HUGO MAKOTO KAMODA
RODRIGO SATO HANADA

nota final 9,0
(nove e zero)

CONTROLE E ACIONAMENTO DE UM ROBÔ PARALELO

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro

São Paulo
2006

HUGO MAKOTO KAMODA
RODRIGO SATO HANADA

CONTROLE E ACIONAMENTO DE UM ROBÔ PARALELO

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro

Área de Concentração:
Engenharia Mecatrônica

Orientador:
Prof. Dr.
Tarcisio Antonio Hess Coelho

São Paulo
2006

FICHA CATALOGRÁFICA

Kamoda, Hugo Makoto
Controle e acionamento de um robô paralelo /H.M. Kamoda, R.S. Hanada. -- São Paulo, 2006.
p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Tempo-real 2.Cinemática 3.Robôs (Controle) 4.Arquiteturas paralelas I.Hanada, Rodrigo Sato II.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos III.t.

Agradecimentos

Agradecemos aos nossos pais pelo constante apoio durante toda essa jornada, Professor Dr. Tarcísio Hess Coelho pela colaboração e orientação neste trabalho assim como nossos colegas pelo apoio. Em especial o colega Marco Aurélio Lins Gomes pelo acompanhamento e pela consultoria prestada na formatação desta.

Resumo

O interesse em desenvolver e dar continuidade ao projeto de um manipulador robótico baseado em um mecanismo de cinemática paralela deve-se ao seu grande potencial para aplicações em equipamentos como "pick and place" e em máquinas-ferramentas. Comparativamente à arquitetura serial, a paralela apresenta alta capacidade de carga, alta rigidez, rapidez, precisão e leveza.

No projeto inicial foram utilizados motores de corrente contínua, porém nesse projeto serão utilizados motores de passo. O motivo de se utilizar motores de passo ao invés dos motores de corrente contínua utilizados originalmente baseia-se no fato de que controlar motores de passos é mais fácil, apresentando algumas vantagens.

Este projeto visa desenvolver o controle de um robô paralelo com motores de passo a partir de um microcomputador, de modo a realizar posicionamento em espaço tridimensional.

ABSTRACT

It is common to find machines using serial kinematics mechanism, where each part of the machine has to support the next parts. Because of that, the mechanism demands heavy and strong structure to keep the stiffness. Otherwise the parallel mechanism comes as a great solution to provide stiffness, fast movements, precise and high accelerations with a high relation on load/weight because of the fact that the weight is supported by more than one arm.

The actuators used in this project are step motors opposite to the DC motors used in the first project because of the facility of the control. Another advantage is that step motors present better speed and acceleration working accuracy.

With a driver, the step motors and a computer it will be possible to control a parallel mechanism in a 3D space.

1. Introdução

Dependendo do conceito a partir do qual foram construídas, as máquinas-ferramenta automatizadas - robôs industriais, tornos, fresas etc. - realizam seus movimentos de forma seqüencial ou simultânea. O movimento simultâneo, tecnicamente conhecido como cinemática paralela, é muito mais rápido e preciso, mas muito mais difícil de ser controlado.

No universo das máquinas-ferramenta, uma estrutura serial é aquela na qual cada movimento vem sempre depois do anterior - é assim que os robôs industriais funcionam - cada eixo movendo-se depois que o anterior completou seu movimento.

Esse movimento em série limita a velocidade dos equipamentos. Resolver esse problema significa incorporar complexos circuitos adicionais, que encarecem os equipamentos e os tornam maiores e menos flexíveis.

A cinemática paralela oferece uma solução que apresenta algumas vantagens. Como o equipamento de cinemática paralela tem que mover uma massa muito menor, ele acelera muito mais rápido do que as máquinas convencionais. As cargas e forças podem ser geradas simultaneamente pelas diversas pernas, o que causa uma deformação muito menor, ou seja, aumenta-se a precisão do equipamento.

Para facilitar a sua compreensão, o robô pode ser dividido em três grandes subsistemas: subsistema mecânico, de controle e eletrônico.

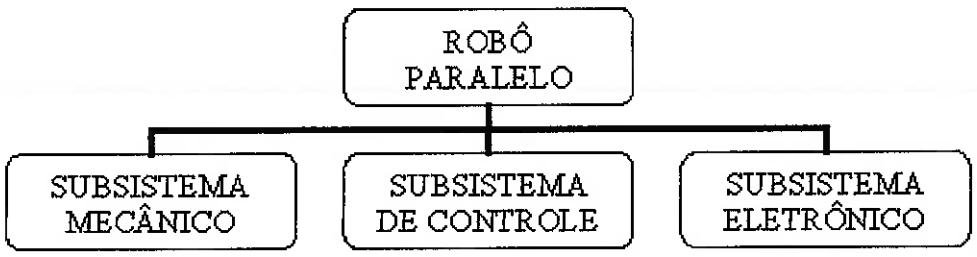


Figura 1.1: Subsistemas do robô.

1.1. Subsistema Mecânico

Inclui o mecanismo, o acoplamento entre os motores e o mecanismo e a fixação dos motores à base.

1.2. Subsistema de Controle

Este subsistema pode, ainda, ser dividido em software, que é composto pelo sistema operacional, software de controle e software de pré-processamento da cinemática inversa, e hardware, formado pelo computador e pela porta de comunicação paralela.

1.3. Subsistema Eletrônico

É composto pelos atuadores, pelo efetuador e por seus respectivos circuitos de acionamento.

2. Objetivos

No presente trabalho realizou-se a familiarização com o mecanismo de cinemática paralela, montagem do driver acionador dos motores de passo, testes de funcionamento da placa e instalação e familiarização com o EMC e o RTLinux. Os objetivos são: estudo de manipuladores robóticos cuja arquitetura se baseia em uma cinemática paralela, controle por motores de passo, domínio do programa EMC e do RTLinux, domínio do driver acionador dos motores de passo, de modo a realizar operações de posicionamento de objetos num espaço tridimensional.

3. Revisão da literatura

3.1. Mecanismos paralelos

"Um *mecanismo de cinemática paralela* pode ser definido como sendo um mecanismo de *cadeia fechada* composto por duas ou mais *cadeias cinemáticas* que conectam a plataforma móvel do efetuador a uma base fixa." (BRANCHINI, 2004, p. 7).

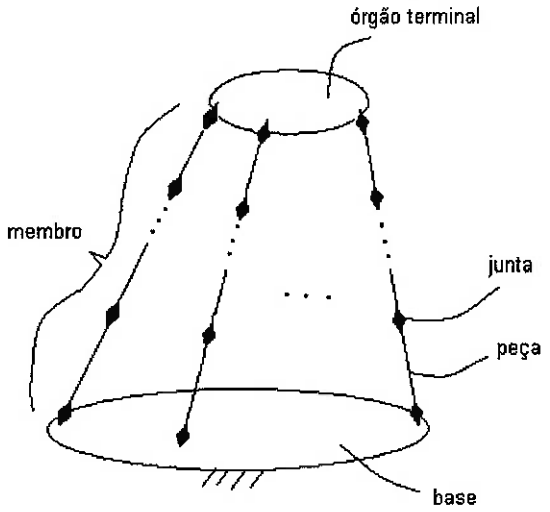


Figura 3.1: Mecanismo paralelo.

Uma cadeia cinemática é um sistema composto por peças que são conectados por intermédio de juntas ou pares cinemáticos. Sendo que uma junta define o movimento relativo entre essas duas peças; e que o tipo de movimento de uma junta é definido pelo o número de graus de liberdade que ela permite ou restringe.

Duas cadeias são dependentes quando o movimento de uma cadeia é determinado pelo da outra. Por outro lado, consideram-se duas cadeias como independentes, no caso em que o movimento de uma cadeia não é afetado pelo movimento da outra.

Um mecanismo é dito como sendo paralelo devido a sua forma de atuação ou acionamento do mecanismo.

O termo cadeia fechada significa que as suas duas extremidades encontram-se unidas, entretanto, quando as duas extremidades da cadeia estão separadas, a cadeia é denominada aberta. (BRANCHINI, 2004, p. 4)

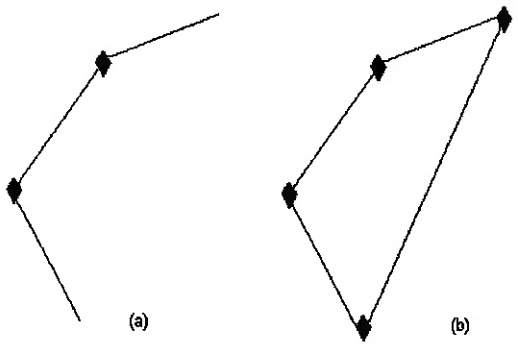


Figura 3.2: Cadeias cinemáticas: (a) aberta; (b) fechada.

Figura 3.3: Estruturas: (a) paralela; (b) serial.

3.2. Aplicações de mecanismos paralelos

Existem hoje diversas aplicações para os mecanismos paralelos, desde a sua aplicação no entretenimento como em brinquedos de parques de diversões, até robustos robôs para as indústrias.

Em 1928, James E. Gwinnett patenteou a primeira plataforma móvel para a indústria de entretenimento com uma grande visão de futuro que posteriormente seria utilizada para a construção de simuladores de vôos e plataformas móveis para cinemas "*in motion*" (www.parallemic.org).

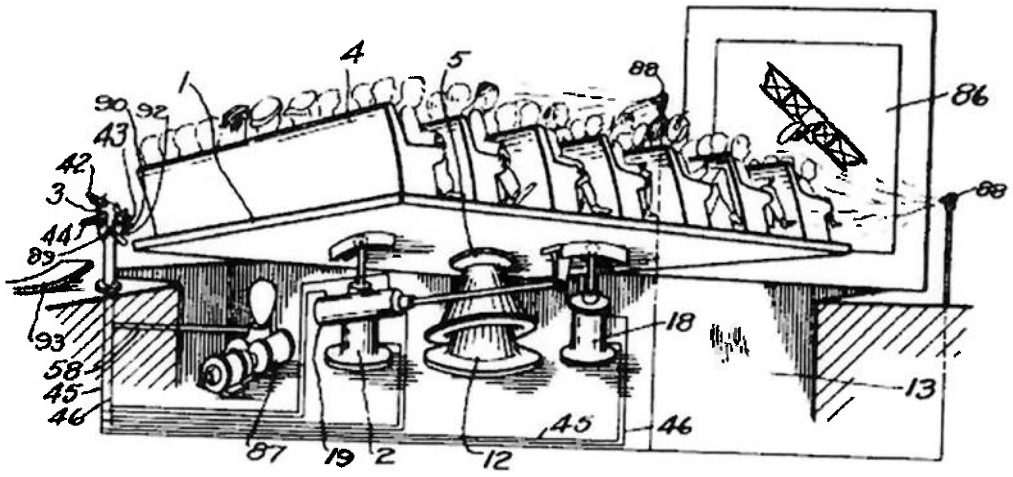


Figura 3.4: Possivelmente o primeiro mecanismo paralelo espacial patenteado em 1931. (US Patent No. 1,789,680)

Na indústria esses tipos de mecanismos são muito utilizados em máquinas CNC de manufatura, prototipagem rápida de peças, manipulação de peças, usinagens. Fora essas aplicações na indústria, temos a sua utilização na medicina como microscópios de nano precisão e dispositivos de cirurgias e assistência à distância e até em mecanismos para testes de pneus.

Figura 3.5: Mecanismo para testes de pneus da Dunlop.

4. Etapas do projeto

Neste trabalho foram realizados a seleção dos motores e do programa a ser utilizado para controle do robô, familiarização do programa, montagem e teste da placa acionadora, estudo da cinemática envolvida no mecanismo paralelo, acoplamento elétrico e mecânico dos motores de passo, instalação e configuração do sistema operacional e do software de controle, instalação de efetuador e testes finais.

4.1. Subsistema Mecânico

4.1.1. Mecanismo

Foram feitas apenas as alterações necessárias para a fixação dos motores de passo à base, acoplamento dos motores ao mecanismo e fixação do efetuador. Portanto, o robô não teve nenhuma parte de seu mecanismo modificada, de forma que a sua cinemática inversa continua a mesma.

4.2. Subsistema de Controle

4.2.1. Seleção do programa a ser utilizado no controle

Das plataformas a serem escolhidas tinha-se o EMC, Enhanced Machine Controller, que funciona em ambiente LINUX e utiliza-se do RTLinux (Real Time Linux) e a outra plataforma é o MatLab/Simulink com a ferramenta de operação em tempo real.

- Matlab/Simulink e Real Time Linux (RTLinux).

O software computacional *Matlab* possui uma ferramenta chamada *Simulink*, que permite modelar sistemas dinâmicos através de diagramas de blocos. O *Simulink* possui um recurso chamado *Real Time Workshop*, que gera um código fonte em lingüagem C correspondente ao sistema modelado. A idéia é modelar um controlador para o robô paralelo e utilizar o código fonte gerado com o robô. Porém, para que o controlador possa ser utilizado, é necessário que o programa seja executado em tempo real. Para isso, podemos utilizar um computador com o ambiente *Linux* e instalar um pacote chamado *Real Time Linux* (RTLinux), que permite ao sistema operaracional a execução em tempo real.

A vantagem desta alternativa é que o código fonte pode ser obtido sem a necessidade de um aprofundamento muito grande em linguagem C. A desvantagem é que nós temos pouca experiência com o ambiente *Linux*. Portanto, um estudo do pacote RTLinux será necessário para sua instalação, configuração e utilização, aumentando o tempo de implementação da alternativa.

Inicialmente ambos os sistemas eram desconhecidos e mereciam ser estudados cautelosamente. No entanto, o EMC já é um programa próprio para o controle de mecanismos diversos, apresenta uma interface gráfica amistosa e encontra-se bastante material na internet a respeito. Uma dificuldade é a familiarização com o Linux e a ferramenta RTLinux de difícil entendimento, instalação e configuração. Os pontos cruciais para a determinação do sistema a ser utilizado foi o fato de o EMC apresentar a interface gráfica que contém um painel de fácil utilização para manipulação dos motores de passo, a não necessidade de nenhuma programação para a comunicação com a porta paralela e o desafio de utilizar um sistema operacional desconhecido pelos componentes da dupla até então.

4.2.2. EMC2 (Enhanced Machine Controller 2).

O EMC2 (Enhanced Machine Controller 2) é um software livre que permite controlar, via computador, desde máquinas como fresadoras, tornos ou até robôs hexapod, dependendo da forma como ele é configurado.

Ele é composto por quatro componentes principais: um controlador de movimentação (EMCMOT) e um controlador de IO (EMCIO), coordenados por um componente chamado EMCTASK, e um conjunto de interfaces gráficas.

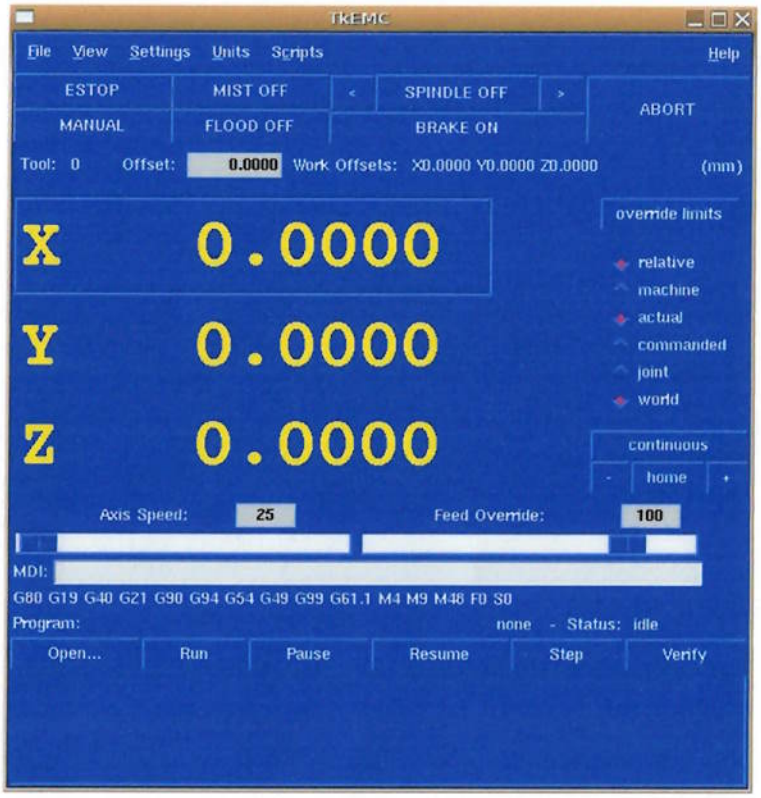


Figura 4.1: Interface gráfica com o usuário.

Por default, o EMC vem configurado para controlar uma máquina de 3 eixos com cinemática inversa trivial como, por exemplo, uma fresadora. No caso de um robô paralelo, porém, a cinemática inversa é bem mais complexa e exigirá uma configuração específica do software, feita através de 6 arquivos:

- *Arquivo com extensão .tbl*:

  Se o EMC estiver controlando um centro de usinagem CNC, este arquivo guarda todas as informações relativas às ferramentas, tais como posições no carrossel, diâmetros e comprimentos. Como queremos controlar um robô, este arquivo não será utilizado.

- *Arquivo com extensão .var:*

  Este arquivo armazena valores de offset para todos os eixos. Para o nosso projeto, também não foi necessário editá-lo.

- *Arquivo com extensão .nml:*

  Este arquivo possibilita configurar o EMC para acesso remoto e, portanto, não será utilizado.

- *Arquivo com extensão .ini:*

É um dos arquivos de configuração principal. Nele, todos os parâmetros da máquina como número de eixos, unidades e velocidades limites estão agrupados por seções e têm um nome associado a um valor, como ilustrado abaixo.

[TRAJ]

AXES = 3

MAX_VELOCITY = .4

Como vemos, o nome de uma seção é definido entre colchetes e, na linha imediatamente abaixo, começam as declarações dos parâmetros pertencentes a ela. Cada parâmetro e seu respectivo valor ocupam uma linha e todas as linhas antecedidas por ponto-e-vírgula ou pelo caractere # são consideradas comentários e, portanto, ignoradas pelo EMC2.

isto é um comentário

; isto é outro comentário

Editando-se os valores altera-se o comportamento do software, permitindo que ele controle a máquina desejada.

- Principais parâmetros de configuração do arquivo *.ini*.
  - Seção [EMC].

    VERSION = $Revision: 1.3$

    Versão do arquivo *.ini*

    NML_FILE = emc.nml

    Nome do arquivo *.nml* que deve ser utilizado pelo controlador.

DEBUG = 0x00000003

Define que tipos de mensagens de debug serão exibidas.

  - Seção [DISPLAY].

    DISLAY = tkemc

    Define a interface do programa com o usuário. Há 6 opções: emcpanel, keysick, tkemc, tkemcts, xemc e yemc.

    CYCLE_TIME = 0.200

    Período de atualização da tela de interface, em segundos.

  - Seção [TASK].

TASK = minimilltask

Define o interpretador de código G. Há duas opções: minimilltask e bridgeporttask.

- Seção [EMCMOT].

  EMCMOT = freqmod.o

  Define o modulo de controle de movimento. Para motores de passo, há três opções: steppermod.o, smdromod.o e freqmod.o, que é o mais geral de todos.

- Seção [HAL]

HALFILE = standard_pinout.hal

Define o arquivo *.hal* que deve ser utilizado.

- Seção [TRAJ].

  AXES = 3

  Número de eixos da máquina.

COORDINATES = X Y Z

Nomes dos eixos.

HOME = 0 0 0

Coordenadas da posição inicial de cada eixo.

LINEAR_UNITS = 1

Este parâmetro determina em que unidade o EMC deve interpretar medidas lineares enviadas por programas externos. Colocando em 1, ele estará configurado para receber unidades em mm. Se os programas externos trabalharem com polegadas, o valor deve ser 1/25,4 = 0.03937007874016.

ANGULAR_UNITS = 1

Análogo ao parâmetro acima, mas para unidade angular. Configurando-o para 1, significa que o software irá receber valores de ângulos em graus. Para radianos, deve-se utilizar o valor PI/180=0.01745329252167.

DEFAULT_VELOCITY = 1
Velocidade inicial para movimentação dos eixos em (unidades do usuário)/segundo.

MAX_VELOCITY = 10
Máxima velocidade dos eixos em (unidades do usuário)/segundo.

DEFAULT_ACCELERATION = 5
Aceleração inicial dos eixos em (unidades do usuário)/segundo^2.

MAX_ACCELERATION = 5
Aceleração máxima dos eixos em (unidades do usuário)/segundo^2.

O EMC permite que cada eixo seja configurado individualmente. Assim, as seções [AXIS_0], [AXIS_1], [AXIS_2]...[AXIS_N-1], onde N é o número de eixos definidos na seção [TRAJ], dizem respeito a cada um dos eixos.

- Seção [AXIS_0]
  TYPE = ANGULAR
  Tipo do eixo. As opções são: LINEAR ou ANGULAR.

  UNITS = 1
  Unidade em mm do eixo, para um eixo linear, ou em graus para um eixo do tipo angular. No caso, ele está configurado para medidas em graus.

  P = 50
  Valor do ganho do compensador proporcional, que é usado pelo controlador freqmod (seção [EMCMOT]) para calcular a freqüencia de clock dos motores de passo.
  MIN_LIMIT = -1000
  Limite mínimo, em unidades do usuário, para a movimentação do eixo.

MAX_LIMIT = 1000

Limite máximo, em unidades do usuário, para a movimentação do eixo.

INPUT_SCALE = 4000

Número de pulsos que deve ser dado ao motor de passo para que o eixo se desloque de uma UNIT.

FERROR = 1.0

Máximo erro permitido entre a posição comandada e a atual.

MIN_FERROR = 0.01

Tem a mesma função de FERROR, mas para velocidades muito baixas.

- Seção [EMCIO]

EMCIO = minimill

Nome do controlador de IO.

- Arquivo em extensão *.hal.*

Neste arquivo são definidas as configurações de hardware do computador, como endereço da porta de comunicação e quais pinos da porta serão utilizados. São definidos também todos os 6 sinais de controle (um sinal de clock e outro de direção para cada um dos 3 atuadores). Diferentemente de um tipo ini, um arquivo tipo hal é configurado através de comandos. Abaixo, temos um exemplo para o nosso projeto:

```
# endereço da porta de comunicação com o robô
loadrt hal_parport cfg="0x0378"
# cria threads
addf parport.0.read base-thread 1
addf parport.0.write base-thread -1
# conecta os sinais de controle aos pinos da porta paralela
linksp Astep parport.0.pin-03-out
linksp Adir parport.0.pin-02-out
linksp Bstep parport.0.pin-05-out
linksp Bdir parport.0.pin-04-out
```

linksp Cstep parport.0.pin-07-out
linksp Cdir parport.0.pin-06-out

- Arquivo com extensão *.emcsh:*

  Aqui é gravado o nome do diretório em que estão todos os arquivos de configuração modificados pelo usuário. Assim que o EMC2 é aberto, é exibida uma tela em que se pode escolher o arquivo *.emcsh* a ser carregado pelo programa.

4.2.3. Sistema operacional para o EMC2

Para que utilizar o EMC2 é necessário que esteja instalado no computador uma distribuição do linux que a princípio pode ser qualquer um.

Devido a sugestões presentes no site do EMC, a distribuição escolhida para que fosse o sistema operacional base para desenvolvimento do projeto, foi o Ubuntu versão 6.06 LTS.

Essa versão de instalação do Ubuntu está disponível em www.ubuntu.com na área de *downloads* e pode ser baixado de forma gratuita. Também existe a opção de solicitar a remessa via correio na página https://shipit.ubuntu.com no link *I want to request free CDs of the Ubuntu 6.06 LTS release*.

Para a instalação do linux, basta fazer o boot do CD no computador depois de configurar adequadamente a BIOS do sistema para isso. Após a inicialização do CD, você será levado para uma página onde haverá um link para instalar o sistema operacional no seu computador. Nesse ponto é possível utilizar o linux sem o instalar já que ele é um *LIVE-CD* (distribuição onde é possível utilizar o linux sem instalar no sistema, funcionado através do CD e utilizando os recursos do seu computador) porém, não é possível utilizar o EMC através do linux funcionando no modo *LIVE*.

Após decidir por instalar o Ubuntu, basta seguir as instruções de instalação, fazer o devido particionamento do HD, caso haja necessidade de rodar outro sistema operacional (windows por exemplo). As instruções podem estar em português se escolhido a língua no momento da instalação.

Para o projeto o HD foi particionado de forma com que se alocasse 1GB para a SWAP (tipo de partição) e 5GB para o sistema.

4.2.4. Instalação do EMC2

O EMC2 está disponível em www.linuxcnc.org em algumas versões. A versão adotada para o projeto foi o EMC2, chamado muitas vezes apenas de EMC no decorrer do projeto. A ferramenta em tempo real foi o RTAPI (Real Time Application Programming Interface) operando sob a distribuição Ubuntu 6.06 do Linux.

Sua instalação consistiu basicamente em três fases: Pré-instalação do EMC pré-compilado com o pacote RTAPI, instalação do EMC a partir do *source code* e a configuração da porta paralela.

A pré-instalação executada a partir da versão *dapper-install* (próprio para o Ubutu 6.06, disponível em http://linuxcnc.org/dapper/emc2-install.sh) serviu apenas para a instalação e configuração do RTAPI e a instalação de alguns pacotes básicos para o EMC2.

Após isso, no prompt do linux foi utilizado o apt para instalação dos pacotes necessários para reconstruir o emc2 a partir do *source code*. Os comandos que foram digitados no prompt do linux para isso foi:

sudo apt-get build-dep emc2

Logo após isso foi instalado o pacote auxiliar de eixos, com o seguinte comando:

sudo apt-get build-dep emc2-axis

Finalizando com a instalação do cliente CVS:

sudo apt-get install cvs

Para atualizar o EMC2 com a última versão CVS, digitou-se:

cvs -z5 -d:ext:anon@cvs.linuxcnc.org:/cvs co -rRELEASE_2_0_4 -d emc2-2.0.4 emc2

Finaliza-se o processo com a recompilação do emc2 com os comandos digitados no diretório src:

./configure –enable-run-in-place

sudo make && sudo make setuid

Por fim a configuração da porta paralela se deu com a substituição da versão do hal_parport.c por um outro arquivo disponível em: http://cvs.linuxcnc.org/cgi-bin/cvsweb.cgi/~checkout~/emc2/src/hal/drivers/hal_parport.c?rev=1.12.2.1.

4.2.5. Configuração do EMC2

Inicialmente o objetivo era configurar o EMC2 com a cinemática inversa do robô, o que tornaria possível controlá-lo diretamente, como mostra a figura 8. Porém, essa abordagem mostrou-se uma tarefa muito complexa e, portanto, foi adotada a alternativa descrita na figura 9, em que utilizou-se um programa auxiliar que efetua os cálculos da cinemática inversa e fornece as posições angulares ($\theta_1, \theta_2, \theta_3$) dos atuadores em função da posição do efetuador no espaço (X,Y,Z). Finalmente, entramos com esses ângulos no EMC2, que se encarregará de girar os motores com esses ângulos e, conseqüentemente, levará o efetuador até a posição desejada.

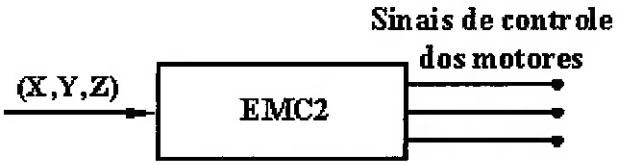


Figura 4.2: Implementação direta.

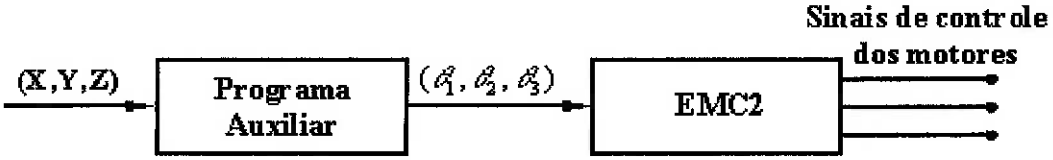


Figura 4.3: Implementação adotada.

Para esta implementação, deve haver uma correspondência direta entre os eixos comandados pelo EMC2 (A, B e C) e os ângulos dos motores (θ_1, θ_2 e θ_3) em graus, ou seja, digitando-se por exemplo a instrução G0 A90 na janela de comando do EMC2, o motor 1 deve girar 90° no sentido positivo. Para isso, deve ser feita a configuração de alguns parâmetros do arquivo .ini.

De acordo com o manual de usuário do EMC2, para a calibração dos eixos, deveriam ser editados os parâmetros UNIT e INPUT_SCALE, ambos da seções [AXIS_0], [AXIS_1] e [AXIS_2]. Conforme foi detalhado em anteriormente, UNIT refere-se à unidade de medida de deslocamento do eixo e INPUT_SCALE define o número de passos necessários para que haja um deslocamento de um UNIT. No caso do robô paralelo, temos as seguintes especificações: 3 eixos angulares movidos diretamente por motores com resolução de 1,8°/passo, o que resultaria em:

UNIT = 1.8

INPUT_SCALE = 1.0

Porém, essas modificações não funcionaram corretamente. Na realidade, alterando-se os valores dos dois parâmetros e medindo-se os deslocamentos dos eixos, foi verificado que apenas o valor de INPUT_SCALE está relacionado ao movimento do eixo, e que esta relação é inversamente proporcional. Assim, a calibração foi feita experimentalmente: para 10 voltas dos motores e INPUT_SCALE = 1.0, as coordenadas mostradas pelo EMC2 foram 2223,8889. Como o desejado era ajustar o INPUT_SCALE para se obter 3600 para as mesmas 10 voltas dos motores (ou seja, 360 por volta), bastou fazer uma regra de três simples:

Input scale	Data
1.8	2223.8889
y	3600.00

Portanto, os valores adotados para os parâmetros foram:

UNIT = 1.0

INPUT_SCALE = - 1.11194445

O valor de INPUT_SCALE é negativo porque o sentido adotado como positivo para a rotação dos motores (sentido horário) é inverso ao sentido default do EMC2.

Após essa alteração também foi necessário editar os parâmetros FERROR e MIN_FERROR, que estavam muito pequenos e provocavam erro.

FERROR = 200

MIN_FERROR = 100

4.2.6. Cálculo de Trajetória

O conjunto de ângulos dos motores (θ_1, θ_2 e θ_3) que compõem uma dada trajetória especificada pelo usuário são calculados por um programa em MATLAB denominado *Trajetoria3.m*, que só pode ser executado através do prompt do MATLAB. As entradas do programa são: ponto inicial (em centímetros), eixo do movimento (x, y ou z), distância a ser percorrida (em centímetros) e discretização da trajetória (número de pontos). O programa fornece como saída um arquivo de nome *Coordenadas.ngc*, que contém o conjunto de coordenadas θ_1, θ_2 e θ_3 (em graus) que posicionam o robô na trajetória desejada. No EMC2, a extensão *.ngc* é utilizada para arquivos que contém instruções em linguagem G. Portanto, as coordenadas do arquivo de saída já estarão no formato adeqüado àquela linguagem.

O programa *Trajetoria3* utiliza a função *CinInv3(x, y, z)*, que implementa as eqüações de cinemática inversa do manipulador, ou seja, dado um ponto no espaço (x,y,z), ele retorna as posições angulares dos atuadores.

Abaixo, segue um exemplo de como utilizar o programa para o cálculo de uma trajetória de distância 5,0 cm ao longo do eixo Y, com ponto inicial Xi = 10, Yi = 0, Zi = -20 e com discretização de 15 pontos.

```
>> trajetoria3(10,0,-20,'y',5.0,15)
```

O arquivo *Coordenadas.ngc* gerado tem o seguinte conteúdo:

```
X12.4 Y20.5 Z20.5
X12.4 Y18.8 Z22.2
X12.5 Y17.2 Z23.9
X12.6 Y15.5 Z25.7
X12.7 Y13.9 Z27.4
X12.8 Y12.3 Z29.2
X13.0 Y10.7 Z31.0
X13.2 Y 9.1 Z32.8
X13.4 Y 7.6 Z34.6
X13.7 Y 6.1 Z36.4
X14.0 Y 4.6 Z38.2
X14.3 Y 3.1 Z40.1
X14.7 Y 1.7 Z42.0
X15.1 Y 0.3 Z43.8
X15.5 Y-1.1 Z45.7
```

O leitor não deve se confundir com as coordenadas acima pois, apesar de estarem sendo chamadas de X, Y e Z, elas são os ângulos dos motores e, portanto, **não** têm correspondência direta com os eixos cartesianos do robô.

4.2.7. Interpolação de Pontos

De forma semelhante ao programa em MATLAB que calcula a trajetória, o programa *Interpola.m* faz a interpolação dos ângulos dos motores de passo de dois pontos no espaço de trabalho do robô. Fornecendo como entrada as coordenadas (xi,yi,zi) do ponto inicial, as coordenadas (xf, yf, zf) do ponto final e o número de pontos de interpolação (discretização), o programa calcula os ângulos dos motores (assim como o *Trajetoria3*) e os escreve no arquivo *Interpola*.ngc.

Como no programa que calcula a trajetória, esse programa de interpolação utiliza-se da função *CinInv3.m* para o cálculo dos ângulos iniciais e finais e por meio da função *linspace* do MATLAB, cria-se uma série de ângulos que interpolam os ângulos iniciais e finais com a quantidade de pontos definidos pelo usuário.

Fazendo um exemplo de interpolação entre os pontos Xi = 10, Yi = 0 e Zi = -17 até o ponto Xf = 4, Yf = 5, Zf = -23, com discretização de 20 pontos, temos o seguinte conteúdo do arquivo *Interpola.ngc*: , lembrando que os valores que são jogados no programa, estão em centímetros e **não** em milímetros como o padrão.

Note que na primeira linha já está incluso o código para avanço de operação G1 e avanço 2000 e no final do programa existe o código M2 que indica o fim do programa:

Para a entrada no prompt do MATLAB:

>>Interpola(10,0,-17,4,5,-23,20)

Temos o código G que foi gerado em *Interpola.ngc*:

```
G1 X-6.2 Y 3.8 Z 3.8 F2000
X-4.8 Y 6.2 Z 8.5
X-3.5 Y 8.5 Z13.2
X-2.1 Y10.9 Z17.9
X-0.8 Y13.3 Z22.6
X 0.6 Y15.6 Z27.3
X 1.9 Y18.0 Z32.0
X 3.3 Y20.4 Z36.7
X 4.6 Y22.7 Z41.4
X 6.0 Y25.1 Z46.1
X 7.3 Y27.5 Z50.8
X 8.7 Y29.8 Z55.5
X10.0 Y32.2 Z60.2
X11.4 Y34.6 Z64.9
X12.7 Y36.9 Z69.6
X14.1 Y39.3 Z74.3
X15.4 Y41.7 Z79.0
X16.8 Y44.1 Z83.7
X18.1 Y46.4 Z88.4
X19.5 Y48.8 Z93.1
M2
```

Vale lembrar novamente que não se deve confundir as coordenadas acima escritas: X, Y e Z como as posições, elas são os ângulos dos motores e, portanto, **não** têm correspondência direta com os eixos cartesianos do robô.

4.3. Subsistema Eletrônico

4.3.1. Seleção dos motores

Existem três tipos de motores que podem ser utilizados para efetuar o "tracking": Os motores comuns de corrente contínua (DC), os de passo e os servo-motores. Motores DC são aqueles normalmente utilizados em eletro-eletrônicos, e a sua principal vantagem é o baixo custo. Um problema é que qualquer variação na corrente, no ciclo ou na carga no eixo do motor pode resultar em variação na sua rotação e, conseqüentemente, na velocidade do manipulador. Portanto, o controle de posição e velocidade deve ser feito em malha fechada. No caso do projeto, necessitamos de três atuadores, e o peso a ser movimentado por cada um deles varia, devido aos ângulos diferentes em que as cadeias cinemáticas do robô irão se posicionar durante a sua movimentação, como ilustrado abaixo. Assim sendo, o uso deste tipo de motor não é aconselhado.

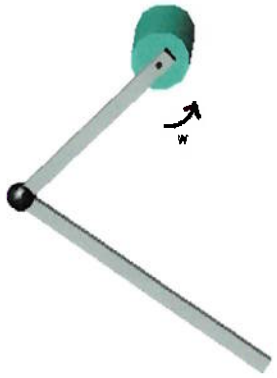


Figura 4.4: Peso da cadeia cinemática a favor do movimento.

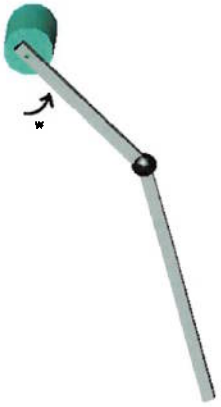


Figura 4.5: Peso da cadeia cinemática contra o movimento.

O servo-motor já vem com sensor e compensador incorporados, facilitando a implementação em malha fechada, mas apresenta o maior custo entre as três opções.

O motor de passo, por sua vez, apresenta uma acurácia muito grande de velocidade e posicionamento, já que o seu acionamento é feito por bobinas independentes que podem ser controladas por um circuito de acionamento ou por um computador, possibilitando que ele seja utilizado em malha aberta. Motores de

passo são encontrados em aparelhos onde a precisão é um fator muito importante. São usados em larga escala em impressoras, plotters, scanners, drivers de disquetes, discos rígidos e muitos outros aparelhos. Além disso, seu custo é intermediário entre o do motor DC e o do servo. A desvantagem do motor de passo é que ele necessita de um circuito de acionamento.

4.3.2. Placa acionadora

A placa acionadora utilizada tem as seguintes características:

- Capacidade de acionar dois motores de passo
- Operação no modo Full ou Half Step
- Controle de Direção CW/CCW
- Aciona apenas motores de 12V
- Corrente máxima de 3A por fase
- Interface com nível lógico TTL
- Alimentação de 12V, sendo +5V gerado internamente
- Proteção de sobrecorrente

Para a aquisição dos componenentes eletrônicos da placa, foi feita uma breve pesquisa de preços através da internet (Anexo B).

A placa acionadora dos motores de passo foi montada e alguns testes realizados. Após a sua montagem foi realizado o primeiro teste juntamente com o programa EMC. Nessa ocasião, a placa pareceu não funcionar corretamente, pois o motor girava apenas em uma direção. Para a completa constatação do funcionamento da placa acionadora, foi realizado um segundo teste em que os pulsos de clock foram aplicados com um gerador de funções, e o sinal de sentido de rotação foi gerado "manualmente", aterrando-se ou ligando-se em +Vcc o ino correspondente. Com este procedimento, foi possível fazer o eixo do motor girar nos dois sentidos.

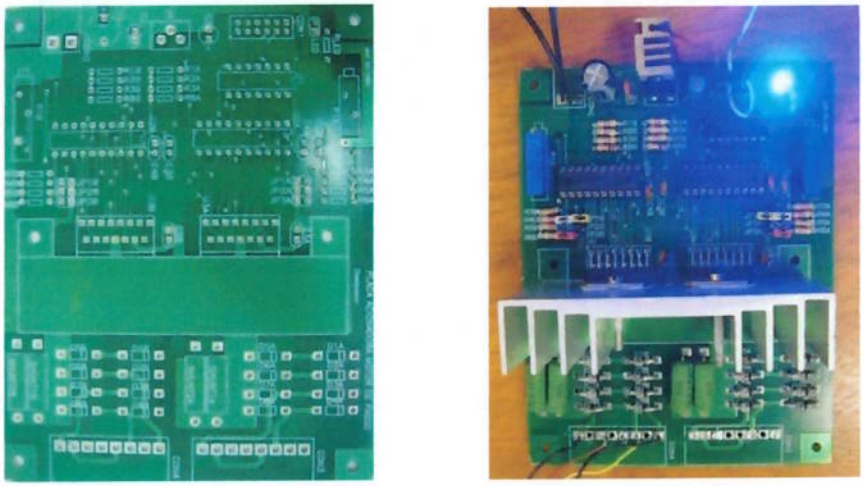


Figura 4.6: Placa antes e após a montagem.

4.3.3. Acionamento do Efetuador

O efetuador utilizado é simplesmente um eletroímã (solenóide) que, quando ativado, é capaz de pegar objetos metálicos. Ele é controlado pelos comandos *M3* e *M5* que, em código G, significam respectivamente ligar e desligar ferramenta.

Devido ao solenóide ter uma baixa resistência elétrica ($14\ \Omega$), ele requisita uma corrente da ordem de centenas de mA e, portanto, não poderia ser ligado diretamente à saída da porta paralela, cuja corrente está limitada a aproximadamente 10 mA. Por essa razão, foi desenvolvido um pequeno circuito elétrico para acionar o efetuador, conforme a figura 4.7.

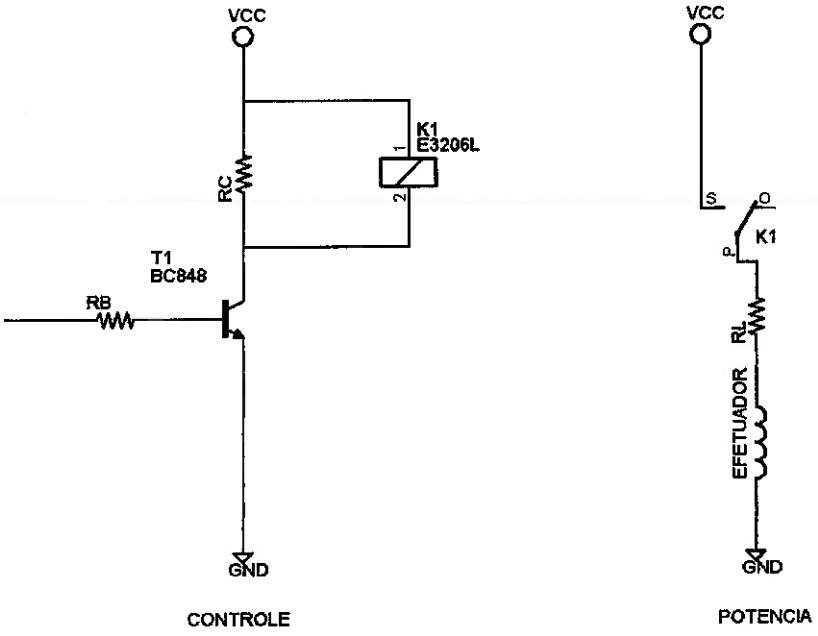


Figura 4.7: Circuito para acionamento do efetuador.

Para isolar a parte de controle da parte de potência, a alimentação do solenóide está ligada ao contato normalmente aberto de um relé. Portanto, o solenóide estará desligado quando a bobina do relé estiver desenergizada. O acionamento do relé, por sua vez, é feito por um circuito que utiliza um transistor operando como chave liga-desliga. Quando a saída da porta paralela está em nível 0, o transistor é cortado e, conseqüentemente, o efetuador é desligado, pois $I_B = 0 \Rightarrow I_C = \beta \cdot I_B = 0 \Rightarrow V_{RELE} = 0$. Por outro lado, se a saída da porta paralela estiver em nível 1, o transistor estará saturado e teremos $I_B \neq 0 \Rightarrow I_C = \beta \cdot I_B \neq 0 \Rightarrow V_{RELE} \cong 12V$. O resistor R_L serve apenas para limitar a corrente no efetuador.

4.3.4. Projeto do circuito

- Especificações do relé.

  V_{RELE} = 12 V, I_{RELE} = 30 mA.

- Especificações do transistor.

  $I_{CMIN} = I_{RELE} = 30mA$, I_{BSAT} = 1 mA. $\Rightarrow$ Transistor adotado: BC548

- Polarização do transistor.

$$R_B = \frac{5 - V_{BE}}{I_{BSAT}} = \frac{5 - 0,8}{10^{-3}} \Rightarrow R_E = 4,2K\Omega$$

$$P_E = (5 - V_{BE}) \cdot I_{BSAT} = (5 - 0,8) \cdot 10^{-3} = 4,2mW$$

$$I_C = \beta \cdot I_B = 140 \cdot 1mA = 140\,mA > I_{CMIN}$$

$$R_C = \frac{12 - V_{CE}}{I_C} = \frac{12 - 0,6}{140 \cdot 10^{-3}} = 81,4\Omega$$

$$P_C = (V_{CC} - V_{CE}) \cdot I_C = (12 - 0,6) \cdot 140 \cdot 10^{-3} = 1,6W$$

5. Ensaios realizados

Para verificar o funcionamento do controle de posição, foram realizados 4 ensaios de movimentação do robô, de acordo com a seguinte metodologia: para cada ensaio, o robô deveria fazer uma trajetória com pontos inicial e final pré – definidos, chamados de pontos teóricos. Então, o robô foi programado com esta trajetória e as posições atingidas por ele foram medidas e comparadas com os valores teóricos. Nos 3 primeiros ensaios, foram considerados movimentos ao longo dos eixos X, Y e Z separadamente, partindo-se do ponto inicial (X_0, Y_0, Z_0) = (10, 0, -20). Nos dois primeiros testes (trajetórias sobre os eixos X e Y), o robô movimentou-se 5 cm no sentido positivo dos eixos, tendo como pontos finais (X_{F1}, Y_{F1}, Z_{F1})= (15, 0, -20) e (X_{F2}, Y_{F2}, Z_{F2})= (10, 5, -20). No eixo Z, ensaiou-se um movimento de 5 cm no sentido negativo, ou seja, (X_{F3}, Y_{F3}, Z_{F3}) = (10, 0, -25). O quarto teste foi realizado com uma trajetória com interpolação, partindo-se novamente do ponto (X_0, Y_0, Z_0) = (10, 0, -25) e indo até (15, 5, -25). Neste procedimento, verificou-se que houve um desvio máximo de 6,67% entre as posições teóricas e medidas. O maior erro ocorreu na trajetória ao longo do eixo X, em que foi medida uma posição final igual a (14, 0, -20) e a posição esperada era (15, 0, -20).

Portanto, foi possível verificar que o controle de posição foi implementado de forma satisfatória.

6. Conclusões

O EMC2 é uma ferramenta muito poderosa para controle de máquinas com cinemática inversa trivial. Porém, para mecanismos complexos como o robô paralelo do projeto, seria necessário um tempo adicional de cerca de dois meses para que a cinemática inversa pudesse ser configurada corretamente. Existem grupos de discussão dos desenvolvedores e usuários do EMC em que são discutidas implementação de cinemática inversa, configurações de parâmetros e outras questões. Embora esses grupos não tenham conseguido responder satisfatoriamente às duvidas em relação à configuração da cinemática inversa, eles foram extremamente eficientes no esclarecimento de dúvidas da parte de instalação do sistema operacional e do EMC2.

Com o programa de interpolação e de cálculo de trajetória, baseados em um arquivo que calcula o ângulo de cada motor de passo, foi possível a implementação satisfatória de movimentos complexos e combinados a partir da utilização do código G.

7. Referências Bibliográficas

EMC HANDBOOK. Tutorial sobre instalação, configuração e uso do EMC. Disponível em <http://www.linuxcnc.org/handbook>. Acesso em: nov. 2006.

PARALLEMIC – THE PARALLEL MECHANISMS INFORMATION CENTER. Informações sobre mecanismos paralelos. Disponível em <http://www.parallemic.org>. Acesso em: nov. 2006.

Branchini, D. M. **Desenvolvimento de um manipulador robótico baseado em um mecanismo de cinemática paralela.** Trabalho de Formatura, Escola Politécnica da USP, 2004.

Dixon, W. E.; Dawson, D. M.; Costic, B. T.; Queiroz, M. S. Q. **A MATLAB-Based Control Systems Laboratory Experience for Undergraduate Students: Toward Standardization and Shared Resources.** IEEE Transactions on education, v. 45, n. 3, ago. 2002.

18º Congresso Internacional de Engenharia Mecânica, 2005, Ouro Preto. **A new family of 3-DOF parallel robot manipulators for pick-and-place operations.**

Apêndice – Equacionamento da Cinemática Inversa

A1. Cinemática Inversa

A partir da cinemática inversa será possível o correto posicionamento dos braços do robô. Recuperando o equacionamento do trabalho anterior, são apresentados a seguir os cálculos da cinemática inversa do modelo tridimensional (3RSS + CP) usando o método algébrico.

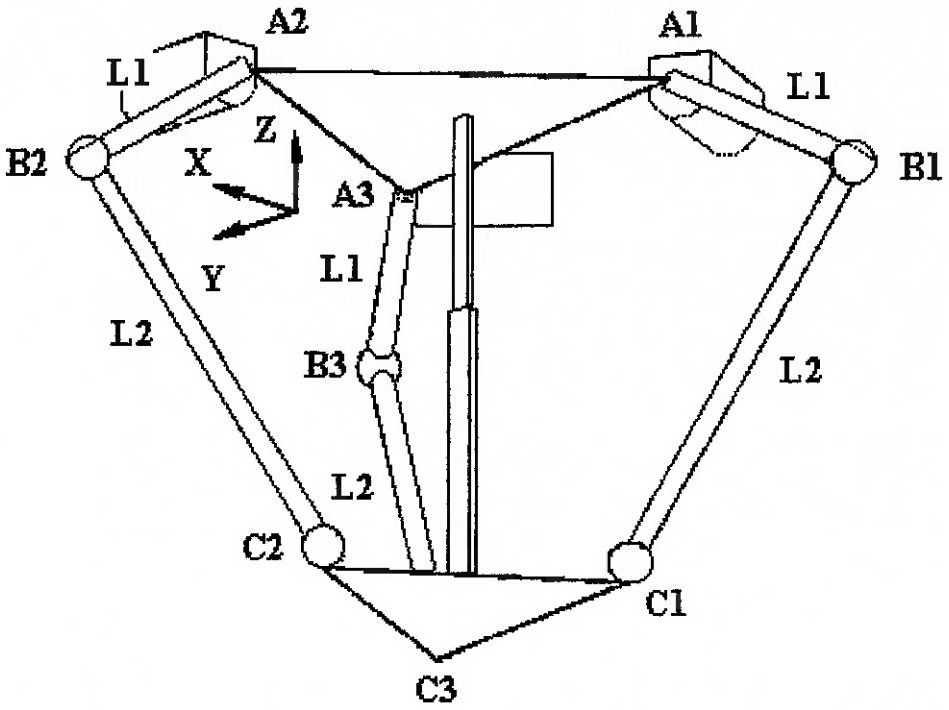
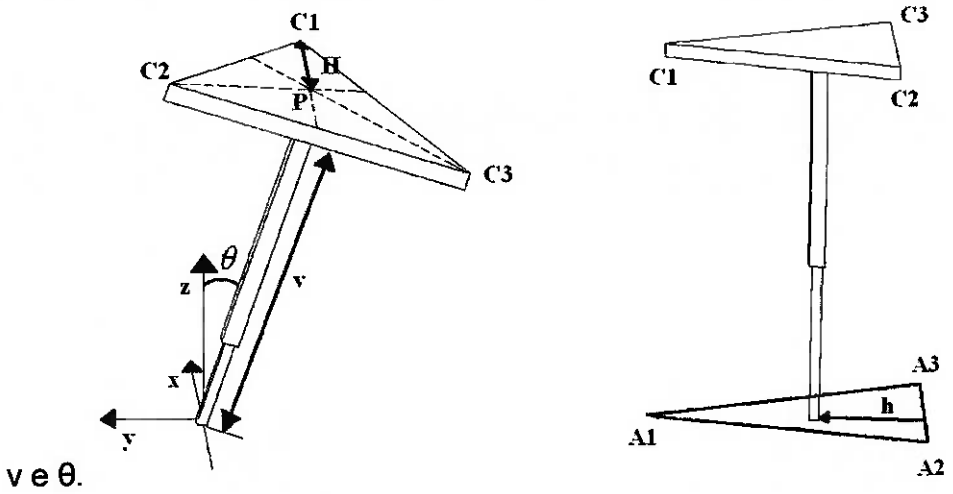


Figura A1.1: Esquema do (3RSS + CP) usado para o calculo da cinemática inversa.

Fato importante que a figura anterior mostra as direções e sentidos dos eixos de movimentação do robô no espaço. As figuras seguintes têm por objetivo mostrar as posições adotadas para: P, h, H, L,



v e θ.

(A) (B)

Figura A1.2: (A) Mostra a posição de P, H, v e θ. (B) Posição de h.

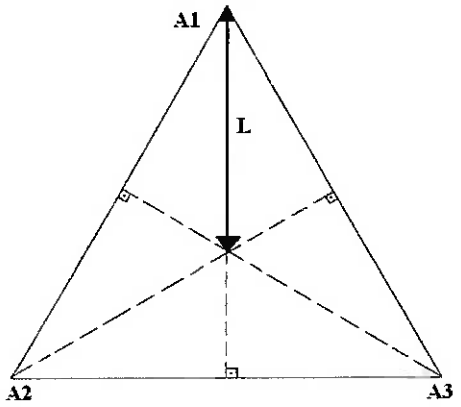


Figura A1.3: Posição de L no triângulo formado por A1, A2, e A3.

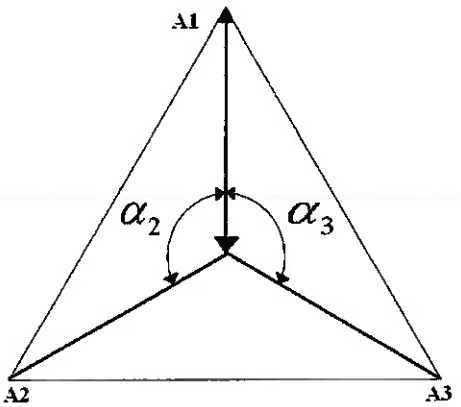


Figura A1.4: Posição dos ângulos α_1, α_2 e α_3.

Os valores para α_1, α_2 e α_3 são:

$\alpha_1 = 0;$

$\alpha_2 = \frac{2\pi}{3};$

$\alpha_3 = \frac{4\pi}{3} ou;$

$\alpha_3 = -\frac{2\pi}{3}.$

A matriz abaixo mostra a posição de P no espaço x, y e z. Sendo que a P está representado na Figura 13 – (A) Mostra a posição de P, H, v e θ. (B) Posição de h.

$$^{b}P = \begin{bmatrix} h \\ -v\sin\theta \\ v\cos\theta \end{bmatrix}$$

Já as duas matrizes seguintes mostram respectivamente a posição de $^{b}C_j$ e $^{b}B_j$ no espaço 3D. Sendo j = 1, 2 e 3.

$$^{b}C_j = \begin{bmatrix} H\cos\alpha_j + h \\ (H\sin\alpha_j)\cos\theta - v\sin\theta \\ (H\sin\alpha_j)\sin\theta + v\cos\theta \\ 1 \end{bmatrix}$$

$$^{b}B_j = \begin{bmatrix} (L1\cos\alpha_j)\cos\theta_j + L\cos\alpha_j \\ (L1\sin\alpha_j)\cos\theta_j + L\sin\alpha_j \\ (L1)\sin\theta_j \\ 1 \end{bmatrix}$$

Como o modulo da distância entre $^{b}B_j - {}^{b}C_j$ resulta no comprimento do braço L2, temos:

$$\left\| {}^{b}B_j - {}^{b}C_j \right\| = L2$$

$$\left({}^{b}B_j - {}^{b}C_j\right)^T \left({}^{b}B_j - {}^{b}C_j\right) = L2^2$$

$$j = 1,2,3.$$

Para calcular a cinemática inversa, precisamos obter os valores dos ângulos θ_j. Para tanto, basta modificar a equação anterior para que ela assuma a seguinte forma.

$$(E_j)\cos(\theta_j) + (F_j)\sin(\theta_j) + (G_j) = 0$$

Como os valores de $\cos(\theta_j)$ e $\sin(\theta_j)$ são:

$$\cos(\theta_j) = \frac{1-u^2}{1+u^2};$$

$$\sin(\theta_j) = \frac{2u}{1+u^2}.$$

Temos:

$$(E_j)\left(\frac{1-u^2}{1+u^2}\right) + (F_j)\left(\frac{2u}{1+u^2}\right) + (G_j) = 0$$

$$(E_j)(1-u^2) + (F_j)(2u) + (G_j)(1+u^2) = 0$$

$$(G_j - E_j)(u^2) + (F_j)(2u) + (G_j + E_j) = 0$$

Então:

$$u = \frac{-F_j \pm \sqrt{F^2{}_j - G_j^{\,2} + E_j^{\,2}}}{(G_j - E_j)}$$

Sendo $u = \tan\left(\frac{\theta_j}{2}\right)$, portanto temos que:

$$\theta_j = 2\arctan(u)$$

Achando θ_j para j = 1, 2 e 3 têm-se os ângulos dos motores em A1, A2 e A3.

Anexos

A. Programas Utilizados

1. Programa Trajetoria3

```
% ------------------------------------------------------------------------
%|  Determina os angulos dos motores de passo para uma trajetoria
%|  Entrada: xo, yo, zo (em cm)
%|           eixo     (x, y ou z)
%|           dist     (distancia a ser percorrida em cm)
%|           n        (numero de pontos)
%|  Saida  : Angulos dos motores (em graus), armazenados em um
%|           arquivo de nome Coordenadas.doc
% ------------------------------------------------------------------------

function Trajetoria3(xo,yo,zo,eixo,dist,n)

%Cria o arquivo Coordenadas.ngc.
fid = fopen('Coordenadas.ngc','w');

%Verifica qual o eixo escolhido.
switch lower(eixo)

 case 'x'
  %Inicia x.
  x = xo;
  for i = 1:n
   %Calcula coordenadas do ponto.
   Angulos = CinInv3(x,yo,zo);
   %Escreve coordenadas no arquivo Coordenadas.ngc.
   fprintf(fid,'X%4.1f Y%4.1f Z%4.1f\n', Angulos);
   %Atualiza x e i para a proxima iteraçao.
   i = i + 1;
   x = x + (dist/n);
  end

 case 'y'
  %Inicia y.
  y = yo;
  for i = 1:n
   %Calcula coordenadas do ponto.
   Angulos = CinInv3(xo,y,zo);
   %Escreve coordenadas no arquivo Coordenadas.ngc.
   fprintf(fid,'X%4.1f Y%4.1f Z%4.1f\n', Angulos);
   %Atualiza y e i para a proxima iteraçao.
   i = i + 1;
   y = y + (dist/n);
  end

 case 'z'
  %Inicia z.
  z = zo;
  for i = 1:n
   %Calcula coordenadas do ponto.
   Angulos = CinInv3(xo,yo,z);
   %Escreve coordenadas no arquivo Coordenadas.ngc.
   fprintf(fid,'X%4.1f Y%4.1f Z%4.1f\n', Angulos);
   %Atualiza z e i para a proxima iteraçao.
```

```
    i = i + 1;
    z = z + (dist/n);
  end

 end

%Fecha o arquivo Coordenadas.doc.
fclose(fid);
```

2. Função CinInv3

```
% ----------------------------------------------
%|  Implementa a cinematica inversa
%|  Entrada: x, y, z (em cm)
%|  Saida  : Angulos dos motores  (em graus)
% ----------------------------------------------

function Angulos = CinInv3(x, y, z)

alfa1 = 0;
alfa2 = 2*pi/3;
alfa3 = 4*pi/3;
L1 = 9;
L2 = 21;
H = 14*sin(pi/3)*2/3;
L = 20*sin(pi/3)*2/3;
v = (y^2 + z^2)^0.5;
teta = atan(y/z);
h = 20*sin(pi/3)-x ;
x11 = 20*sin(pi/3)-h-H;

if x11 >= 0
   gama = acos(-(21^2-81-(x11^2+v^2))/(2*9*(x11^2+v^2)^0.5))*180/pi;
   beta = atan(v/x11)*180/pi;
   m11 = 180-(beta+gama);
else
   gama = acos(-(21^2-81-(x11^2+v^2))/(2*9*(x11^2+v^2)^0.5))*180/pi;
   beta = atan((x11^2)^.5/v)*180/pi;
   m11 = 180-(90+beta+gama);
end

E2  = [(-2*cos(alfa2)*(H-L)*L1)-(2*h*L1*cos(alfa2))-
(2*(H*cos(teta)L)*L1*(sin(alfa2))^2)+(2*L1*sin(alfa2)*v*sin(teta))];
F2  = [(-2*L1*H*sin(alfa2)*sin(teta))-(2*v*L1*cos(teta))];
G2  = (v^2+h^2+((cos(alfa2))^2*(H-L)^2)+2*h*(cos(alfa2))*(H-
L)+(2*H*v*sin(alfa2)*sin(teta)*cos(teta))+((H^2)*((sin(alfa2))^2)*((sin(teta))^2))+((sin(alfa2))^2)*(((H*co
s(teta)-L))^2)-(2*v*sin(teta)*sin(alfa2)*(H*cos(teta)-L))-L2^2);
u22 = (-F2-(F2^2-G2^2+E2^2)^0.5)/(G2-E2);
tetam22 = 2*atan(u22);
m22 = tetam22*180/pi;
E3  = [(-2*cos(alfa3)*(H-L)*L1)-(2*h*L1*cos(alfa3))-(2*(H*cos(teta)-
L)*L1*(sin(alfa3))^2)+(2*L1*sin(alfa3)*v*sin(teta))];
F3  = [(-2*L1*H*sin(alfa3)*sin(teta))-(2*v*L1*cos(teta))];
G3  = (v^2+h^2+((cos(alfa3))^2*(H-L)^2)+2*h*(cos(alfa3))*(H-
L)+(2*H*v*sin(alfa3)*sin(teta)*cos(teta))+((H^2)*((sin(alfa3))^2)*((sin(teta))^2))+((sin(alfa3))^2)*(((H*co
s(teta)-L))^2)-(2*v*sin(teta)*sin(alfa3)*(H*cos(teta)-L))-L2^2);
u32 = (-F3-(F3^2-G3^2+E3^2)^0.5)/(G3-E3);
tetam32 = 2*atan(u32);
m32 = tetam32*180/pi;
% Resultados
Angulos = [m11, m22, m32];
```

3. Programa Interpola

```
% ------------------------------------------------------------
%|  Determina os angulos dos motores de passo para uma trajetoria  |
%|  Entrada: xi, yi, zi, xf, yf, zf (em centimetros)              |
%|          n                (numero de pontos)                   |
%|  Saida  : Angulos dos motores (em graus), armazenados em um     |
%|          arquivo de nome Interpola.ngc                         |
% ------------------------------------------------------------


function Interpola(xi, yi, zi, xf, yf, zf, n)

%Cria o arquivo Interpola.ngc
fid = fopen('Interpola.ngc','w');

%Calcula angulos iniciais
Angulos_i=CinInv3(xi, yi, zi);

%Calcula angulos finais
Angulos_f=CinInv3(xf, yf, zf);

%Gera n angulos entre angulos iniciais e angulos finais
X=linspace(Angulos_i(1),Angulos_f(1),n);
Y=linspace(Angulos_i(2),Angulos_f(2),n);
Z=linspace(Angulos_i(3),Angulos_f(3),n);

fprintf(fid,'G1 X%4.1f Y%4.1f Z%4.1f F2000\n',X(1), Y(1), Z(1));

for i = 2:n
   %Escreve os angulos no arquivo Interpola.ngc
   fprintf(fid,'X%4.1f Y%4.1f Z%4.1f\n', X(i), Y(i), Z(i));
   %Atualiza z e i para a proxima iteraçao.
   i = i + 1;
end

fprintf(fid,'M2\n');

%Fecha o arquivo Interpola.gnc.
fclose(fid);
```

B. Tabela de preços

	Qtdade	Preço unitário	Custo Parcial	Soma Parcial
Motor de passo	3	R$ 250,00	R$ 750,00	R$ 750,00
Placa Motor Passo	2	R$ 71,72	R$ 143,44	R$ 893,44
Fonte 12V estab.	1	R$ 200,00	R$ 200,00	R$ 1.093,44
CI L297	4	R$ 20,17	R$ 80,68	R$ 1.174,12
CI L298	4	R$ 15,69	R$ 62,76	R$ 1.236,88
CI 7807	2	R$ 7,00	R$ 14,00	R$ 1.250,88
CI 7805	2	R$ 1,00	R$ 2,00	R$ 1.252,88
Soquete	10	R$ 1,50	R$ 15,00	R$ 1.267,88
Resistor	34	R$ 0,10	R$ 3,40	R$ 1.271,28
Res. Pot.	8	R$ 1,00	R$ 8,00	R$ 1.279,28
Capacitor	18	R$ 1,00	R$ 18,00	R$ 1.297,28
Conector	8	R$ 3,00	R$ 24,00	R$ 1.321,28
Jumper	12	R$ 0,10	R$ 1,20	R$ 1.322,48
Res. Var.	4	R$ 1,50	R$ 6,00	R$ 1.328,48
Led	2	R$ 0,50	R$ 1,00	R$ 1.329,48
Diodo	32	R$ 1,00	R$ 32,00	R$ 1.361,48
			TOTAL	R$ 1.361,48

C. Fotos da fabricação do protótipo

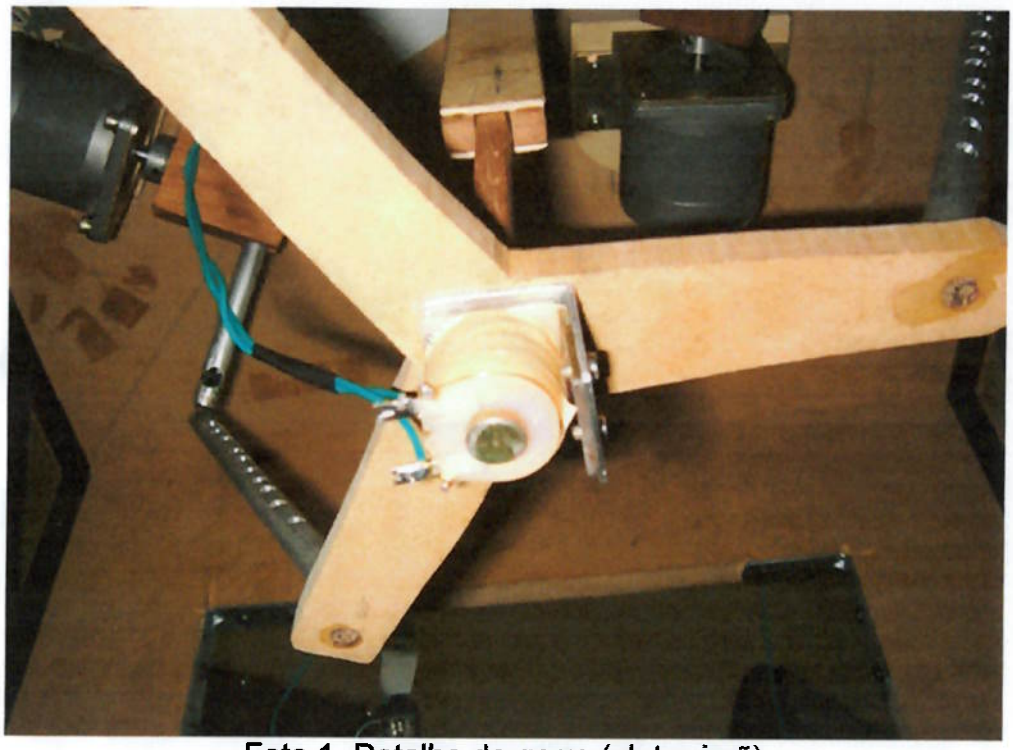


Foto 1. Detalhe da garra (eletro-imã)

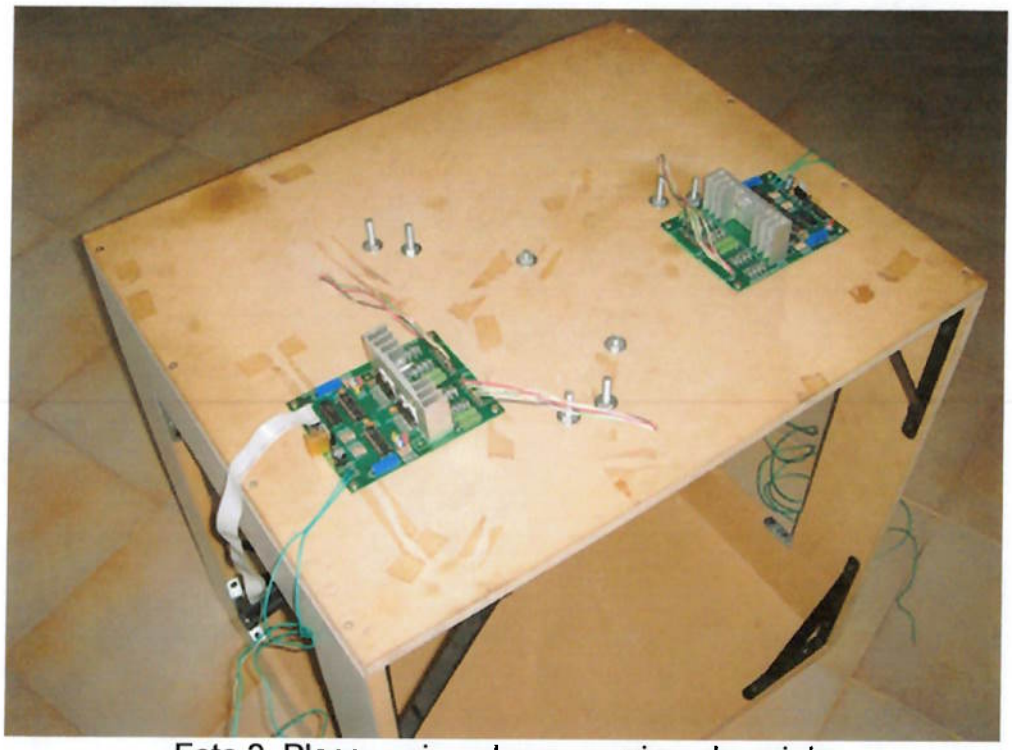


Foto 2. Placas acionadoras em cima da gaiola

Foto 3. Ajustando o ângulo dos motores

Foto 4. Confeccionando o suporte do motor

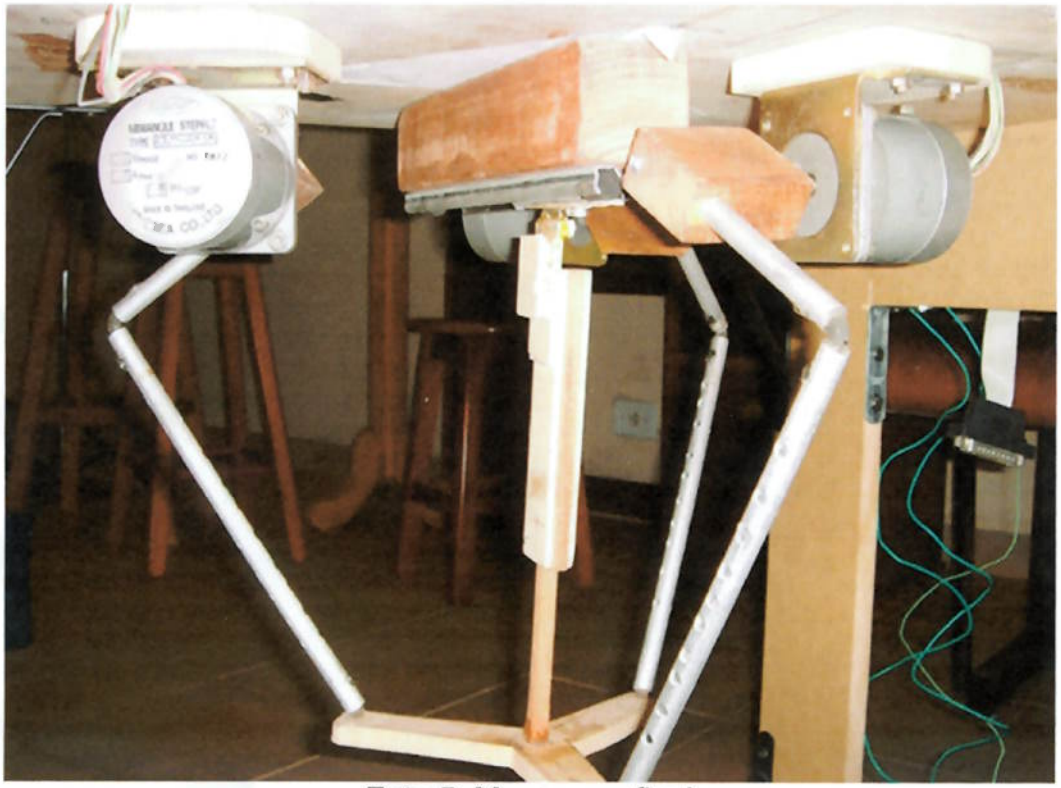


Foto 5. Montagem final