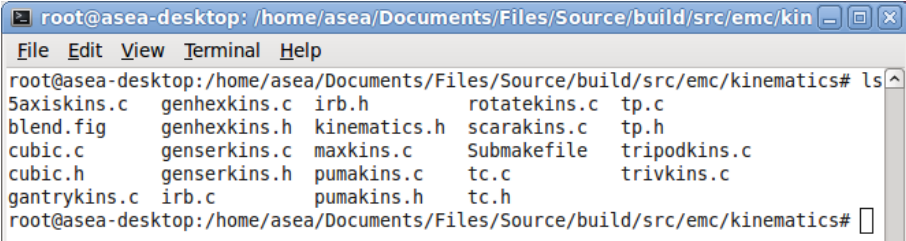
../rtlib/pumakins$(MODULE_EXT): $(addprefix objects/rt, $(pumakins-objs))
../rtlib/irb$(MODULE_EXT): $(addprefix objects/rt, $(irb-objs))

# Fim edição arquivo "Makefile"

```

Figura 4.29: Edição arquivo "Makefile" para a nova configuração

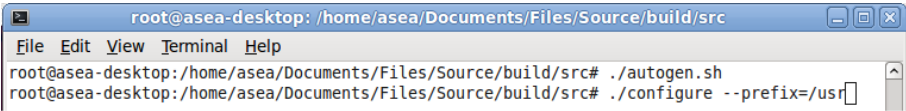
7. Configurar o arquivo com a cinemática: o jeito mais fácil para incluir no pacote fonte do controlador as equações cinemáticas da nova configuração é copiar e modificar um arquivo existente na pasta chamada "*kinematics*", com as diferentes equações homogêneas do manipulador ASEA IRB6-S2. Existem dois arquivos com a informação da nova configuração do manipulador: um arquivo com as funções da cinemática inversa e direta, baseadas nos módulos em tempo real do controlador com extensão *.c (LINUXCNC.ORG, 2015c), assim como outro arquivo com definições de variáveis adicionais para trabalhar com as equações cinemáticas conhecido como arquivo cabeçalhos com extensão *.h. Este tipo de arquivo tem os valores das variáveis da notação DH, no caso do manipulador ASEA IRB6-S2. Na Figura 4.30 são apresentados os arquivos associados a cinemáticas do robô ASEA IRB6-S2 na pasta das configurações de máquinas usadas na plataforma LinuxCNC.



```
root@asea-desktop: /home/asea/Documents/Files/Source/build/src/emc/kin
File Edit View Terminal Help
root@asea-desktop:/home/asea/Documents/Files/Source/build/src/emc/kinematics# ls
5axiskins.c  genhexkins.c  irb.h          rotatekins.c  tp.c
blend.fig    genhexkins.h  kinematics.h  scarakins.c  tp.h
cubic.c      genserkins.c  maxkins.c     Submakefile   tripodkins.c
cubic.h      genserkins.h  pumakins.c   tc.c          trivkins.c
gantrykins.c irb.c         pumakins.h   tc.h
root@asea-desktop:/home/asea/Documents/Files/Source/build/src/emc/kinematics#
```

Figura 4.30: Configuração do arquivo com a cinemática do manipulador

8. Configurações iniciais para compilação: foram criados dois arquivos com a informação do modelo matemático para o robô definidos como "irb.c" e "irb.h" os quais podem ser consultados no Apêndice J. Após identificar os arquivos é necessário compilar o pacote fonte com a nova configuração para validar se existem erros e para instalar posteriormente a nova configuração no sistema. Na Figura 4.31 apresentam-se os comandos iniciais para configurar a compilação e instalação da plataforma com a configuração da nova cinemática.



```
root@asea-desktop: /home/asea/Documents/Files/Source/build/src
File Edit View Terminal Help
root@asea-desktop:/home/asea/Documents/Files/Source/build/src# ./autogen.sh
root@asea-desktop:/home/asea/Documents/Files/Source/build/src# ./configure --prefix=/usr
```

Figura 4.31: Comandos iniciais para compilação da plataforma

9. Compilação e instalação da plataforma: o compilador informa que está configurado para continuar com o procedimento para instalar o LinuxCNC com as modificações feitas. Na Figura 4.32 é apresentada a resposta do compilador antes de avançar com a instalação do sistema.

4.4.3 Gerar e Configurar Novo Projeto de Manipulador em LinuxCNC

Com a cinemática do manipulador ASEA IRB6-S2 introduzida no controlador foi gerado um novo projeto para associar os arquivos executáveis gerados depois da compilação do pacote fonte do LinuxCNC junto com os elementos de controle do manipulador. Continuando desta forma com as atividades apresentadas na Figura 3.8, associadas a metodologia apresentada no Capítulo 3.


```

#####
#                               LinuxCNC - Enhanced Machine Controller                               #
#####
#                               #
# LinuxCNC is a software system for computer control of machine #
# tools such as milling machines. LinuxCNC is released under the #
# GPL. Check out http://www.linuxcnc.org/ for more details. #
# #
# #
# It seems that ./configure completed successfully. #
# This means that RT is properly installed #
# If things don't work check config.log for errors & warnings #
# #
# warning: If you already have an installed linuxcnc, this will #
# replace an existing installation. If you have installed #
# a linuxcnc package, this will damage the package. #
# hint: To test a self-built version of linuxcnc without damaging #
# the package version, don't specify a --prefix #
# #
# Next compile by typing #
# make #
# then install it by typing #
# sudo make install #
# #
# To run the software type #
# linuxcnc #
# #
#####

```

Figura 4.32: Compilar e instalar a plataforma com a nova configuração do manipulador

O projeto da configuração do manipulador ASEA IRB6-S2 é armazenado em uma pasta com arquivos da configuração do manipulador. Desta forma o projeto é parte da biblioteca de exemplos do controlador, permitindo usar os recursos disponíveis da plataforma. Toda a informação do projeto é copiada junto com os outros programas exemplos no pacote fonte, para posteriormente compilar e instalar o controlador como o procedimento seguido na Subsubseção 4.4.2.2.

O controlador LinuxCNC é usado para o controle de máquinas CNC, com cinemáticas triviais. Portanto, os assistentes de configuração desenvolvidos até hoje foram projetados para configurar uma máquina CNC e não para manipuladores industriais. Como consequência, foi criado um projeto com um dos assistentes da plataforma chamado "*Stepconf*" para gerar os arquivos de configuração com a informação básica para três eixos (GUTIERREZ, 2013), editando os arquivos e adaptando os parâmetros necessários ao manipulador ASEA IRB6-S2 para cinco graus de liberdade.

A seguir é apresentado o roteiro básico usado para gerar o projeto do manipulador com os arquivos de configuração necessários para o controle do sistema por meio do assistente de configuração para máquina CNC, chamado "*Stepconf*", o mesmo é instalado junto ao controlador LinuxCNC. Todas as configurações criadas através deste assistente são situadas no diretório "*linuxcnc/config*" do sistema operacional Linux.

1. A tela inicial do assistente é apresentada na Figura 4.33. "*Stepconf*" faz a configuração do controle de máquinas CNC ou robôs por meio da placa paralela padrão, com sinais tipo *step* e *direction*.
2. Na seguinte tela do assistente, Figura 4.34, é possível criar uma nova configuração de máquina (projeto) ou editar o já existente.



Figura 4.33: Tela inicial "Stepconf"

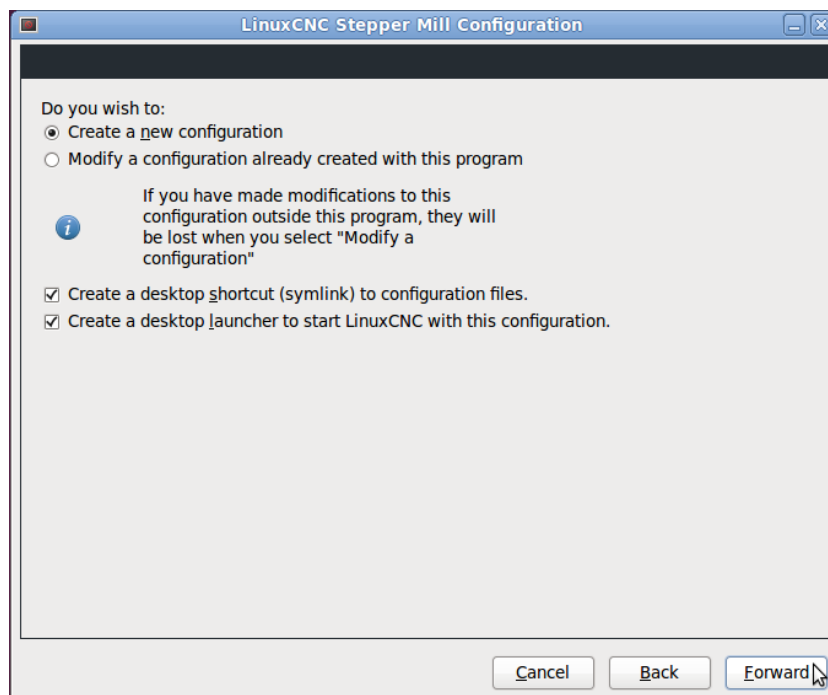


Figura 4.34: Escolha da configuração do projeto

3. Na seguinte tela do assistente da configuração para uma nova máquina, são necessários dados básicos de alguns componentes da parte hardware do subsistema de controle. Nesta seção da configuração inicia-se a integração da plataforma com os elementos escolhidos na Seção 4.3. A tela apresentada na Figura 4.35 está dividida em quatro seções: a primeira identificando o projeto, as unidades, que

para o robô é em milímetros, e a configuração dos eixos. O número máximo de eixos permitido pelo assistente é de quatro (X, Y, Z e A), ou quatro graus de liberdade. Para a configuração de manipuladores esta informação é modificada diretamente nos arquivos de configuração gerados pelo assistente, o que evidencia uma limitação que deveria ser desenvolvida facilitando a configuração dos nove eixos suportados pelo LinuxCNC através de um assistente.

The screenshot shows a window titled "LinuxCNC Stepper Mill Configuration" with a sub-header "Basic machine information". The form contains the following fields and settings:

- Machine Name: irb
- Configuration directory: ~/linuxcnc/configs/irb
- Axis configuration: XYZ
- Machine units: Millimeter
- Driver characteristics: (Multiply by 1000 for times specified in μ s or microseconds) Additional signal conditioning or isolation such as optocouplers and RC filters can impose timing constraints of their own, in addition to those of the driver.
- Driver type: Other
- Driver Timing Settings:
 - Step Time: 1000 ns
 - Step Space: 2500 ns
 - Direction Hold: 200 ns
 - Direction Setup: 200 ns
- Parallel Port Settings:
 - First Parport Base Address: 0x378 Out
 - Second Parport Address: Enter Address Out
 - Third Parport Address: Enter Address Out
- Base Period Maximum Jitter: 20000 ns Min Base Period: 28500 ns
- Onscreen prompt for tool change Test Base Period Jitter Max step rate: 35087 Hz

Buttons at the bottom: Cancel, Back, Forward.

Figura 4.35: Informação básica da máquina ou projeto

A seguinte seção, da tela dos dados básicos da máquina, apresenta as características da sincronização dos drives de controle dos motores com LinuxCNC. Existem drives predefinidos no assistente junto com a informação das variáveis de sincronização. No caso da configuração para o manipulador é necessário editar estas informações com relação ao tipo de drive. Estes dados de sincronização estão disponíveis no manual de cada fornecedor ou podem ser consultados em <http://wiki.linuxcnc.org/cgi-bin/wiki.pl?Stepper_Drive_Timing>. O drive selecionado para o projeto é *Gecko G320X*.

Deve ser informado para o assistente o endereço em memória da porta paralela, tendo um valor padrão 0x378 em sistema numérico hexadecimal. Finalmente é necessário informar ao assistente uma variável de tempo chamada "*Latency*", definida como o tempo que precisa o computador para parar as tarefas que esta executando para possibilitar a resposta de um requerimento externo, que para o caso do LinuxCNC é usado como a oscilação periódica para a sincronização dos sinais tipo *step*. (LINUXCNC.ORG, 2015a).

Na Figura 4.36 observa-se o teste de "latência" próprio do controlador para conhecer o período de tempo máximo usado no projeto, neste caso é de 19.407 ns. Enquanto o teste é executado, é necessário usar a maior quantidade de recursos do computador para avaliar a pior condição do valor de latência. Fatores como a velocidade da CPU, tipo de *Motherboard*, placa de vídeo, portos USB, dentre outros, determinam a resposta do sistema. Com um maior número de recursos *hardware* usados (processos executados) o valor da latência incrementa (LINUXCNC.ORG, 2015a).

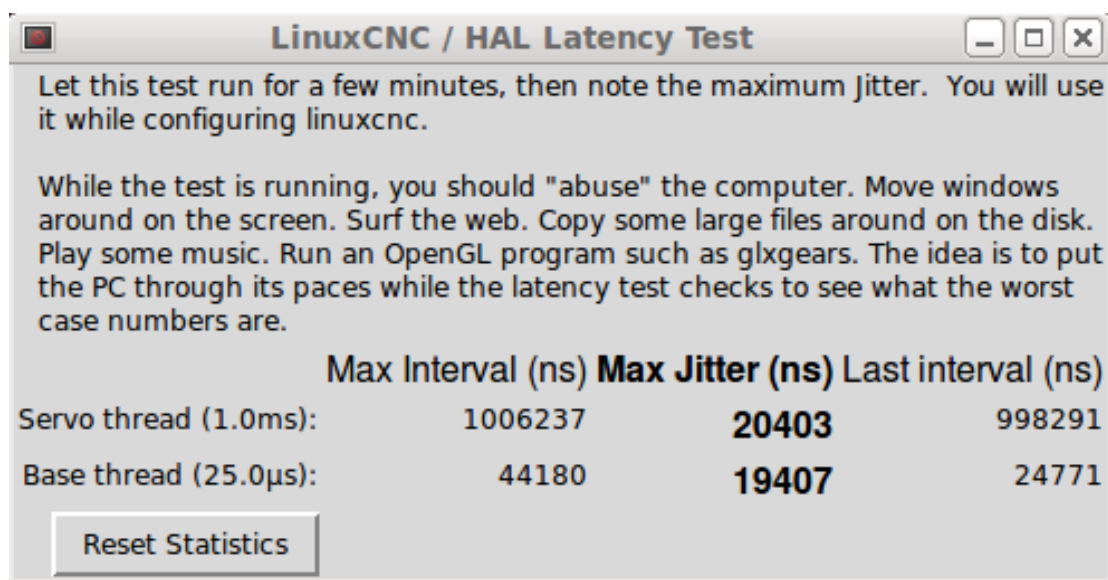


Figura 4.36: Teste de latência no LinuxCNC

Para o projeto foi necessário usar um computador com porta paralela, além de garantir a operação do controle do manipulador por meio das especificações de hardware recomendadas pelo LinuxCNC em sua versão 2.5.4 na Tabela 4.13.

Tabela 4.13: Especificação do computador para LinuxCNC

Componente	Marca	Característica
<i>Motherboard</i>	Itautec	ST4160
Processador	Intel	Core 2 Duo
RAM	SMART	4 GB - DDR2
Disco rígido	Western Digital	500 GB - SATA
Placa de vídeo	NVIDIA	512 MB - DDR2

4. Configuração entradas/saídas: o assistente "*Stepconf*" permite ajustar os sinais processados através da placa paralela, para cada um dos graus de liberdade do manipulador. Neste caso foram configurados quatro saídas, por causa da limitação do assistente para aceitar outro tipo de máquina com um maior número de eixos para controlar. Na Figura 4.37 é apresentada a configuração das entradas e saídas para o projeto do manipulador ASEA IRB6-S2. As outras saídas e entradas requeridas pelo projeto para o controle de cinco graus de liberdade são configuradas diretamente nos arquivos de configuração gerados desde o assistente.

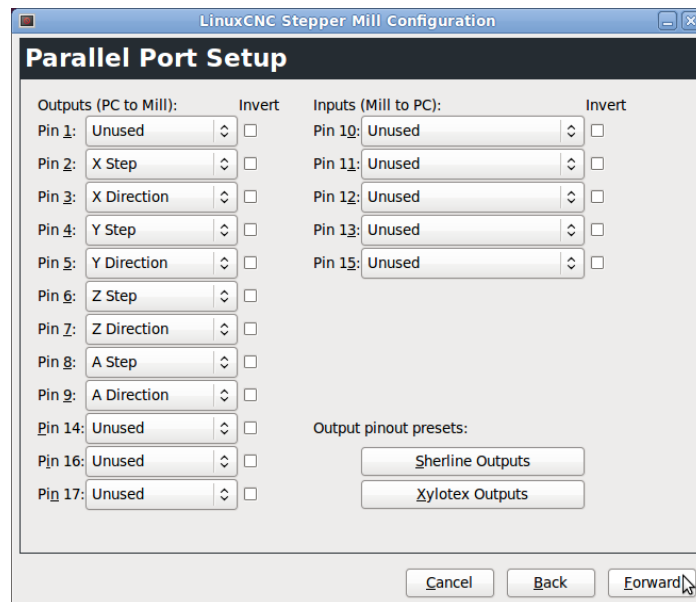


Figura 4.37: Configuração de entradas e saídas

5. Ajuste dos parâmetros dos eixos: cada um dos graus de liberdade do manipulador é configurado a partir do assistente, levando em consideração que os dados são baseados em máquinas CNC, onde cada eixo tem um deslocamento linear o que implica modificar fora do assistente a característica rotacional das juntas, assim como a inclusão de um quinto eixo. Desta forma, para o manipulador ASEA IRB6-S2 o ajuste dinâmico de cada eixo é ajustado com a operação do robô, garantindo a calibração das cinco juntas rotacionais associadas aos cinco graus de liberdade do manipulador. Na Figura 4.38 são apresentados os valores básicos para configurar cada um dos eixos desde o assistente.

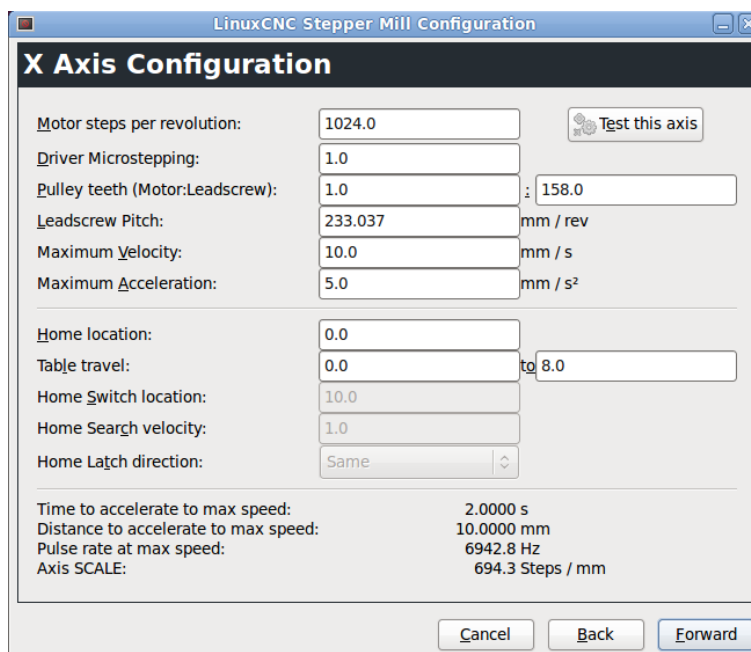


Figura 4.38: Configuração de entradas e saídas

O valor de passo do motor por cada revolução (*Motor steps per revolution*) depende da configuração do *encoder* escolhido na Subseção 4.3.1. A resolução deste dispositivo digital pode ser ajustado em relação as indicações do *drive* usado. É apresentado na Eq. 4.2 o valor mínimo da resolução do *encoder* recomendado pelo fornecedor do *drive* escolhido e o valor definido segundo as características do *encoder* (GECKODRIVE, 2011).

$$Velocidade_{max-servomotor} = 4700rpm \quad (4.2a)$$

$$Velocidade_{80\%} = 3760rpm = 63rps \quad (4.2b)$$

$$Frequência\ de\ passo_{max} = 45kHz \quad (4.2c)$$

$$Pulsos\ por\ revolução = \frac{45kHz}{63rps} \approx 180pulsos/rev \quad (4.2d)$$

$$Resolução\ encoder = 256ppr \quad (4.2e)$$

6. Fazendo referência às instruções do fabricante do *drive* para determinar a resolução ótima do *encoder*, foram ajustados todos os eixos do manipulador, quatro através do assistente e o quinto eixo diretamente nos arquivos de configuração. O valor calculado anteriormente é multiplicado por quatro, por causa da característica de quadratura apresentada por este tipo de *encoder* (CUI INC, 2014). Desta forma, o processo é finalizado permitindo aplicar os ajustes incluídos no assistente, gerando o projeto e os arquivos associados para o controle do manipulador.
7. É necessário copiar a pasta gerada pelo assistente no diretório do pacote fonte do LinuxCNC, modificar os arquivos de configuração para ajustar o projeto ao manipulador ASEA IRB6-S2. Na Figura 4.39 é apresentada a pasta do diretório do pacote fonte, antes de modificar os arquivos de configuração, compilação e instalação com o novo projeto gerado.

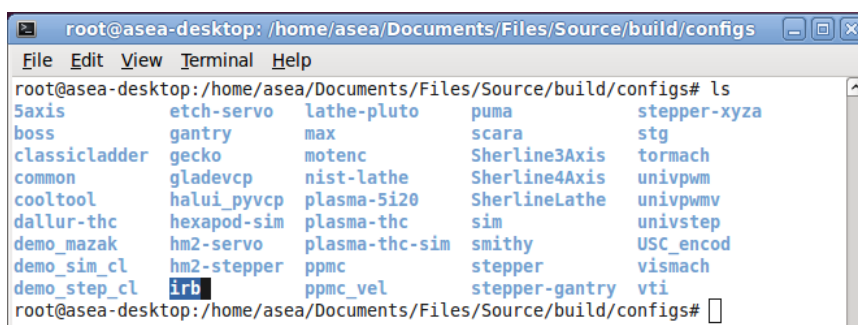


Figura 4.39: Projeto no diretório das configurações padrões

8. Após as alterações nos arquivos de configuração do projeto o controlador é compilado e instalado repetindo o procedimento feito em Subsubseção 4.4.2.2. Após a instalação, esta garantida a integração do projeto com as dependências usadas pelo LinuxCNC para o controle do manipulador, além de fazer parte da biblioteca de exemplos disponíveis na plataforma.

O projeto gerado pelo LinuxCNC é configurado através de arquivos de texto. Estes arquivos podem

ser lidos e editados em qualquer aplicativo para edição, compatíveis com a maioria de distribuições Linux. Os arquivos são usados pelo controlador no momento de executar o projeto, assim quando a máquina ou manipulador está em operação. Desta forma, as modificações feitas nos arquivos de configuração para o manipulador ASEA IRB6-S2 tem cópias de respaldo no caso de apresentar falhas de execução do LinuxCNC com a nova configuração (LINUXCNC.ORG, 2015c).

4.4.4 Configuração Arquivo INI

O arquivo disponível em cada um dos projetos do controlador através do LinuxCNC modifica e estabelece os ajustes padrões das máquinas criadas pelo assistente de configuração. A informação contida neste arquivo é compilada diretamente pelo código do controlador, além de conter uma seção usada pela camada de simulação hardware, conhecida como HAL.

Foi necessário modificar o arquivo INI gerado pelo assistente "*Stepconf*" para incluir as características da configuração do manipulador ASEA IRB6-S2, por meio das seções definidas para um melhor entendimento do usuário com as características de LinuxCNC.

EMC: nesta seção é apresentada a informação geral do projeto, como a versão no momento de modificar o arquivo permitindo ter um histórico das mudanças feitas. Da mesma forma é possível personalizar o nome da máquina e a opção de gerar mensagens do desenvolvedor no momento de executar a plataforma. Na Figura 4.40 se apresenta a seção do arquivo INI para o manipulador.

```
#####  
# General section  
#####  
[EMC]  
  
#- Version of this INI file  
VERSION =          $Revision$  
  
#+ Name of machine, for use with display, etc.  
MACHINE =          LinuxCNC-ASEA-IRB6/S2  
  
#+ Debug level, 0 means no messages.  
DEBUG = 0
```

Figura 4.40: Valores da seção [EMC] do arquivo INI

DISPLAY: a plataforma LinuxCNC possibilita escolher opções de interfaces de usuário para o controle das máquinas. No projeto para o manipulador ASEA IRB6-S2 foi usada a interface padrão do controlador, assim como os dados gerados desde o assistente para esta seção.

RS274NGC: através de um arquivo NC, o interpretador da plataforma converte as instruções da norma RS274 para funções padrões da máquina (KRAMER; PROCTOR; MESSINA, 2000). No caso do manipulador a maioria das funções usadas estão associadas com a movimentação do efetuador do robô.

EMCMOT: são valores usados pelo controlador de movimento em tempo real. Para a configuração do manipulador não foram modificados os valores gerados pelo assistente de configuração.

HAL: nesta seção são relacionados os arquivos de configuração da camada HAL para serem executados ao iniciar o controlador com o projeto do manipulador. Na Figura 4.41 são apresentados os arquivos de configuração usados.

```
#####  
# Hardware Abstraction Layer section  
#####  
[HAL]  
  
# list of hal config files to run through halcmd  
# files are executed in the order in which they appear  
  
# Simulation halfile  
HALFILE =                irb_sim.hal  
# Pendant halfile  
HALFILE =                vc-p4s.hal  
# Widget and indicator halfile  
POSTGUI_HALFILE =       irb_postgui.hal
```

Figura 4.41: Valores da seção [HAL] do arquivo INI

TRAJ: ajustes adicionais dos parâmetros da máquina, que no caso do manipulador são indicados nesta seção. Valores como o número máximo de juntas, assim como a definição das unidades lineares e angulares. Na Figura 4.42 se observa os valores definidos associados ao projeto de robô ASEA IRB6-S2.

```
[TRAJ]  
AXES =                    5  
COORDINATES =            X Y Z A B  
HOME =                   0 0 0 0 0  
LINEAR_UNITS =           mm  
ANGULAR_UNITS =          deg
```

Figura 4.42: Valores da seção [TRAJ] do arquivo INI

AXES nesta seção se configuram os cinco eixos do manipulador individualmente. Variáveis como o tipo de junta, a velocidade e a aceleração máxima, a relação entre o deslocamento angular do motor e a junta são modificados nesta parte do arquivo INI. Na Figura 4.43 são apresentados os valores iniciais para cada um dos cinco graus de liberdade, ajustados no momento da operação do manipulador.

Com os valores relacionados ao arquivo INI, foi possível associar a configuração básica do manipulador em relação aos parâmetros gerados pelo assistente para máquinas CNC com o máximo de quatro graus de liberdade.

```

[AXIS_0]
TYPE = ANGULAR
HOME = 0.0
MAX_VELOCITY = 5.00
MAX_ACCELERATION = 5.00
STEPGEN_MAXACCEL = 6.25
SCALE = 450
MIN_LIMIT = -360.0
MAX_LIMIT = 360.0
FERROR = 1.000
MIN_FERROR = 0.250
HOME_OFFSET = 0.0
HOME_SEARCH_VEL = 0.0
HOME_LATCH_VEL = 0.0
HOME_USE_INDEX = NO
HOME_IGNORE_LIMITS = NO
HOME_SEQUENCE = 0

```

Figura 4.43: Valores da seção [AXES] do arquivo INI

4.4.5 Configuração Arquivo HAL

O arquivo tem a capacidade de fazer a interligação via software dos componentes físicos e digitais usados para controlar a movimentação do manipulador. Com ajuda do assistente "*Stepconf*" foi gerado o arquivo para a configuração da camada HAL do projeto associado ao manipulador ASEA IRB6-S2. Foi necessário editar e adicionar parâmetros no arquivo HAL, permitindo a integração do módulo da cinemática específica do robô, os componentes hardware escolhidos e a configuração feita no arquivo INI. A seguir são descritas as modificações feitas para a customização da camada HAL no projeto de *retrofitting*, detalhando a atividade associada na Figura 3.8 da metodologia descrita no Capítulo 3.

- A parte inicial do arquivo contém os módulos em tempo real usados pelo LinuxCNC para controlar os parâmetros cinemáticos do manipulador. Na Figura 4.44 apresentam-se as linhas de código carregando os módulos necessários. É carregado o módulo "*irb*" gerado na compilação do arquivo com as equações homogêneas do manipulador, assim como o módulo *motion* responsável por controlar cada um dos sinais de controle dos *drives* motores. Os valores são obtidos do arquivo INI.

```

#####
# load RT modules
#####
loadrt irb
loadrt [EMCMOT]EMCMOT
base_period_nsec=[EMCMOT]BASE_PERIOD
servo_period_nsec=[EMCMOT]SERVO_PERIOD
traj_period_nsec=[EMCMOT]TRAJ_PERIOD
num_joints=[TRAJ]AXES

```

Figura 4.44: Módulos RT do arquivo HAL

- Foi necessário configurar no arquivo HAL as saídas físicas que controlam cada uma das juntas do manipulador, neste caso os sinais enviados através da interface paralela aos *drives* de controle. Foi adicionado uma saída adicional as geradas pelo assistente "*Stepconf*", por causa da falta de suporte para máquinas que tenham mais de quatro eixos. Na Figura 4.45 são apresentadas as configurações para a geração de sinais tipo *step* e *dir*.

```
#####
# output pins
#####

# joint 0
net xdir => parport.0.pin-03-out
net xstep => parport.0.pin-02-out
setp parport.0.pin-02-out-reset 1

# joint 1
net ydir => parport.0.pin-05-out
net ystep => parport.0.pin-04-out
setp parport.0.pin-04-out-reset 1

# joint 2
net zdir => parport.0.pin-07-out
net zstep => parport.0.pin-06-out
setp parport.0.pin-06-out-reset 1

#joint 3
net adir => parport.0.pin-09-out
net astep => parport.0.pin-08-out
setp parport.0.pin-08-out-reset 1

# joint 4
net bdir => parport.0.pin-17-out
net bstep => parport.0.pin-14-out
setp parport.0.pin-14-out-reset 1
#####
```

Figura 4.45: Configuração saídas da placa paralela no arquivo HAL

- Assim como foram configuradas as saídas físicas da interface paralela para o controle das juntas do manipulador, foi ajustado o arquivo para configurar a geração de sinais a partir da camada HAL do quinto grau de liberdade. Na Figura 4.46 são apresentados os parâmetros do eixo adicional.

```
#####
# AXIS 4 - Signals configuration
#####

setp stepgen.4.position-scale [AXIS_4]SCALE
setp stepgen.4.steplen 1
setp stepgen.4.stepspace 0
setp stepgen.4.dirhold 15200
setp stepgen.4.dirsetup 15200
setp stepgen.4.maxaccel [AXIS_4]STEPGEN_MAXACCEL
net bpos-cmd axis.4.motor-pos-cmd => stepgen.4.position-cmd
net bpos-fb stepgen.4.position-fb => axis.4.motor-pos-fb
net bstep <= stepgen.4.step
net bdir <= stepgen.4.dir
net benable axis.4.amp-enable-out => stepgen.4.enable
```

Figura 4.46: Parâmetros para geração de sinais dos eixos via HAL

4.4.6 Modelo Virtual do Robô ASEA IRB6-S2 para Simulação em LinuxCNC

Na plataforma LinuxCNC é possível gerar um modelo virtual de qualquer máquina, permitindo visualizar a simulação dos movimentos comandados a partir da interface de usuário em paralelo a operação da máquina física. Neste caso foram modeladas as juntas principais do robô ASEA IRB6-S2 em um software CAD chamado FreeCAD, para simular a operação junta com a interface de usuário padrão *AXIS-GUI*, da plataforma. A simulação foi possível com ajuda de um pacote baseado em funções *OpenGL* e *Python* chamado *Vismach*. O simulador permite gerar variáveis compatíveis com a camada HAL para cada uma das juntas ou grau de liberdade do manipulador. Garantindo desta forma a execução simultânea com as variáveis cinemáticas do robô real e o modelo virtual. A seguir será detalhado os passos básicos para gerar e integrar a simulação com o projeto do manipulador ASEA IRB6-S2, como parte das atividades apresentadas na Figura 3.8, associadas à metodologia apresentada no Capítulo 3

- Cada umas das cinco juntas foram modeladas em um software CAD com formato compatível com o simulador *Vismach*. As peças são importadas mediante um *script* feito em *python* com as configurações necessárias para gerar a simulação. O arquivo está disponível no Apêndice K.
- A forma mais simples de gerar o *script* no modelo virtual é modificar um arquivo existente na instalação do LinuxCNC, neste caso `"/usr/bin"` é o diretório padrão para os *scripts* em linguagem *python*. O nome do novo arquivo que contém a configuração do modelo virtual é `"irbgui"`, o qual tem que ser carregado no arquivo HAL após finalizar a edição do *script*.
- Foram criadas as variáveis compatíveis com a camada HAL, para permitir a integração com o controlador LinuxCNC. Cada uma das juntas deverá ter uma variável que será carregada no arquivo HAL para associar o modelo virtual com os valores das variáveis cinemáticas do manipulador. Na Figura 4.47 se apresentam as definições da simulação e das juntas no *script*.

```
c = hal.component("irbgui")
c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint1", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint2", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint3", hal.HAL_FLOAT, hal.HAL_IN)
c.newpin("joint4", hal.HAL_FLOAT, hal.HAL_IN)
c.ready()
```

Figura 4.47: Definição das variáveis HAL

- O simulador *vismach* tem uma lógica própria para importar e fixar cada uma das peças em um só conjunto. É fundamental iniciar a importação da última peça do conjunto, neste caso a quinta junta do manipulador. Posteriormente é importada a quarta junta, fazendo um conjunto entre as duas peças. Desta forma, até a junta da base devesse ter o procedimento para construir o modelo do manipulador. Todas as vezes que a peça é importada define-se o eixo de rotação em relação as coordenadas de origem do sistema. Para maior informação da configuração do simulador pode-se consultar em <http://linuxcnc.org/docs/html/gui/vismach.html>.

- Na Figura 4.48 é apresentado o modelo virtual do manipulador ASEA IRB6-S2 após a execução do *script*.

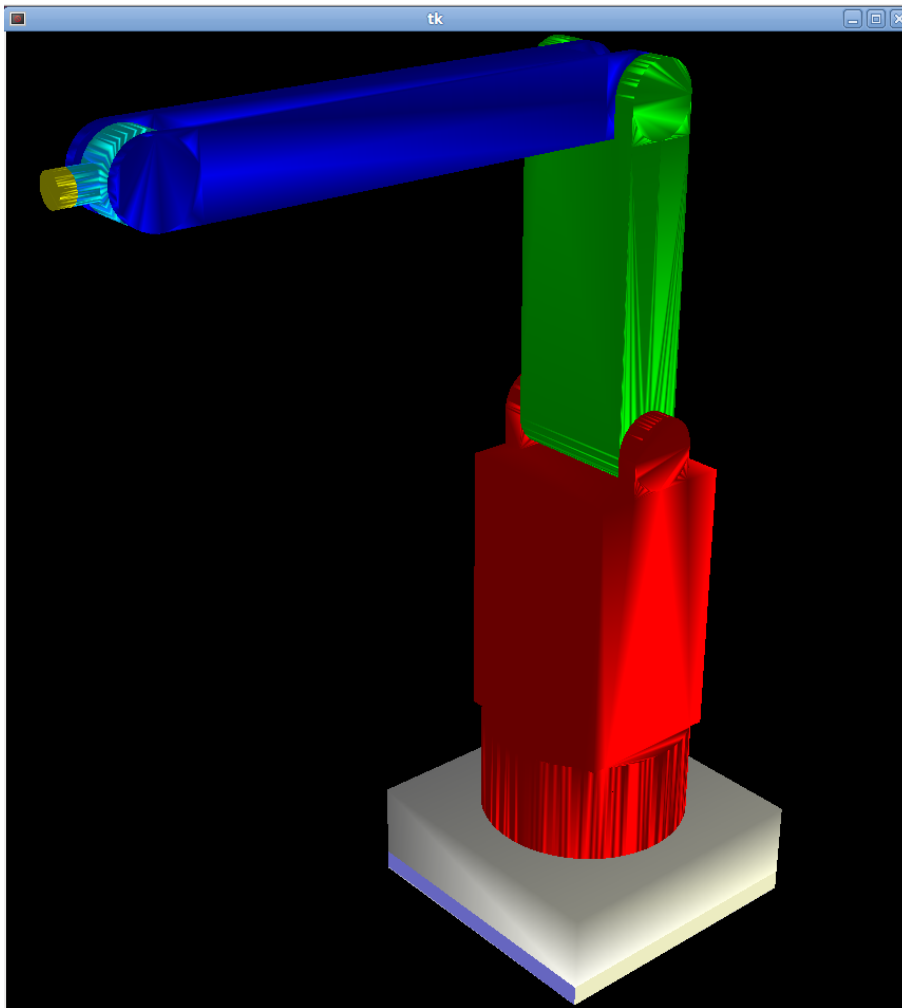


Figura 4.48: Modelo virtual do manipulador ASEA IRB6-S2

- Para possibilitar o controle desde a execução de LinuxCNC das juntas do modelo virtual do manipulador é necessário editar o arquivo HAL, relacionando aos componentes gerados no *script*. Na Figura 4.49 se apresentam as instruções para integrar o modelo virtual na camada HAL da nova configuração gerado para o manipulador ASEA IRB6-S2.

#	signal_name	signal_source	Destination
net	J0scaled	xpos-fb	irbgui.joint0
net	J1scaled	ypos-fb	irbgui.joint1
net	J2scaled	zpos-fb	irbgui.joint2
net	J3scaled	apos-fb	irbgui.joint3
net	J4scaled	bpos-fb	irbgui.joint4

Figura 4.49: Instruções no arquivo HAL para o controle da simulação

4.5 Montagem dos Novos Componentes no Manipulador ASEA IRB6-S2

Prosseguindo com a técnica do *retrofitting* associada às atividades do Capítulo 3, especificamente na Figura 3.9, é apresentado nesta seção o procedimento para a montagem dos componentes adquiridos e fabricados na estrutura mecânica do manipulador, assim como a adequação do gabinete de controle. Os subsistemas que integram o controle da movimentação das juntas do robô estão dispostos em duas partes: os componentes dos subsistemas de alimentação e alguns associados ao subsistema de controle foram situados no gabinete de controle original do manipulador; enquanto o restante dos elementos do subsistema de controle e locomoção estão localizados na estrutura mecânica do manipulador.

4.5.1 Montagem dos Conjuntos de Junta

Na estrutura mecânica do manipulador foram montados os conjuntos das cinco juntas com os novos componentes, o que possibilita o controle a partir da plataforma LinuxCNC. A seguir é apresentado o procedimento empregado para acoplar os motores com os diferentes componentes adicionais, que integram os conjuntos de cada uma das cinco juntas do robô.

1. Inicialmente foi montado os conjuntos das juntas pertencentes aos graus de liberdade um, quatro e cinco. Este grupo tem uma configuração semelhante a diferença das juntas dois e três. Apresentam engrenagens tipo *Harmonic-Drive*, sendo diferente onde a redução varia para cada junta. Na Figura 4.50 são apresentados os conjuntos das juntas fixadas na estrutura mecânica. Foi possível aproveitar os cabos originais do robô, para os sinais dos *encoder* e a tensão dos motores.

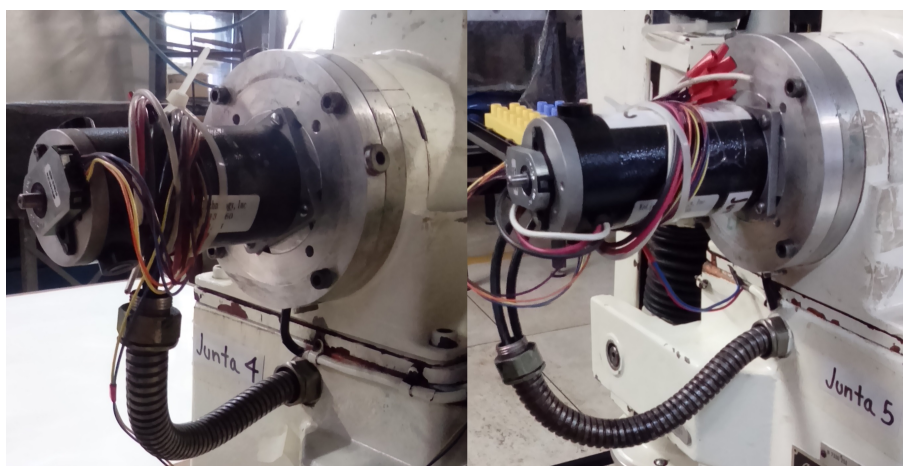


Figura 4.50: Montagem juntas quatro e cinco

2. Finalizado a montagem dos conjuntos das juntas um, quatro e cinco, foram fixados os elementos das juntas dois e três à estrutura mecânica do manipulador. A engrenagem deste tipo de juntas é de acoplamento direto no parafuso de esferas. Também foi aproveitado o cabeamento original do robô para os sinais dos *encoder* e a tensão de alimentação dos servomotores. Na Figura 4.51 são apresentados os dois conjuntos de juntas associados ao segundo e terceiro graus de liberdade na estrutura do manipulador.

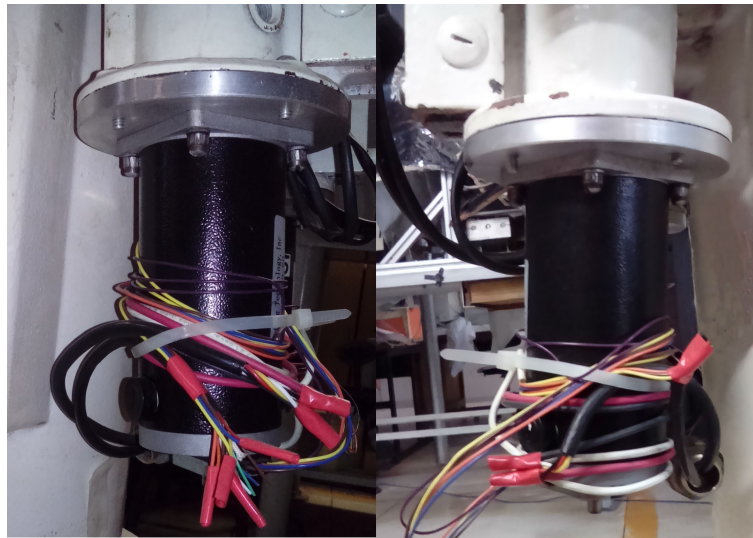


Figura 4.51: Montagem juntas dois e três

4.5.2 Montagem do Gabinete de Controle

Para finalizar a montagem dos diferentes elementos do sistema responsáveis pelo controle do manipulador, foi usado o gabinete do controle projetado pelo fornecedor para instalar os componentes associados aos subsistemas de alimentação e controle. Na Figura 4.52 se apresenta a distribuição interna do gabinete.



Figura 4.52: Distribuição interna do gabinete de controle

Fonte de alimentação: foram aproveitados fusíveis originais do subsistema de alimentação, usados para alimentar os *drives* escolhidos responsáveis pelo controle dos motores. É apresentada na Figura 4.53 a fonte de alimentação junto com a placa de capacitores no gabinete de controle do manipulador. Cada fusível tem valor nominal de 10 A, limitando de forma independente variações inesperadas de corrente para cada motor, protegendo desta forma os *drives* de controle e evitando falhas no sistema.

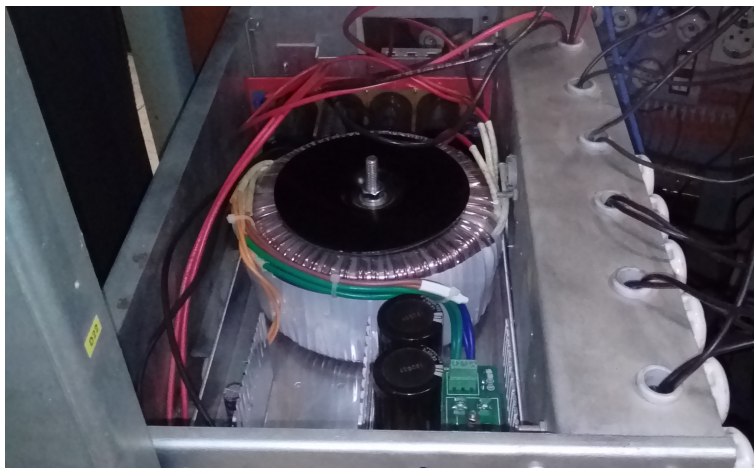


Figura 4.53: Fonte de alimentação no gabinete de controle

Interface de comunicação e drives: é apresentada na Figura 4.54 a distribuição feita para situar a interface de comunicação paralela junto com os *drives* de controle para cada um dos cinco motores do manipulador. A ligação dos componentes de controle associados a placa paralela, *drives* de controle, os motores, *encoders* e a fonte de alimentação foi baseada no diagrama providenciado pelo fornecedor dos componentes. Para maiores detalhes o desenho elétrico está disponível no Apêndice I.



Figura 4.54: Disposição da interface paralela e os *drives* de controle

Gabinete após o retrofit: com os componentes dos subsistemas de controle e alimentação instalados no gabinete, foi adaptada a parte externa para permitir interação com o usuário do sistema. Na Figura 4.55 é apresentada a modificação feita ao gabinete original do manipulador.



Figura 4.55: Parte frontal do gabinete de controle

Na Figura 4.56 observa-se o manipulador ASEA IRB6-S2 após a finalização do processo de montagem.

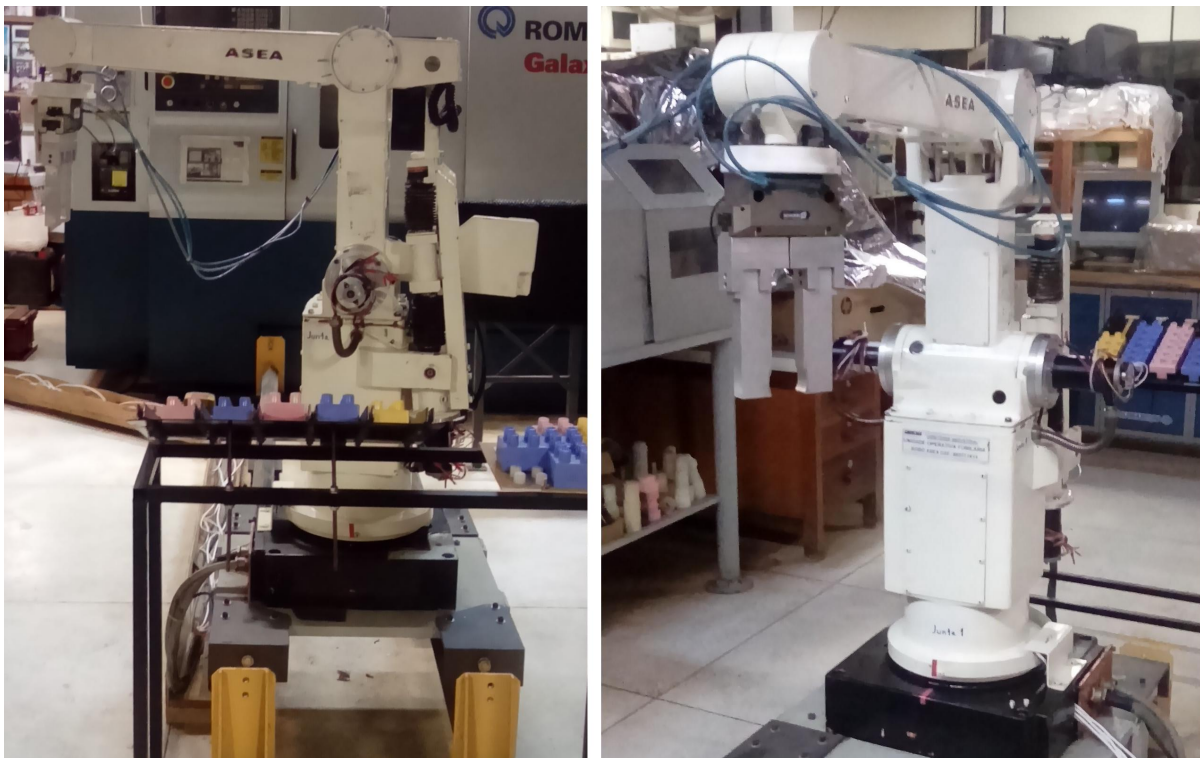


Figura 4.56: Manipulador ASEA IRB6-S2 após *retrofitting*

4.6 Testes Iniciais após o *Retrofitting*

Com o robô equipado com os componentes funcionais para os subsistemas envolvidos na operação do manipulador, é necessário testar e ajustar as variáveis que afetam a movimentação desejada das cinco juntas por meio da configuração associada ao manipulador ASEA IRB6-S2 no controlador LinuxCNC. Variáveis como a validação da cinemática com a execução de programas NC, a calibração da posição angular de cada uma das juntas do robô, a sintonia dos parâmetros do controlador PID dos *drives* associados a cada grau de liberdade, assim como o posicionamento inicial que deverá ter o manipulador antes de iniciar alguma tarefa.

4.6.1 Banco de Testes para Operação Inicial com LinuxCNC

Antes da montagem final no manipulador e para testar os componentes adquiridos junto com o controlador LinuxCNC, foi construído um banco de testes onde foi possível realizar a operação de cada um dos graus de liberdade do robô, mediante aos motores e demais componentes próprios dos cinco subconjuntos de junta. Com banco de testes desenvolvido foram integrados na plataforma LinuxCNC os componentes associados aos subsistemas de alimentação e controle, garantindo a compatibilidade do sistema para controlar o manipulador ASEA IRB6-S2.

Na Figura 4.57 é apresentado o banco de testes fabricado, junto com o gabinete de controle original do robô, fazendo testes de geração e recepção de sinais por meio do LinuxCNC aos componentes físicos adquiridos para o *retrofitting*.

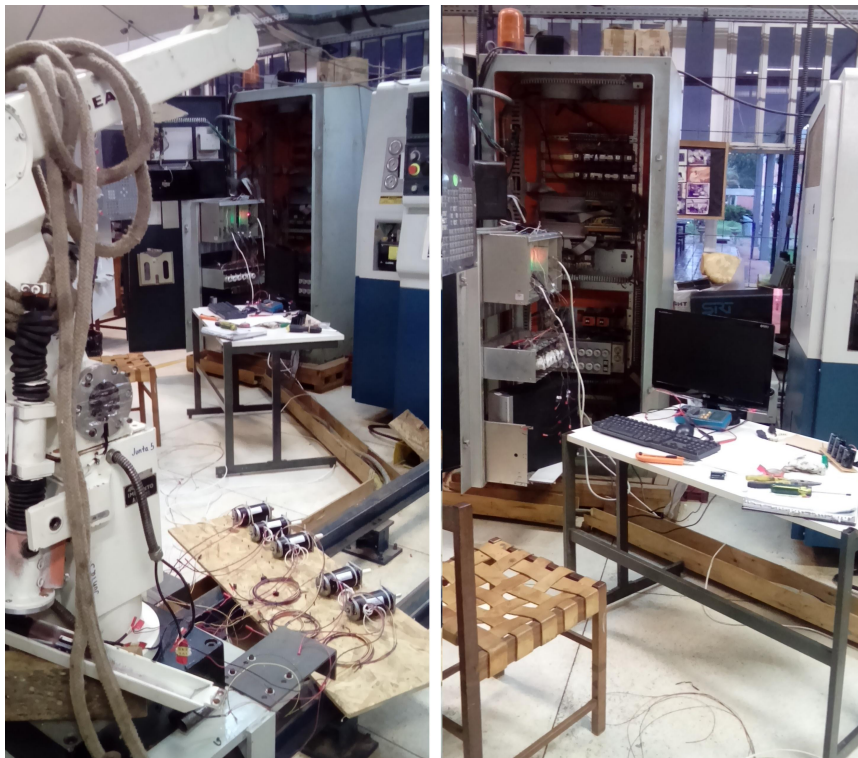


Figura 4.57: Banco de testes dos atuadores do manipulador

Em uma tabela de madeira foram fixados cinco motores associados aos graus de liberdade do robô, ligados através de cabos ao gabinete de controle original do manipulador onde foram colocados os componentes associados ao subsistema de controle tais como: placa paralela; *drives*; fonte de alimentação e computador com LinuxCNC.

O procedimento executado com o banco de testes consistiu em fazer a conexão entre os componentes envolvidos na geração de movimento para cada grau de liberdade do manipulador. Posteriormente foi possível integrar o banco de teste com a plataforma LinuxCNC permitindo gerar sinais de controle para cada servomotor de forma independente. Os componentes para cada junta são apresentados na Figura 4.58.

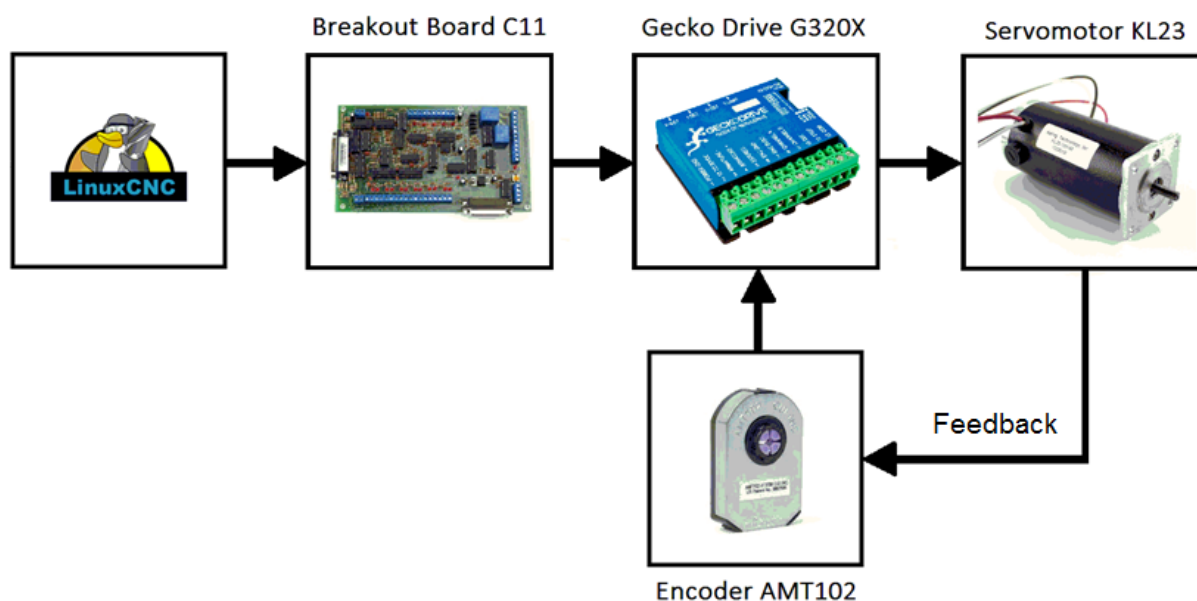


Figura 4.58: Componentes necessários para o controle das juntas

Desta forma foi possível integrar cada um dos componentes necessários para o controle das juntas no controlador LinuxCNC antes de fazer a montagem final dos subsistemas no manipulador. Os testes foram feitos a partir da interface do usuário da plataforma de controle, com a configuração associada ao robô ASEA IRB6-S2. Na Figura 4.59 é apresentada a interface gráfica para o controle do usuário da plataforma, assim como a seção onde foram geradas as ordens para movimentar cada motor.

4.6.2 Simulação da Execução Programa NC

O modelo cinemático definido para permitir o controle do robô ASEA IRB6-S2, foi incluído no controlador LinuxCNC por meio de arquivos em linguagem C e compilados posteriormente com o pacote fonte da versão do controlador, tal como foi descrito na Seção 4.4. Com as equações homogêneas dispostas no controlador LinuxCNC, foi possível gerar uma simulação da movimentação do robô, permitindo verificar o deslocamento correto na trajetória definida do modelo virtual do manipulador. Na Figura 4.60 apresenta-se o modelo de robô virtual executando as instruções geradas pelo LinuxCNC.

Após finalizar a simulação do programa NC, se observa, na Figura 4.61, como o robô realizou de forma correta as instruções geradas, fazendo a simulação de forma satisfatória.

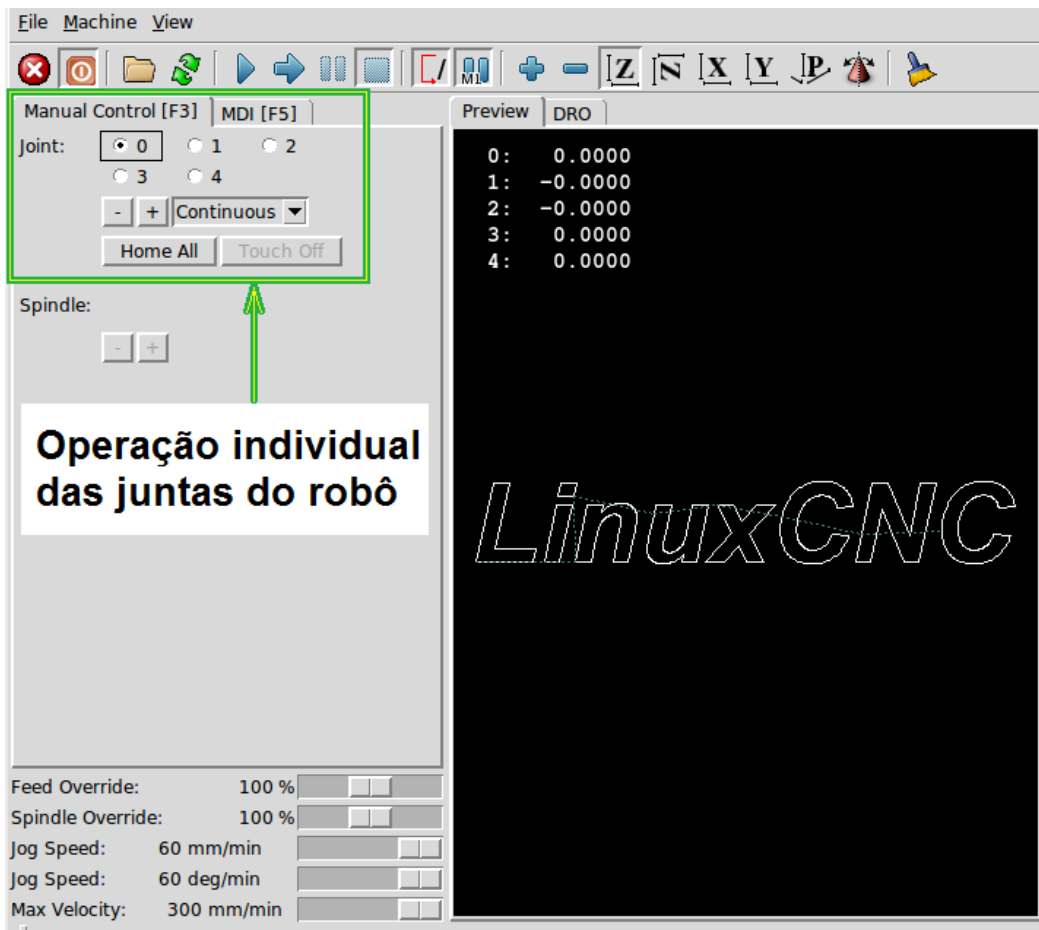


Figura 4.59: Movimentação das juntas desde a interface de usuário do LinuxCNC

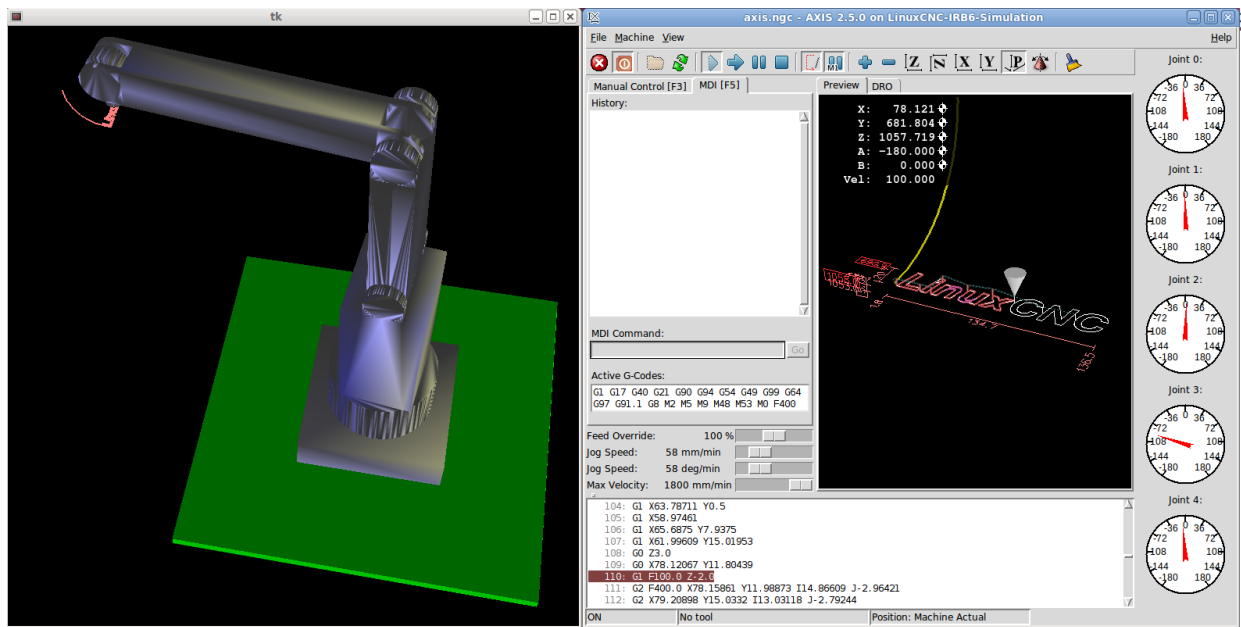


Figura 4.60: Simulação da operação do manipulador em LinuxCNC

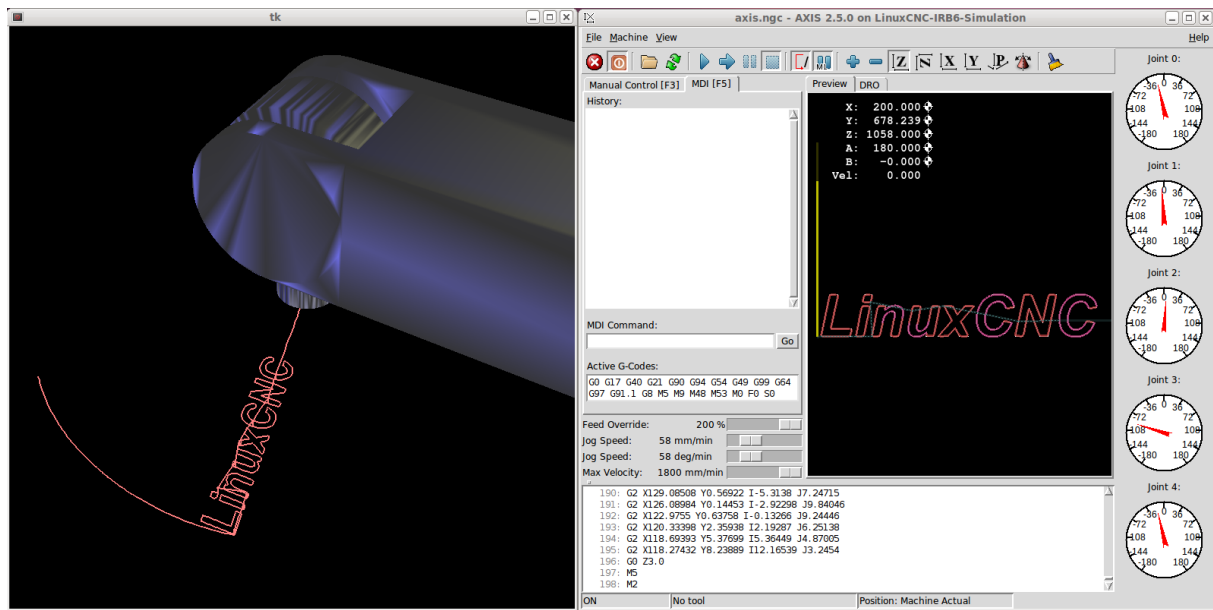


Figura 4.61: Resultado da simulação do manipulador em LinuxCNC

Desta forma foi possível observar que a cinemática incluída no controlador teve uma resposta adequada na execução das instruções de forma virtual. Também foi possível comprovar a integração entre a simulação do modelo do robô e os arquivos gerados após a compilação da configuração do manipulador incluída na instalação da plataforma LinuxCNC. É necessário esclarecer que o tempo de amostragem em um sistema baseado em tempo real depende de múltiplas variáveis associadas a tempos de leitura e aquisição de dados.

4.6.3 Calibração da Posição de Cada Junta

A partir da plataforma LinuxCNC foi efetuado o procedimento para ajustar o número aproximado de pulsos que deveria enviar o controlador, permitindo assim que cada uma das juntas se movimentassem conforme as instruções programadas. O LinuxCNC está em capacidade de calibrar cada uns dos graus de liberdade, independente da configuração na redução mecânica específica entre o motor e a junta do manipulador. Para coordenar esta relação, o controlador dispõe de uma variável no arquivo de configuração INI, chamado *SCALE*. A variável é definida com a quantidade de pulsos ou *steps* necessários para o avanço da junta por unidade de usuário (LINUXCNC.ORG, 2015a).

No caso da configuração para o manipulador ASEA IRB6-S2, a variável *SCALE* se define como a quantidade de *steps* necessários para o deslocamento angular de cada junta com um valor igual a um grau ($SCALE = steps/grau$). LinuxCNC facilita a calibração de cada uns dos eixos do manipulador por intermédio de um assistente de calibração na interface de usuário do controlador. A Figura 4.62 apresenta o módulo de calibração em LinuxCNC, junto com a interface de usuário padrão do controlador.

O processo de ajuste para cada eixo foi feito pelo valor conhecido como rotação, por exemplo 90°, permitindo ajustar facilmente o valor da variável. Posteriormente o assistente possibilita gravar diretamente no arquivo de configuração INI o valor para a variável *SCALE*. São apresentados na Tabela 4.14, os valores das cinco juntas, obtidos com a ajuda do assistente de configuração incluído no LinuxCNC.

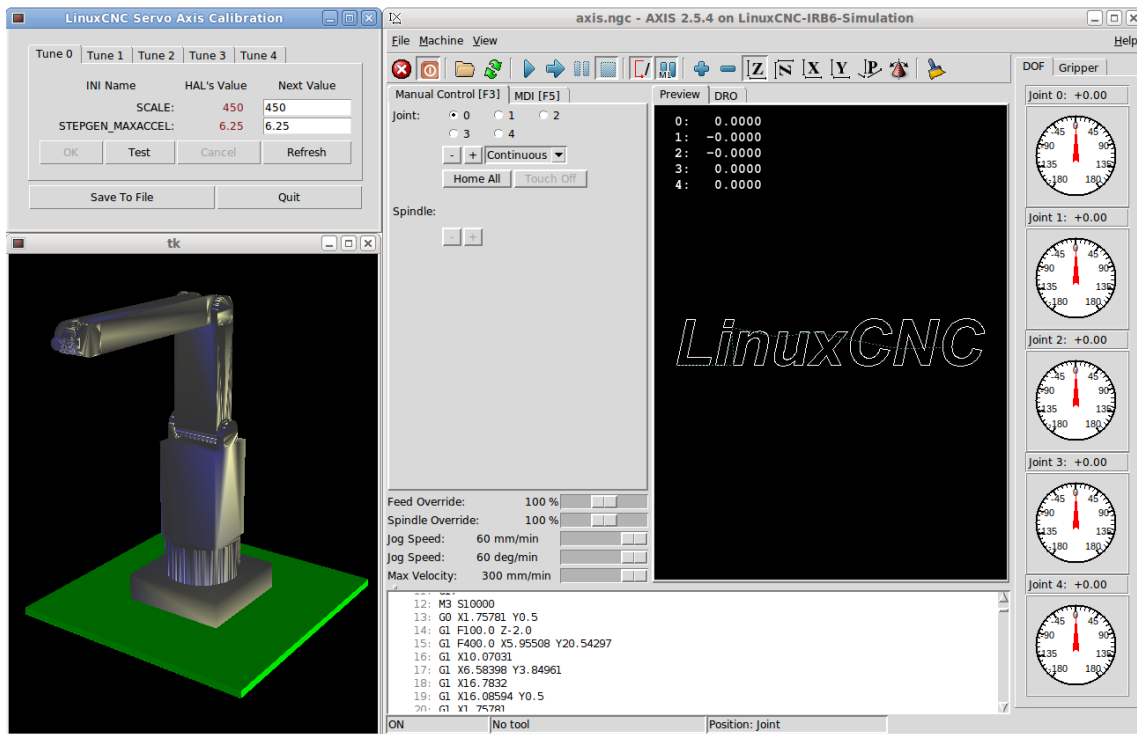


Figura 4.62: Assistente de calibração na interface de usuário do LinuxCNC

Tabela 4.14: Valores de calibração das juntas

Variável	Junta (<i>steps/grau</i>)				
	1	2	3	4	5
<i>SCALE</i>	450	467	495	370	220

Com a ajuda do assistente de calibração dos eixos da plataforma LinuxCNC, foram deduzidas e incluídas no arquivo INI da configuração do manipulador ASEA IRB6-S2, os valores da variável denominada *SCALE*. Desta forma é garantido que os valores das variáveis articulares relativas as juntas, são proporcionais ao movimento rotacional de cada grau de liberdade do robô (LINUXCNC.ORG, 2015b). Em consequência foi simplificado o recurso computacional empregado no calculo dos valores das variáveis articulares do manipulador, devido a que foram substituídas as equações que relacionam o valor de rotação das juntas em função da posição angular dos atuadores (Subseção 2.6.3) com a variável *SCALE* do arquivo INI. Na Figura 4.63 e apresentada a analogia mencionada anteriormente.

4.6.4 Estratégia de Sintonia do PID

O projeto para o controle de manipuladores industriais tradicionalmente é compreendido como sintonia de controladores PID de cada um dos *drives* que gerenciam os motores dos diferente graus de liberdade do robô. Um controle simples é prioridade em relação a um controle complexo, desde a perspectiva do setor industrial (SICILIANO et al., 2008). Assim sendo, o jeito mais simples e efetivo usado na atua-

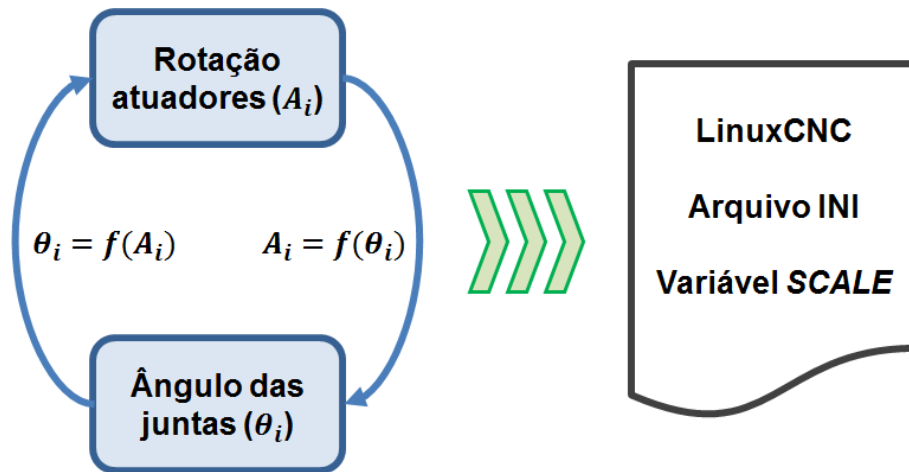


Figura 4.63: Variável SCALE para relacionar rotação servomotores/juntas

lidade para o controle do posicionamento final do efetuator de um robô é a estratégia para o controle independente de juntas. Neste tipo de técnica, cada eixo do robô é controlado como um sistema com uma-única-entrada/uma-única-saída (SISO). O objetivo deste controle independente é comparar e compensar a saída desejada em relação a um sinal de referência (SPONG; HUTCHINSON; VIDYASAGAR, 2005).

Em relação ao controle do manipulador ASEA IRB6-S2, está diretamente associado com os *drives* dos servomotores adquiridos para o subsistema de controle. A compensação requerida para cada junta é ajustada diretamente no *drive* manualmente, procedimento feito por meio de três potenciômetros correspondentes aos resultados dos parâmetros do controlador PID, ou seja, o proporcional (K_p), derivativo (K_d) e o integral (K_i). Na Figura 4.64 são apresentados os potenciômetros ajustáveis dos *drives*, usados para o *retrofitting* do manipulador.

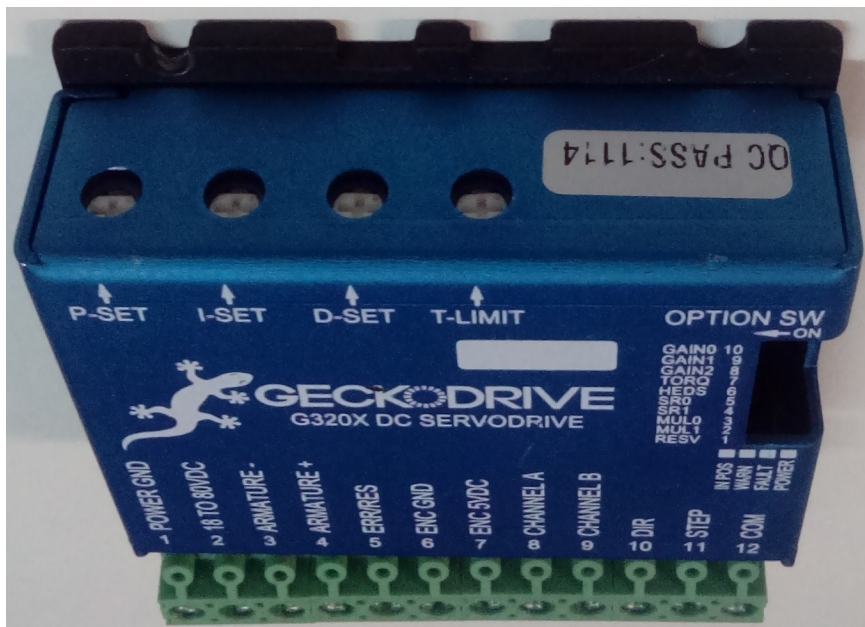


Figura 4.64: Parâmetros do PID ajustáveis no *drive* empregado

O fornecedor do dispositivo responsável pelo controle do posicionamento das juntas do manipulador elaborou um roteiro para sintonizar cada um dos parâmetros do PID, diretamente no *drive*, garantindo que ao final do procedimento cada um dos motores conservem sua posição independente da carga usada no efetuator do manipulador. Por conseguinte, foram sintonizados os *drives* com a carga máxima projetada para o robô. A carga usada neste caso foi uma garra pneumática referida na Seção 4.2, utilizada para o posicionamento de peças. A seguir são apresentados os passos empregados para a sintonia dos controladores de cada junta, do manipulador ASEA IRB6-S2 (GECKODRIVE, 2011).

1. Foi girado totalmente o potenciômetro associado com o parâmetro Integral, "I", no sentido anti-horário, para zerar o valor da constante.
2. O termo Derivativo "D", deve ser ajustado proporcionalmente ao parâmetro Proporcional "P". Logo, o primeiro termo a variar foi o "D", posteriormente o potenciômetro relacionado ao parâmetro "P". Se a sequência não é aplicada dessa forma, o motor vai oscilar violentamente, prejudicando a junta associada.
3. Foram fixados os termos "P" e "D", aproximadamente 25% da escala total dos potenciômetros. O termo "D", segundo o fornecedor, é o causador do barulho do motor. Desta forma, quando o parâmetro Derivativo do *drive* foi aumentado, o "zumbido" do motor conseqüentemente aumentou.
4. A rigidez dos motores é proporcional a variação do potenciômetro associado ao termo "P". Como consequência, foi estabelecido para cada *drive* um ponto médio, relacionando uma rigidez tal que os motores mantivessem a posição das juntas tentando reduzir o mínimo de barulho do servomotor.
5. Finalmente, foi ajustado o parâmetro "I", responsável por eliminar o erro em regime permanente da posição comandada em relação a posição atingida (LEWIS; DAWSON; ABDALLAH, 2004). A variação deste parâmetro é proporcional à rigidez do motor. O que foi ajustado em relação ao potenciômetro do parâmetro "P".

Foi sintonizado o controlador PID em cada um dos *drives* responsáveis pelos servomotores do manipulador, segundo as instruções do fornecedor. É necessário esclarecer que os ajustes feitos nos parâmetros dos PID não afetam a precisão do posicionamento próprio dos *drives*. O único efeito evidente é o tempo de estabelecimento de cada um dos servomotores do manipulador *gecko320x*. O escopo do trabalho não contempla uma análise profunda na parte da sintonização do controlador PID, por isso só foi levando em consideração a sugestão do fornecedor *Gecko* para uma sintonia básica tendo como característica fundamental o tempo de posicionamento comandado para as juntas.

4.6.5 Configuração Posicionamento *home* do Manipulador

O posicionamento do manipulador industrial em uma posição *home* é um desafio quando o *encoder* usado não proporciona uma posição absoluta ao sistema de controle (DIETRICH et al., 2010). Desta forma, para o controle do manipulador ASEA IRB6-S2 foi necessário configurar na plataforma LinuxCNC uma rotina para o posicionamento *home* das cinco juntas do manipulador.

A sequência foi feita diretamente nos arquivos de configuração do projeto criado para o robô ASEA IRB6-S2 no LinuxCNC, especificamente nos arquivos INI e HAL. O arquivo INI possibilita habilitar a sequência *home* e ajustar as velocidades para cada junta, enquanto o arquivo HAL permite gerenciar os sinais dos sensores de posicionamento para cada junta do manipulador.

Os sensores originais usados junto com LinuxCNC para o posicionamento *home* do manipulador são *reed switch*. São cinco sensores que foram localizados para garantir um posicionamento inicial para cada uma das juntas do robô (ASEA, 2003).

É apresentada na Figura 4.65 a distribuição da configuração feita nos arquivos INI e HAL, para implementar a sequência *home* na plataforma LinuxCNC.

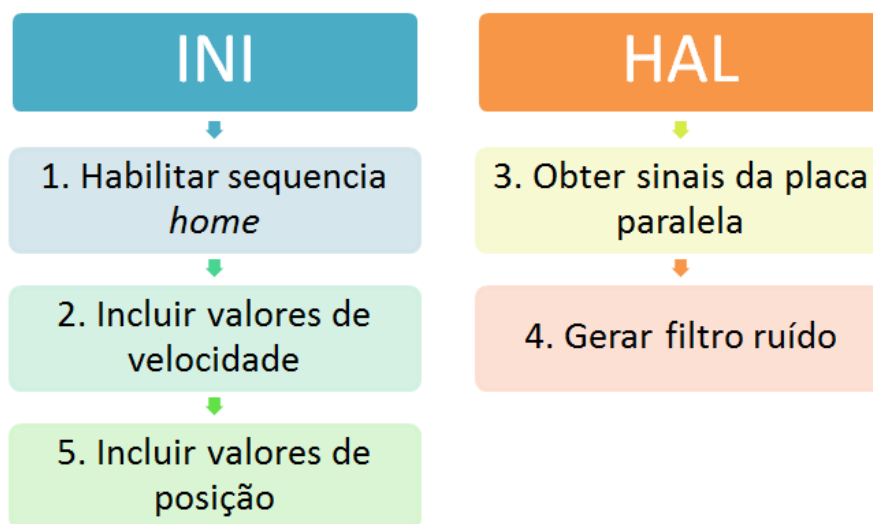


Figura 4.65: Configuração sequência *home* no LinuxCNC

A seguir são explicados cada um dos passos seguidos, permitindo obter uma sequência de *home* para a operação do manipulador ASEA IRB6-S2.

1. **Habilitar sequência *home*:** foi modificado no arquivo INI o valor da variável chamada *HOME SEARCH VEL*. O valor padrão é zero, modificando o valor é possível habilitar a opção *home* de cada junta. O valor não deve ser alto, pois no momento em que é detectado o pulso enviado desde o sensor, a junta suspende seu movimento, sendo o erro de posicionamento proporcional a velocidade fixada (LINUXCNC.ORG, 2015c).
2. **Incluir valores de velocidade:** existe outra variável indispensável para o posicionamento inicial do manipulador, chamada *HOME LATCH VEL*. A variável deverá ter um valor diferente de zero, se a sequência *home* é habilitada. É necessário que o valor seja menor que a variável *HOME SEARCH VEL*, pois é a velocidade final para detectar a posição adequada do sensor da junta. Na Tabela 4.15 são apresentados os valores para cada juntas das variáveis modificadas no arquivo INI.
3. **Obter sinais da placa paralela:** no arquivo HAL foram configuradas as entradas da placa paralela. Foram usadas quatro entradas da placa paralela: três entradas para os sensores associados as três

Tabela 4.15: Valores das variáveis de velocidade para a sequência *home*

Variável/Junta	1 (mm/s)	2 (mm/s)	3 (mm/s)	4 (mm/s)	5 (mm/s)
<i>HOME SEARCH VEL</i>	3.75	0.7	0.7	1.0	1.0
<i>HOME LATCH VEL</i>	-0.5	-0.5	-0.5	0.8	-0.8

primeiras juntas do manipulador, enquanto as juntas quatro e cinco compartilham uma entrada da placa paralela. Na Figura 4.66 se apresentam as linhas de código usadas na plataforma LinuxCNC para obter os sinais dos sensores das duas primeiras juntas do manipulador.

```
net home-x <= parport.0.pin-13-in-not
net home-y <= parport.0.pin-11-in-not
net home-x => axis.0.home-sw-in
net home-y => axis.1.home-sw-in
```

Figura 4.66: Configuração juntas um e dois arquivo HAL

4. **Gerar filtro de ruído:** para obter os sinais das juntas: três, quatro e cinco, foi necessário gerar desde o LinuxCNC dois filtros de ruído, devido a instabilidade do sinal gerado pelos sensores. Essa característica na leitura desde o LinuxCNC produzia falhas no momento de fazer a sequência *home* das juntas. O módulo responsável para criar os filtros desde LinuxCNC é intitulado *debounce*. O filtro realizado gera um atraso na leitura das entradas, neste caso das juntas três, quatro e cinco. Na Figura 4.67 é apresentada a configuração no arquivo HAL.

```
#Carregar o modulo debounce

loadrt debounce cfg=2 # Dois filtro gerados
addf debounce.0 base-thread
setp debounce.0.delay 15 # Tempo de atraso de 15 nS

#Configurar leitura da porta paralela

#Home Sensor Joint 3
net sw-in2 parport.0.pin-12-in-not => debounce.0.1.in
net sw-debounced2 debounce.0.1.out => axis.2.home-sw-in

#Home Sensor Joint 4 e 5
net sw-in3 parport.0.pin-15-in-not => debounce.0.0.in
net sw-debounced3 debounce.0.0.out => axis.3.home-sw-in
net sw-debounced3 debounce.0.0.out => axis.4.home-sw-in
```

Figura 4.67: Configuração juntas três, quatro e cinco arquivo HAL

5. **Incluir valores da posição:** com as velocidades incluídas no arquivo INI para procurar a posição

do sensor em cada junta e a configuração para adquirir os sinais dos sensores no arquivo HAL foram ajustados os valores finais do posicionamento *home* do manipulador no arquivo INI. Estes valores estão baseados na posição *home* da Figura 4.68.



Figura 4.68: Configuração *home* robô ASEA IRB6-S2

Desta forma, a sequência do posicionamento foi configurada para cada junta, iniciando na base do manipulador e finalizando no efetuador final ou quinto grau de liberdade. Os arquivos INI e HAL estão disponíveis no Apêndice M e Apêndice N, enquanto o vídeo da sequência do posicionamento *home* do manipulador está disponível em <<https://youtu.be/cpx9kQ3-Bxg>>.

Capítulo 5

Estudo de Casos e Análise de Capabilidade

5.1 Introdução

No presente capítulo será apresentado uma abordagem baseada em um Estudo de Casos com o robô ASEA IRB6-S2, permitindo avaliar o comportamento e desempenho após a finalização do *retrofitting*, através da metodologia proposta no Capítulo 3. O objetivo dos experimentos é de quantificar e conhecer parâmetros que possibilitem estimar se é factível a integração com um processo industrial com esquemas de operações industriais específicas. Avaliar o desempenho do robô permite verificar as capacidades de desenvolver tarefas definidas segundo a aplicação (ROMANO, 2002).

Alguns dos parâmetros que serão apresentados neste capítulo por meio dos diferentes estudos de casos estão relacionados com uma medida que quantifica o desempenho do manipulador industrial, denominada *exatidão* dos movimentos onde o robô pode realizar a partir de um processo predeterminado.

Parâmetros como a repetibilidade e capabilidade do manipulador em condições não variáveis de montagem, meio ambiente, instrumentos de medição, configuração do sistema de controle e tipos de trajetórias a serem executadas pelo robô, garantem a avaliação desejada.

O manipulador deve ter a capacidade de repetir a mesma tarefa várias vezes, e para quantificar isso é necessário o parâmetro de avaliação do desempenho do robô chamado *repetibilidade*, que pode ser definida como a habilidade que o manipulador tem em atingir de forma consistente um ponto específico (ROMANO, 2002).

Há outros parâmetros que serão apresentados ao longo do presente capítulo, a fim de conhecer o desempenho do robô é a capabilidade ou capacidade do processo para produzir produtos ou serviços livres de defeito de forma controlada, permitindo conhecer se o robô está em capacidade de fazer um processo específico com requisitos de qualidade definidos. Uma das técnicas que permite saber se o processo pode falhar dentro de limites específicos, é a análise de capabilidade (STERN, 2016).

5.2 Validação *Retrofitting* Baseado em Programação por *teach-in*

No presente estudo de caso é empregada uma técnica de programação *on-line* denominada *teach-in*, a qual possibilita a introdução de um programa específico no controlador de um manipulador industrial. Este método de programação é o mais usado atualmente, baseado no aprendizado de trajetória e pontos (GRZEJSZCZAK; LEGOWSKI; NIEZABITOWSKI, 2015). O princípio desse tipo de programação é deslocar manualmente o efetuator do robô por meio de uma sequência desejada de pontos sob a supervisão do operador (NOF, 1999), permitindo armazenar uma ou várias trajetórias que são executadas pelo manipulador as vezes que seja necessário. Para o caso da validação do *retrofitting*, a sequência de pontos específicos fez com que o robô conseguisse trasladar uma peça tipo *LEGO* de um local para o outro, ressaltando que o robô tem a capacidade de atingir as instruções do controlador *LinuxCNC* baseadas na programação *teach-in*.

O módulo *teach-in* responsável por gravar ou adquirir os pontos desejados em termos de coordenadas cartesianas da área de trabalho do robô (X,Y,Z,A,B), foi gerado por meio de um aplicativo desenvolvido em linguagem *Python* e pode ser consultado no Apêndice L. Os pontos são guardados em um arquivo de edição, possibilitando a modificação segundo a aplicação final. Nesse caso foi editado o arquivo de saída do módulo *teach-in* para obter dois programas NC compatíveis com o interpretador *LinuxCNC* da norma RS-274. A Figura 5.1 apresenta os passos para obter a programação do robô com o módulo *teach-in*:

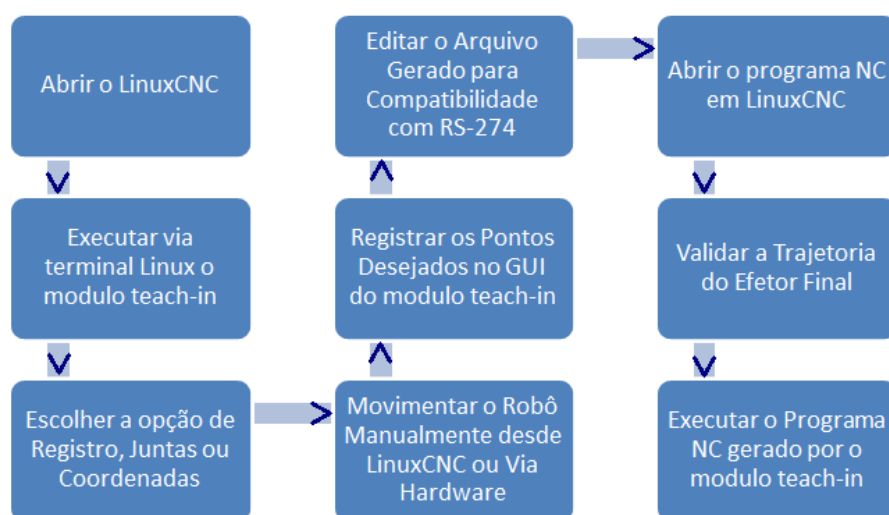


Figura 5.1: Processo de obtenção de pontos com o módulo *teach-in* em *LinuxCNC*

Para a captura de pontos é necessário abrir a interface GUI do *LinuxCNC* e em paralelo executar o módulo *teach-in* a partir do terminal Linux, possibilitando desta forma ter um registro dos pontos finais desejados para uma aplicação específica. É apresentada na Figura 5.2 a interface GUI no módulo *teach-in*:

O módulo *teach-in* tem a opção de fazer o monitoramento dos pontos finais desejados ou armazenar os pontos em um arquivo para edição. Também tem a alternativa de conhecer o deslocamento em graus de cada uma das juntas do robô, assim como os valores das coordenadas cartesianas da área de trabalho. O processo de registro dos pontos deve ser feito de forma manual com um botão disposto para essa tarefa. O detalhe das opções descritas anteriormente são apresentadas na Figura 5.3

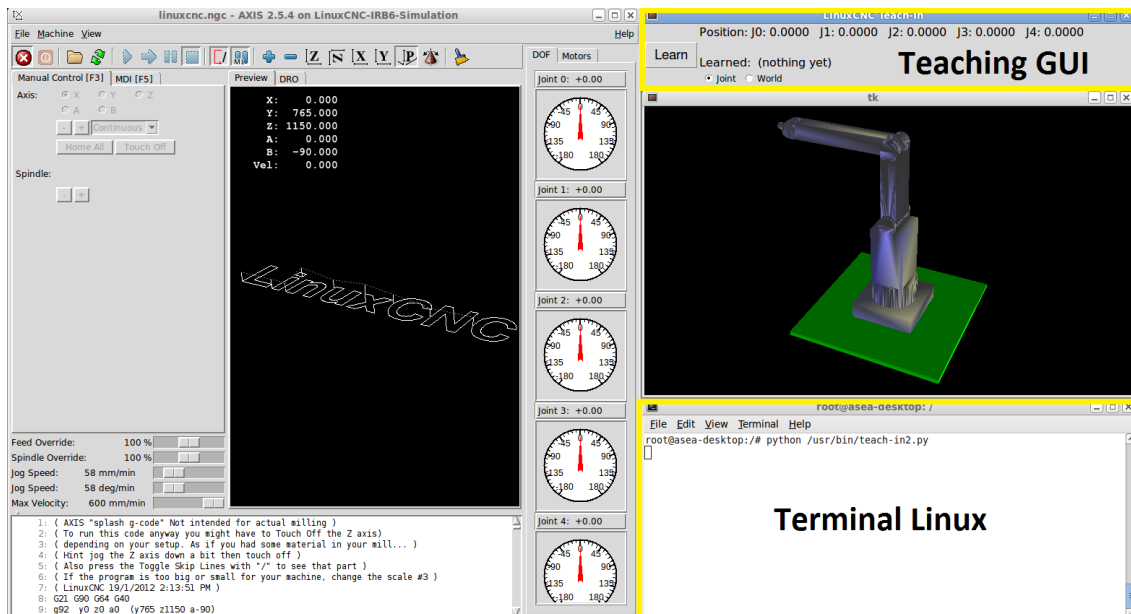


Figura 5.2: GUI inicial *teach-in* com *LinuxCNC*

Botão para Registrar

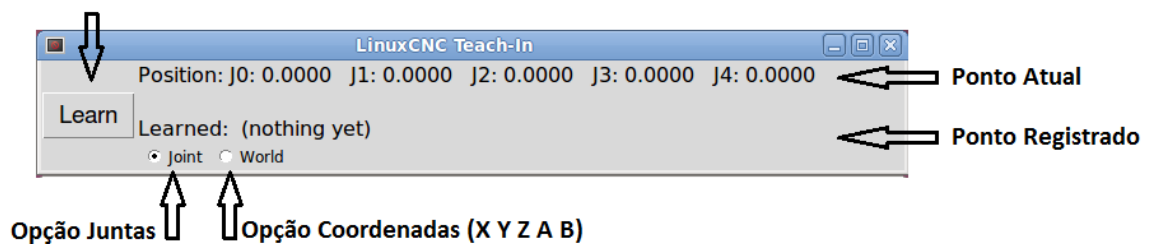


Figura 5.3: GUI do módulo *teach-in*

Através da interface GUI do *LinuxCNC* ou via hardware, é necessário movimentar o robô até os pontos desejados segundo a aplicação final. Nesse caso, o manipulador ASEA recebeu o comando com um *pendant* P4-S do fornecedor VistaCNC, no qual pode-se observar na Figura 5.4. O uso desse dispositivo facilita o controle do robô por qualquer usuário, garantindo ergonomia e flexibilidade no momento de localizar os pontos desejados para o efetuator do robô, além de ter compatibilidade com *LinuxCNC* por meio do padrão USB.

O arquivo de edição gerado pelo módulo *teach-in* tem a informação dos pontos desejados segundo a opção escolhida na interface GUI do módulo. Se é *Joint*, o arquivo de edição terá a informação de cada junta do manipulador em graus, mas se a opção escolhida é *World* o arquivo gerado vai ter as variáveis cartesianas da área de trabalho do robô (X, Y, Z, A, B).

A Figura 5.5 apresenta a comparação do arquivo editável com os pontos adquiridos, com um exemplo de como são registrados os valores das variáveis articulares do robô, ou as juntas e as coordenadas cartesianas das áreas de trabalho (X, Y, Z, A, B), gerado por meio do módulo *teach-in*, e da mesma forma é apresentado o formato do arquivo compatível com a norma RS-274.



Figura 5.4: Pendant P4S compatível com *LinuxCNC*
 Fonte: VistaCNC (2016, p. 1)

```
point 1 => J0: 0.0000 J1: 0.0000 J2: 0.0000 J3: -90.0000 J4: 0.0000
point 2 => J0: 20.0714 J1: -10.0054 J2: 5.0820 J3: -90.0000 J4: 0.0000
N15 G01 X-258.7535 Y708.1737 Z1107.8788 A180.0000 B-5.0820
N20 G01 X-258.7535 Y708.1737 Z1107.8788 A180.0000 B-5.0820
N25 G01 X-500.2634 Y564.0922 Z1107.8788 A180.0000 B-5.0820
```

Opção Juntas
 Opção Coordenadas

```
1: G21 G90 G64 G40
2:
3: (Programa 1)
4: g0 X0 Y0 Z0 F450
5: g1 X0.00 y-1.069 z-82.947 A-8.028 B3.747
6: M101
7:
8: g1 X0 Y0 Z0 A0 B0
9: g1 X-216.187 Y7.603 Z44.899 A13.926 B10.303
10: g1 X-217.187 y11.388 z-78.732 A2.407 B13.788
11: M102
12: g1 X-216.187 Y7.603 Z44.899 A13.926 B10.303
13:
14: (programa2)
15: (g0 X0 Y0 Z0 F450)
16: (g1 x-0.009 y-10.359 z-52.330 a-0.737 b3.994)
17: (M101)
18: (g1 x0 y0 z0 a0 b0)
19: (g1 x-198.874 y44.940 z45.254 a49.543 b4.438)
20: (g1 x-195.820 y32.556 z-39.112 a36.897 b8.669)
21: (M102)
22: (g1 x-198.874 y44.940 z45.254 a49.543 b4.438)
```

Figura 5.5: Arquivos gerados em *LinuxCNC* com a opção *teach-in*
 a) Arquivo pontos adquiridos; b) Arquivo código G

Após ter os dois programas NC, foi possível executá-los com o controlador *LinuxCNC* para movimentar o robô segundo os pontos finais obtidos pelo módulo *teach-in*, tendo como resultado final a movimentação de uma peça entre dois locais diferentes com o efetuador do robô.

É possível observar na Figura 5.6 a distribuição física disposta para garantir a movimentação de uma peça tipo *LEGO* de um ponto a outro, de acordo com as instruções interpretadas pelo controlador *LinuxCNC*, do programa NC gerado com a estratégia de programação *teach-in*.

A movimentação feita pelo robô para mover a peça *LEGO* tem um registro gráfico na interface GUI do *LinuxCNC* apresentando a trajetória do efetuador do robô, com isso a garra pneumática fixada na junta final possibilita a execução da aplicação. A Figura 5.7 apresenta a trajetória do efetuador que deverá atingir o manipulador para levar a peça *lego* segundo os pontos programados via *teach-in*.



Figura 5.6: Estudo de caso pick and place

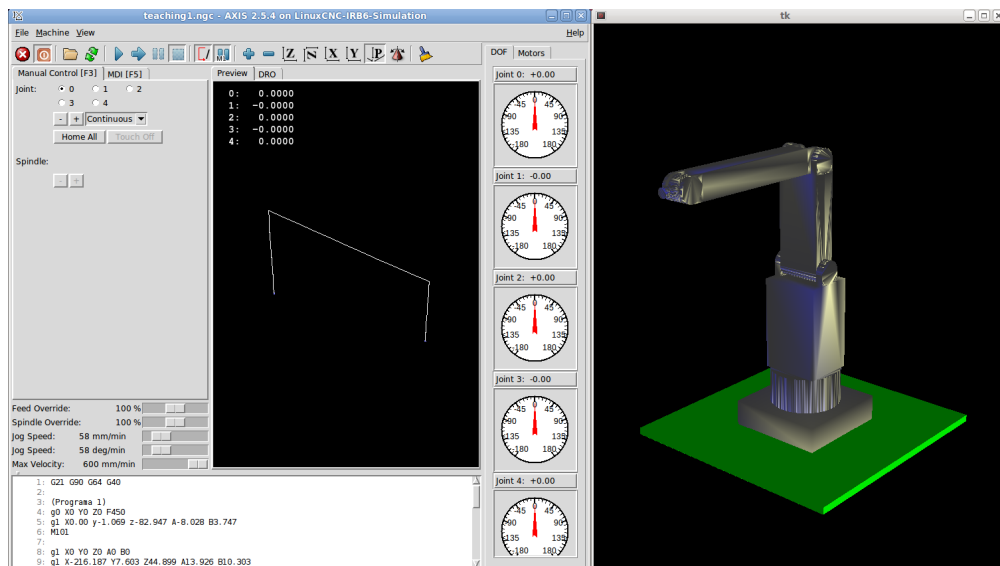


Figura 5.7: Trajetória desejada via *teach-in* programa NC número 1

Após a execução do primeiro programa NC gerado pelo módulo *teach-in*, é possível validar a trajetória seguida pelo efetuador do manipulador na GUI do controlador *LinuxCNC*, assim como uma simulação básica do modelo do robô IRB6-S2, conforme apresentado na Figura 5.8.

Para acionar a garra pneumática foram gerados dois comandos compatíveis com a norma RS-274 para abrir e fechar a garra quando estivesse na posição adequada com a peça *LEGO*. Esses comandos são M101 (Abrir) e M102 (Fechar) adicionados ao programa NC. É possível observar na Figura 5.9 a segunda simulação antes de executar o segundo programa NC obtido a partir do módulo *teach-in*, permitindo verificar por meio do controlador *LinuxCNC* se existem linhas de código com erros que devam ser corrigidos, possibilitando que o robô movimente a peça entre os pontos desejados.

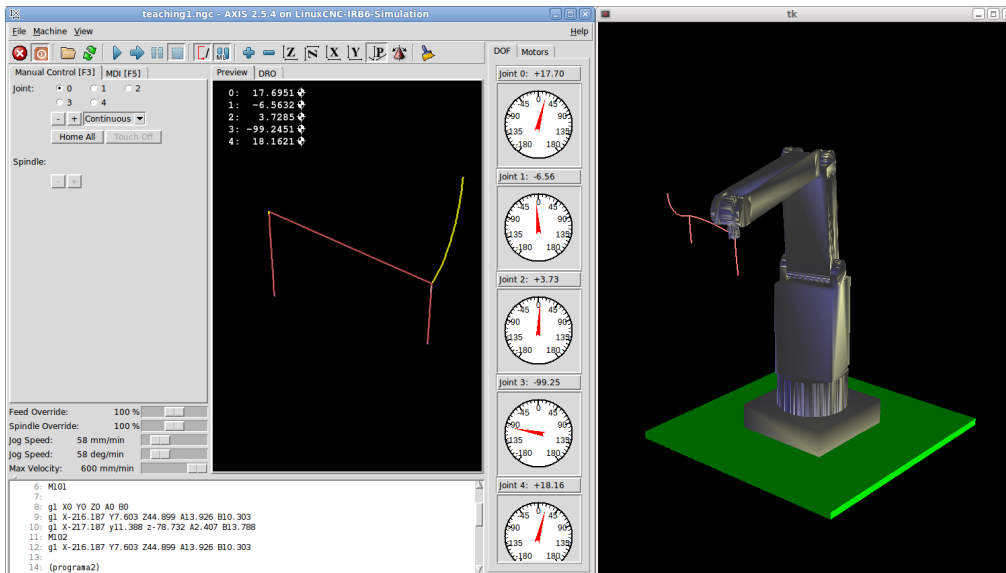


Figura 5.8: Simulação posicionamento peça *LEGO* Programa NC número 1

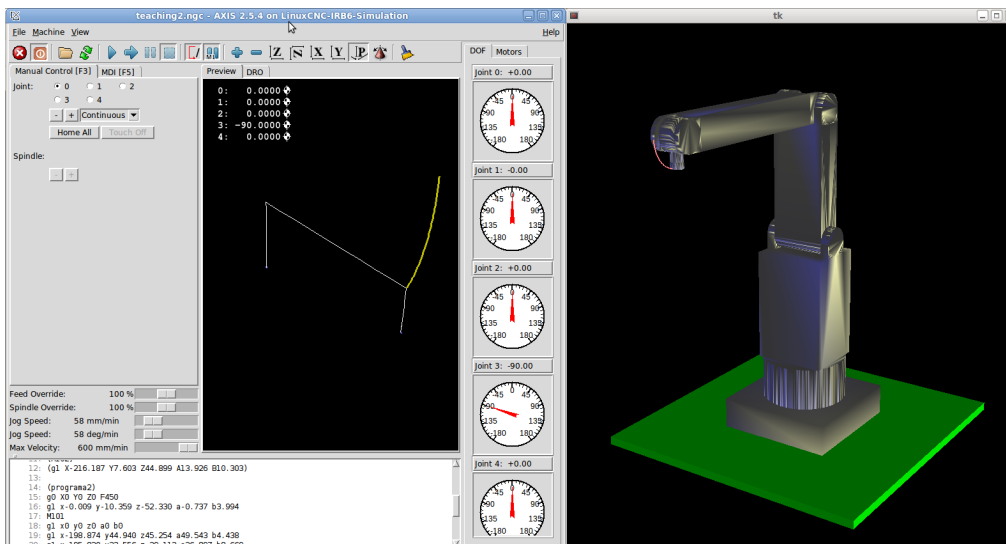


Figura 5.9: Trajetória desejada via *teach-in* programa NC número 2

Finalmente pode-se observar na Figura 5.10, no GUI do controlador *LinuxCNC* a trajetória atingida pelo efetuador do manipulador do segundo programa NC gerado a partir do módulo *teach-in*, permitindo o deslocamento de uma peça tipo *LEGO* entre dois pontos estabelecidos pelo operário na programação *on-line* com o robô IRB6-S2.

Os dois programas NC gerados a partir da estratégia de programação de robôs *teach-in*, foram verificados antes de serem executados para determinar possíveis erros nas linhas de código através do *LinuxCNC*, e posteriormente executados e simulados em tempo real, permitindo validar que o manipulador após o *retrofiting* tem a capacidade de se deslocar manualmente com a supervisão de um operador para permitir a programação *on-line* do módulo *teach-in*, além de atingir pontos desejados registrados e interpretados de um arquivo compatível com a norma RS-274.

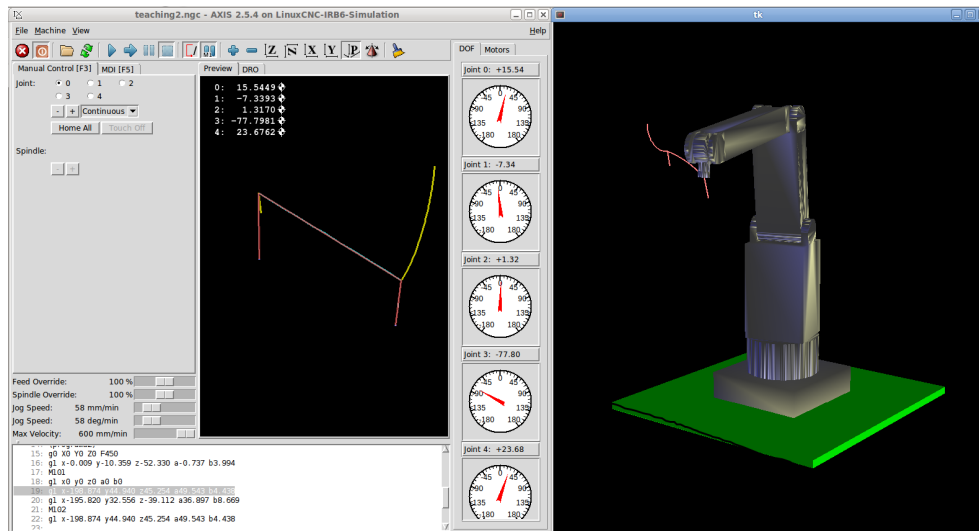


Figura 5.10: Simulação posicionamento peça *LEGO* programa NC número 2

Os programas NC usados para movimentar uma peça de um ponto inicial até um ponto final estão disponíveis no apêndice A. Os vídeos da movimentação do robô para os dois programas NC estão disponíveis para consulta em <<https://youtu.be/ttGI1hPhhWQ>>.

5.3 Validação *Retrofitting* com Programa NC Gerado por CAD/CAM

Com o programa exemplo do *LinuxCNC* gerado por CAD/CAM compatível com a norma RS-274, foi possível validar a cinemática do robô através da movimentação do efetuador, desenhando em uma folha de papel a figura interpretada pelo controlador com ajuda de uma caneta fixada na junta final do manipulador. É necessário esclarecer que o programa gerado por CAD/CAM contém uma sequência de comandos em código G com pontos em coordenadas X, Y e Z os quais devem ser atingidos pelo efetuador. É apresentada na Figura 5.11 a trajetória realizada pela caneta fixada ao manipulador IRB6-S2.

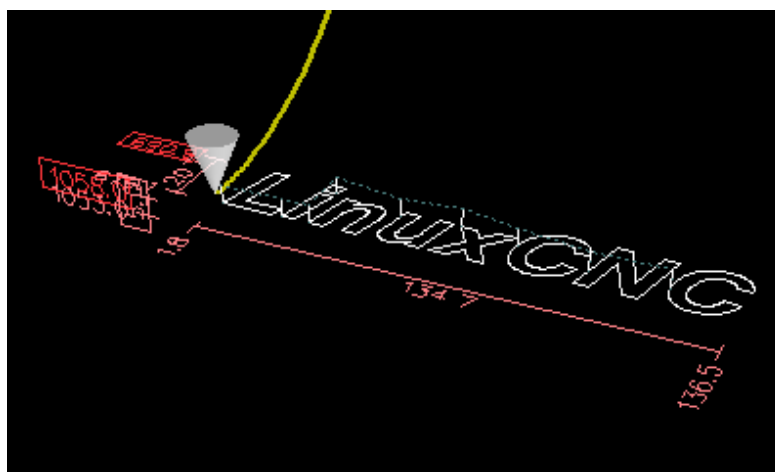


Figura 5.11: Trajetória a ser atingida pelo manipulador

Terminada a execução do programa exemplo no controlador *LinuxCNC*, na Figura 5.12 é apresentada a simulação da trajetória atingida pela ferramenta na interface GUI do controlador. A simulação que foi executada em tempo real junto com a movimentação física do manipulador, para que a caneta, como o efetuator do presente experimento, conseguisse desenhar sobre a folha de papel o traço determinado pelo programa NC carregado e interpretado pelo controlador *LinuxCNC*, gerando as coordenadas para o efetuator em termos de X, Y e Z para cada linha de código do programa NC.

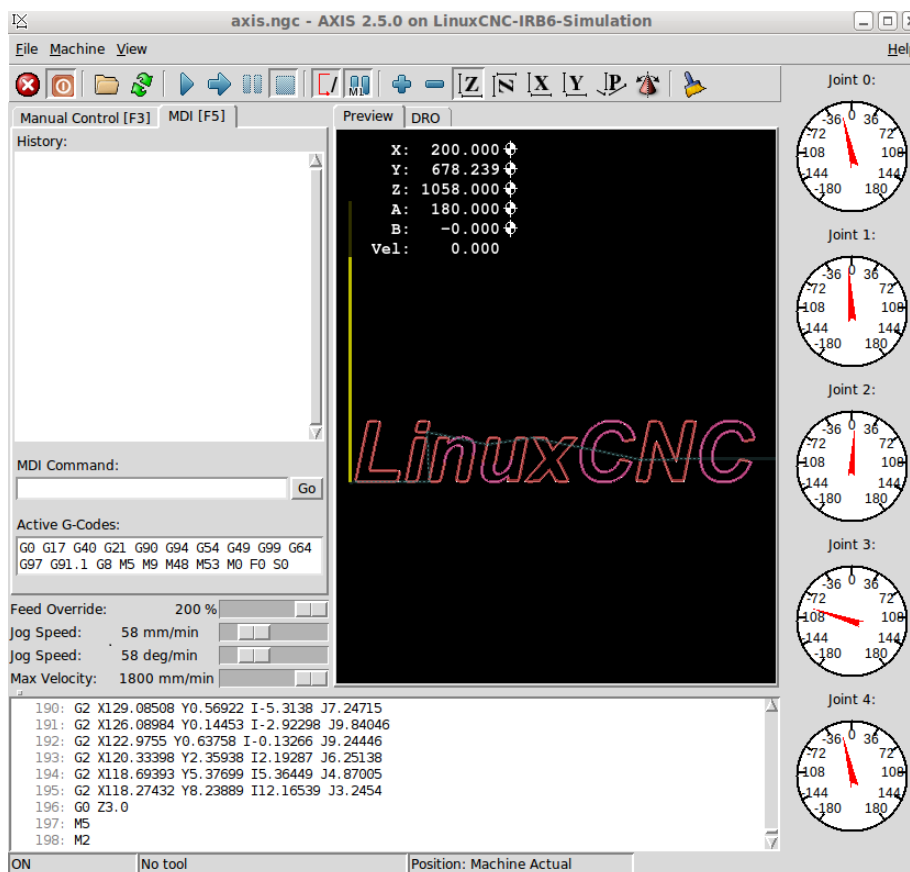


Figura 5.12: Simulação da trajetória gerada pelo programa NC

Paralelamente a simulação no controlador *LinuxCNC*, o robô fez a movimentação de cada uma das juntas segundo a cinemática integrada própria do manipulador, esperando ter ao final da execução da rotina um resultado semelhante ao mostrado pela simulação. É possível observar na fig. 5.13 a operação do manipulador, tal como o acondicionamento feito para possibilitar a validação da cinemática inversa do manipulador através do desenho feito em uma folha de papel com uma caneta comum.

Terminada a execução da rotina foi possível observar graficamente na folha de papel se o resultado gerado pelo robô é semelhante as instruções geradas pelo controlador baseado no programa NC. O resultado é apresentado na Figura 5.14.

Graficamente é possível contrastar o resultado obtido no desenho da trajetória feita pela caneta fixada ao robô, e garantir que a cinemática implementada no controlador -nesse estudo de caso- está funcionando virtualmente e fisicamente com a configuração do robô ASEA. O vídeo da execução do presente estudo de caso está disponível para consulta em <<https://youtu.be/ttGI1hPhhWQ>>.



Figura 5.13: Robô em operação executando o programa NC

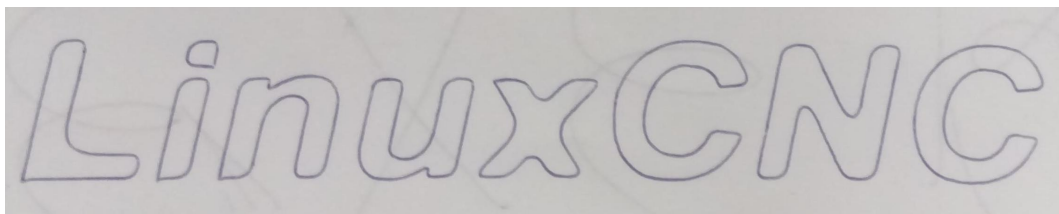


Figura 5.14: Resultado do traço feito pela caneta como efetuator do robô

5.4 Estudo de Repetibilidade com Peça de Geometria Complexa em 2D Gerada por CAD/CAM

Para esse experimento mede-se a diferença entre a espessura do traço gerado por vários percursos feitos pelo efetuator e também a mesma trajetória desenhada em um único percurso, tudo comandado por meio do *LinuxCNC*, possibilitando estimar a diferença entre as espessuras das linhas feitas pela caneta e assim permitindo avaliar a *repetibilidade* como uma estimativa da distância máxima esperada da diferença entre as posições atingidas pelo manipulador de um único ponto programado previamente. Neste caso, Para o presente experimento foi usada uma distribuição de *T-Student* para extrair o número de amostras necessárias a serem executadas pelo robô, nesse caso foram escolhidas 10 repetições, ou seja, 9 graus de liberdade e que junto a um índice de confiança (IC) de 99.5% (próximo a 99.7%), se estabelece uma incerteza considerável com um $t = 3,69$ (WALPOLE; MYERS; MYERS, 1999).

O programa NC gerado para este estudo de caso foi escolhido de um exemplo do controlador *LinuxCNC* onde foram removidas algumas linhas de código visando diminuir o tempo da execução total dos testes. É possível observar na Figura 5.15 a simulação da trajetória gerada pelo efetuador do manipulador, nesse caso foi usada uma caneta fixada na última junta. O programa NC pode ser consultado no apêndice B.

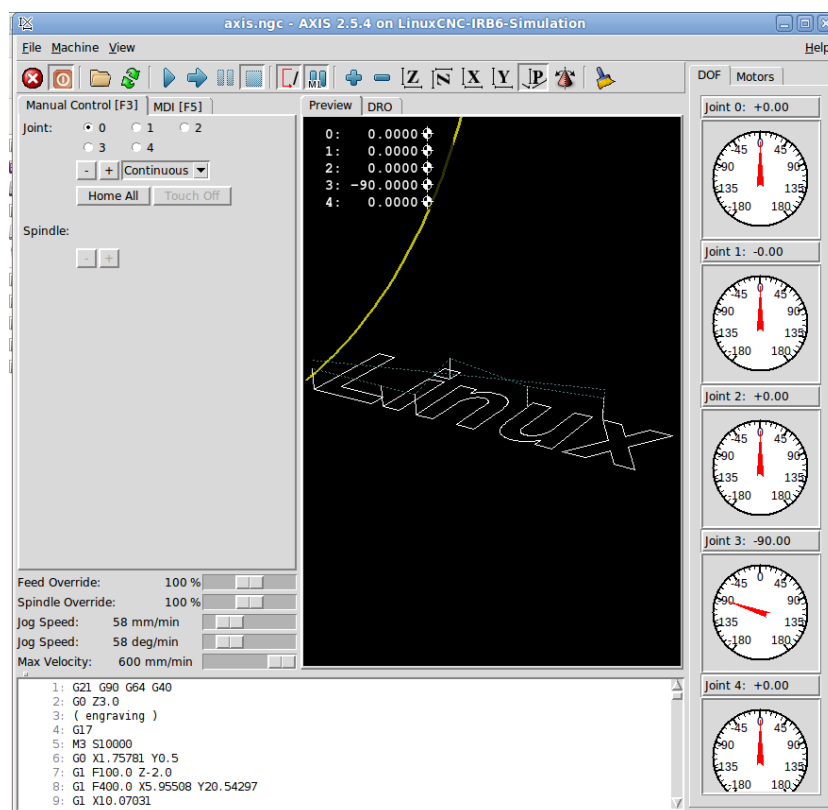


Figura 5.15: Simulação inicial da trajetória do efetuador

Após verificar que o controlador *LinuxCNC* conseguiu interpretar o programa NC é possível executar as 10 rotinas, cada uma sobre o mesmo plano de trabalho, evitando movimentar a folha de papel onde a caneta fez o determinado desenho. Foi usado o acelerômetro de um celular para diminuir a inclinação dos eixos X e Y, diminuindo o erro do *setup* no processo proposto para obter o erro aproximado de repetibilidade. Na Figura 5.16 apresenta a primeira e algumas outras repetições posteriores ao início deste estudo de caso.

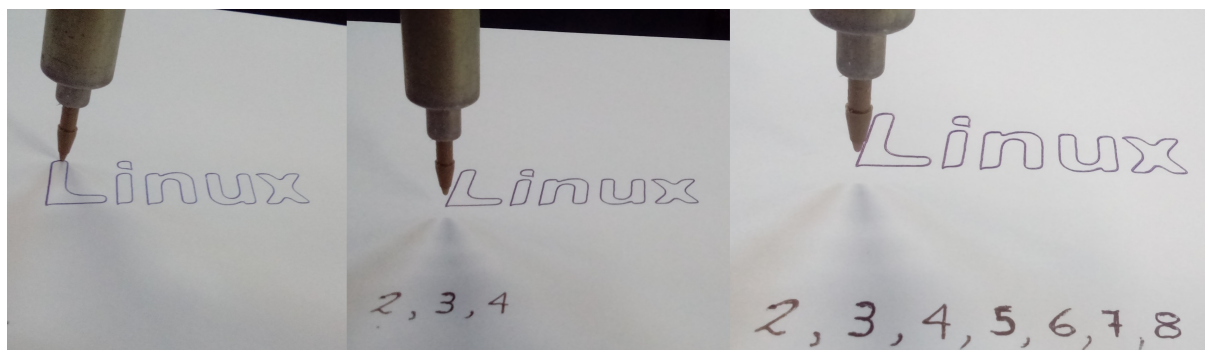


Figura 5.16: Registro digital da repetibilidade

Finalizando as dez repetições, é apresentado na Figura 5.17 o desenho feito pela caneta demonstrando-se sempre constante em qualquer uma das amostras feitas, garantindo que o robô pode fazer a mesma atividade programada n vezes, reduzindo a incerteza do processo no qual pode haver falhas na operação se as condições externas pudessem mudar durante o processo de marcação da folha de papel.

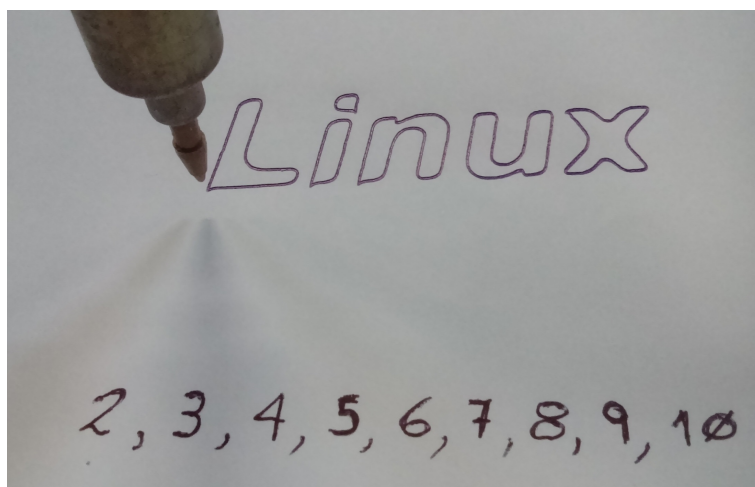


Figura 5.17: Resultado final da repetibilidade do programa NC

Para quantificar o erro na repetibilidade do processo com o robô ASEA foi feita uma última corrida na mesma folha de papel usada para os 10 procedimentos. A Figura 5.18 apresenta o resultado geral no momento de fazer a marcação das figuras desenhadas pelo robô.

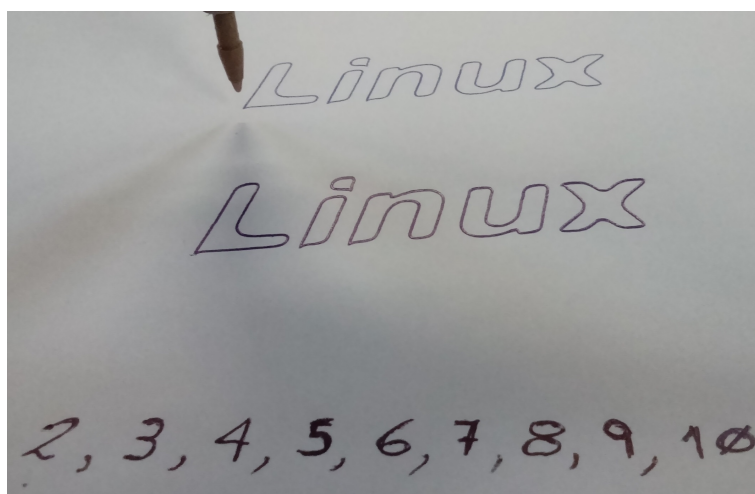


Figura 5.18: Resultado gráfico para medição das espessuras

Para estimar o erro médio das espessuras do traço feito nas 10 repetições em comparação a um traço feito com a mesma trajetória, foram medidos 50 pontos diferentes para cada um dos dois traços, levando em conta que para uma distribuição normal de dados é necessário ter mais de 30 amostras (WALPOLE; MYERS; MYERS, 1999), e assim definir um intervalo de confiança de 95% para este experimento de *repetibilidade*.

Com ajuda do medidor/projetor de perfil *Mitutoyo PJ-A3000* foi possível fazer as medições das espessuras do traço em pontos específicos de cada uma das letras desenhadas pela caneta. O projetor permite ter uma imagem em escala do traço permitindo obter uma boa estimativa do valor das espessuras. Na Figura 5.19 é possível observar o equipamento usado no momento da tomada das amostras na folha de papel que foi usado no experimento.



Figura 5.19: Projetor de perfil *Mitutoyo PJ-A3000*

O projetor de perfil usado para o experimento tem uma interface para facilitar a medição de coordenadas 2D de forma manual, sendo assim, foi usada a função chamada "*line to point*", a qual permite obter a distância entre um ponto e uma linha através da reta normal, neste caso, para achar a espessura do traço. Na Figura 5.20 é apresentada a função e a interface do usuário disposta à fazer as medições.

É possível observar na Tabela 5.1 os valores das espessuras em pontos diferentes do desenho feito em uma repetição, para cada uma das letras geradas através do programa NC. O medidor/projetor de perfil *Mitutoyo PJ-A3000* apresenta uma incerteza de medição de $\pm 0.15\%$ do valor obtido em cada amostra (MITUTOYO, 2011).

Tabela 5.1: Medição das espessuras do traço único

Letra	1 (mm)	2 (mm)	3 (mm)	4 (mm)	5 (mm)	6 (mm)	7 (mm)	8 (mm)	9 (mm)	10 (mm)
L	0.435	0.413	0.441	0.408	0.434	0.391	0.432	0.390	0.391	0.433
I	0.426	0.425	0.463	0.475	0.414	0.328	0.326	0.334	0.353	0.460
N	0.434	0.345	0.462	0.361	0.412	0.410	0.347	0.395	0.416	0.429
U	0.401	0.452	0.396	0.331	0.371	0.421	0.413	0.463	0.345	0.367
X	0.403	0.438	0.424	0.381	0.413	0.393	0.391	0.420	0.397	0.385

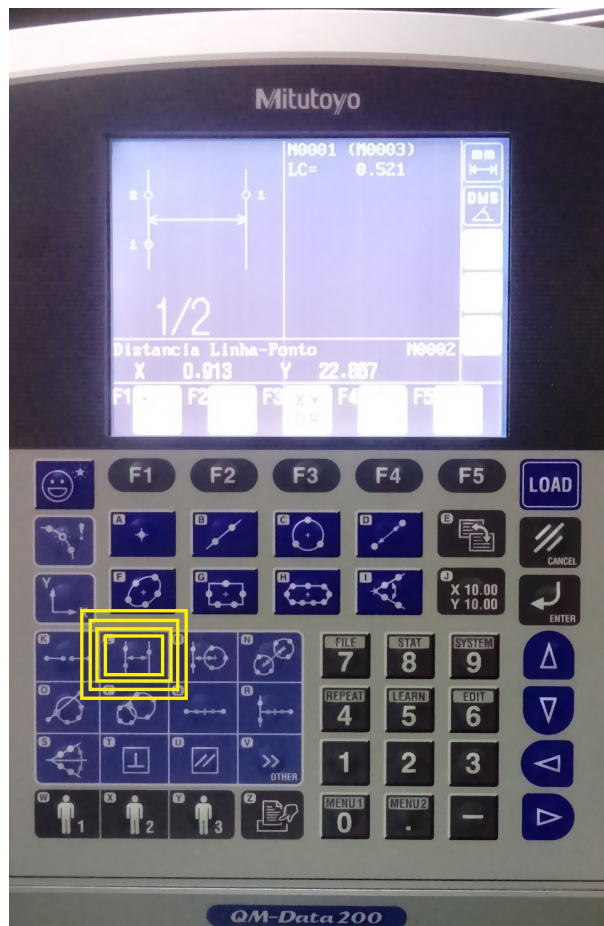


Figura 5.20: Opção "line to point" da interface de usuário do projetor de perfil

Para analisar as amostras obtidas na medição dos pontos, verifica-se o comportamento dos dados em uma curva normal, como mostra a Figura 5.21, observando dois tipos de resultados da qualidade do processo feito. Neste caso a repetição das marcas registradas na folha de papel. Essas análises levam em conta um grupos de dados, ou seja, grupos de amostras do mesmo experimento chamado *Overall Capability*. Entretanto, a seguinte análise denominada *Potential Capability* gera resultados da capacidade de fazer um processo com condições iniciais específicas de todos os dados coletados, sem importar os subgrupos.

Visando as análises do experimento desenvolvido pelo manipulador no presente estudo de caso, foi usado e analisado valores relacionados com os resultados que agrupam todos os dados como um único conjunto, ou seja, (*Potential Capability*). A Figura 5.21 foi gerada pelo software de estatística *Minitab17*.

Com a análise de capacidade feito pelo software de estatística *Minitab*, é possível saber se o processo desenvolvido pelo robô tem a capacidade de produzir falhas controladas, com os indicadores gerados pela análise, como C_p e C_{pk} . Estas duas variáveis podem ser usadas juntas para saber o quão capaz é o processo, além de validar que o processo seja centralizado (STAROVESKI et al., 2009).

C_p é a capacidade potencial do processo e define se o processo é capaz de cumprir especificações, enquanto o C_{pk} é o índice de capacidade, que permite conhecer se o processo está centrado segundo os limites ou requerimentos. Um processo com capacidade de produzir produtos nos limites de desenho, não

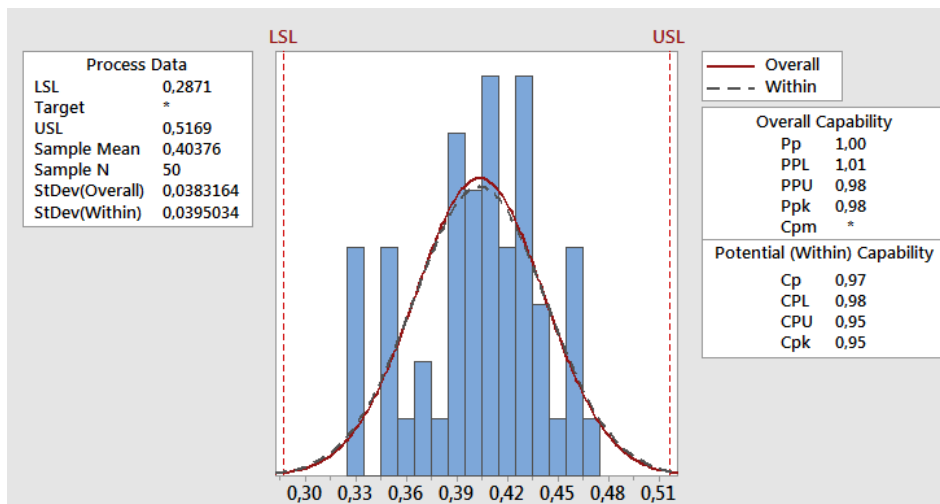


Figura 5.21: Curva normal das espessuras medidas do traço único

necessariamente, pode ser aceito pois os resultados podem não ser centralizados aos limites do processo, o que geraria alternativas para melhorar os resultados e garantir a capacidade do processo.

Se o indicador de capacidade C_p é medido, deve haver um valor igual ou superior a 1, sendo que em manufatura, frequentemente, o valor nominal deve ser igual ou maior que 1,33, para que o processo seja considerado estável (STERN, 2016). Desta forma com os resultados apresentados na Figura 5.21, o C_p tem um valor de 0,97, segundo os dados coletados, sendo possível inferir se o processo está perto de ser estável para um C_p igual a 1, além disso o indicador C_{pk} é de 0,95, indicando que o processo está centralizado segundo os limites de tolerância. Para melhorar os indicadores apresentados anteriormente é necessário implementar como trabalhos futuros uma estratégia de calibração mais robusta em relação a que foi proposta neste projeto.

Com os valores apresentados anteriormente é possível conhecer o valor médio das espessuras feitas no desenho de referência, assim como o desvio padrão (σ). Sendo que o valor médio ou média é o promédio numérico de um número determinado de dados em uma amostra, enquanto σ pode-se definir como a variação ou dispersão em relação a média (\bar{x}) (WALPOLE; MYERS; MYERS, 1999). O resultado da média para os valores das espessuras apresentadas na Tabela 5.1 está detalhado na Eq. 5.1, enquanto o desvio padrão segundo a análise de capacidade tem um valor de $\sigma_1 = 0.039 \text{ mm}$.

$$\bar{x}_1 = \frac{x_1 + x_2 + \dots + x_n}{n} = 0.403 \text{ mm} \quad (5.1)$$

A espessura de um único traço tem valor nominal de 0,403 mm, ou seja, este é o valor do traço esperado quando o efetuador do robô com a caneta executa uma marcação no papel. Este valor será usado como referência pra estimar o erro sistemático de múltiplos traços gerados pelo efetuador. Quanto melhor a precisão do posicionamento do Robô na marcação do traço sobre o traço já existente, melhor será a repetibilidade estimada pelo Robô. Assim este valor de traço será utilizado como valor nominal pra estimar a parcela de erro sistemático da avaliação de desempenho usando o conceito/métrica de repetibilidade.

Foi necessário medir em pontos parecidos em cada letra desenhada sobre o papel para conhecer a

espessura dos traços no experimento de *repetibilidade* com as 10 repetições do mesmo programa NC. Os valores são apresentados na Tabela 5.2.

Tabela 5.2: Medição das espessuras do traço com 10 repetições

Letra	1 (mm)	2 (mm)	3 (mm)	4 (mm)	5 (mm)	6 (mm)	7 (mm)	8 (mm)	9 (mm)	10 (mm)
L	0.517	0.493	0.503	0.632	0.621	0.478	0.643	0.456	0.504	0.739
I	0.508	0.502	0.491	0.518	0.595	0.501	0.493	0.475	0.516	0.663
N	0.507	0.508	0.502	0.518	0.593	0.516	0.479	0.423	0.446	0.640
U	0.465	0.498	0.595	0.464	0.448	0.511	0.511	0.607	0.472	0.473
X	0.549	0.616	0.627	0.565	0.526	0.509	0.522	0.620	0.552	0.515

Novamente o uso do *Minitab17* para obter a análise de capacidade e o gráfico dos valores normalizados obtidos na medição é indispensável, com o objetivo de continuar as análises dos resultados no presente estudo de caso, podendo-se observar na Figura 5.22:

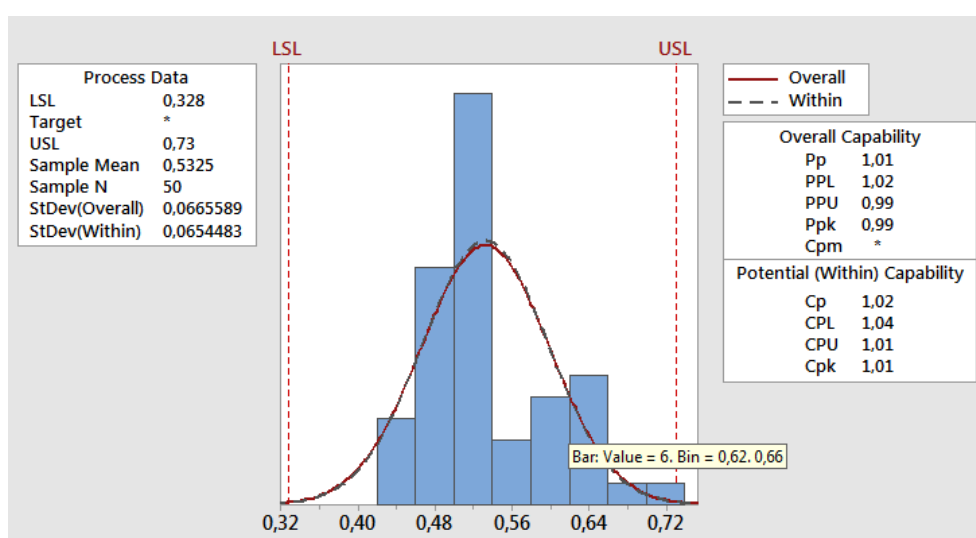


Figura 5.22: Curva normal das espessuras medidas com repetição

Da curva normal apresentada na Figura 5.22 é possível inferir que os dados coletados do traço feito em 10 repetições tem valores fora dos limites da função gaussiana, ou seja, dados afastados do valor da espessura do traço desejado quando o robô faz várias repetições do programa escolhido. Além disso, novamente como na análise de capacidade para um único traço os valores do limite superior e inferior de engenharia foram calculados para um C_p igual a 1, o que indica que o processo é estável. A Figura 5.22 apresenta o valor C_{pk} próximo a C_p , por tanto o processo é centralizado.

A partir da análise do Capabilidade Potencial (*Potencial Capability*), é possível conhecer o desvio padrão com um valor aproximado de $\sigma_2 = 0.065 \text{ mm}$, e uma média \bar{x}_2 para o experimento de *repetibilidade* apresentada na Equação 5.2:

$$\bar{x}_2 = \frac{x_1 + x_2 + \dots + x_n}{n} \quad (5.2)$$

$$\bar{x}_2 = 0.532 \text{ mm}$$

Com os valores da média \bar{x}_1 e \bar{x}_2 , e o desvio padrão σ_1 e σ_2 obtidos da análise de capacidade anteriormente mencionados é possível avaliar e estimar a repetibilidade da tarefa executada várias vezes em referência ao programa executado uma vez através da Eq. (5.3), para um intervalo de confiança igual a 95%.

$$\text{ErroSistemático}(\bar{x}_2, \bar{x}_1) = \bar{x}_2 - \bar{x}_1 = 0.129mm \quad (5.3a)$$

$$\sigma_{\text{repetibilidade}} = \sqrt{\sigma_2^2 + \sigma_1^2} = 0.076mm \quad (5.3b)$$

$$\text{Repetibilidade}(IC\text{ de }95\%) = \text{ErroSistemático}(\bar{x}_2, \bar{x}_1) \pm 2\sigma_{\text{repetibilidade}} \quad (5.3c)$$

$$\text{Repetibilidade} = 0.129 \pm 0.151 \quad (5.3d)$$

$$\text{Repetibilidade}_{\min} = -0.02mm \quad (5.3e)$$

$$\text{Repetibilidade}_{\max} = 0.28mm \quad (5.3f)$$

O valor negativo de estimação de repetibilidade não faz sentido físico, pois não há traço negativo da linha. Assim sendo, o valor estimado para a repetibilidade avaliada em relação ao traço de uma única linha é de 0,28 mm, ou seja, quando vários posicionamentos são repetidos n vezes, o erro associado à repetibilidade do efetuador do robô que executa a marcação do traço é de 0,28 mm, mostrando um grau de precisão/variabilidade considerável, comparado a manipuladores comerciais novos e modernos como KUKA KR6/2 que é de 0,1 mm (ABREU, 2002) e o ABB 6660 que tem repetibilidade de 0,07 mm (OLIVEIRA, 2013).

Considerando tratar-se de um robô de 1975 modernizado com um valor de repetibilidade original de 0.2 mm (ASEA, 2003), o valor estimado para repetibilidade surpreende positivamente, pois segundo a ABB seus Robôs evoluíram significativamente desde de 1974 (Asea IRB6) até os dias atuais, quando apresentavam uma precisão de posicionamento de 1 mm e hoje está na ordem de 10 microns (ABB, 2014).

O vídeo da execução do presente estudo de caso esta disponível para consulta em <<https://youtu.be/ttGI1hPhhWQ>>.

5.5 Ensaio Geométrico com Peça Padrão Moldura: Erro de Retilidade e Erro de Perpendicularidade

Geometricamente é possível analisar os resultados do manipulador em uma determinada tarefa visando um feedback para melhorar o desempenho em futuros testes ou aplicações específicas. Desta forma, na presente seção verifica-se através de um programa NC executado pelo manipulador o parâmetro geométrico denominado retilidade para um plano de trabalho 2D, baseado em uma peça moldura definida para avaliar o parâmetro nos eixos X e Y do robô IRB6-S2. A retilidade pode ser definida como uma representação qualitativa de uma superfície em termos de variação/desvio da sua geometria fazendo referência a uma linha reta pré definida (BEWOOR, 2009). O desenho feito por uma caneta fixada na última junta do manipulador pode-ser observado na Figura 5.23.

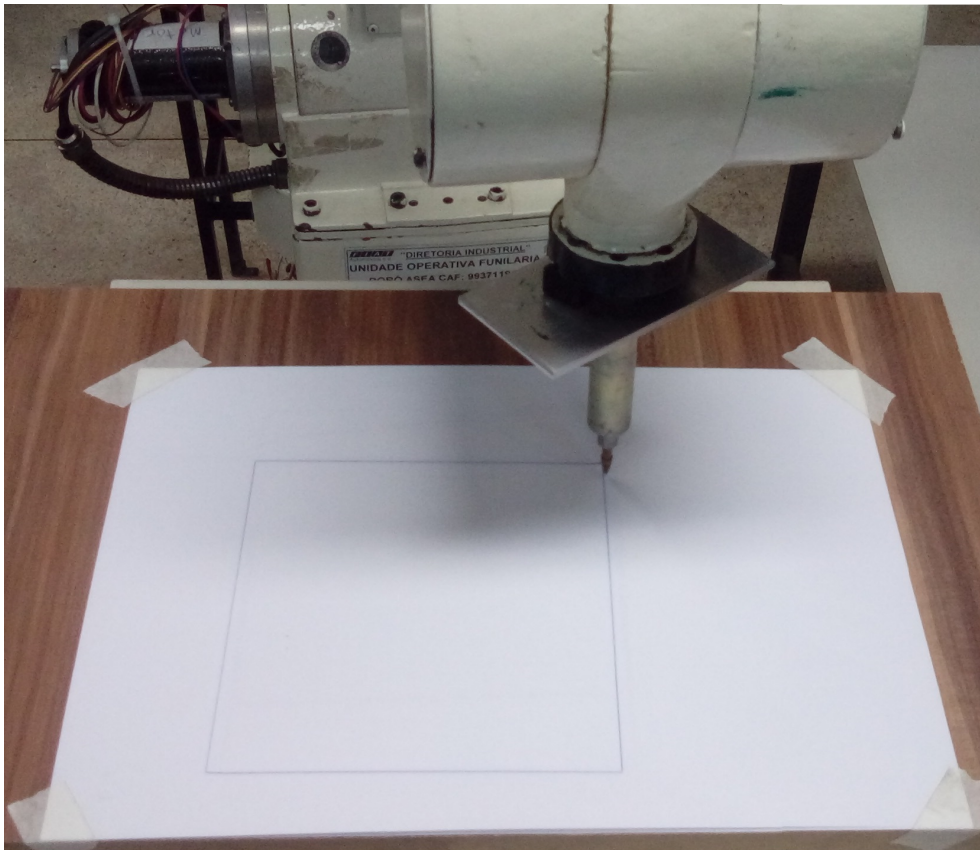


Figura 5.23: Marcação da caneta na área de trabalho

Com o quadro desenhado em uma folha de papel pela caneta fixada na última junta do robô, é possível fazer medições do comprimento em diferentes posições em referência aos eixos X e Y. As amostras obtidas foram medidas com um calibrador digital *SHAN*, com uma precisão de $\pm 0.03\text{mm}$. O instrumento usado pode ser observado na Figura 5.24.



Figura 5.24: Paquímetro digital usado para as medições

Foram definidos 14 pontos de referência para cada eixo (X e Y) e desta forma obter uma boa precisão das amostras medidas. A medição de cada comprimento foi uma média de 10 medições para cada um dos pontos, com uma aproximação ao ponto do meio do traço feito pela caneta, garantindo uniformidade do conjunto de valores. É possível observar na Figura 5.25 como foi feita a distribuição dos pontos de referência para medir cada um dos comprimentos de cada eixo.

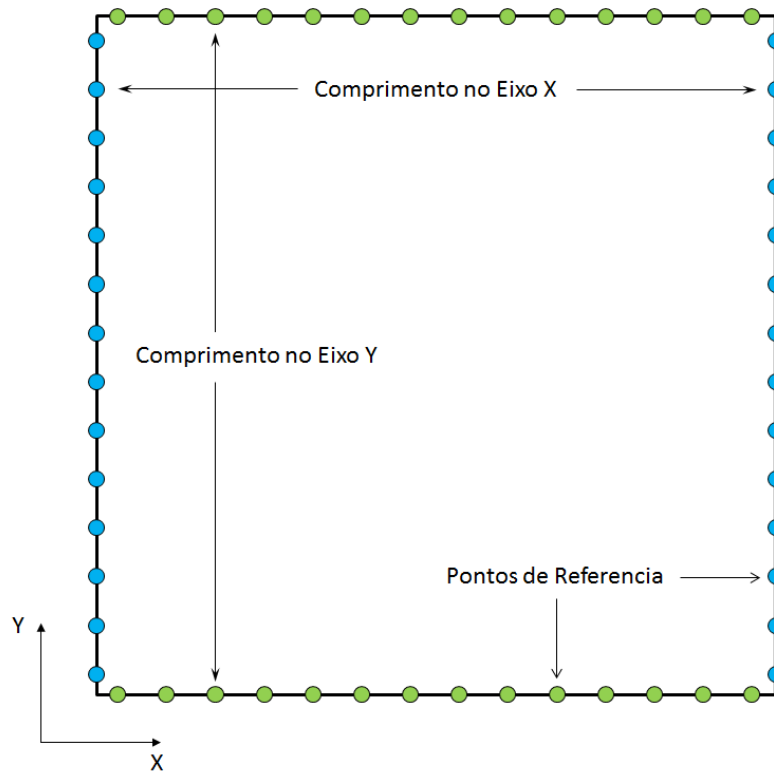


Figura 5.25: Pontos de referências para as medições

Para gerar um gráfico de regressão linear dos dados obtidos na medição do comprimento ao longo dos eixos X e Y do quadro desenhado pelo robô, é necessário calcular o erro de cada um dos valores médios, medidos com o calibrador digital. É apresentada a Eq. 5.4, a qual permite obter o valor do erro para cada amostra. O valor nominal programado no arquivo NC foi de 140 mm.

$$Erro_{x,y} = ValorNominal_{x,y} - ValorMedido_{x,y} \quad (5.4)$$

Com os valores obtidos de cada uma das medições obtidas com o calibrador digital, foi possível calcular os erros de retlineidade para os eixos X e Y. Na Tabela 5.3 são apresentados os valores para cada um dos eixos analisados no presente experimento, assim como os erros fazendo referência ao Valor Nominal projetado para a movimentação do efetuador do robô.

Com os valores dos erros foi possível fazer a análise da regressão linear de cada um dos eixos X e Y, permitindo conhecer uma aproximação da magnitude do erro de retlineidade que atualmente tem o manipulador. Na Figura 5.26 é apresentada a linha de regressão para os valores correspondentes ao eixo X.

Segundo os resultados da Figura 5.26, é possível observar que os valores das posições medidas estão perto da linha de regressão, ou seja, que o Erro Padrão da Regressão (S) tem um valor pequeno, representando desta forma a distância promédia dos valores observados em referência à linha de regressão. Enquanto o valor do coeficiente de determinação, R^2 , com um valor de 95.9% (perto de 100%), indica que os dados se ajustam ao modelo de regressão linear calculado através dos dados obtidos (SCHUMACKER, 2014).

Tabela 5.3: Medição do comprimento ao longo dos eixos X e Y

Posição	X (mm)	Y (mm)	Ex (mm)	Ey (mm)
1	140,86	140,04	-0,86	-0,04
2	140,81	140,11	-0,81	-0,11
3	140,79	140,12	-0,79	-0,12
4	140,74	140,19	-0,74	-0,19
5	140,66	140,20	-0,66	-0,20
6	140,54	140,21	-0,54	-0,21
7	140,36	140,23	-0,36	-0,23
8	140,30	140,26	-0,30	-0,26
9	140,16	140,37	-0,16	-0,37
10	140,08	140,39	-0,08	-0,39
11	140,07	140,41	-0,07	-0,41
12	139,98	140,42	0,02	-0,42
13	139,97	140,42	0,03	-0,42
14	139,97	140,43	0,03	-0,43

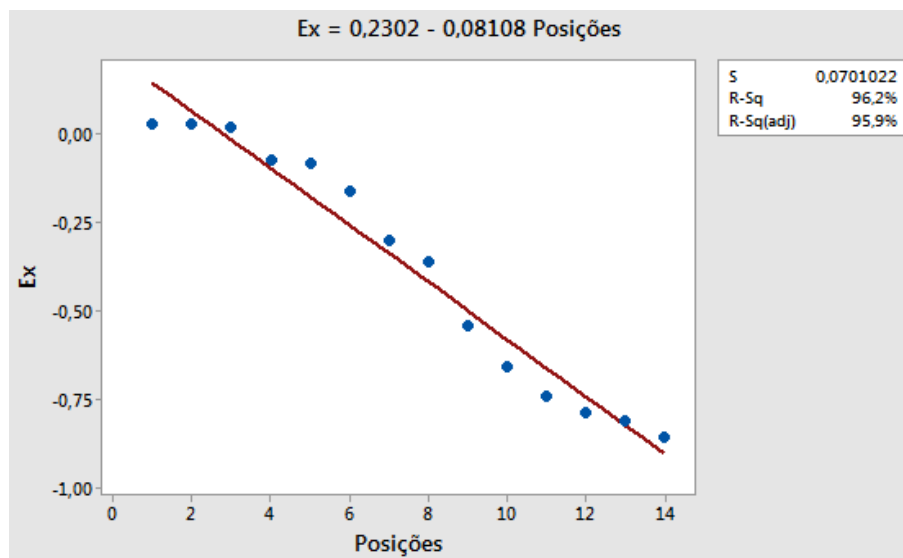


Figura 5.26: Regressão linear do erro de retilidade em X

Na Figura 5.27 apresenta-se a análise de regressão para o eixo Y, com base nas amostras coletadas no desenho feito pelo manipulador na folha de papel.

Em comparação com o resultado do eixo X, na Figura 5.27 apresenta pouca dispersão dos dados em referência ao modelo de regressão linear, o que se traduz um valor de 95,2% para R^2 . Por outro lado, os dados apresentam pouca distância em relação à linha de regressão, justificando esta observação desde o valor de (S), menor ao valor calculado para o eixo X.

Na Eq. 5.5 é possível observar as equações das linhas de regressão para cada um dos eixos analisados na presente seção.

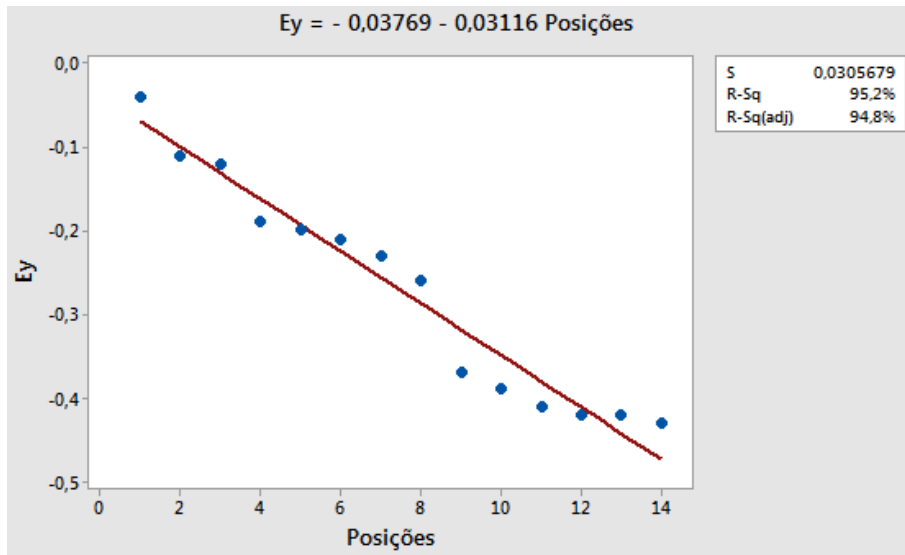


Figura 5.27: Regressão lineal do erro de reticidade em Y

$$E_x = -0.08p + 0.2302 \quad (5.5a)$$

$$E_y = -0.03p + 0.0376 \quad (5.5b)$$

Com as equações das retas de regressão é possível achar os valores máximos de erro de reticidade para os eixos X e Y, com base no cálculo de erro de resíduo, o qual é possível observar na Eq. 5.6 como é possível calcular os valores máximo segundo os resíduos ou distância entre o valor nominal e o valor da reta de regressão (BRUCE, 2015).

$$E_{x-máx} = |X_i - \bar{X}| = 0.119 \quad (5.6a)$$

$$E_{y-máx} = |Y_i - \bar{Y}| = 0.361 \quad (5.6b)$$

O grau de erro de reticidade é aceitável para um robô que foi desenvolvido na década de 80, em comparação a robôs modernos e com modelos de calibração avançados como o ABB IRB140 que apresenta um erro aproximado de reticidade de 0.115 mm (PAZIANI; GIACOMO; TSUNAKI, 2009), ou o manipulador IRB 6400 com valores que não excedem os 0.5 mm (YOUNG; PICKIN, 2000).

Com as equações de regressão apresentadas anteriormente (5.5), é possível conhecer a inclinação das retas de regressão em referência aos eixos X e Y, segundo o desempenho do manipulador no momento de fazer a peça moldura. Sendo m o coeficiente angular da reta, a Eq. 5.7 pode estimar o valor da inclinação da equação de regressão.

$$Y = mX + b \quad (5.7)$$

Com os valores $m_x = 0.08$ e $m_y = 0.03$ é possível estimar que existe baixa inclinação das retas de regressão em referência aos eixos X e Y respectivamente, segundo o traço gerado na folha de papel da peça moldura planejada no programa NC.

Com as equações das linhas em relação ao referencial cartesiano de abcissas e ordenadas, é possível estimar se as duas retas são perpendiculares, para isso, é necessário calcular o ângulo entre as linhas, ou seja, devem ter um ponto de interseção, o que na literatura é denominada como concorrência (RYAN, 1991). Para que duas retas sejam perpendiculares deve-se cumprir com a condição que o ângulo formado pelas duas linhas sejam igual a 90° (RYAN, 1991). É necessário trocar na equação de regressão do erro em Y (E_y) os valores das coordenadas próprias da abscissa e ordenada, para que a referência seja o eixo vertical e não o horizontal (ROHDE et al., 2012). A continuação na Figura 5.28 é apresentada a regressão para E_y com os valores trocados entre os eixos de referência.

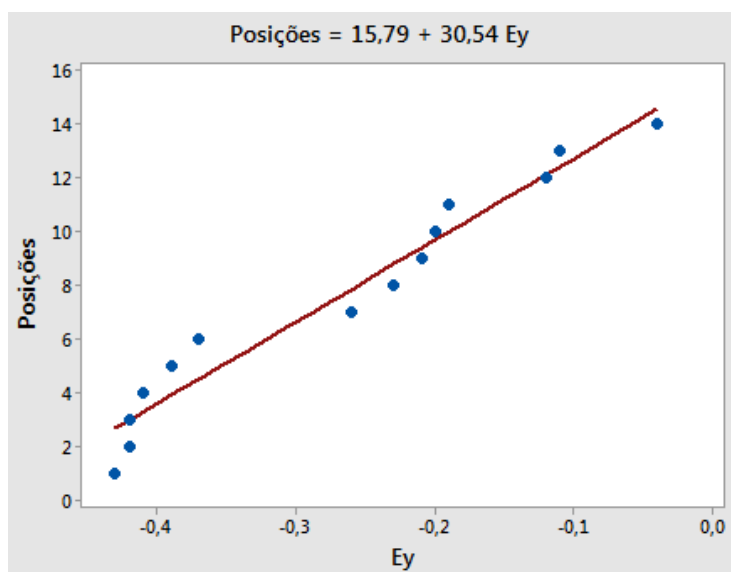


Figura 5.28: Regressão linear do erro em Y com eixos trocados

Para validar se as duas linhas que correspondem aos eixos X e Y do programa NC executado pelo manipulador IRB6-S2 são perpendiculares, é necessário obter o ângulo θ que forma as duas retas matematicamente descritas na Eq. 5.5. Para obter o ângulo é usada a Eq. 5.8 junto com coeficiente m_x e m_y de cada uma das retas de regressão descritas anteriormente (LARSON, 2010).

$$\tan \theta = \left| \frac{m_2 - m_1}{1 + m_1 \cdot m_2} \right| \quad (5.8)$$

Segundo os valores dos coeficientes das retas de regressão, $m_x = -0.08108$ e $m_y = 30.54$, é possível calcular o ângulo entre as retas como é apresentado na Eq. 5.9.

$$\theta = 87.24^\circ \quad (5.9)$$

Desta forma pode-se assumir que os eixos X e Y na execução do programa NC através do robô IRB6-S2 têm um erro de perpendicularidade, definido como a diferença entre o desvio das linhas de regressão é

90°, obtendo um valor de erro aproximado de 2.76°, assumindo assim a condição de perpendicularidade existente para a peça moldura desenhada para o presente experimento (STONE, 2012). Para o caso do manipulador ASEA IRB6-S2 experimentalmente o erro de perpendicularidade permite conhecer o ângulo máximo de erro conhecido com um valor aproximado menor a 3°, em relação a uma linha reta nos eixos de referência X e Y segundo o programa NC executado.

O vídeo da movimentação do robô para executar o programa NC da peça moldura está disponível para consulta no <<https://youtu.be/ttGI1hPhhWQ>>.

5.6 Análise dos Resultados Obtidos

Segundo os resultados obtidos em cada um dos experimentos é possível identificar as causas que afetam o desempenho do robô após do retrofitting nas tarefas feitas e que permitiram obter os valores para as variáveis de desempenho identificadas ao longo do presente capítulo. A síntese dos resultados é apresentada na Tabela 5.4.

Tabela 5.4: Síntese dos resultados obtidos

Parâmetro	Erro
Programação via <i>teach-in</i>	Foi gerado e executado um programa NC por meio da técnica de programação on-line <i>teach-in</i>
Programação via CAD/CAM	Foi executado um programa NC exemplo gerado através de CAD/CAM
Repetibilidade	Foi obtido um erro de 0.28mm após o <i>retrofitting</i>
Retilidade	Foi estimado um erro de retilidade para o eixo X de 0.119 e para Y de 0.361 mm.
Perpendicularidade	O erro de perpendicularidade estimado foi de 2.76°

É possível fazer uma programação *off-line* do robô desde a metodologia de aprendizagem de máquinas *teach-in*, obtendo pontos desejados do elemento final do manipulador comandado em uma interface tipo *pendant*, permitindo que o operador possa obter um posicionamento exato devido à possibilidade de observar de perto os pontos específicos para o deslocamento do manipulador. O robô tem capacidade de se movimentar segundo o programa NC, gerado através do módulo *teach-in* do controlador *LinuxCNC* aos pontos capturados na programação *on-line*.

Baseado em um programa NC gerado por CAD/CAM e interpretado pelo controlador *LinuxCNC*, foi possível estimar que a repetibilidade do manipulador depois de fazer mais de 10 traços da mesma trajetória, consegue ter um erro de repetibilidade aproximado a 0.28 mm em relação com o erro original de 0.2 mm (DUYSINX; GERADIN, 2004). Sendo um valor aceitável para um robô que foi construído há mais de 40 anos e que foi operado de forma contínua em uma fábrica automotiva da *FIAT* antes de ser doado à Universidade de Brasília.

Originalmente o manipulador ASEA IRB6-S2 apresenta um erro de repetibilidade de 0.2 mm (DUYSINX; GERADIN, 2004) próximo ao valor experimental obtido de 0.28 mm, permitindo validar após o *retrofitting* uma comparação com um dos valores disponíveis na literatura do desempenho do robô, em relação aos robôs modernos com um erro de repetibilidade de 0.1 mm, visando diminuir o erro aproximado

de repetibilidade com técnicas de calibração focadas no aprimoramento de parâmetros específicos do desempenho do manipulador. Com o erro de repetibilidade obtido no presente trabalho é garantido que o manipulador possua a capacidade de fazer a mesma tarefa várias vezes com uma variável de desempenho estimada, com um erro de posicionamento conhecido e aceitável.

Foi possível identificar variáveis que afetam o desempenho do robô no experimento proposto de repetibilidade, como por exemplo a instabilidade da mesa com a folha de papel onde o robô fez a marcação em varias opções; assim como a geração de espessuras do taco longe da média calculada quando o robô atingia uma curva em cada uma das geometrias das letras planejadas no programa NC. É possível melhorar os resultados, por exemplo aprimorando o *setup* do experimento, poderia ter um número de amostras de no mínimo 500 pontos para cada uma das repetições, conforme a norma ISO 9283, segundo os critérios de desempenho e métodos de testes correlatos para manipuladores industriais (SIRINTERLIKCI et al., 2009).

Mediante as análises geométricas foi possível conhecer o erro de retilineidade entre a distância comandada e a distância executada pelo manipulador ao longo dos eixos X e Y. O erro de retilineidade ficou entre 0.2 e 0.4 milímetros aproximadamente, garantindo desta forma que o robô IRB6-S2 após o procedimento de *retrofitting* estivesse em capacidade de fazer linhas retas com um erro controlado e relativamente baixo em comparação aos robôs industriais modernos e que tiveram métodos de calibração avançados, os quais não são parte do escopo do presente trabalho mas representam a base para futuros desenvolvimentos.

Outro parâmetros avaliados no manipulador após o processo de *retrofitting* foi a perpendicularidade entre duas retas desenhadas com a caneta, obtendo um erro baixo de aproximadamente 3 % em comparação com os 90° graus que deveria ter idealmente duas retas com este tipo de variável de desempenho avaliada. O valor serve como referencia para diminuir o erro de perpendicularidade com instrumentos de medição e técnicas específicas em trabalhos futuros, devido a que não faz parte do escopo do projeto.

Variáveis que afetavam este parâmetro de desempenho foram identificadas tais como a instabilidade da área de trabalho onde o robô fez a marcação e as limitações mecânicas do manipulador e que devido a movimentação das juntas serem mecanicamente dependentes entre si, permitindo compensar com valores aproximados o movimento das juntas de forma manual diretamente nas equações homogêneas do controlador *LinuxCNC*.

Capítulo 6

Conclusões

Este capítulo apresenta as conclusões associadas ao trabalho realizado com o manipulador ASEA IRB6-S2, assim como sugestões para trabalhos futuros que possam complementar a solução proposta baseada em uma arquitetura aberta de controle para manipuladores industriais.

6.1 Conclusões do Trabalho

No trabalho apresentado foi proposta uma metodologia para a implementação da técnica *retrofitting* em manipuladores industriais, e que foi validada usando o manipulador ASEA IRB6-S2 para integrar o robô a uma arquitetura aberta de controle baseada na plataforma LinuxCNC. Foram projetados e fabricados os diferentes componentes dos subsistemas de controle, alimentação e os conjuntos de juntas, de acordo com o diagnóstico realizado do estado original do manipulador. Permitindo a instalação de uma nova arquitetura de controle aberta compatível com LinuxCNC, aproveitando a estrutura mecânica e o gabinete original do manipulador.

Com a atualização da parte *Software* do sistema de controle foi integrado e configurado o controlador LinuxCNC com as equações cinemáticas próprias do manipulador ASEA IRB6-S2. Permitindo identificar e definir de forma detalhada os passos necessários para incluir qualquer tipo de modelo matemático característico de um manipulador dentro da plataforma.

Foram identificados dois arquivos indispensáveis para a configuração do projeto desenvolvido em LinuxCNC, como são o arquivo HAL e INI. O primeiro foi configurado para interconectar de forma virtual os sinais de controle do manipulador tais como os pulsos enviados para os *drive* dos motores; os sensores de posicionamento das juntas e o acionamento da garra pneumática adaptada como efetuator. Enquanto o arquivo INI é o encarregado em armazenar as configurações do projeto associado com o manipulador, permitindo determinar o número e tipo de graus de liberdade para controlar o que permite estabelecer a distinção de uma máquina CNC.

A validação da implementação da metodologia proposta foi feita com a execução de programas NC interpretados pela plataforma LinuxCNC. O primeiro foi realizado através da programação *teach-in* onde foram gravadas posições determinadas mediante a movimentação manual do efetuator final desde o Li-

nuxCNC. As coordenadas cartesianas gravadas para cada ponto no espaço de trabalho permitiu gerar um arquivo compatível com a norma RS-274 o qual foi executado em modo *off-line* para que o manipulador movimentasse a peça de um ponto a outro.

O segundo método de validação para validar o *retrofitting* foi mediante a execução de um programa exemplo NC gerado através de CAD/CAM dez vezes com uma caneta adaptada como efetuator. O programa NC foi executado dez vezes para obter um erro de repetibilidade aproximado de 0.28 mm, que se comparando com os 0.2 mm do valor original especificado pelo fornecedor do manipulador ASEA IRB6-S2. Em consequência, é fatível afirmar que é possível obter um maior ajuste da variável de desempenho de repetibilidade com a nova arquitetura de controle aberta funcional de um robô de mais de quarenta anos de serviço, e mais de quatro anos inativo.

Com uma peça moldura em um plano de trabalho 2D foram obtidos os valores estimados dos erros de retilineidade e perpendicularidade para os eixos X e Y da trajetória do efetuator. O erro estimado de retilineidade foi entre 0.2 e 0.4 milímetros, o que permite afirmar que o manipulador após o *retrofitting* esta em capacidade de executar trajetórias em linha reta com um erro conhecido e relativamente baixo dependendo da aplicação desejada.

Enquanto a perpendicularidade foi estimado um erro de 3% em relação aos 90° como valor ideal entre os eixos X e Y. Estes valores devem ser considerados no momento de aplicar-as técnicas de calibração específicas segundo as tarefas do manipulador, levando em consideração a calibração feita for focada na relação entre os pulsos gerados pelo controlador para cada junta do manipulador.

Com a plataforma LinuxCNC foi possível validar com o modelo virtual do robô desenvolvido em *Python* a simulação da trajetória interpretada pelo controlador mediante a um programa NC, gerando a movimentação do efetuator com a cinemática direta e inversa vinculadas diretamente no controlador LinuxCNC. Desta forma foi possível conhecer a resposta da solução gerada sem a execução de testes no robô físico, o que evita possíveis falhas na estrutura mecânica do manipulador ou nos subsistemas de controle e alimentação. Uma das limitações da simulação é a falta de associação entre a relação mecânica projetada pelo fornecedor que integra as juntas dois e três por uma parte, assim como as juntas quatro e cinco.

Para a implementação do *retrofitting* em manipuladores robóticos e para a construção de sistemas de controle industriais a tendência é o uso de arquiteturas tipo *open source*, com o objetivo de gerar soluções industriais não proprietárias, o que torna os custos de investimento no *retrofitting* mais acessíveis em comparação a soluções integradas por componentes *software* e *hardware* licenciados e com arquiteturas de controle fechadas. Em consequência a plataforma LinuxCNC se torna uma solução viável pois precisa de poucos recursos *hardware* para seu funcionamento, graças as características do módulo HAL integrado na plataforma. Além de redução de custos financeiros dado que o LinuxCNC usa a licença tipo GNU-GPL.

A plataforma LinuxCNC é recomendada para aplicações industriais, onde não é necessário a aquisição de licenças ou precise de um sistema com baixa susceptibilidade a falhas ou travamento do sistema operacional, além da compatibilidade com resposta da comunicação em tempo real. É uma solução completa com a integração das equações cinemáticas de qualquer tipo de manipulador industrial, além de controlar e calcular a orientação de cada junta a partir de programas NC gerados por um sistema CAD/CAM, em um mesmo ambiente de desenvolvimento.

Foi disponibilizada uma solução baseada em uma arquitetura aberta de controle para a integração do manipulador ASEA IRB6-S2 com qualquer tipo de sistema industrial, no caso do presente projeto para a integração com uma célula de manufatura flexível (FMC) localizada no laboratório GRACO da Universidade de Brasília, onde o robô é responsável pela movimentação de peças para usinagem entre as diferentes estações de trabalho presentes na célula.

A aplicação da técnica *retrofitting* através da metodologia proposta para manipuladores industriais permitiu observar que os custos do projeto representam aproximadamente 10% do valor de um manipulador novo com características semelhantes ao robô ASEA IRB6-S2, ou seja com capacidade para manipular até 6 kg, justificando economicamente o uso da solução proposta para aproveitar todas as capacidades dos manipuladores em aplicações industriais ou acadêmicas, diferentes das concebidas no momento de serem projetados pelos fornecedores de soluções proprietárias em automação industrial como KUKA ou ABB.

A modernização do manipulador possibilita o desenvolvimento de novos projetos que garantam a consolidação do uso de arquiteturas abertas de controle ao invés de sistemas proprietários que limitem as aplicações e a integração com novos componentes de controle que melhoram o desempenho dos manipuladores, obtendo uma ampla faixa de aplicações industriais, evitando a dependência com os fornecedores de tecnologia para a modificação da arquitetura licenciada na parte *software e hardware*.

Os resultados e o progresso detalhado da implementação da técnica *retrofitting*, como os documentos técnicos dos projetos gerados para os subsistemas de controle, alimentação e os componentes escolhidos para os conjuntos das juntas, permitindo que o manipulador ASEA IRB6-S2 voltasse novamente a ser operacional, foram disponibilizados para quem queira contribuir, usar e desenvolver novas aplicações baseados nos resultados obtidos no presente trabalho, incentivando a consolidação dinâmica de uma arquitetura aberta de controle para manipuladores industriais. A informação do projeto esta disponível em <<https://plus.google.com/116960304375198732734>>.

A distribuição do LinuxCNC customizada pra controlar o manipulador ASEA IRB6-S2 está disponível em <<https://sourceforge.net/projects/linuxcnc-robot-asea-irb6s2/>>, onde está disponibilizada a distribuição tipo imagem (ArquivoImagemLinuxCnCRoboAseaIRB6-S2), assim como toda a informação associada com o desenvolvimento do *retrofitting*. Desta forma é possível herdar a configuração por qualquer pessoa interessada no trabalho realizado com o manipulador ASEA IRB6-S2, contribuindo desta forma na consolidação de uma arquitetura aberta de controle especifica para manipuladores industriais baseada em comando numérico.

6.2 Sugestões para Trabalhos Futuros

A seguir são apresentados algumas sugestões para trabalhos futuros a serem desenvolvidos para melhorar a solução da arquitetura de controle proposta usando LinuxCNC:

- Implementar os controladores PID via software mediante o LinuxCNC, possibilitando a comparação do desempenho do robô com os controladores PID dos *drives* implementados para cada motor. Este tipo de controle via software é uma das características não exploradas no presente trabalho, mas que deve ser levada em conta como uma alternativa para a sintonização dos controladores usados.

- É possível gerar novas equações cinemáticas através do método geométrico e implementá-las como solução alternativa ao que foi mencionado no presente trabalho. Desta forma é possível comparar os erros de repetibilidade, retilineidade e perpendicularidade com o objetivo de conhecer o desempenho do manipulador com outra concepção para solucionar o problema da cinemática inversa.
- Com o manipulador funcional deve-se implementar e comparar estratégias de calibração que permitam reduzir os erros indicados no presente trabalho, aproveitando a arquitetura aberta proposta baseada no controlador LinuxCNC.
- Implementar um servidor baseado no padrão *MtConnect* integrado ao controlador LinuxCNC para monitorar os parâmetros do manipulador ASEA IRB6-S2, e tornar-os dados disponíveis em plataformas não proprietárias e interoperáveis.
- Controlar o posicionamento do manipulador através do processamento de imagens 3D usando o sensor *Kinect* através da programação *teach-in*, permitindo gravar o posicionamento final do atuador ou a configuração das juntas para depois reproduzir o movimento mediante um programa NC com as trajetórias definidas.
- Integrar a nova arquitetura aberta de controle do manipulador com as FMC de Usinagem e Montagem presentes no laboratório GRACO da Universidade de Brasília, usando o padrão *MtConnect* e OPC para modelar uma máquina de estados, eventos discretos, ao invés de trabalhar com sinais digitais (I/O) de 24 volts entre os CLP dos equipamentos das FMC de Usinagem e Montagem.

REFERÊNCIAS BIBLIOGRÁFICAS

ABB. *A century of ABB Review*. [S.l.], 2014. 80 p.

ABREU, P. *Robótica Industrial: Especificação de robôs e células robotizadas*. 2002. Slides.

ASADA, H.; KANADE, T.; TAKEYAMA, I. Control of a direct-drive arm. *Journal of Dynamic Systems, Measurement, and Control*, American Society of Mechanical Engineers, v. 105, n. 3, p. 136–142, 1983.

ASATO, O. et al. Analysis of open cnc architecture for machine tools. *Journal of the Brazilian Society of Mechanical Sciences*, SciELO Brasil, v. 24, n. 3, p. 208–212, 2002.

ASEA. *Manual ASEA*. [S.l.], 2003.

BARRIENTOS, A. *Fundamentos de Robótica*. Segunda. [S.l.]: McGraw-Hill, 2007.

BECERRA, V. M. et al. Hardware retrofit and computed torque control of a puma 560 robot updating an industrial manipulator. *IEEE control systems*, IEEE, v. 24, n. 5, p. 78–82, 2004.

BEWOOR, A. K. *Metrology & Measurement*. [S.l.]: McGraw-Hill Education (India) Pvt Limited, 2009. 558 p. ISBN 9780070140004.

BOMFIM, M. et al. A low cost methodology applied to remanufacturing of robotic manipulators. *Congresso Brasileiro de Autom?tica*, v. 20, p. 1506–1513, September 2014.

BOMFIM, M. H. S. *Remanufatura de Manipuladores Robóticos com Arquitetura Aberta*. Dissertação (Mestrado), jan. 2013.

BOSTICK, W. *Energy Storage, Compression, and Switching*. [S.l.]: Springer US, 2013. ISBN 9781468422146.

BRETTEL, M. et al. How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective. *International Journal of Mechanical, Industrial Science and Engineering*, v. 8, n. 1, p. 37–44, 2014.

BRUCE, P. C. *Introductory Statistics and Analytics: A Resampling Perspective*. [S.l.]: Wiley, 2015. 312 p. ISBN 9781118881330.

CANGELOSI, A.; SCHLESINGER, M.; SMITH, L. B. *Developmental robotics: From babies to robots*. [S.l.]: MIT Press, 2015. ISBN 9780262028011.

CHUKWUEKWE, D. O. et al. Reliable, robust and resilient systems: Towards development of a predictive maintenance concept within the industry 4.0 environment. In: *EFNMS Euro Maintenance Conference*. [S.l.: s.n.], 2016.

CRAIG, J. *Introduction to Robotics: Mechanics and Control*. [S.l.]: Pearson/Prentice Hall, 2005. (Addison-Wesley series in electrical and computer engineering: control engineering). ISBN 9780201543612.

- CUI INC. *Modular Incremental Encoder*: Amt10. 20050 SW 112th Ave. Tualatin, OR 97062, 2014.
- DIETRICH, F. et al. *An Autonomous and Safe Homing Strategy for Parallel Kinematic Five-Bar Manipulators*. Dordrecht: Springer Netherlands, 2010. 501–508 p. ISBN 978-90-481-9262-5.
- DUYSINX, P.; GERADIN, M. *An introduction to robotics: mechanical aspects*. 2004.
- EBENHOFER, G. et al. A system integration approach for service-oriented robotics. In: IEEE. *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. [S.l.], 2013. p. 1–8.
- ELMARAGHY, H.; MONOSTORI, L. Variety management in manufacturing cyber-physical production systems: Roots, expectations and r & d challenges. *Procedia CIRP*, v. 17, p. 9 – 13, 2014. ISSN 2212-8271.
- EMMANUEL, A. C.; INYIAMA, H. A survey of controller design methods for a robot manipulator in harsh environments. *European Journal of Engineering and Technology Vol*, v. 3, n. 3, 2015.
- FESTO. *Solenoid coils*: Msfg-24/42-50/60-od. [S.l.], 2014. 1 p.
- FUJITA, S.; YOSHIDA, T. Ose: open system environment for controller. In: *7th International Machine Tool Engineers Conference*. [S.l.: s.n.], 1996. p. 234–243.
- GECKODRIVE. *G320X Servo Drive*. [S.l.], 2011.
- GILCHRIST, A. *Industry 4.0: The Industrial Internet of Things*. [S.l.]: Springer, 2016. ISBN 978-1-4842-2047-4.
- GOTO, S. *Robot Arms*. [S.l.]: InTech, 2011. ISBN 978-953-307-160-2.
- GOWDY, J. *A qualitative comparison of interprocess communications toolkits for robotics*. [S.l.]: Citeseer, 2000.
- GRZEJSZCZAK, T.; LEGOWSKI, A.; NIEZABITOWSKI, M. Robot manipulator teaching techniques with use of hand gestures. In: IEEE. *2015 20th International Conference on Control Systems and Computer Science*. [S.l.], 2015. p. 71–77.
- GUPTA, K. K.; YU, Y. On eye-sensor based path planning for robots with non-trivial geometry/kinematics. In: IEEE. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. [S.l.], 2001. v. 1, p. 265–270.
- GUTIERREZ, M. *Desenvolvimento de uma Fresadora CNC Aderente á Norma STEP-NC Baseado no Controlador de Máquina Avanzado (EMC2)*. [S.l.]: Universidade de Brasília, 2013.
- HAMILTON, K.; HASCOËT, J.; RAUCH, M. D3. 1.1 description of the smc open controller. *Deliverable FoFdration, Novembre*, 2011.
- HARI-Team. *iGETRobot*: Canal youtube igetrobot interface. 2016. Disponível em: <<https://www.youtube.com/user/taxicnc>>. Acesso em: 18 sep. 2016.
- HASCOET, J.-Y.; RAUCH, M. Enabling advanced cnc programming with opennc controllers for hsm machines tools. *High Speed Machining*, v. 2, n. 1, p. 1–14, 2016.
- HORN, C.; KRÜGER, J. Feasibility of connecting machinery and robots to industrial control services in the cloud. In: IEEE. *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. [S.l.], 2016. p. 1–4.
- HUNT, V. *Robotics Sourcebook*. [S.l.]: Elsevier Science, 2012. ISBN 9780444601612.

HUO, F.; HONG, G.; POO, A. Extended development of linuxcnc for control of a delta robot. In: IEEE. *2015 International Conference on Advanced Mechatronic Systems (ICAMechS)*. [S.l.], 2015. p. 114–118.

IEEE. *IEEE Guide to the POSIX Open System Environment (OSE): Sponsor Portable Applications Standards Committee of the IEEE Computer Society Approved May 2, 1995, IEEE Standards Board*. [S.l.]: IEEE, 1995.

II, E. L. *Soldagem robotizada com eletrodo revestido*. Tese (Doutorado), 2006.

International Federation of Robotics. *World Robotics 2005*. [S.l.]: United Nations Publications, 2005. ISBN 9789211011005.

KANG, S. et al. *Robot Development Using Microsoft Robotics Developer Studio*. [S.l.]: CRC Press, 2016. ISBN 9781439821664.

Keling Technology Inc. *Manual NEMA23 - KL23-130-60*. [S.l.], 2016. 1 p.

KELLY, R.; DAVILA, V.; PEREZ, J. *Control of Robot Manipulators in Joint Space*. [S.l.]: Springer London, 2006. (Advanced Textbooks in Control and Signal Processing). ISBN 9781852339999.

KRAMER, T. R.; PROCTOR, F. M.; MESSINA, E. Internal Report, *The NIST RS274NGC Interpreter*. 2000.

KULKARNI, A.; TAI, K.; ABRAHAM, A. *Probability Collectives: A Distributed Multi-agent System Approach for Optimization*. [S.l.]: Springer International Publishing, 2015. (Intelligent Systems Reference Library). ISBN 9783319160009.

LAGES, W.; BAYAN, R. V.; Q., B. A. Arquitetura aberta para retrofitting de robôs. *UFRGS*, S?o Paulo, 2012.

LAGES, W.; BRACARENSE, A. Robot retrofitting: A perspective to small and medium size enterprises. *Austrian-Brazilian Automation Day, III*, p. 1–11, April 2003.

LARSON, R. *Trigonometry*. [S.l.]: Cengage Learning, 2010. 624 p. (Available 2011 Titles Enhanced Web Assign). ISBN 9781439049075.

LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, v. 3, p. 18 – 23, 2015. ISSN 2213-8463.

LEWIS, F. L.; DAWSON, D. M.; ABDALLAH, C. T. *Robot Manipulator Control: Theory and Practice: Automation and control engineering*. 2. ed. [S.l.]: CRC Press, 2004. ISBN 9780824740726.

LIMA, A. D. et al. Atulização de hardware e software de um robô industrial. *Congresso Brasileiro de Autom?tica*, XVIII, p. 4403–4410, 2010.

Lima II, E. et al. Sensoring for retrofitting of an industrial robot. *Information Control Problems in Manufacturing*, v. 1, p. 545–550, April 2004.

LINUXCNC.ORG. *Getting Started V2.5*. [S.l.], 2015.

LINUXCNC.ORG. *HAL Manual V2.5*. [S.l.], 2015.

LINUXCNC.ORG. *Integrator Manual V2.5*. [S.l.], 2015.

LIPTAK, B. *Instrument Engineers' Handbook, Fourth Edition, Volume Two: Process Control and Optimization*. [S.l.]: CRC Press, 2005. (Instrument Engineers' Handbook). ISBN 9781420064001.

LLC Harmonic Drive. *Harmonic drive general catalog*. [S.l.], 2016. 76 p.

- MAIA, D. et al. Retrofitting do robô asea irb6. 2003.
- MARTINEZ, A.; FERNÁNDEZ, E. *Learning ROS for Robotics Programming*. [S.l.]: Packt Publishing, 2013. ISBN 9781782161455.
- MILUTINOVIC, D. Reconfigurable robotic machining system controlled and programmed in a machine tool manner. *The International Journal of Advanced Manufacturing Technology*, v. 53, p. 1217–1229, April 2011.
- MITUTOYO. *PJ-A3000 Profile Projector*. 2021. ed. [S.l.], 2011. 6 p.
- MOCTEZUMA, L. E. G. et al. Retrofitting a factory automation system to address market needs and societal changes. In: IEEE. *IEEE 10th International Conference on Industrial Informatics*. [S.l.], 2012. p. 413–418.
- MORTIMER, J.; ROOKS, B. *The International Robot Industry Report*. [S.l.]: Springer Berlin Heidelberg, 2013. ISBN 9783662131749.
- NATARAJAN, R. *Power System Capacitors*. [S.l.]: Taylor & Francis Group, 2005. ISBN 9781574447101.
- NICOLAIDES, A. *Pure mathematics: Trigonometry*. [S.l.]: P.A.S.S, 2007. (Success in Pure Mathematics, v. 2). ISBN 9781872684871.
- NOF, S. *Handbook of Industrial Robotics*. [S.l.]: Wiley, 1999. 1348 p. (Electrical and electronic engineering, v. 1). ISBN 9780471177838.
- OGATA, K. *Modern Control Engineering*. [S.l.]: Prentice Hall, 2010. (Instrumentation and controls series). ISBN 9780136156734.
- OLIVEIRA A., D. P.; MORENO, U. An open-architecture robot controller applied to interaction tasks. *Advances in Robot Manipulators*, p. 99–112, 2010.
- OLIVEIRA, A. S. d. *Retrofitting de robôs manipuladores com incorporação de controle de posição e força: aplicação em um robô industrial*. 147 p. Dissertação (mathesis) — Universidade Federal de Santa Catarina, 2007.
- OLIVEIRA, L. C. P. L. *Maquinagem de superfícies complexas com recurso a sistema robótico*. 129 p. Dissertação (Mestrado) — Universidade do Porto, 2013.
- PAZIANI, F. T.; GIACOMO, B. D.; TSUNAKI, R. H. Robot measuring form errors. *Robotics and Computer-Integrated Manufacturing*, v. 25, n. 1, p. 168 – 177, 2009. ISSN 0736-5845.
- PREEZ, R. 3d 6-dof serial armn robot - kinematics and implementation in linuxcnc. *ASM*, Pretoria, 2014.
- RIBEIRO, A. et al. Metodologia para implementação de retroffiting de controladores de equipamentos industriais. 2007.
- ROHDE, U. et al. *Introduction to Differential Calculus: Systematic Studies with Engineering Applications for Beginners*. [S.l.]: John Wiley & Sons, 2012. ISBN 9781118130148.
- ROMANO, V. F. *Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos*. Primeira. [S.l.]: Edgard Blücher, 2002. ISBN 85-212-0315-2.
- RYAN, D. *CAD/CAE Descriptive Geometry*. [S.l.]: Taylor & Francis, 1991. 224 p. ISBN 9780849342738.
- SCHUMACKER, R. E. *Learning Statistics Using R*. [S.l.]: SAGE Publications, 2014. 648 p. ISBN 9781483324777.

- SCHUNK. *SCHUNK Grippers pneumatic: Pgn-plus universal gripper*. [S.l.], 2014. 83-94 p.
- SHACKLEFORD, W. P.; PROCTOR, F. M.; MICHALOSKI, J. L. The neutral message language: A model and method for message passing in heterogeneous environments. In: CITESEER. *Proceedings of the 2000 World Automation Conference*. [S.l.], 2000. p. 11–16.
- SHIN, S.-J. et al. Developing a virtual machining model to generate mtconnect machine-monitoring data from step-nc. *International Journal of Production Research*, Taylor & Francis, p. 1–19, 2015.
- SICILIANO, B.; KHATIB, O. *Springer Handbook of Robotics*. [S.l.]: Springer Berlin Heidelberg, 2008. (Springer Handbook of Robotics). ISBN 9783540239574.
- SICILIANO, B. et al. *Robotics: Modelling, Planning and Control*. [S.l.]: Springer London, 2008. (Advanced Textbooks in Control and Signal Processing). ISBN 9781846286421.
- SIRINTERLIKCI, A. et al. Repeatability and accuracy of an industrial robot: Laboratory experience for a design of experiments course. 2009.
- SPONG, M.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot Modeling and Control*. [S.l.]: Wiley, 2005. ISBN 9780471649908.
- STAROVESKI, T. et al. Implementation of a linux-based cnc open control system. *International Scientific Conference on Production Engineering*, XII, p. 209–216, June 2009.
- STERN, T. *Lean Six Sigma: International Standards and Global Guidelines*. Second edition. [S.l.]: CRC Press, 2016. ISBN 9781498739610.
- STONE, H. W. *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. [S.l.]: Springer US, 2012. 224 p. (The Springer International Series in Engineering and Computer Science). ISBN 9781461319993.
- STRASSER, T.; ROOKER, M.; EBENHOFER, G. Distributed control concept for a 6-dof reconfigurable robot arm. In: *Innovation production machines and systems: fourth Proms virtual international conference; 1st–14th July*. [S.l.: s.n.], 2008.
- SZKODNY, T. Forward and inverse kinematics of irb-6 manipulator. *Mechanism and machine theory*, XXX, p. 1039–1056, 1995.
- TOQUICA, J. S.; ALVARES, A. Desarrollo e implementación de un controlador basado en tecnología cnc para el robot asea irb6-s2 usando linuxcnc. *CIBIM*, IEEE, n. XII, p. 981–989, 2015.
- TOQUICA, J. S.; ÁLVARES, J. A. Implementación de la técnica retrofitting para el robot asea irb6-s2 usando linuxcnc. In: *Congresso Nacional de Engenharia Mecânica - CONEM2016*. [S.l.: s.n.], 2016. p. 10.
- VISTACNC. *VistaCNC P4-S(E) CNC Control Pendant for LinuxCNC: P4-s pendant linuxcnc manual v. 1.0*. [S.l.], 2016. 12 p.
- WALPOLE, R.; MYERS, R.; MYERS, S. *Probabilidad y estadística para ingenieros*. Sexta. [S.l.]: Pearson Educación, 1999. ISBN 9789701702642.
- WEINBERGER, M.; BILGERI, D.; FLEISCH, E. Iot business models in an industrial context. *at-Automatisierungstechnik*, v. 64, n. 9, p. 699–706, 2016.
- YOUNG, K.; PICKIN, C. G. Accuracy assessment of the modern industrial robot. *Industrial Robot: An International Journal*, v. 27, n. 6, p. 427–436, 2000.

YU, D. et al. An open cnc system based on component technology. *IEEE Transactions on Automation Science and Engineering*, IEEE, v. 6, n. 2, p. 302–310, 2009.

ŽIVANOVIĆ, S.; GLAVONJIĆ, M. Methodology for implementation scenarios for applying protocol step-nc. *Journal of Production Engineering*, v. 17, n. 1, p. 71–74, 2014.

APÊNDICES

A. PROGRAMA NC - TEACH-IN

Neste apêndice se apresentam dois programas NC, em linguagem compatível com a norma RS274 (código G) para o estudo de caso *teach-in*. Estes programas NC são interpretados pelo controlador LinuxCNC para a geração e envio de sinais de controle ao manipulador ASEA IRB6-S2.

```
1 (Programa um)
2
3 G21 G90 G64 G40
4
5 G0 X0 Y0 Z0 F450
6 G1 X0.00 Y-1.069 Z-82.947 A-8.028 B3.747
7 M101
8
9 G1 X0 Y0 Z0 A0 B0
10 G1 X-216.187 Y7.603 Z44.899 A13.926 B10.303
11 G1 X-217.187 Y11.388 Z-78.732 A2.407 B13.788
12 M102
13 G1 X-216.187 Y7.603 Z44.899 A13.926 B10.303
14
15 M30
```

```
1 (Programa dois)
2
3 G21 G90 G64 G40
4
5 G0 X0 Y0 Z0 F450
6 G1 X-0.009 Y-10.359 Z-52.330 A-0.737 B3.994
7 M101
8
9 G1 X0 Y0 Z0 A0 B0
10 G1 X-198.874 Y44.940 Z45.254 A49.543 B4.438
11 G1 X-195.820 Y32.556 Z-39.112 A36.897 B8.669
12 M102
13 G1 X-198.874 Y44.940 Z45.254 A49.543 B4.438
14
15 M30
```

B. PROGRAMA NC - REPETIBILIDADE

Neste apêndice se apresenta um programa NC, em linguagem compatível com a norma RS274 (código G) para o estudo de caso de repetibilidade. Este programa NC é interpretado pelo controlador LinuxCNC para a geração e envio de sinais de controle ao manipulador ASEA IRB6-S2.

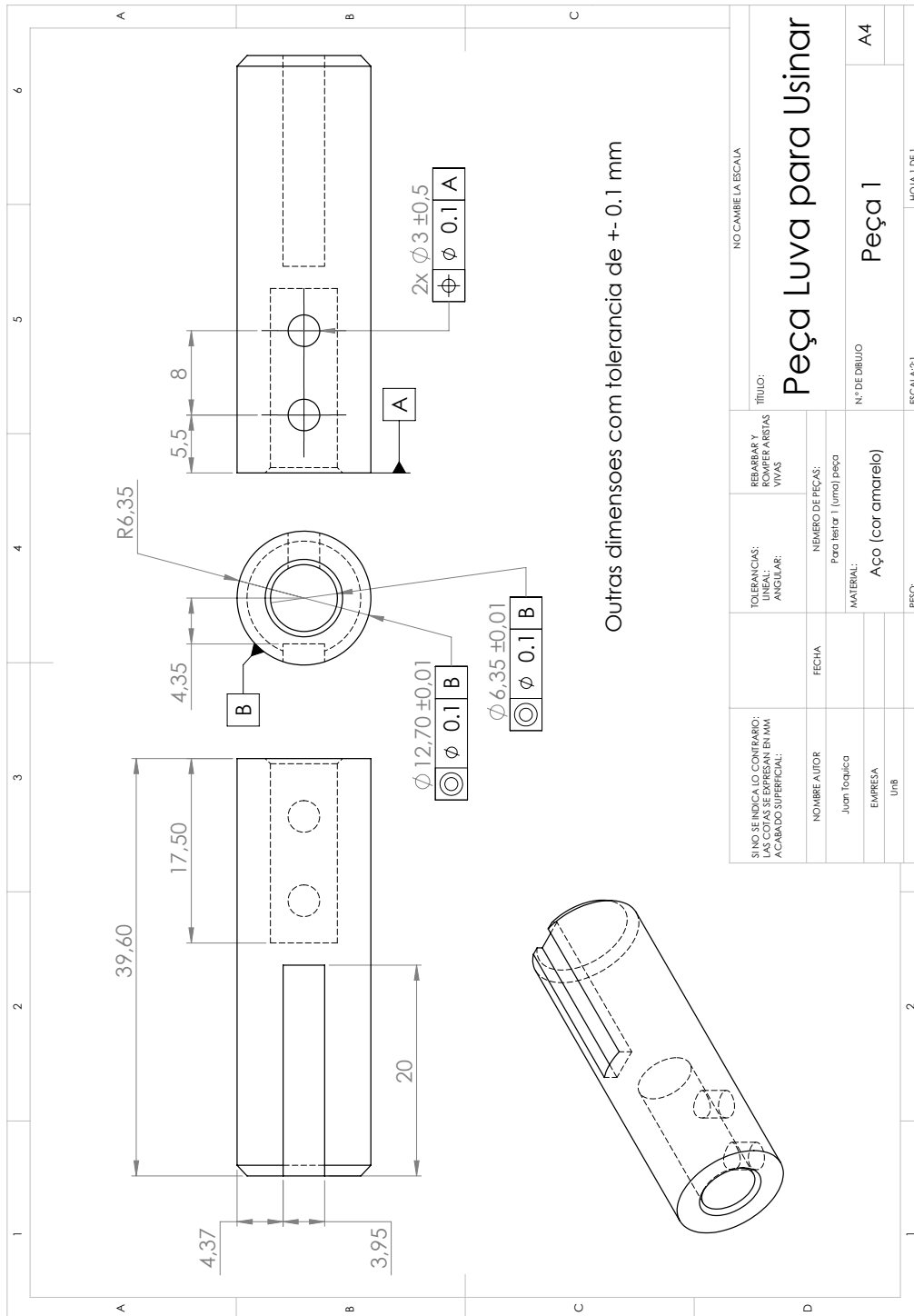
```
1  AXIS "splash g-code" Not intended for actual milling. To run this code anyway you might have to
2  Touch Off the Z axis depending on your setup. As if you had some material in yourmill. Hint jog the
3  Z axis down a bit then touch off. Also press the Toggle Skip Lines with "/" to see that part.
4  If the program is too big or small for your machine, change the scale to 3.
5  Copyright 2000–2016 LinuxCNC – GNU License. Edited from original example file June/2016 by:
6  Juan Sebastian Toquica Arenas and Prof. Alberto J. Alvares
```

```
1  G21 G90 G64 G40 G17
2  M3 S10000
3  G0 Z3.0
4  G0 X1.75781 Y0.5
5  G1 F100.0 Z-2.0
6  G1 F400.0 X5.95508 Y20.54297
7  G1 X10.07031
8  G1 X6.58398 Y3.84961
9  G1 X16.7832
10 G1 X16.08594 Y0.5
11 G1 X1.75781
12 G0 Z3.0
13 G0 X18.72461
14 G1 F100.0 Z-2.0
15 G1 F400.0 X21.75977 Y15.01953
16 G1 X25.68359
17 G1 X22.64844 Y0.5
18 G1 X18.72461
19 G0 Z3.0
20 G0 X26.55859
21 G1 F100.0 Z-2.0
22 G1 F400.0 X29.59375 Y15.01953
23 G1 X33.3125
24 G1 X32.92969 Y13.13281
25 G2 X34.16342 Y14.08624 I8.82141 J-10.13994
26 G2 X35.52734 Y14.8418 I4.53823 J-6.58354
27 G2 X38.08398 Y15.36133 I2.53506 J-5.9247
28 G2 X39.5966 Y15.13543 I0.06403 J-4.74845
29 G2 X40.90039 Y14.33594 I-1.02874 J-3.14049
30 G2 X41.7019 Y13.08328 I-2.33045 J-2.37388
31 G2 X41.93945 Y11.61523 I-4.07102 J-1.41199
32 G2 X41.76899 Y10.15744 I-10.17473 J0.45091
33 G2 X41.48828 Y8.7168 I-39.45138 J6.93932
34 G1 X39.7793 Y0.5
35 G1 X35.85547
36 G1 X37.57813 Y8.74414
37 G3 X37.79665 Y9.84001 I-62.81729 J13.09579
38 G3 X37.96094 Y10.94531 I-9.6524 J1.99958
39 G3 X37.50977 Y12.12109 I-1.54162 J0.0829
40 G3 X36.2793 Y12.55859 I-1.13356 J-1.23903
41 G3 X35.26888 Y12.33731 I0.05277 J-2.65845
42 G3 X34.36523 Y11.83398 I1.956 J-4.57455
```

43 G3 X32.71094 Y9.91992 I2.86418 J-4.1474
44 G3 X32.13267 Y8.21493 I8.76492 J-3.92328
45 G3 X31.72656 Y6.46094 I36.34493 J-9.33906
46 G1 X30.48242 Y0.5
47 G1 X26.55859
48 G0 Z3.0
49 G0 X26.09375 Y16.98828
50 G1 F100.0 Z-2.0
51 G1 F400.0 X22.16992
52 G1 X22.9082 Y20.54297
53 G1 X26.83203
54 G1 X26.09375 Y16.98828
55 G0 Z3.0
56 G0 X46.14777 Y12.78778
57 G1 F100.0 Z-2.0
58 G1 F400.0 X46.61523 Y15.01953
59 G1 X50.53906
60 G1 X48.74805 Y6.41992
61 G3 X48.55485 Y5.46101 I39.83359 J-8.52447
62 G3 X48.41992 Y4.49219 I7.34343 J-1.51652
63 G3 X48.88477 Y3.41211 I1.45252 J-0.01493
64 G3 X50.07422 Y2.96094 I1.13093 J1.18803
65 G3 X51.09961 Y3.15234 I-0.00663 J2.87782
66 G3 X52.13867 Y3.75391 I-1.85377 J4.40013
67 G3 X53.0957 Y4.68359 I-3.51724 J4.57812
68 G3 X53.88867 Y6.05078 I-4.71119 J3.64605
69 G3 X54.44922 Y8.10156 I-12.97687 J4.64901
70 G1 X55.89844 Y15.01953
71 G1 X59.82227
72 G1 X56.78711 Y0.5
73 G1 X53.12305
74 G1 X53.5332 Y2.46875
75 G2 X51.14513 Y0.79202 I-6.21919 J6.3187
76 G2 X48.29688 Y0.1582 I-2.84268 J6.05776
77 G2 X46.78426 Y0.3841 I-0.06403 J4.74845
78 G2 X45.48047 Y1.18359 I1.02874 J3.14049
79 G2 X44.68637 Y2.45262 I2.34744 J2.35189
80 G2 X44.45508 Y3.93164 I4.23866 J1.42044
81 G2 X44.63379 Y5.43705 I10.83187 J-0.52256
82 G2 X44.91992 Y6.92578 I45.15644 J-7.90718
83 G1 X46.14777 Y12.78778
84 G0 Z3.0
85 G0 X61.99609 Y15.01953
86 G1 F100.0 Z-2.0
87 G1 F400.0 X66.16602
88 G1 X68.28516 Y10.87695
89 G1 X71.96289 Y15.01953
90 G1 X76.73438
91 G1 X69.99414 Y7.48633
92 G1 X73.6582 Y0.5
93 G1 X69.48828
94 G1 X67.39648 Y4.57422
95 G1 X63.78711 Y0.5
96 G1 X58.97461
97 G1 X65.6875 Y7.9375
98 G1 X61.99609 Y15.01953
99 G0 Z5.0
100 G0 X1.75781 Y0.5
101 M5
102 M30

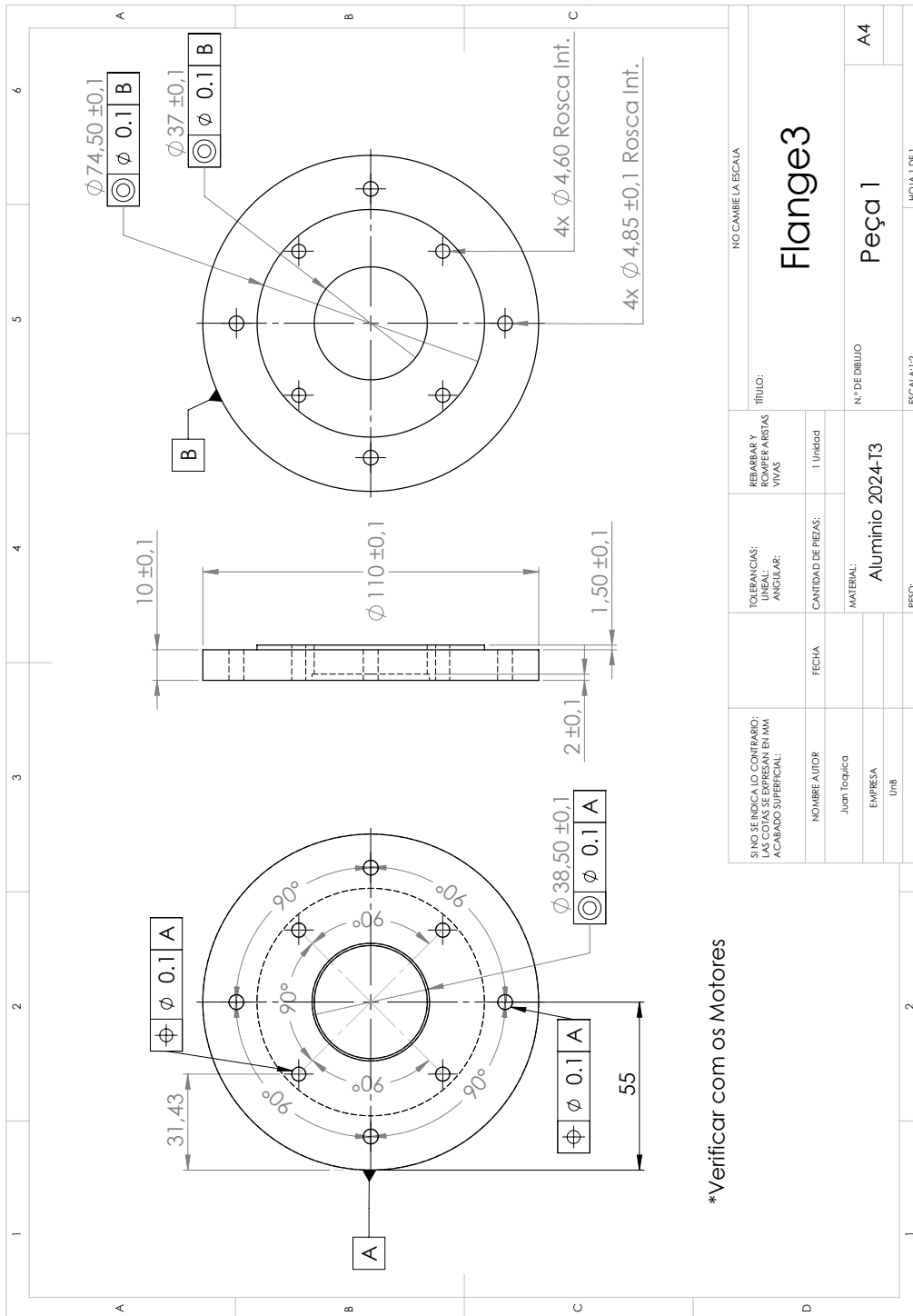
C. DESENHO MECÂNICO DAS LUVAS

Neste apêndice se apresenta o desenho mecânico das luvas fabricadas para acoplar os eixos dos motores adquiridos ao *Harmonic-drive* e aos parafusos de esferas das cinco juntas do manipulador ASEA IRB6-S2.



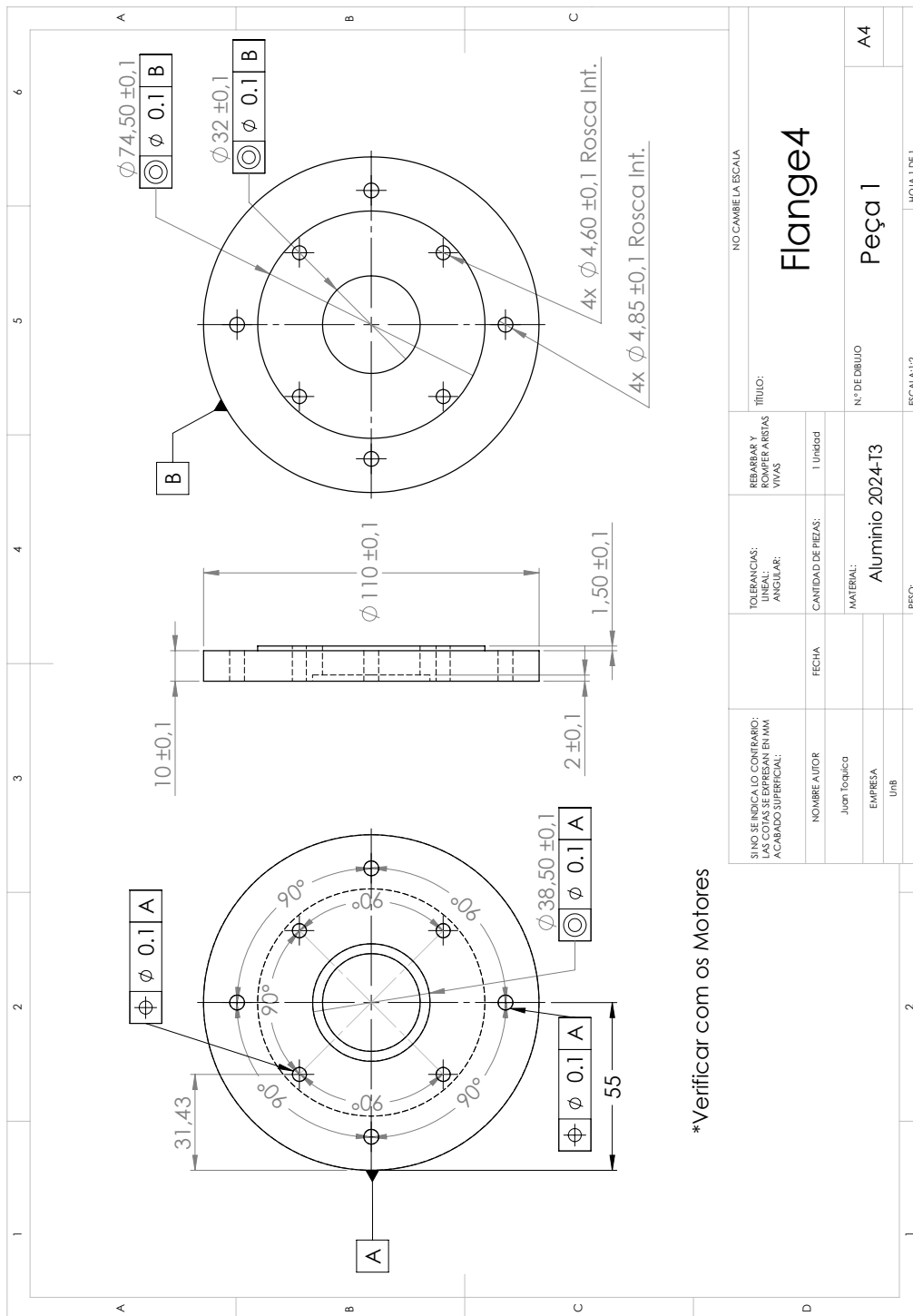
D. DESENHO MECÂNICO FLANGE: JUNTA DOIS

Neste apêndice se apresenta o desenho mecânico da flange da junta dois, responsável por unir a carcaça do motor com a estrutura mecânica do manipulador ASEA IRB6-S2.



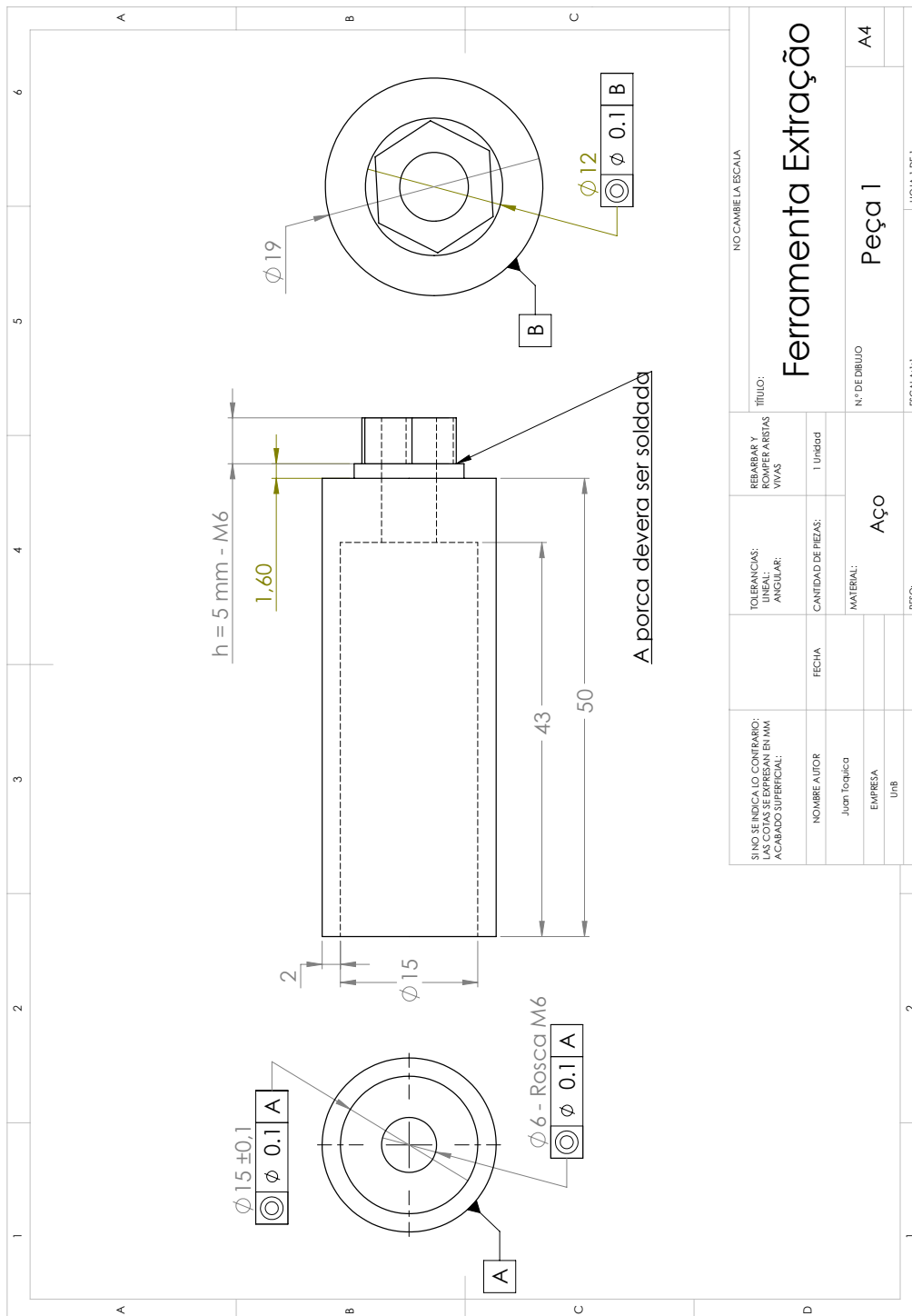
E. DESENHO MECÂNICO FLANGE: JUNTA TRÊS

Neste apêndice se apresenta o desenho mecânico da flange da junta três, responsável por unir a carcaça do motor com a estrutura mecânica do manipulador ASEA IRB6-S2.



F. FERRAMENTA DE EXTRAÇÃO DE PINOS

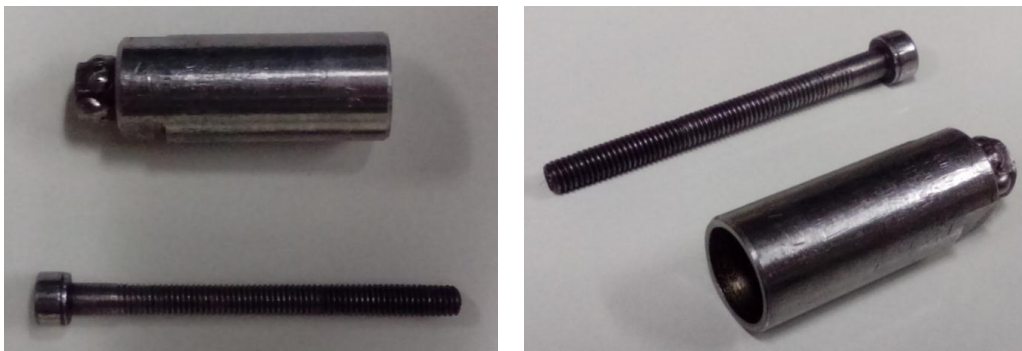
Neste apêndice se apresenta o desenho mecânico da ferramenta de extração projetada e fabricada para tirar os pinos que fixam o base do manipulador com a estrutura mecânica das ultimas quatro juntas.



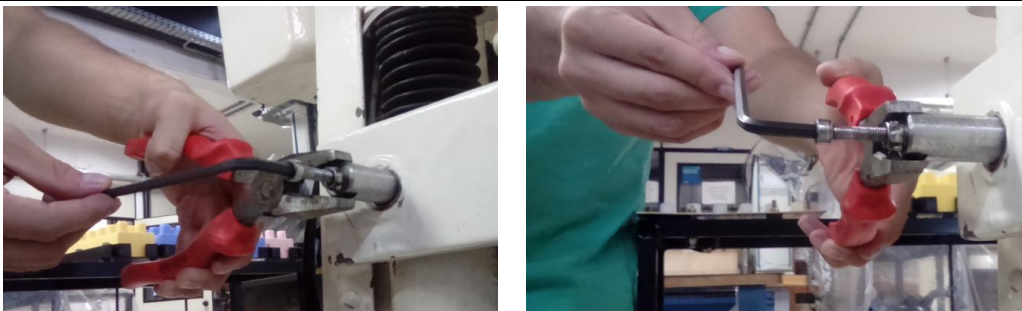
G. USO DA FERRAMENTA DE EXTRAÇÃO DE PINOS

Neste apêndice se apresenta em detalhe o uso da ferramenta de extração projetada e fabricada para afastar a estrutura mecânica das ultimas quatro juntas da base do manipulador.

Vista da ferramenta de extração de pinos



Vista da ferramenta no momento da extração do pino

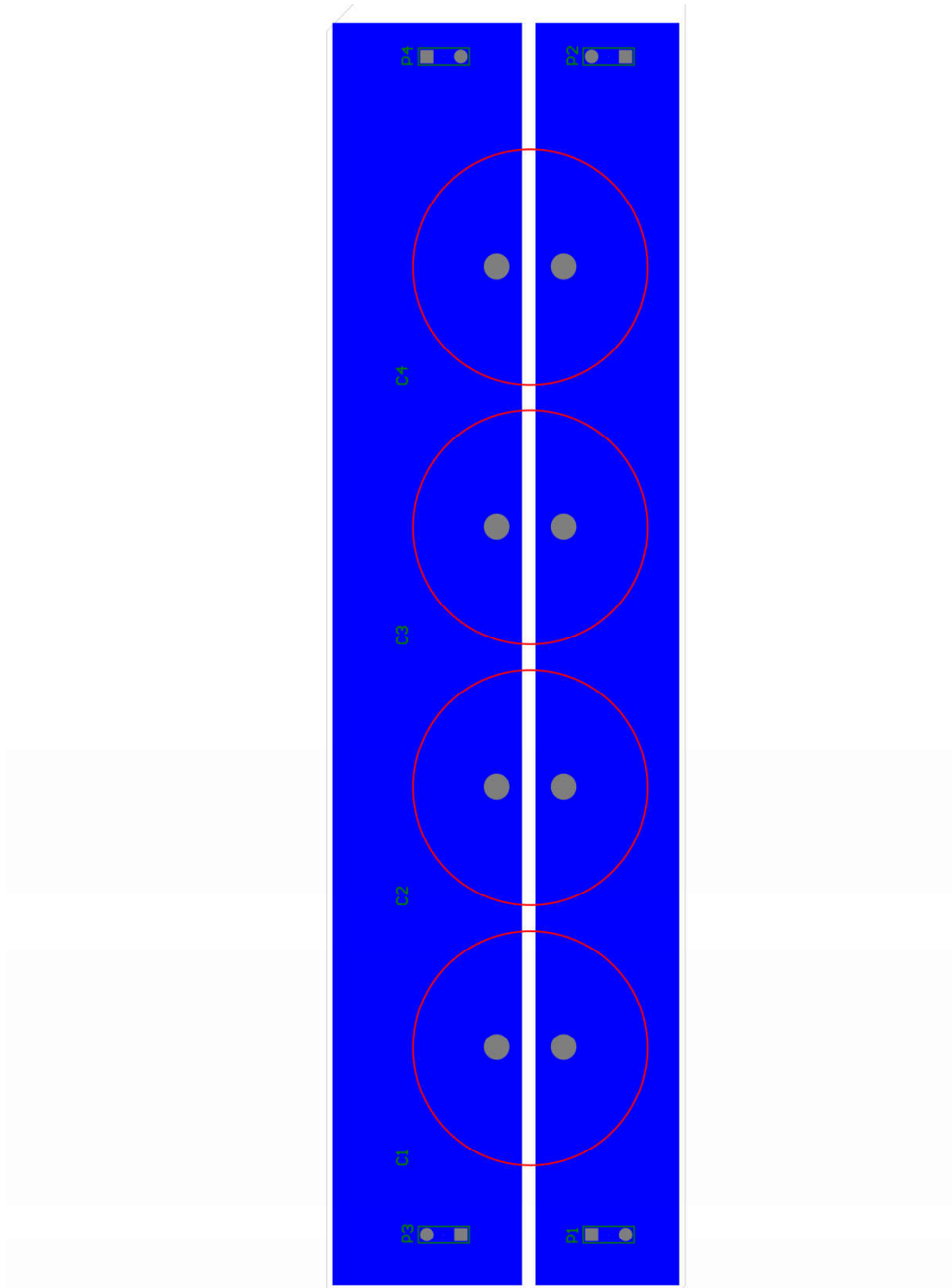


Vista da ferramenta de extração com o Alicate e chave Allen de 5 mm



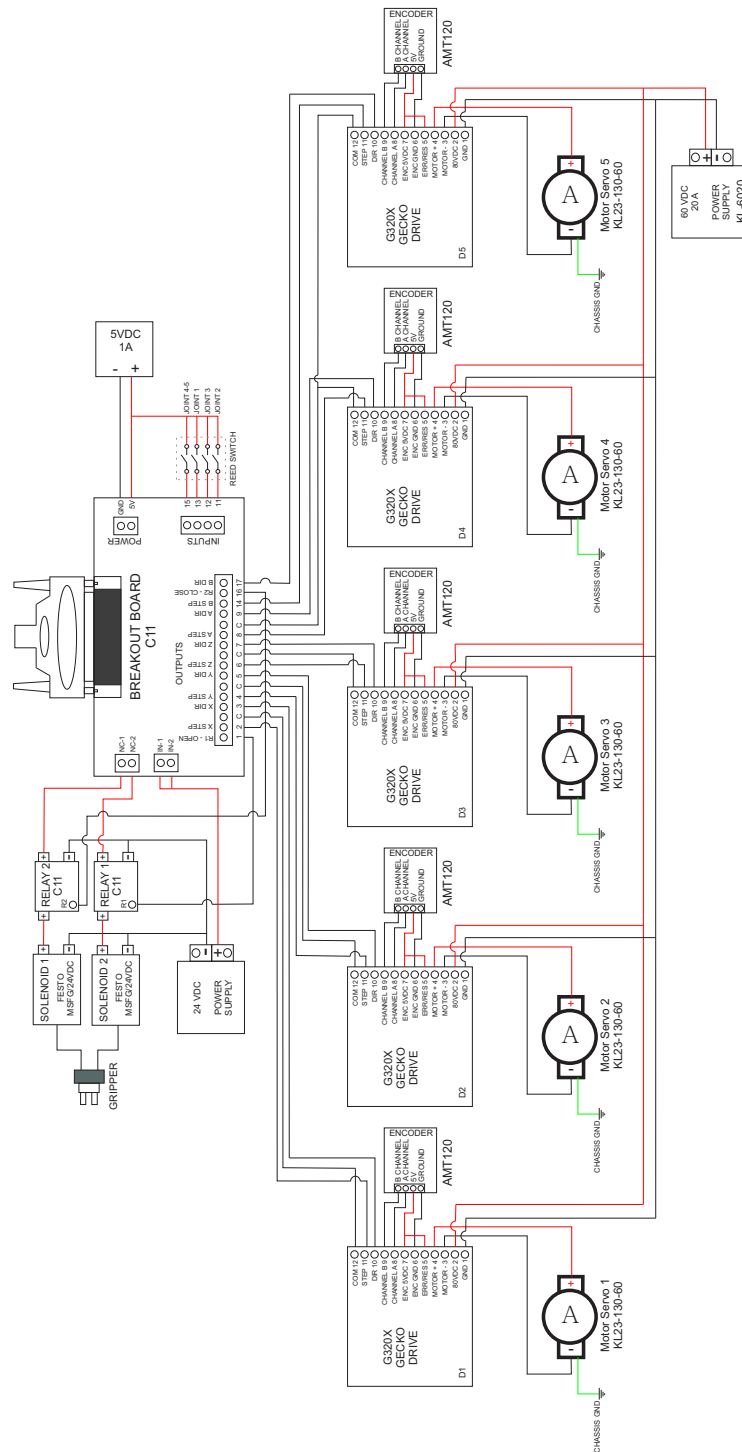
H. DESENHO TÉCNICO PLACA CAPACITORES

Neste apêndice se apresenta o desenho técnico da placa de capacitores fabricada para a integração com a fonte de tensão do sub-sistema de alimentação do manipulador ASEA IRB6-S2.



I. DIAGRAMA ELÉTRICO NAS CONEXÕES PLACAS/-MOTORES/DRIVES/ENCODERS/FONTE/COMPUTADOR

Neste apêndice se apresenta o desenho elétrico do sistema de controle após o *retrofitting*.



J. ARQUIVOS COM A CINEMÁTICA DO MANIPULADOR

Neste apêndice se apresenta o código em linguagem C do arquivo "irb.c", com as equações da cinemática direta e inversa do manipulador ASEA IRB6-S2.

```
1  /*****
2  * Description: irb.c
3  * Kinematics for irb6 typed robots, edited from pumakinks.c
4  * Set the params using HAL to fit your robot.
5  * Derived from a work by Tadeusz Szkodny(1995)
6  * Authors: Alberto Alvares – GIAI/UnB and Juan Toquica – UnB
7  * License: GPL Version 2 – System: Linux
8  * Copyright (c) 2015 All rights reserved.
9  *****/
10 */
11 #include "rtapi_math.h"
12 #include "posemath.h"
13 #include "irb.h"
14 #include "kinematics.h"          /* decls for kinematicsForward, etc. */
15
16 #include "rtapi.h"              /* RTAPI realtime OS API */
17 #include "rtapi_app.h"         /* RTAPI realtime module decls */
18 #include "hal.h"
19
20 struct haldata {
21     hal_float_t *a2, *a3, *d1, *d5; //Constant Variables DH IRB6-S2
22     hal_float_t *th1, *th2, *th3, *th4, *th5; //Theta Angles for VM Simulation
23     hal_float_t *ba, *bc, *bd, *bj, *dj, *fe, *fg, *fh; //IRB6-S2 Dimensions
24     hal_float_t *k1, *k2, *k3, *k4, *k5, *k6; //Reduction Gears
25 } *haldata = 0;
26
27 #define A2 (*(haldata->a2))
28 #define A3 (*(haldata->a3))
29 #define D1 (*(haldata->d1))
30 #define D5 (*(haldata->d5))
31
32 #define BA (*(haldata->ba))
33 #define BC (*(haldata->bc))
34 #define BD (*(haldata->bd))
35 #define BJ (*(haldata->bj))
36 #define DJ (*(haldata->dj))
37 #define FE (*(haldata->fe))
38 #define FG (*(haldata->fg))
39 #define FH (*(haldata->fh))
40
41 #define K1 (*(haldata->k1))
42 #define K2 (*(haldata->k2))
43 #define K3 (*(haldata->k3))
44 #define K4 (*(haldata->k4))
45 #define K5 (*(haldata->k5))
46 #define K6 (*(haldata->k6))
47
48 int kinematicsForward(const double * joint,
49 EmcPose * world,
50 const KINEMATICS_FORWARD_FLAGS * fflags,
```

```

51 KINEMATICS_INVERSE_FLAGS * iflags)
52 {
53
54 double s1, s2, s3, s4, s5;
55 double c1, c2, c3, c4, c5;
56 double s23;
57 double c23;
58
59 double s234;
60 double c234;
61
62 double t1, t2, t3, t4, t5, t6, t7, t8, t9;
63
64 double th3, th2, th1;
65
66 PmHomogeneous hom;
67 PmPose worldPose;
68 PmRpy rpy;
69
70 // #####
71 // Modify by robot mechanical design
72 // It was modified in inverse kinematic
73 th3 = joint[2] - (joint[1]* K1);
74 // #####
75
76 th2 = joint[1];
77 th1 = joint[0];
78
79 /* Calculate sin of joints for future use */
80 //s1 = sin(joint[0]*PM_PI/180);
81 s1 = sin(th1*PM_PI/180);
82 //s2 = sin(joint[1]*PM_PI/180);
83 s2 = sin(th2*PM_PI/180);
84 s3 = sin(th3*PM_PI/180);
85 //s3 = sin(joint[2]*PM_PI/180);
86 s4 = sin(joint[3]*PM_PI/180);
87 s5 = sin(joint[4]*PM_PI/180);
88
89 /* Calculate cos of joints for future use */
90 //c1 = cos(joint[0]*PM_PI/180);
91 c1 = cos(th1*PM_PI/180);
92 //c2 = cos(joint[1]*PM_PI/180);
93 c2 = cos(th2*PM_PI/180);
94 c3 = cos(th3*PM_PI/180);
95 //c3 = cos(joint[2]*PM_PI/180);
96 c4 = cos(joint[3]*PM_PI/180);
97 c5 = cos(joint[4]*PM_PI/180);
98
99 s23 = c2 * s3 + s2 * c3;
100 c23 = c2 * c3 - s2 * s3;
101
102 s234 = s2*c3*c4 + c2*s3*c4 + c2*c3*s4 - s2*s3*s4;
103 c234 = c2*c3*c4 - s2*s3*c4 - s2*c3*s4 - c2*s3*s4;
104
105 /* Calculate terms to be used in definition of... */
106 /* first column of rotation matrix. */
107 t1 = s1 * s234 * c5;
108 t2 = c1 * s5;
109 t3 = -c1 * s234 * c5;
110 t4 = s1 * s5;

```

```

111
112 /* Define first column of rotation matrix */
113 hom.rot.x.x = t1 + t2;
114 hom.rot.x.y = t3 + t4;
115 hom.rot.x.z = c234 * c5;
116
117 /* Calculate terms to be used in definition of... */
118 /* second column of rotation matrix. */
119 t1 = -s1 * s234 * s5;
120 t2 = c1 * c5;
121 t3 = c1 * s234 * s5;
122 t4 = s1 * c5;
123
124 /* Define second column of rotation matrix */
125 hom.rot.y.x = t1 + t2;
126 hom.rot.y.y = t3 + t4;
127 hom.rot.y.z = -c234 * s5;
128
129 /* Calculate term to be used in definition of... */
130 /* third column of rotation matrix. */
131 // no terms
132
133 /* Define third column of rotation matrix */
134 hom.rot.z.x = -s1 * c234;
135 hom.rot.z.y = c1 * c234;
136 hom.rot.z.z = s234;
137
138 /* Calculate term to be used in definition of... */
139 /* position vector. */
140 t1 = A2 * s1 * s2;
141 t2 = A3 * s1 * c23;
142 t3 = D5 * s1 * c234;
143
144 t4 = A2 * c1 * s2;
145 t5 = A3 * c1 * c23;
146 t6 = D5 * c1 * c234;
147
148 t7 = A2 * c2;
149 t8 = A3 * s23;
150 t9 = D5 * s234;
151
152 /* Define position vector */
153 hom.tran.x = t1 - t2 - t3;
154 hom.tran.y = -t4 + t5 + t6;
155 hom.tran.z = D1 + t7 + t8 + t9;
156
157 *iflags = 0;
158
159 /* convert hom.rot to world->quat */
160 pmHomPoseConvert(hom, &worldPose);
161 pmQuatRpyConvert(worldPose.rot,&rpy);
162 world->tran = worldPose.tran;
163 world->a = rpy.r * 180.0/PM_PI;
164 world->b = rpy.p * 180.0/PM_PI;
165 world->c = rpy.y * 180.0/PM_PI;
166
167 /* return 0 and exit */
168 return 0;
169 }
170

```

```

171 int kinematicsInverse(const EmcPose * world,
172 double * joint,
173 const KINEMATICS_INVERSE_FLAGS * iflags,
174 KINEMATICS_FORWARD_FLAGS * fflags)
175 {
176 PmHomogeneous hom;
177 PmPose worldPose;
178 PmRpy rpy;
179
180 double t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12;
181
182 double th1, th2, th3, th4, th5, th34;
183 double th1_m, th2_m, th3_m, th4_m, th5_m;
184
185 double s1, c1;
186 double s2, c2;
187 double s3, c3;
188 double s34, c34;
189 double s5, c5;
190
191 double w1, w2;
192
193 double lambda_2, lambda_3;
194
195 /* reset flags */
196 *fflags = 0;
197
198 /* convert pose to hom */
199 worldPose.tran = world->tran;
200 rpy.r = world->a*PM_PI/180.0;
201 rpy.p = world->b*PM_PI/180.0;
202 rpy.y = world->c*PM_PI/180.0;
203 pmRpyQuatConvert(rpy,&worldPose.rot);
204 pmPoseHomConvert(worldPose, &hom);
205
206 /* Joint 1 (2 independent solutions) */
207
208 th1 = atan2(-hom.tran.x,hom.tran.y);
209
210 /* save sin, cos for later calcs */
211 s1 = sin(th1);
212 c1 = cos(th1);
213
214 /* Joint 3 (2 independent solutions) */
215 t1 = D5 * s1 * hom.rot.z.x;
216 t2 = D5 * c1 * hom.rot.z.y;
217
218 w1 = -s1 * hom.tran.x + c1 * hom.tran.y + t1 - t2;
219 w2 = hom.tran.z - D1 - D5 * hom.rot.z.z;
220
221 /*
222 t3 = pow(w1,2);
223 t4 = pow(w2,2);
224 t5 = pow(A2,2);
225 t6 = pow(A3,2);
226 */
227
228 t3 = w1 * w1;
229 t4 = w2 * w2;
230 t5 = A2 * A2;

```



```

231 t6 = A3 * A3;
232 t7 = 2.0 * A2 * A3;
233
234 s3 = (t3 + t4 - (t5 + t6))/t7;
235
236 //t7 = pow(s3,2); sr/share/glade3/pixmaps/
237
238 t7 = s3 * s3;
239
240 c3 = sqrt(1.0 - t7);
241
242 th3 = atan2(s3, c3);
243 //th3 = asin(s3);
244
245 s3 = sin(th3);
246 c3 = cos(th3);
247
248 /* Joint 2 */
249
250 t1 = (A3 * s3) + A2;
251 t2 = w2 * A3 * c3;
252 t3 = w1 * t1;
253 //t4 = pow(A3, 2) * pow(c3, 2) ;
254 //t5 = pow(t1, 2);
255 t4 = A3 * A3 * c3 * c3;
256 t5 = t1 * t1;
257
258 t6 = w1 * A3 * c3;
259 t7 = w2 * t1;
260
261 s2 = (t2 - t3)/(t4 + t5);
262 c2 = (t6 + t7)/(t4 + t5);
263
264 th2 = atan2(s2, c2);
265
266 s2 = sin(th2);
267 c2 = cos(th2);
268
269 /* compute th34*/
270
271 t1 = s1 * s2 * hom.tran.x;
272 t2 = c1 * s2 * hom.tran.y;
273 t3 = s1 * c2 * hom.tran.x;
274 t4 = c1 * c2 * hom.tran.y;
275
276 s34 = t1 - t2 + (c2 * hom.tran.z) - (D1 * c2) - A2 - (A3 * s3);
277 c34 = -t3 + t4 + (s2 * hom.tran.z) - (D1 * s2) - (A3 * c3);
278
279 t5 = D5 * s34;
280 t6 = D5 * c34;
281
282 th34 = atan2(s34, c34);
283
284 /* Joint 4 */
285
286 th4 = th34 - th3;
287
288 /* Joint 5 */
289
290 s5 = (c1 * hom.rot.x.x) + (s1 * hom.rot.x.y);

```

```

291 c5 = (c1 * hom.rot.y.x) + (s1 * hom.rot.y.y);
292
293 th5 = atan2(s5, c5);
294
295 // #####
296 // Modify by robot mechanical design
297 // It was modified in forward kinematic
298 th3_m = th3 + (th2* K1);
299 // #####
300
301 th1 = th1 * 180 / PM_PI;
302 th2 = th2 * 180 / PM_PI;
303 th3 = th3_m * 180 / PM_PI;
304 th4 = th4 * 180 / PM_PI;
305 th5 = th5 * 180 / PM_PI;
306
307 joint[0] = th1;
308 joint[1] = th2;
309 joint[2] = th3;
310 joint[3] = th4;
311 joint[4] = th5;
312
313 return 0;
314 }
315
316 int kinematicsHome(EmcPose * world,
317 double * joint,
318 KINEMATICS_FORWARD_FLAGS * fflags,
319 KINEMATICS_INVERSE_FLAGS * iflags)
320 {
321 /* use joints, set world */
322
323 // *fflags = 0;
324 // *iflags = 0;
325
326 return kinematicsForward(joint, world, fflags, iflags);
327 // return kinematicsInverse(joint, world, fflags, iflags);
328
329 }
330
331 KINEMATICS_TYPE kinematicsType()
332 {
333
334 return KINEMATICS_BOTH;
335 }
336
337
338 EXPORT_SYMBOL(kinematicsType);
339 EXPORT_SYMBOL(kinematicsForward);
340 EXPORT_SYMBOL(kinematicsInverse);
341 MODULE_LICENSE("GPL");
342
343 int comp_id;
344
345 int rtapi_app_main(void) {
346 int res=0;
347
348 comp_id = hal_init("irb");
349 if (comp_id < 0) return comp_id;
350

```

```

351 haldata = hal_malloc(sizeof(struct haldata));
352 if (!haldata) goto error;
353
354 if((res = hal_pin_float_new("irb.A2", HAL_IO, &(haldata->a2), comp_id)) < 0) goto error;
355 if((res = hal_pin_float_new("irb.A3", HAL_IO, &(haldata->a3), comp_id)) < 0) goto error;
356 if((res = hal_pin_float_new("irb.D1", HAL_IO, &(haldata->d1), comp_id)) < 0) goto error;
357 if((res = hal_pin_float_new("irb.D5", HAL_IO, &(haldata->d5), comp_id)) < 0) goto error;
358
359 A2 = IRB_A2;
360 A3 = IRB_A3;
361 D1 = IRB_D1;
362 D5 = IRB_D5;
363
364 if((res = hal_pin_float_new("irb.BA", HAL_IO, &(haldata->ba), comp_id)) < 0) goto error;
365 if((res = hal_pin_float_new("irb.BC", HAL_IO, &(haldata->bc), comp_id)) < 0) goto error;
366 if((res = hal_pin_float_new("irb.BD", HAL_IO, &(haldata->bd), comp_id)) < 0) goto error;
367 if((res = hal_pin_float_new("irb.BJ", HAL_IO, &(haldata->bj), comp_id)) < 0) goto error;
368 if((res = hal_pin_float_new("irb.DJ", HAL_IO, &(haldata->dj), comp_id)) < 0) goto error;
369 if((res = hal_pin_float_new("irb.FE", HAL_IO, &(haldata->fe), comp_id)) < 0) goto error;
370 if((res = hal_pin_float_new("irb.FG", HAL_IO, &(haldata->fg), comp_id)) < 0) goto error;
371 if((res = hal_pin_float_new("irb.FH", HAL_IO, &(haldata->fh), comp_id)) < 0) goto error;
372
373 BA = IRB_BA;
374 BC = IRB_BC;
375 BD = IRB_BD;
376 BJ = IRB_BJ;
377 DJ = IRB_DJ;
378 FE = IRB_FE;
379 FG = IRB_FG;
380 FH = IRB_FH;
381
382 if((res = hal_pin_float_new("irb.K1", HAL_IO, &(haldata->k1), comp_id)) < 0) goto error;
383 if((res = hal_pin_float_new("irb.K2", HAL_IO, &(haldata->k2), comp_id)) < 0) goto error;
384 if((res = hal_pin_float_new("irb.K3", HAL_IO, &(haldata->k3), comp_id)) < 0) goto error;
385 if((res = hal_pin_float_new("irb.K4", HAL_IO, &(haldata->k4), comp_id)) < 0) goto error;
386 if((res = hal_pin_float_new("irb.K5", HAL_IO, &(haldata->k5), comp_id)) < 0) goto error;
387 if((res = hal_pin_float_new("irb.K6", HAL_IO, &(haldata->k6), comp_id)) < 0) goto error;
388
389 K1 = IRB_K1;
390 K2 = IRB_K2;
391 K3 = IRB_K3;
392 K4 = IRB_K4;
393 K5 = IRB_K5;
394 K6 = IRB_K6;
395
396 res = hal_pin_float_new("irb.th1", HAL_OUT, &(haldata->th1), comp_id);
397 if(res < 0) goto error;
398 res = hal_pin_float_new("irb.th2", HAL_OUT, &(haldata->th2), comp_id);
399 if(res < 0) goto error;
400 res = hal_pin_float_new("irb.th3", HAL_OUT, &(haldata->th3), comp_id);
401 if(res < 0) goto error;
402 res = hal_pin_float_new("irb.th4", HAL_OUT, &(haldata->th4), comp_id);
403 if(res < 0) goto error;
404 res = hal_pin_float_new("irb.th5", HAL_OUT, &(haldata->th5), comp_id);
405 if(res < 0) goto error;
406
407 *(haldata->th1) = 0;
408 *(haldata->th2) = 0;
409 *(haldata->th3) = 0;
410 *(haldata->th4) = 0;

```

```

411 *(haldata->th5) = 0;
412
413 hal_ready(comp_id);
414 return 0;
415
416 error:
417 hal_exit(comp_id);
418 return res;
419 }
420
421 void rtapi_app_exit(void) { hal_exit(comp_id); }

```

Arquivo "irb.h" com os valores DH do manipulador ASEA IRB6-S2.

```

1  /*****
2  * Description: irb.h
3  *****/
4  * This is the header file of irb.c.
5  *****/
6  * Authors: Alberto Alvares – GIAI/UnB
7  *          Juan Toquica – UnB
8  * License: GPL Version 2
9  * System: Linux
10 *
11 * Copyright (c) 2015 All rights reserved.
12 *****/
13 */
14 #ifndef IRB_H
15 #define IRB_H
16
17 /* the default values for a IRB6 type robot, these can be changed as HAL parameters */
18 #define IRB_D1 700
19 #define IRB_A2 450
20 #define IRB_A3 670
21 #define IRB_D5 95
22
23 #define IRB_BA 190
24 #define IRB_BC 240
25 #define IRB_BD 140
26 #define IRB_BJ 450
27 #define IRB_DJ 430
28 #define IRB_FE 190
29 #define IRB_FG 240
30 #define IRB_FH 140
31
32 /* A cinemática dos atuadores/motores considerou reduções igual a 1 (k1=k2=k3=k4=k5=1),
33 e os valores das reduções foram compensadas através da configuração do arquivo de
34 configuração INI (irb.ini) do LinuxCNC, conforme o descrito no Capítulo 4 – "Retrofitting
35 do Robô ASEA IRB6-S2", na Seção 4.6.3 – "Calibração da Posição de Cada Junta"*/
36
37 #define IRB_K4 1.0
38 #define IRB_K5 1.0
39 #define IRB_K6 1.0
40 #define IRB_K1 1.0
41 #define IRB_K2 1.0
42 #define IRB_K3 1.0
43
44 #endif /* IRB_H */

```

K. ARQUIVO DO MODELO VIRTUAL

Neste apêndice se apresenta o código em linguagem Python do arquivo "irbgui", o qual permite gerar o modelo virtual e a simulação do manipulador ASEA IRB6-S2.

```
1  #!/usr/bin/python
2  # Authors: Alberto Alvares – GIAI/UnB
3  #           Juan Toquica – UnB
4  # License: GPL Version 2
5  # System: Linux
6
7  from vismach import *
8  import hal
9  import math
10 import sys
11
12 c = hal.component("irbgui")
13 c.newpin("joint0", hal.HAL_FLOAT, hal.HAL_IN)
14 c.newpin("joint1", hal.HAL_FLOAT, hal.HAL_IN)
15 c.newpin("joint2", hal.HAL_FLOAT, hal.HAL_IN)
16 c.newpin("joint3", hal.HAL_FLOAT, hal.HAL_IN)
17 c.newpin("joint4", hal.HAL_FLOAT, hal.HAL_IN)
18 c.ready()
19
20 floor = Collection([Box(-205,-205,-30,205,205,0)])
21 floor = Color([1,1,1,1],[floor])
22
23 work = Capture()
24
25 #tool goes here.. maybe later
26 tool = Capture()
27
28 # "tooltip" for backplot will be the tip of the tool, for now link7
29 tooltip = Capture()
30 tool = Collection([tooltip, tool])
31
32 # link 5
33 link5 = AsciiOBJ(filename="/home/asea/linuxcnc/configs/irb/5.obj")
34 link5 = Color([1,1,0,1],[link5])
35
36 link5 = Translate([link5],0,0,-20)
37 link5 = Collection([link5, tool])
38 link5 = Translate([link5],0,0,20)
39
40 link5 = HalRotate([link5],c,"joint4",1,0,0,1)
41
42 # link 4
43 link4 = AsciiOBJ(filename="/home/asea/linuxcnc/configs/irb/4.obj")
44 link4 = Color([0,1,1,1],[link4])
45 link4 = Rotate([link4],-90,0,1,0)
46 link4 = Translate([link4],20,0,-75)
47
48 link4 = Collection([link5, link4])
49 #apply HAL DOF
50 link4 = Translate([link4],0,0,75)
```

```

51
52 link4 = HalRotate ([ link4 ],c," joint3 ",1,1,0,0)
53
54 # link 3
55 link3 = AsciiOBJ(filename="/home/asea/linuxcnc/configs/irb/3.obj")
56 link3 = Color([0,0,1,1],[link3])
57 link3 = Rotate([link3],90,0,1,0)
58 link3 = Translate([link3],-50,-50,-50)
59 link3 = Collection([link4, link3])
60 #apply HAL DOF
61 link3 = Translate([link3],0,0,670)
62 link3 = Rotate([link3],-90,1,0,0)
63
64 link3 = HalRotate([link3],c," joint2 ",1,1,0,0)
65
66
67 # link 2
68 link2 = AsciiOBJ(filename="/home/asea/linuxcnc/configs/irb/2.obj")
69 link2 = Color([0,1,0,1],[link2])
70 link2 = Translate([link2],-95,-50,-450)
71
72 link2 = Collection([link3, link2])
73
74 link2 = Translate([link2],0,0,450)
75
76 #apply HAL DOF
77 link2 = HalRotate([link2],c," joint1 ",1,1,0,0)
78
79 # link 1
80 link1 = AsciiOBJ(filename="/home/asea/linuxcnc/configs/irb/1.obj")
81 link1 = Color([1,0,0,1],[link1])
82 link1 = Rotate([link1],90,0,0,1)
83 link1 = Translate([link1],0,0,-590)
84
85 #assemble
86 link1 = Collection([link2, link1])
87 #apply
88 link1 = Translate([link1],0,0,590)
89 link1 = HalRotate([link1],c," joint0 ",1,0,0,1)
90
91 # base
92 base = AsciiOBJ(filename="/home/asea/linuxcnc/configs/irb/base.obj")
93 base = Color([1,1,1,1],[base])
94 base = Rotate([base],90,1,0,0)
95 base = Translate([base],-205,205,-110)
96 base = Collection([link1, base])
97 base = Translate([base],0,0,110)
98
99 # stationary base
100 model = Collection([tooltip, base, floor, work])
101
102 main(model, tooltip, work,1000)

```

L. MODULO TEACH-IN

Neste apêndice se apresenta o modulo *teach-in*, em linguagem Python para gravar os dados de maquina no manipulador ASEA IRB6-S2 em um arquivo de edição.

```
1  #!/usr/bin/python
2  """ Usage:
3  python teach.py nmlfile outputfile
4  If outputfile is not specified, writes to standard output.
5  You must ". scripts/rip-environment" before running this script, if you use
6  run-in-place.
7  """
8  #   Copyright 2007 Jeff Epler <jepler@unpythonic.net>
9  #
10 #   This program is free software; you can redistribute it and/or modify
11 #   it under the terms of the GNU General Public License as published by
12 #   the Free Software Foundation; either version 2 of the License, or
13 #   (at your option) any later version.
14 #
15 #   This program is distributed in the hope that it will be useful,
16 #   but WITHOUT ANY WARRANTY; without even the implied warranty of
17 #   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
18 #   GNU General Public License for more details.
19 #
20 #   You should have received a copy of the GNU General Public License
21 #   along with this program; if not, write to the Free Software
22 #   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
23
24 import linuxcnc
25 import Tkinter
26 import sys
27
28 linenumber = 1;
29
30 #if len(sys.argv) > 1:
31 #   linuxcnc.nmlfile = sys.argv[1]
32
33 if len(sys.argv) > 1:
34   outfile = sys.argv[1]
35   sys.stdout = open(outfile, 'w')
36
37 s = linuxcnc.stat()
38
39 #def get_cart():
40 #   s.poll()
41 #   position = " ".join(["%-8.4f"] * s.axes)
42 #   return position % s.position[:s.axes]
43
44 def get_cart():
45   s.poll()
46   position = "X%-8.4f Y%-8.4f Z%-8.4f A%-8.4f B%-8.4f \n" % (s.position[0], s.position[1], s.position[2], s.positi
47   return position
48
```

```

49 #def get_joint():
50 #     s.poll()
51 #     position = " ".join(["%-8.4f"] * s.axes)
52 #     return position % s.joint_actual_position[:s.axes]
53
54 def get_joint():
55     s.poll()
56     position = "J0: %-8.4f J1: %-8.4f J2: %-8.4f J3: %-8.4f J4: %-8.4f \n" % (s.joint_actual_position[0], s.joint_a
57     return position
58
59
60 def log():
61     global linenumber;
62     if world.get():
63         p = get_cart()
64         print "N%-3.0f" % (linenumber*5), "G01", p;
65
66     else:
67         p = get_joint()
68         print "point", linenumber, "=>" , p;
69
70     label1.configure(text='Learned: %s' % p)
71     #     print linenumber, p, s.flood, s.mist, s.lube, s.spindle_enabled;
72     #     print "N%-3.0f" % (linenumber*5), "G01", p;
73
74     linenumber += 1;
75
76 def show():
77     s.poll()
78     if world.get():
79         p = get_cart()
80     else:
81         p = get_joint()
82     label2.configure(text='Position: %s' % p)
83     app.after(100, show)
84
85     app = Tkinter.Tk(); app.wm_title('LinuxCNC Teach-In')
86
87     world = Tkinter.IntVar(app)
88
89     button = Tkinter.Button(app, command=log, text='Learn', font=("helvetica", 14))
90     button.pack(side='left')
91
92     label2 = Tkinter.Label(app, width=60, font='fixed', anchor="w")
93     label2.pack(side='top')
94
95     label1 = Tkinter.Label(app, width=60, font='fixed', text="Learned: (nothing yet)", anchor="w")
96     label1.pack(side='top')
97
98     r1 = Tkinter.Radiobutton(app, text="Joint", variable=world, value=0)
99     r1.pack(side='left')
100    r2 = Tkinter.Radiobutton(app, text="World", variable=world, value=1)
101    r2.pack(side='left')
102
103    show()
104    app.mainloop()

```


M. ARQUIVO DE CONFIGURAÇÃO INI

Neste apêndice se apresenta o código em linguagem Python do arquivo "irb.ini", o qual permite configurar os parâmetros específicos do manipulador ASEA IRB6-S2, posteriormente lidos pela camada HAL no momento da execução do LinuxCNC.

```
1 # EMC controller parameters for a simulated and operated irb6-s2 robot
2 # General note: Comments can either be preceded with a # or ; - either is
3 # acceptable, although # is in keeping with most linux config files.
4 # Settings with a + at the front of the comment are likely needed to get
5 # changed by the user.
6 # Settings with a - at the front are highly unneeded to be changed
7 # Edited from puma 560 robot
8 # Prof. Alberto Alvares
9 # Juan Sebastian Toquica Arenas
10 # Copyright 2000-2016 LinuxCNC - GNU License.
11 #####
12 # General section
13 #####
14
15 # General section -----
16 [EMC]
17
18 #- Version of this INI file
19 VERSION =          $Revision$
20
21 #+ Name of machine, for use with display, etc.
22 MACHINE =          LinuxCNC-IRB6-Simulation
23
24 #+ Debug level, 0 means no messages. See src/emc/nml_int/emcglb.h for others
25 DEBUG = 0
26 # DEBUG =          0x00000007
27 #DEBUG =           0x7FFFFFFF
28
29 #####
30 # Sections for display options
31 #####
32 [DISPLAY]
33
34 #+ Name of display program, e.g., xemc
35 DISPLAY =          axis
36 # DISPLAY =         usrmot
37 # DISPLAY =         mini
38 # DISPLAY =         tkemc
39
40 #- Cycle time, in seconds, that display will sleep between polls
41 CYCLE_TIME =       0.200
42
43 #- Path to help file
44 HELP_FILE =        tklinucnc.txt
45
46 #- Initial display setting for position, RELATIVE or MACHINE
47 POSITION_OFFSET =   MACHINE
48
```

```

49 #- Initial display setting for position , COMMANDED or ACTUAL
50 POSITION_FEEDBACK =      ACTUAL
51
52 #+ Highest value that will be allowed for feed override , 1.0 = 100%
53 MAX_FEED_OVERRIDE =      2.0
54
55 #+ Prefix to be used
56 PROGRAM_PREFIX = /home/asea/linuxcnc/nc_files
57
58 #- Introductory graphic
59 INTRO_GRAPHIC = linuxcnc.gif
60 INTRO_TIME =      5
61 PYVCP =          irb_b.xml
62
63 # Editor to be used with Axis
64 EDITOR = gedit
65
66 #####
67 # Task controller section
68 #####
69 [TASK]
70
71 # Name of task controller program , e.g. , milltask
72 TASK =          milltask
73
74 #- Cycle time , in seconds , that task controller will sleep between polls
75 CYCLE_TIME =          0.010
76
77 #####
78 # Part program interpreter section
79 #####
80 [RS274NGC]
81
82 #- File containing interpreter variables
83 PARAMETER_FILE =          irb.var
84
85 #####
86 # Motion control section
87 #####
88 [EMCMOT]
89
90 EMCMOT =          motmod
91
92 #- Timeout for comm to emcmot , in seconds
93 COMM_TIMEOUT =          1.0
94
95 #- Interval between tries to emcmot , in seconds
96 COMM_WAIT =          0.010
97
98 #- Servo task period , in nanoseconds
99 SERVO_PERIOD =          1000000
100 #- Trajectory Planner task period , in nanoseconds - will be rounded to an
101 # integer multiple of SERVO_PERIOD
102 TRAJ_PERIOD =          10000000
103
104 BASE_PERIOD =          100000
105
106 #####
107 # Hardware Abstraction Layer section
108 #####

```

```

109 [HAL]
110
111 # The run script first uses halcmd to execute any HALFILE
112 # files , and then to execute any individual HALCMD commands.
113 #
114
115 # list of hal config files to run through halcmd
116 # files are executed in the order in which they appear
117 HALFILE =          irb_sim.hal
118 HALFILE =          vc-p4s.hal
119 POSTGUI_HALFILE =  irb_postgui.hal
120
121 #- list of halcmd commands to execute
122 # commands are executed in the order in which they appear
123 #HALCMD =          save neta
124
125 #load halui to enable
126 HALUI =           halui
127
128 #####
129 # HALUI Section
130 #####
131
132 [HALUI]
133
134 MDI_COMMAND = M101
135 MDI_COMMAND = M102
136 MDI_COMMAND = M103
137 MDI_COMMAND=...
138 MDI_COMMAND=...
139 MDI_COMMAND=...
140 MDI_COMMAND=G10 L20 P1 X0
141 MDI_COMMAND=G10 L20 P1 Y0
142 MDI_COMMAND=G10 L20 P1 Z0
143 MDI_COMMAND=G0 X0 Y0 Z0
144 MDI_COMMAND=G10 L20 P1 A0
145 MDI_COMMAND=G10 L20 P1 B0
146 MDI_COMMAND=G10 L20 P1 C0
147
148
149 #####
150 # Trajectory planner section
151 #####
152 [TRAJ]
153 #+ machine specific settings
154 AXES =            5
155 COORDINATES =    X Y Z A B
156 HOME =           0 0 0 0 0
157 LINEAR_UNITS =   mm
158 ANGULAR_UNITS =  deg
159 CYCLE_TIME =     0.010
160 DEFAULT_VELOCITY = 5.0
161 #MAX_VELOCITY =  10.0
162 DEFAULT_ACCELERATION = 3.00
163 #MAX_ACCELERATION = 5.00
164
165 #####
166 # Axes sections
167 #####
168

```

```

169 #+ First axis
170 [AXIS_0]
171 TYPE = ANGULAR
172 HOME = 0.0
173 #HOME = 0.0
174 #MAX_VELOCITY = 1.50
175 #MAX_ACCELERATION = 0.250
176 #STEPGEN_MAXACCEL = 0.3125
177 MAX_VELOCITY = 5.00
178 MAX_ACCELERATION = 5.00
179 STEPGEN_MAXACCEL = 6.25
180 SCALE = 450
181 BACKLASH = 0.000
182 #INPUT_SCALE = 4000
183 #OUTPUT_SCALE = 1.000
184 MIN_LIMIT = -360.0
185 MAX_LIMIT = 360.0
186 FERROR = 1.000
187 MIN_FERROR = 0.250
188 HOME_OFFSET = 70.0
189 HOME_SEARCH_VEL = 3.75
190 HOME_LATCH_VEL = -0.5
191 #HOME_OFFSET = 0.0
192 #HOME_SEARCH_VEL = 0.0
193 #HOME_LATCH_VEL = 0.0
194 HOME_USE_INDEX = NO
195 HOME_IGNORE_LIMITS = NO
196 HOME_SEQUENCE = 0
197
198 #+ Second axis
199 [AXIS_1]
200 TYPE = ANGULAR
201 HOME = -1.250
202 #HOME = 0.0
203 #MAX_VELOCITY = 1.5
204 #MAX_ACCELERATION = 0.250
205 #STEPGEN_MAXACCEL = 0.3125
206 MAX_VELOCITY = 10.00
207 MAX_ACCELERATION = 5.00
208 STEPGEN_MAXACCEL = 6.25
209 SCALE = -467
210 BACKLASH = 0.000
211 #INPUT_SCALE = 4000
212 #OUTPUT_SCALE = 1.000
213 MIN_LIMIT = -100.00
214 MAX_LIMIT = 100.00
215 FERROR = 1.000
216 MIN_FERROR = 0.250
217 HOME_OFFSET = -1.00
218 HOME_SEARCH_VEL = 0.7
219 HOME_LATCH_VEL = -0.5
220 #HOME_OFFSET = 0.0
221 #HOME_SEARCH_VEL = 0.0
222 #HOME_LATCH_VEL = 0.0
223 HOME_USE_INDEX = NO
224 HOME_IGNORE_LIMITS = NO
225 HOME_SEQUENCE = 1
226
227 #+ Third axis
228 [AXIS_2]

```

```

229 TYPE = ANGULAR
230 #HOME = 0.0
231 HOME = 1.750
232 #MAX_VELOCITY = 2.00
233 #MAX_ACCELERATION = 0.250
234 MAX_VELOCITY = 10.00
235 MAX_ACCELERATION = 5.00
236 BACKLASH = 0.000
237 #STEPGEN_MAXACCEL = 0.3125
238 STEPGEN_MAXACCEL = 6.25
239 SCALE = -495
240 #INPUT_SCALE = 4000
241 #OUTPUT_SCALE = 1.000
242 MIN_LIMIT = -40.00
243 MAX_LIMIT = 10.00
244 FERROR = 1.000
245 MIN_FERROR = 0.200
246 HOME_OFFSET = 0.250
247 HOME_SEARCH_VEL = 0.7
248 HOME_LATCH_VEL = -0.5
249 #HOME_OFFSET = 0.0
250 #HOME_SEARCH_VEL = 0.0
251 #HOME_LATCH_VEL = 0.0
252 HOME_USE_INDEX = NO
253 HOME_IGNORE_LIMITS = NO
254 HOME_SEQUENCE = 2
255
256 #+ Fourth axis
257 [AXIS_3]
258 TYPE = ANGULAR
259 #HOME = 0.00
260 HOME = -19.00
261 #MAX_VELOCITY = 2.00
262 #MAX_ACCELERATION = 0.250
263 MAX_VELOCITY = 10.00
264 MAX_ACCELERATION = 5.00
265 BACKLASH = 0.000
266 #STEPGEN_MAXACCEL = 0.3125
267 STEPGEN_MAXACCEL = 6.25
268 SCALE = 370
269 #INPUT_SCALE = 4000
270 #OUTPUT_SCALE = 1.000
271 MIN_LIMIT = -200.00
272 MAX_LIMIT = 200.0
273 FERROR = 1.000
274 MIN_FERROR = 0.200
275 #HOME_OFFSET = 0.00
276 #HOME_SEARCH_VEL = 0.00
277 #HOME_LATCH_VEL = 0.00
278 HOME_OFFSET = 1.00
279 HOME_SEARCH_VEL = 1.0
280 HOME_LATCH_VEL = 0.8
281 HOME_USE_INDEX = NO
282 HOME_IGNORE_LIMITS = NO
283 HOME_SEQUENCE = 3
284
285 #+ Fifth axis
286 [AXIS_4]
287 TYPE = ANGULAR
288 HOME = 0.000

```

```

289 #HOME = -10.000
290 MAX_VELOCITY = 10.00
291 MAX_ACCELERATION = 5.00
292 BACKLASH = 0.000
293 #STEPGEN_MAXACCEL = 6.25
294 STEPGEN_MAXACCEL = 6.25
295 SCALE = 220
296 #INPUT_SCALE = 4000
297 #OUTPUT_SCALE = 1.000
298 MIN_LIMIT = -200.00
299 MAX_LIMIT = 180.0
300 FERROR = 1.000
301 MIN_FERROR = 0.200
302 #HOME_OFFSET = -1.0
303 #HOME_SEARCH_VEL = 1.00
304 #HOME_LATCH_VEL = -0.80
305 HOME_OFFSET = 0.000
306 HOME_SEARCH_VEL = 0.00
307 HOME_LATCH_VEL = 0.00
308 HOME_USE_INDEX = NO
309 HOME_IGNORE_LIMITS = NO
310 HOME_SEQUENCE = 4
311
312
313 #####
314 # section for main IO controller parameters
315 #####
316 [EMCIO]
317
318 #- Name of IO controller program, e.g., io
319 EMCIO = io
320
321 #- cycle time, in seconds
322 CYCLE_TIME = 0.100
323
324 #- tool table file
325 TOOL_TABLE = irb.tbl

```

N. ARQUIVO DE CONFIGURAÇÃO HAL

Neste apêndice se apresenta o código em linguagem Python do arquivo "irb_sim.hal", o qual permite configurar e interligar virtualmente na camada HAL, as variáveis dos componentes *hardware* do sub-sistema de controle, integrados ao *retrofitting* do manipulador ASEA IRB6-S2.

```
1 # core HAL config file for simulation - 5 axis
2 # ASEA IRB6-S2 ROBOT
3 # Prof. Alberto Alvares - UnB
4 # Juan Sebastian Toquica Arenas - UnB
5 # Copyright 2000-2016 LinuxCNC - GNU License.
6 #####
7 # load RT modules
8 loadrt irb
9 loadrt [EMCMOT]EMCMOT
10 base_period_nsec=[EMCMOT]BASE_PERIOD
11 servo_period_nsec=[EMCMOT]SERVO_PERIOD
12 traj_period_nsec=[EMCMOT]TRAJ_PERIOD
13 num_joints=[TRAJ]AXES
14
15 loadrt debounce cfg=2
16
17 loadrt probe_parport
18 loadrt hal_parport cfg="0x378 out"
19 setp parport.0.reset-time 1000
20 loadrt stepgen step_type=0,0,0,0,0
21
22 addf parport.0.read base-thread
23 addf stepgen.make-pulses base-thread
24 addf parport.0.write base-thread
25 addf parport.0.reset base-thread
26
27 addf debounce.0 base-thread
28 setp debounce.0.delay 15
29
30 addf stepgen.capture-position servo-thread
31 addf stepgen.update-freq servo-thread
32
33 net spindle-cmd <= motion.spindle-speed-out
34
35 #in/out pins definition
36
37 #outputs pins
38
39 #net estop-out => parport.0.pin-01-out
40
41 net xdir => parport.0.pin-03-out
42 net xstep => parport.0.pin-02-out
43 setp parport.0.pin-02-out-reset 1
44
45 net ydir => parport.0.pin-05-out
46 net ystep => parport.0.pin-04-out
47 setp parport.0.pin-04-out-reset 1
48
```

```

49 net zdir => parport.0.pin-07-out
50 net zstep => parport.0.pin-06-out
51 setp parport.0.pin-06-out-reset 1
52
53 net adir => parport.0.pin-09-out
54 net astep => parport.0.pin-08-out
55 setp parport.0.pin-08-out-reset 1
56
57 net bdir => parport.0.pin-17-out
58 net bstep => parport.0.pin-14-out
59 setp parport.0.pin-14-out-reset 1
60
61 #inputs pins
62
63 net home-x <= parport.0.pin-13-in-not
64 net home-y <= parport.0.pin-11-in-not
65
66 #Home Sensor Joint 2
67 net sw-in2 parport.0.pin-12-in-not => debounce.0.1.in
68
69 #Home Sensor Joint 3 e 4
70 net sw-in3 parport.0.pin-15-in-not => debounce.0.0.in
71
72 #AXIS parameter definitions
73 #AXIS_0
74
75 setp stepgen.0.position-scale [AXIS_0]SCALE
76 setp stepgen.0.steplen 1
77 setp stepgen.0.stepspace 0
78 setp stepgen.0.dirhold 15200
79 setp stepgen.0.dirsetup 15200
80 setp stepgen.0.maxaccel [AXIS_0]STEPGEN_MAXACCEL
81 net xpos-cmd axis.0.motor-pos-cmd => stepgen.0.position-cmd
82 net xpos-fb stepgen.0.position-fb => axis.0.motor-pos-fb
83 net xstep <= stepgen.0.step
84 net xdir <= stepgen.0.dir
85 net xenable axis.0.amp-enable-out => stepgen.0.enable
86
87 net home-x => axis.0.home-sw-in
88
89 #AXIS_1
90
91 setp stepgen.1.position-scale [AXIS_1]SCALE
92 setp stepgen.1.steplen 1
93 setp stepgen.1.stepspace 0
94 setp stepgen.1.dirhold 15200
95 setp stepgen.1.dirsetup 15200
96 setp stepgen.1.maxaccel [AXIS_1]STEPGEN_MAXACCEL
97 net ypos-cmd axis.1.motor-pos-cmd => stepgen.1.position-cmd
98 net ypos-fb stepgen.1.position-fb => axis.1.motor-pos-fb
99 net ystep <= stepgen.1.step
100 net ydir <= stepgen.1.dir
101 net yenable axis.1.amp-enable-out => stepgen.1.enable
102
103 net home-y => axis.1.home-sw-in
104
105 #AXIS_2
106
107 setp stepgen.2.position-scale [AXIS_2]SCALE
108 setp stepgen.2.steplen 1

```



```

109 setp stepgen.2.stepspace 0
110 setp stepgen.2.dirhold 15200
111 setp stepgen.2.dirsetup 15200
112 setp stepgen.2.maxaccel [AXIS_2]STEPGEN_MAXACCEL
113 net zpos-cmd axis.2.motor-pos-cmd => stepgen.2.position-cmd
114 net zpos-fb stepgen.2.position-fb => axis.2.motor-pos-fb
115 net zstep <= stepgen.2.step
116 net zdir <= stepgen.2.dir
117 net zenable axis.2.amp-enable-out => stepgen.2.enable
118
119 net sw-debounced2 debounce.0.1.out => axis.2.home-sw-in
120
121 #AXIS_3
122
123 setp stepgen.3.position-scale [AXIS_3]SCALE
124 setp stepgen.3.steplen 1
125 setp stepgen.3.stepspace 0
126 setp stepgen.3.dirhold 15200
127 setp stepgen.3.dirsetup 15200
128 setp stepgen.3.maxaccel [AXIS_3]STEPGEN_MAXACCEL
129 net apos-cmd axis.3.motor-pos-cmd => stepgen.3.position-cmd
130 net apos-fb stepgen.3.position-fb => axis.3.motor-pos-fb
131 net astep <= stepgen.3.step
132 net adir <= stepgen.3.dir
133 net enable axis.3.amp-enable-out => stepgen.3.enable
134
135 net sw-debounced3 debounce.0.0.out => axis.3.home-sw-in
136
137 #AXIS_4
138
139 setp stepgen.4.position-scale [AXIS_4]SCALE
140 setp stepgen.4.steplen 1
141 setp stepgen.4.stepspace 0
142 setp stepgen.4.dirhold 15200
143 setp stepgen.4.dirsetup 15200
144 setp stepgen.4.maxaccel [AXIS_4]STEPGEN_MAXACCEL
145 net bpos-cmd axis.4.motor-pos-cmd => stepgen.4.position-cmd
146 net bpos-fb stepgen.4.position-fb => axis.4.motor-pos-fb
147 net bstep <= stepgen.4.step
148 net bdir <= stepgen.4.dir
149 net benable axis.4.amp-enable-out => stepgen.4.enable
150
151 net sw-debounced3 debounce.0.0.out => axis.4.home-sw-in
152
153 # create a signal for the estop loopback
154 net estop-out <= iocontrol.0.user-enable-out
155 net estop-out => iocontrol.0.emc-enable-in
156
157 # create a signals for manual tool-change
158 loadusr -W hal_manualtoolchange
159 net tool-change iocontrol.0.tool-change => hal_manualtoolchange.change
160 net tool-changed iocontrol.0.tool-changed <= hal_manualtoolchange.changed
161 net tool-number iocontrol.0.tool-prep-number => hal_manualtoolchange.number
162 net tool-prepare-loopback iocontrol.0.tool-prepare => iocontrol.0.tool-prepared
163
164
165 #####
166
167 #add motion controller functions to servo thread
168 addf motion-command-handler servo-thread

```

```
169 addf motion-controller servo-thread
170
171 loadusr -W irbgui
172
173 loadrt scale count=5
174
175 addf scale.0 servo-thread
176 addf scale.1 servo-thread
177 addf scale.2 servo-thread
178 addf scale.3 servo-thread
179 addf scale.4 servo-thread
180
181 net xpos-fb scale.0.in
182 net ypos-fb scale.1.in
183 net zpos-fb scale.2.in
184 net apos-fb scale.3.in
185 net bpos-fb scale.4.in
186
187 setp scale.0.gain 1
188 setp scale.1.gain 1
189 setp scale.2.gain 1
190 setp scale.3.gain 1
191 setp scale.4.gain 1
192
193 net J0scaled scale.0.out irbgui.joint0
194 net J1scaled scale.1.out irbgui.joint1
195 net J2scaled scale.2.out irbgui.joint2
196 net J3scaled scale.3.out irbgui.joint3
197 net J4scaled scale.4.out irbgui.joint4
```