Universidade do Vale do Paraíba
Faculdade de Ciência da Computação
Curso de Engenharia de Computação

Implementação do Sistema de Controle de um Manipulador Revoluto

Fernanda Mara Brito
Monica Fiumana Martin Falcon

Relatório do Trabalho de Conclusão de Curso apresentado à Banca Avaliadora da Faculdade de Ciência da Computação da Universidade do Vale do Paraíba, como parte dos requisitos para obtenção do Título de Bacharel em Engenharia de Computação.

São José dos Campos – SP
12/2005

Implementação do Sistema de Controle de um Manipulador Revoluto

Fernanda Mara Brito
Monica Fiumana Martin Falcon

Banca Avaliadora

Presidente Alderico de Paula

Orientador Emerson de Góes

Gabriel Hickel

_____

Emerson de Góes
Orientador Acadêmico

_____

Alderico de Paula
Coordenador da Disciplina de TCC

Data:

Agradecimentos

Agradecemos primeiramente aos nossos Pais, que nos proporcionaram a educação, nos ensinando a importância do conhecimento, que nos deram apoio, acreditando e confiando em nossa capacidade. Agradecemos também aos professores que nos ensinaram e nos incentivaram a aprendermos sempre mais.

Dedicatória

Dedicamos este trabalho primeiramente aos nossos Pais, que sempre nos deram apoio em tudo, nos orientando e nos ajudando. Aos professores em geral, e principalmente ao nosso orientador, Emerson de Góes.

Resumo

Este trabalho descreve a implementação de uma interface gráfica para gerar trajetórias e controle para um robô de três graus de liberdade. A linguagem de programação utilizada é o Visual Basic. A comunicação com o robô é feita através da porta paralela.

O robô utilizado foi construído com componentes do kit Robix, e é composto por três servomotores FUTABA (deslocamento angular de 0^o a 180^o), elos de alumínio de tamanhos variados e conexões de juntas. A construção mecânica do robô está descrita ao longo do trabalho. Os atuadores do robô estão ligados diretamente na porta paralela do PC, sem a necessidade de qualquer outro tipo de interface eletrônica.

Para definir qual trajetória o robô deve seguir, o usuário utiliza o *software* desenvolvido em Visual Basic. O *software* é usado para a entrada de dados (posição e orientação do órgão terminal do robô), cálculo das cinemáticas direta e inversa, visualização do deslocamento do elos através de gráficos, armazenamento de dados de trajetórias no *Access*, comunicação e envio de dados ao robô via porta paralela.

Definida a trajetória, o *software* converte as coordenadas do espaço cartesiano, informadas pelo usuário, em coordenadas de junta (ângulos θ_1, θ_2 e θ_3) obtidas através da cinemática inversa. Esses ângulos são enviados aos atuadores do robô via porta paralela, usando modulação por largura de pulso (*PWM-pulse width modulation*). A base de tempo para o sistema é definida pelo temporizador do PC.

Enquanto os pulsos são gerados para os atuadores do robô, o *software* apresenta a trajetória percorrida por cada elo, e ao mesmo tempo o robô se movimenta, percorrendo a trajetória estipulada pelo usuário. Os pontos da trajetória plotada são calculados através da cinemática direta.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Símbolos, Acrogramas e Abreviações

PWM	*Pulse-Width Modulation*
IHM	Interface Homem Máquina
DBA	*Data Base Administrator*
SGBDs	Sistemas de Gerenciamento e Bancos de Dados
RUR	*Rossum´s Universal Robots*
AMF	*American Machine & Foundry Company*
PBE	*Programming By Example*
E-R	Entidade Relacionamento
SPP	*Standard Parallel Port*
EPP	*Enhanced Parallel Port*
CC	Corrente Contínua
FIFO	*First In First Out*
MLP	Modulação por Largura de Pulso
BCD	*Binary Coded Decimal*
IBM	*Intl Business Mach*
DMA	*Directy Memory Access*
SQL	*Structured Query Language*
DLL	*Dynamic Link Library*
PC	*Personal Computer*
CPU	*Central Processing Unit*
3D	Tridimensional
D-H	Denavit-Hartenberg
RAM	*Random Acess Memory*
DRAM	*Dynamic Random Acess Memory*
CD	*Compact Disc*
ECP	*Enhanced Capabilities Port*
DC	*Direct Current*

Capítulo 1
Introdução

1.1 Antecedentes

Antigamente, os trabalhadores eram obrigados a executar atividades manuais, onde estas eram perigosas e repetitivas, colocando assim a vida do homem em risco, e proporcionando altos índices de acidente de trabalho e falhas no produto final.

No século XVIII, houve a revolução industrial, tendo a burguesia industrial ávida por maiores lucros, menores custos e produção acelerada, e assim buscou alternativas para melhorar a produção de mercadorias utilizando a mecanização dos sistemas de produção.

No início dos anos 60, os primeiros robôs começaram a ser usados com o objetivo de substituir o homem em tarefas que ele não podia realizar por envolverem condições desagradáveis, tipicamente contendo altos níveis de calor, ruído, gases tóxicos, esforço físico extremo e trabalhos monótonos e repetitivos.

Cada vez mais as linhas de produção colocam robôs ao invés de homens, devido ao fato de o ser humano apresentar limites físicos (como fadiga, cansaço etc) e mentais (inteligência emocional, estresse etc), diferente das máquinas, que podem trabalhar por horas seguidas.

Com o aumento dos robôs, perde-se a necessidade de contratar pessoas para executar tais tarefas, tendo os postos de trabalho migrado para manipular os robôs, fazer manutenção, alterar a programação da máquina e assim por diante. Praticamente os trabalhadores viraram gerenciadores de máquinas, e assim deixaram de ser produtores.

1.2 Descrição do Problema

O trabalho a ser desenvolvido irá implementar uma interface gráfica usada para definir uma trajetória a ser executada pelo manipulador. O aplicativo mostra o percurso realizado pelo robô e gera os sinais de controle para os atuadores, de modo que o órgão terminal do manipulador execute a trajetória definida pelo usuário do programa. As

juntas do robô são acionadas através de pulsos gerados pelo computador e enviados através da porta paralela.

Para a definição da trajetória, o usuário deverá fornecer a posição e orientação do órgão terminal no espaço cartesiano. O *software* será desenvolvido em linguagem Visual. Para a execução da estrutura mecânica do robô serão utilizados componentes do kit Robix (servomotores, elos de alumínio, conexões de juntas etc). O sistema deverá funcionar, independentemente da plataforma de *software* e *hardware* do computador.

1.3 Objetivo

Este trabalho tem como objetivo a implementação de uma interface gráfica para gerar trajetórias e controle para um robô de três graus de liberdade. A linguagem de programação a ser utilizada é o Visual Basic. A comunicação com o robô será feita através da porta paralela. Os conceitos de robótica, desenvolvimento de *software* e manipulação de *hardware* deverão estar integrados.

1.4 Restrições

O *software* somente poderá ser instalado em computadores com sistema operacional Windows. Para o funcionamento prático do projeto é necessário o kit Robix da FCC-UNIVAP, montado igual ao projetado.

1.5 Organização do Trabalho

O trabalho é dividido em cinco capítulos. O capítulo um, é a introdução que versa sobre a motivação e os principais objetivos do trabalho. O capítulo dois apresenta um breve histórico da robótica, o estado da arte e os conceitos básicos para a compreensão deste trabalho. No terceiro capítulo, encontram-se as especificações do sistema, o projeto preliminar, o projeto detalhado e os testes realizados. O capítulo quatro informa os resultados obtidos com o desenvolvimento deste trabalho, apresentando telas do *software*, fotos, código e validações do sistema. No capítulo cinco encontra-se a conclusão final do projeto.

Capítulo 2

Embasamento Teórico

2.1 Histórico

Com o avanço das pesquisas sobre Inteligência Artificial, em breve já será possível ter um robô (“robotnik” palavra de origem tcheca que significa servo), que pense e aja como o seu dono [1].

A seguir, há uma descrição dos fatos históricos que contribuíram para o desenvolvimento da Robótica [1], [2]:

IV a.C. Grécia	Aristóteles relata os primeiros princípios da robótica, referentes à utilização de instrumentos dedicados a trabalhos determinados, sem o auxílio das mãos humanas, o que reduziria os esforços do homem, com ênfase no conceito de mestre e escravo.
Séc. XIII	Inicia-se a Revolução Industrial, com a evolução de novas fontes de energia, novos mecanismos e instrumentos. Os novos conceitos industriais tornam possível a evolução da máquinaria capaz de controlar uma serie de ações seqüenciadas.
Séc. XIX	No final do século XIX inicia-se o desenvolvimento da máquina. Exposições de máquinas, para promover os mais recentes eventos tecnológicos, são realizadas com freqüência. O motor elétrico é introduzido na indústria. A máquina substitui o homem.
1914-1918	A Primeira Guerra Mundial acarreta várias mudanças. O poder da máquina mostra sua forma negativa e destrutiva.
1921	O dramaturgo Karel Capek usa pela primeira vez a palavra “robot” na peça teatral Rossum´s Universal Robots (R.U.R.), que retrata a criação de robôs para substituir o homem nos trabalhos pesados. O robô começa a ser visto como uma máquina “humana” com inteligência e personalidade.
1928	Um robô mecânico abre uma exposição de modelos técnicos, em Londres.
1940	O grande escritor americano de ficção científica Isaac Asimov

estabelece quatro leis muito simples para a robótica:

1ª lei: "Um robô não pode ferir um ser humano ou, permanecendo passivo, deixar um ser humano exposto ao perigo".

2ª lei: "O robô deve obedecer às ordens dadas pelos seres humanos, exceto se tais ordens estiverem em contradição com a primeira lei".

3ª lei: "Um robô deve proteger sua existência na medida em que essa proteção não estiver em contradição com a primeira e a segunda lei".

4ª lei: “Um robô não pode causar mal à humanidade nem permitir que ela própria o faça" (lei escrita por Asimov em 1984).

Entretanto, cabe observar que os robôs têm braços e articulações capazes de trabalhos repetitivos e autônomos, mas não sensibilidade para se controlar a si próprios e resolver os problemas que poderão surgir.

Meados 1950 — A mecânica é substituída pelo poder elétrico e hidráulico. George C. Devol desenvolve uma invenção, a qual, chama de “programmed articulated transfer device"; um autômato cujas operações (uma seqüência de operações determinadas pelas instruções) são programadas.

1959 — Devol e Joseph F. Engelberger desenvolvem o primeiro robô industrial pela Unimation Inc, capaz de executar automaticamente uma variedade de tarefas. Ele difere dos demais autômatos devido à possibilidade de ser reprogramado e remodelado para outras tarefas, com um nível de custos pouco elevado.

1960 — Nos anos 60, torna-se significativo o fato de a flexibilidade destas máquinas aumentar mediante o emprego de diferentes tipos de sensores. Dessa década em diante a investigação a respeito da robótica começa a incidir sobre o tema robótica móvel.

1962 — H.A.Ernest inicia o desenvolvimento de um computador capaz de controlador uma mão mecânica com sensores táteis (MH-1). Essa invenção consegue mover e “sentir” blocos, tornando possível, por

meio de controle da mão, o empilhamento de blocos sem ajuda humana. Tomovic e Boni desenvolvem um protótipo equipado com um sensor de pressão que, após “sentir” o objeto, transmite para o computador informações referentes ao seu tamanho.

1963 A American Machine & Foundry Company (AMF) introduz no mercado uma versão de um robô comercial (Versatran). São desenvolvidos diversos projetos de braços para manipuladores, tais como o braço Roehampton e o braço Edinburgh.

1968 McCarthy e colaboradores, no Laboratório de Inteligência Artificial de Stanford, desenvolvem um computador com “mãos, olhos, pernas e ouvidos” (manipuladores, câmeras de vídeo e microfones), demonstrando capacidade de identificação, reconhecimento e manipulação de blocos espalhados sobre uma mesa mediante a decodificação de mensagens faladas. Pieper estuda o problema de cinemática de um manipulador controlado por computador.

1969 O homem pisa no solo lunar pela primeira vez. Nessa época já se utilizam manipuladores para recolher amostras e executar pequenas tarefas em resposta ao comando de um controle remoto. O modo de tele-operação serve para efetuar escavações e outras tarefas de complexidade reduzida.

1970 A Robótica começa a incidir na pesquisa do uso de sensores para facilitar operações manuais.

1971 Kahn e Roth analisam a dinâmica e o controle de braços flexíveis usando controladores do tipo bang-bang (“near minimum time”).

1973 Em Standford, Balles e Paul utilizam um sensor visual e um sensor de peso demonstrando um braço controlado por computador para a montagem de bombas-d’água de automóveis.

1974 A Cincinnati Milacron introduz o primeiro robô industrial controlado por computador, denominado T3 ("The Tomorrow Tool", ou Ferramenta do Futuro), que move objetos numa linha de montagem. Inoue, no Laboratório de Inteligência Artificial,

	aprofunda estudos sobre a utilização de sensores de peso (força), enquanto Nevins, no Draper Laboratory, investiga diferentes técnicas de sensoriamento.
1975	Will e Grossman desenvolvem na IBM um manipulador, controlado por computador com sensores táteis e de peso, para realizar uma montagem mecânica de 20 partes de uma máquina de escrever.
1980	General Motors, em Detroit, nos Estados Unidos, introduz um robô industrial com "inteligência" eletrônica, capaz de reconhecer diferentes componentes numa tela transportadora e de escolher aqueles de que necessita.

“A idéia de se construir robôs começou a tomar força no início do século XX com a necessidade de aumentar a produtividade e melhorar a qualidade dos produtos. É nesta época que o robô industrial encontrou suas primeiras aplicações, o pai da robótica industrial foi George Devol. De 1980 até os dias de hoje, devido aos inúmeros recursos que os sistemas de microcomputadores nos oferecem, a robótica atravessa uma época de contínuo crescimento que permitirá, em um curto espaço de tempo, o desenvolvimento de robôs inteligentes fazendo assim a ficção do homem antigo se tornar a realidade do homem atual” [1].

2.2 Estado da Arte

A automação possibilita grandes incrementos na produtividade do trabalho, possibilitando que as necessidades básicas da população possam ser atendidas. Além de aumentar a produção, os equipamentos automatizados possibilitam uma melhora na qualidade do produto, uniformizando a produção, eliminando perdas e refugos.

A automação também permite a eliminação de tempos perdidos, ou seja permite a existência de "operários" que trabalhem 24 horas por dia sem requerer direitos trabalhistas, o que leva a um grande crescimento na rentabilidade dos investimentos.

“Sem dúvida a automação industrial foi e é um grande impulsionador da tecnologia de robótica. Cada vez mais tem se procurado aperfeiçoar os dispositivos, dotando-os de inteligência para executar as tarefas necessárias. Por exemplo, as redes

neurais objetivam linearizar os acionamentos eletromecânicos; com a lógica *Fuzzy* pode-se planejar a trajetória para robôs redundantes e, com os sistemas especialistas, é possível detectar vazamentos de água a partir da aquisição remota de consumo” [1].

Além dos vários benefícios que a Robótica nos proporciona, há também os impactos sociais que ela pode causar, por exemplo, o desemprego. “Este temor de desemprego vem aumentando a cada dia que passa. A queda nos custos dos robôs tornando-os acessíveis para muitos setores das indústrias, fez com que eles (os robôs) pudessem competir com a mão de obra barata, como a existente nos países do terceiro mundo, ameaçando o emprego de muitas pessoas. Muitas empresas multinacionais, que se instalavam em países subdesenvolvidos para utilizar-se do recurso "mão de obra barata", já estão pensando em reverter essa tendência e concentrar suas operações nos seus próprios países de origem, utilizando robôs para baratear seus custos.

O uso de robôs para as indústrias passa a ser uma questão de sobrevivência, assim, resistir ao uso dos robôs é uma batalha perdida, principalmente devido à forma acelerada com que eles caem de preço. Além disso, o sucesso que as empresas e países usuários de robôs vem obtendo é alto. O Japão, por exemplo, em 10 anos conseguiu quadruplicar a sua produção de automóveis, mantendo praticamente a mesma força de trabalho” [1].

2.3 Conceitos Básicos

Para o desenvolvimento do trabalho, alguns conceitos básicos são necessários, por exemplo: linguagem visual, banco de dados, geração de trajetórias, temporização com o 8254, PWM, porta paralela, servomotores, modelagem de robô (parâmetros de Denavit-Hartenberg, cinemática direta e inversa).

2.3.1 Linguagem Visual

Em um estudo sobre linguagens visuais, Shu [1998] afirmava que, tradicionalmente, as estruturas das linguagens de programação eram baseadas em representações textuais (unidimensionais, comando após comando) em decorrência da organização interna dos computadores. Para ele, as linguagens convencionais continuariam a ser usadas no futuro, mas para encorajar o desenvolvimento da “computação por usuários finais” (*end-user computing*), deveriam existir representações

mais amenas para a compreensão humana. Shu (1998) mencionava três premissas com base nas quais, as linguagens de programação visual cumpririam esse objetivo, embora ele mesmo admitisse não saber até que ponto elas se aplicavam no ambiente computacional:

1. Pessoas, em geral, preferem Figuras ao invés de palavras.

2. Figuras são mais poderosas do que palavras como meio de comunicação. Elas podem conter mais significado em uma unidade de expressão mais concisa.

3. Figuras não têm as barreiras de idioma que as linguagens naturais têm. Elas são entendidas pelas pessoas a despeito de que idioma elas falam.

Um estudo mais recente mostra que há um grande número de aspectos culturais dos quais depende a interpretação que se faz das imagens, uma vez que a associação do elemento visual com outras informações depende de um conhecimento prévio [3].

Há algum tempo atrás, quase todos os microcomputadores vinham acompanhados de um interpretador BASIC. Aprender a usar o computador incluia também o aprendizado dessa linguagem, de modo que cada novo usuário era, potencialmente, um novo programador.

HyperCard , lançado em 1987, foi o primeiro ambiente popular de programação baseado inteiramente em uma interface gráfica, destinado a substituir uma linguagem de programação convencional por uma linguagem visual. Antes disso, porém, VisiCalc já havia propiciado aos usuários um sistema para programação de cálculos, baseado em uma metáfora de planilha e na técnica de *programação por exemplo* (*Programming By Example — PBE*).

A importância histórica do VisiCalc transcende a simples definição de uma nova classe de aplicativos: ele tornou viável a pessoas comuns o uso do computador para a solução de problemas para os quais não existiam programas específicos, sem que tivessem de escrever uma única linha de código em BASIC ou outra linguagem convencional. O que caracteriza um sistema de programação para usuários finais, fundamentalmente, é o fato de ele permitir que a lógica do programa gerado seja deduzida a partir de uma estrutura (um *modelo*) montada através de uma interface visual [4].

A combinação das técnicas de visualização com interfaces gráficas e manipulação direta propiciou a criação de linguagens visuais destinadas à “condução de

computações científicas", [5], tais como AVS, DX e Khoros. Essas três ferramentas popularizaram o paradigma *caixas e setas*, usado posteriormente em Tioga [7], e constituem sistemas de programação voltados para usuários técnicos.

2.3.2 Banco de Dados

O Banco de Dados consiste numa base de dados ou coleção integrada de dados e informações armazenadas de maneira lógica e estruturada de forma a atender às necessidades daquele que se utiliza dessa base.

O Sistema de Gerenciamento de Banco de Dados é uma coleção de dados inter-relacionados e em conjunto com programas para acessar esses dados. Assim, estes são concebidos para gerenciar grandes quantidades de dados e informações com vistas a proporcionar maiores conhecimentos. Analisando as informações contidas no SGBD pode-se embasar as tomadas de decisões.

Segue abaixo a descrição da manipulação dos dados:

1) Recuperação da informação armazenada no banco de dados;
2) A inserção de novas informações no banco de dados;
3) A remoção de informações do banco de dados.

As tabelas do banco de dados contêm uma ou mais chaves primárias que indentificam cada linha da tabela atribuindo a ela um nome. Cada linha precisa ter sua própria identidade, ou seja, duas linhas não podem ter a mesma chave primária. A chave primária é composta por várias colunas de uma tabela. Como regra, essas são, normalmente, as primeiras colunas. A chave primária é como um substantivo, porque nomeia o objeto de cada linha, as outras colunas como os adjetivos, uma vez que oferecem informações adicionais sobre o objeto.

O nome da coluna é considerado parte da definição da tabela. Em contrapartida, o nome da linha, que é a chave primária da linha, é considerada parte do dado da tabela. Há duas regras para organizar as colunas da chave primária de uma tabela:

1) Nenhuma das colunas da chave primária pode ser Null. Isso faz sentido, porque uma chave Null contém um valor desconhecido, por essa razão, se uma coluna de chave primária fosse Null, isso significaria que não é possível saber a indentidade do objeto ou da linha.

2) Cada linha precisa ter uma identidade que seja diferente de todas as outras linhas da tabela, isto é, duas linhas não podem ter a mesma identidade – os mesmos

valores em todas as colunas da chave primária. Duas linhas de uma tabela precisam pelo menos ter uma coluna primária em que os valores sejam diferentes.

O modelo entidade-relacionamento do banco de dados baseia-se em uma percepção de mundo real que consiste em uma coleção de objetos chamados de entidades, e de relacionamentos entre esses objetos. Assim, uma entidade é um objeto existente e é distinguível de outro objeto, ou seja: ele é uma entidade única. A essa distinção é associado um conjunto de atributos que descrevem esse objeto. Por exemplo: Uma pessoa tem um nome e é de um sexo (mulher chamada Maria e é do sexo feminino).

A forma de representação da entidade-relacionamento é E-R, assim, em adição a entidade e relacionamento o modelo E-R representa certas restrições as quais o conteúdo do banco de dados deve-se adequar. Uma restrição importante é o mapeamento de cardinalidades que expressa o número de entidades às quais outras entidades podem ser associadas por meio de um conjunto de relacionamentos.

A estrutura conceitual global de um banco de dados pode ser expressa graficamente por um diagrama E-R, como pode ser visto na Figura 2.1, que consiste nos seguintes componentes:

1) Retângulos – que representam conjuntos de entidades;

2) Elipses – que representam os atributos;

3) Losangos – que representam relacionamentos entre conjuntos de entidades;

4) Linhas – que ligam atributos a conjuntos de entidades e relacionamentos.

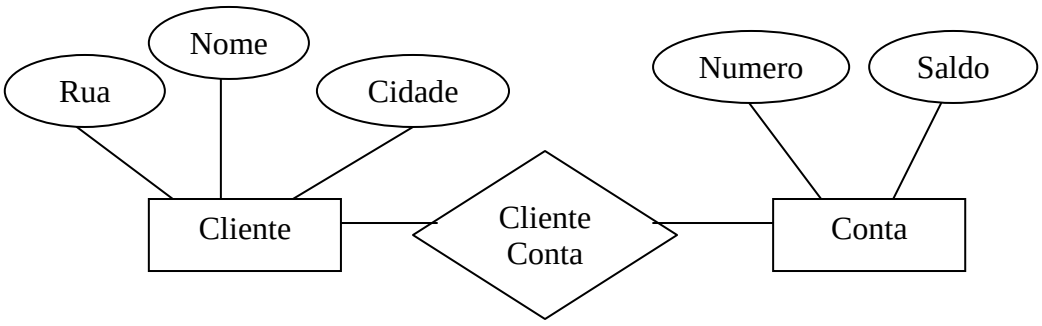


FIGURA 2.1 - DIAGRAMA E-R

2.3.3 Temporização com o 8254

O 8254 (programmable Timer/Counter – ver Anexo D), permite gerar interrupções em intervalos de tempo regulares. Incluído nos computadores, este

dispositivo permite controlar eventos em tempo real. Como é possível observar na Figura 2.2 extraída de [8], o 8254 é composto por três contadores decrementais de 16 bits cada um. Estes contadores podem ser utilizados independentemente um do outro. Cada contador é capaz de contar em binário ou em BCD e possui freqüência máxima de entrada de 10MHz, e pode operar em 6 modos distintos:

modo 0: interrupção em contagem final

modo 1: monoestável redisparável por *hardware*

modo 2: gerador de taxa

modo 3: gerador de onda quadrada

modo 4: *strobe* disparável por *software*

modo 5: *strobe* disparável por *hardware* (redisparável)

Estes modos podem ser programados no 8254 através da escrita da palavra de controle.

O 8254 permite controlar o relógio em tempo real do computador (gera uma interrupção a cada 1/18,2 s), efetuar contagens de eventos, iniciar o *refresh* da memória DRAM, proporciona sinais de som para o *speaker* interno e pode servir como temporizador para um usuário.

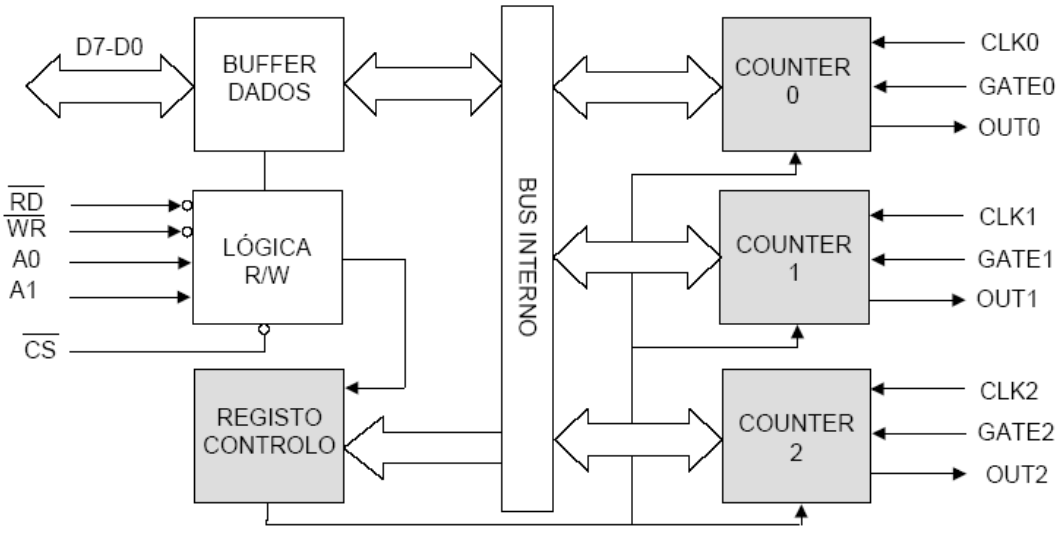


FIGURA 2.2 - O 8254 - CONTADOR PROGRAMÁVEL

As duas entradas A0 e A1 permitem habilitar cada um dos três contadores e registro de controle.

Os sinais de Gate permitem iniciar ou terminar uma contagem através de um sinal de *hardware* externo.

O contador 0 gera as interrupções do relógio. O CLK0 tem uma freqüência de 1,19318 MHz. Assim, o registro é decrementado a intervalos constantes (1.193.180 vezes/s). Quando o registro é decrementado até zero, um sinal de interrupção no IRQ0 do 8259 A é gerado. Isto significa que o contador gera interrupções a cada 1.193.180/65536 segundos, logo são geradas 18,2 interrupções por segundo.

O contador 1 encontra-se programado para enviar interrupções a cada 15µs, utilizadas para pedir uma operação de DMA de *refresh* de memória DRAM.

O contador 2 está associado à criação de tons para o *speaker*.

Seleciona-se o contador e o Registro de controle, de acordo com o endereço e valores de A1 e A0, como é possível ver na Tabela 2.1 extraída de [8].

Tabela 2.1- Endereçamento do 8254.

A1	**A0**	**Porta**	**Seleciona**
0	0	40h	Contador 0
0	1	41h	Contador 1
1	0	42h	Contador 2
1	1	43h	Registro de Controle

A porta 61H ativa ou desativa o *speaker* do computador (Figura 2.3). Caso este esteja ativo, não será possível acionar os contadores.

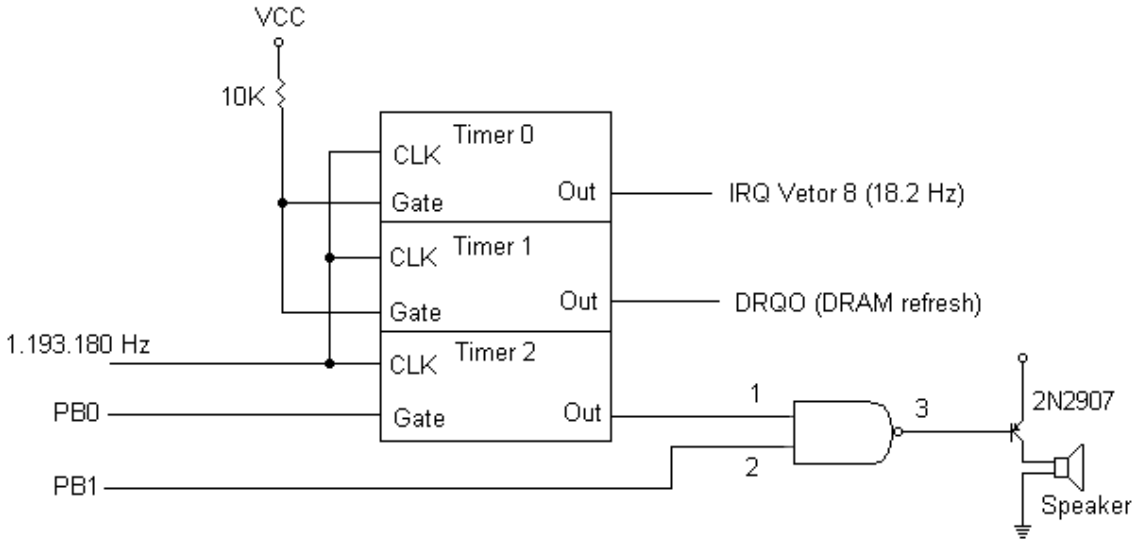


FIGURA 2.3 - A PORTA 61H PROGRAMA PB0 E PB1

2.3.4 Porta Paralela

A porta paralela é uma interface de comunicação entre o computador e um periférico. Quando a IBM criou seu primeiro PC (*Personal Computer*) ou Computador Pessoal, a idéia era conectar a essa porta uma impressora, mas atualmente, são vários os periféricos que se utilizam desta porta para enviar e receber dados para o computador (exemplos: *scanners*, câmeras de vídeo, unidade de disco removível e outros).

A Porta Paralela está ligada diretamente à placa mãe do computador. Deve-se ter muito cuidado ao conectar circuitos eletrônicos a essa porta, pois, uma descarga elétrica ou um componente com a polaridade invertida, poderá causar danos irreparáveis ao seu computador [9].

2.3.4.1 Modelos de Porta Paralela

2.3.4.1.1 Transmissão unidirecional

A porta paralela SPP (Standard Parallel Port) pode chegar a uma taxa de transmissão de dados de 150KB/s. Comunica-se com a CPU utilizando um barramento de dados de 8 bits [9]. Para a transmissão de dados entre periféricos são usados 4 bits por vez.

2.3.4.1.2 Transmissão bidirecional

A porta avançada EPP (*Enhanced Parallel Port*) chega a atingir uma taxa de transferência de 2MB/s. Para atingir essa velocidade, é necessário um cabo especial. Comunica-se com a CPU utilizando um BUS de dados de 32 bits. Para a transmissão de dados entre periféricos são usados oito bits por vez.

A porta avançada ECP (*Enhanced Capabilities Port*) tem as mesmas características que a EPP, porém, utiliza DMA (acesso direto à memória), sem a necessidade do uso do processador, para a transferência de dados. Utiliza também um buffer FIFO de 16 bytes [9].

2.3.4.2 Endereços da Porta Paralela

O computador nomeia as portas paralelas, chamando-as de LPT1, LPT2, LPT3 etc, mas, a porta física padrão do computador é a LPT1, e seus endereços são: 378h (para enviar um byte de dados pela porta), 378+1h (para receber um valor através da Porta) e, 378+2h (para enviar dados). Às vezes pode estar disponível a LPT2, e seus

endereços são: 278h, 278+1h e 278+2h, com as mesmas funções dos endereços da porta LPT1 respectivamente. Na Tabela 2.2 extraída de [9] é possível ver maiores detalhes dos endereçamentos.

Tabela 2.2 - Portas paralelas e endereços.

Nome da Porta	Endereço de memória	Endereço da Porta	Descrição
LPT1	0000:0408	888(d) 78(H)	Endereço base
LPT2	0000:040A	632(d) 78(H)	Endereço base

2.3.4.3 Registradores

Utilizando a Porta Paralela conectada a uma impressora, os endereços terão nomes, como segue na Tabela 2.3 [9].

Tabela 2.3 - Registradores da Porta Paralela.

Nome	Endereços LPT1	Endereços PT2	Descrição
Registro de Dados	378h	278h	Envia um byte para a impressora
Registro de Status	379h	279h	Ler o Status da impressora
Registro de Controle	37Ah	27Ah	Envia dados de controle para a impressora

2.3.4.4 O Conector DB25

O DB25 é um conector que fica na parte de trás do gabinete do computador, e é através deste, que o cabo paralelo (Figura 2.6 – conector macho do cabo paralelo) se conecta ao computador para poder enviar e receber dados [9].

No DB25, um pino está em nível lógico 0 quando a tensão elétrica está entre 0 a 0,4V. Um pino se encontra em nível lógico 1 quando a tensão elétrica está entre 3.1V e 5V.

As Figuras 2.4 e 2.5 [9] mostram o conector padrão DB25, com 25 pinos, onde cada pino tem um nome que o identifica.

Devido aos diversos modos que a porta paralela pode trabalhar, mostra-se na Figura 2.7 o funcionamento no modo SPP.

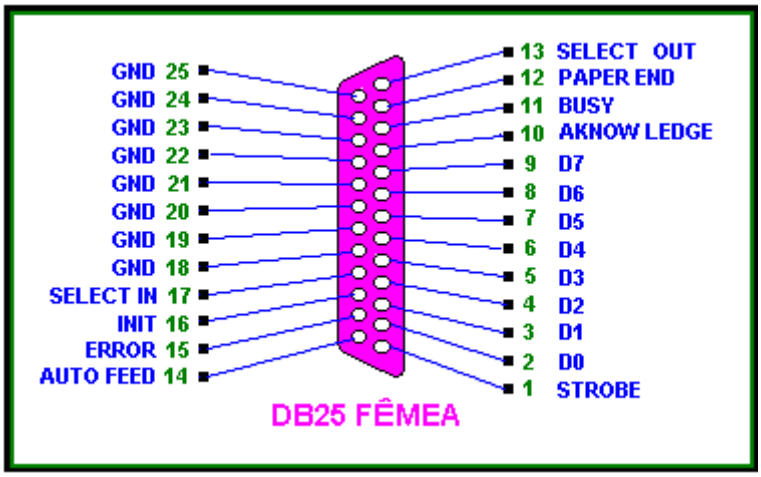


FIGURA 2.4 - CONECTOR DB25 FÊMEA.

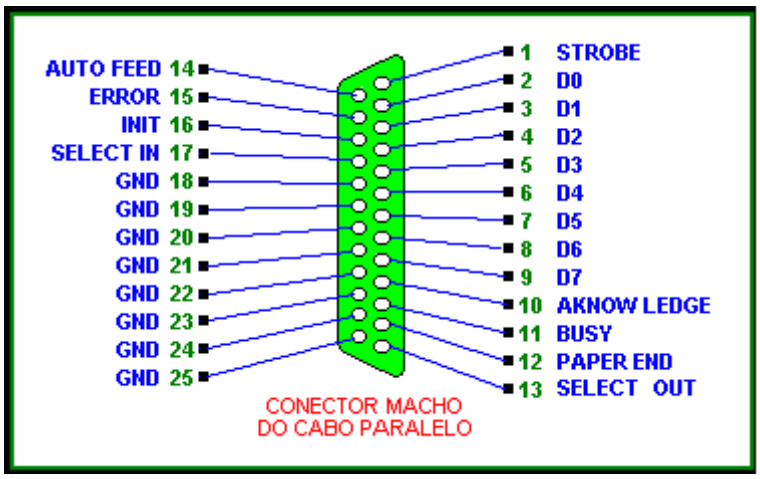


FIGURA 2.5 - CONECTOR DB25 MACHO.

FIGURA 2.6 - FOTO DO CONECTOR DB25 MACHO DO CABO PARALELO.

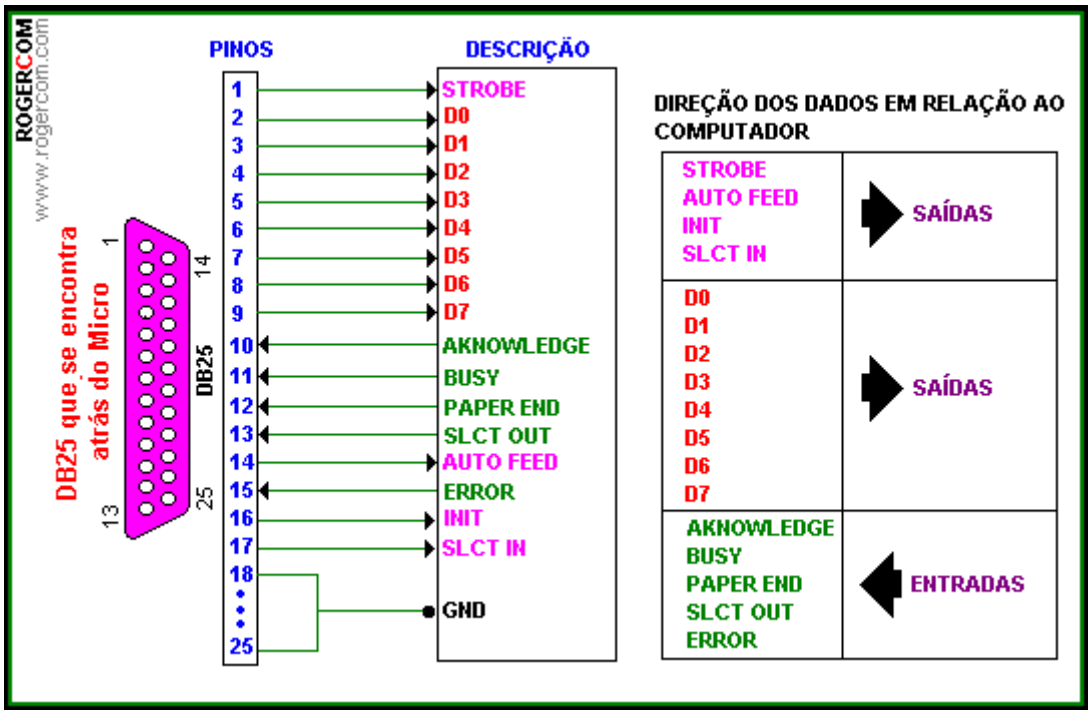


FIGURA 2.7 - ESQUEMA DE FUNCIONAMENTO DO DB25 NO MODO SPP.

2.3.5 PWM

"A **Modulação por largura de pulso** -mais conhecida pela sigla em inglês "PWM" (*Pulse-Width Modulation*)- de um sinal ou em fontes de alimentação, envolve a modulação de sua razão cíclica (*duty cycle*) para transportar qualquer informação sobre um canal de comunicação ou controlar o valor da alimentação entregue a carga." [10].

PWM é uma maneira de obter sinal alternado de baixa freqüência, através de uma modulação em alta freqüência. Consiste basicamente em aplicar uma onda quadrada de amplitude Vcc e freqüência alta no lugar da tensão contínua. A tensão média varia em função do tempo que a onda fica em nível alto e do tempo que a onda fica em nível baixo. A relação entre o tempo que a onda fica em nível alto e o período total é conhecido como *Duty Cycle*, que é normalmente expresso em percentual. Portanto, para uma modulação PWM com *Duty Cycle* igual a 50%, por exemplo, metade do tempo a tensão fica em nível alto e metade do tempo em nível baixo, [11].

Consegue-se obter este tipo de modulação, comparando uma tensão de referência (que seja imagem da tensão de saída buscada), com um sinal triangular simétrico, cuja freqüência determine a freqüência de chaveamento. "A freqüência da onda triangular

(chamada portadora) deve ser, no mínimo 20 vezes superior à máxima freqüência da onda de referência, para que se obtenha uma reprodução aceitável da forma de onda desejada, após efetuada a filtragem. A largura do pulso de saída do modulador varia de acordo com a amplitude relativa da referência em comparação com a portadora (triangular). Tem-se, assim, uma Modulação por Largura de Pulso" [12].

A tensão de saída é obtida através de uma sucessão de ondas retangulares de duração variável e amplitude igual à tensão de alimentação CC.

O sinal modulado é um trem de pulsos com a largura variando de acordo com a amplitude da informação, quanto maior a amplitude maior a largura e vice-versa. A freqüência do sinal modulado é igual a freqüência do sinal de portadora.

2.3.6 Servomotores

São motores que acionam um redutor que está acoplado a um potenciômetro que irá dar o "feedback" de posicionamento ao sistema eletrônico interno.

Para movimentar o servomotor, é necessário gerar um sinal PWM. Dependendo da largura do pulso, o eixo do servomotor terá um deslocamento angular diferente [13].

Segue na Figura 2.8, adaptada de [13], um servomotor que é muito utilizado em aeromodelismo.

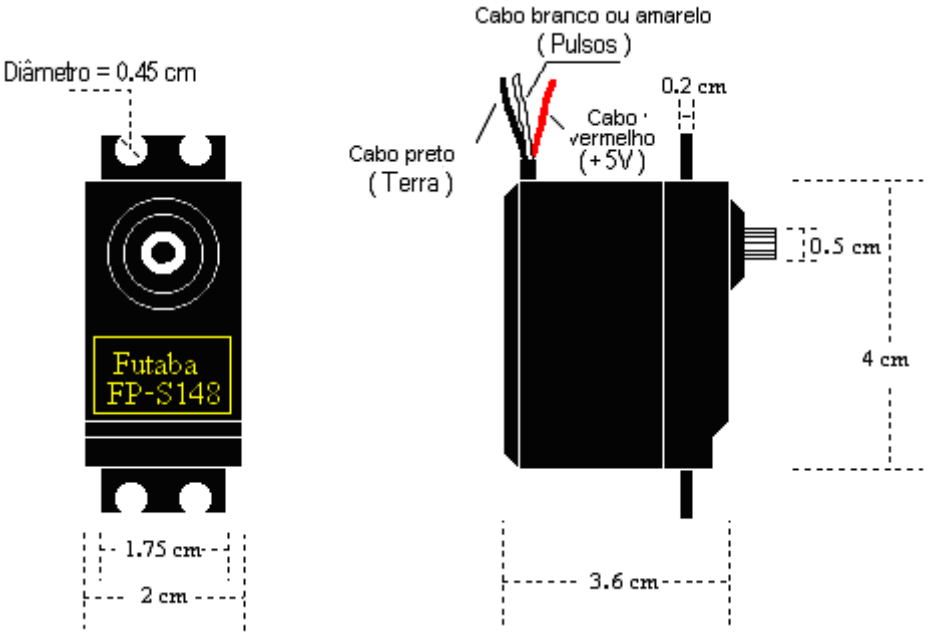


FIGURA 2.8 - MEDIDAS DO SERVOMOTOR FP-S148.

Um servomotor deste tipo é basicamente um motor elétrico DC cujo o eixo pode girar aproximadamente 180 graus (não realizam voltas completas como os motores normais). Note que tem três cabos que saem de sua caixa. O vermelho é alimentação de tensão (+5V), o preto é o terra (0V ou GND). O cabo branco (ou amarelo) é o cabo que recebe os pulsos relativos à rotação (0 a 180 graus).

Dentro do servomotor, um circuito controlador comanda um motor de corrente continua indicando quantas voltas ele deve girar para posicionar o eixo externo na posição que foi solicitada [13].

Na Figura 2.9, adaptada de [13] vemos como estão posicionadas essas peças dentro do servomotor.

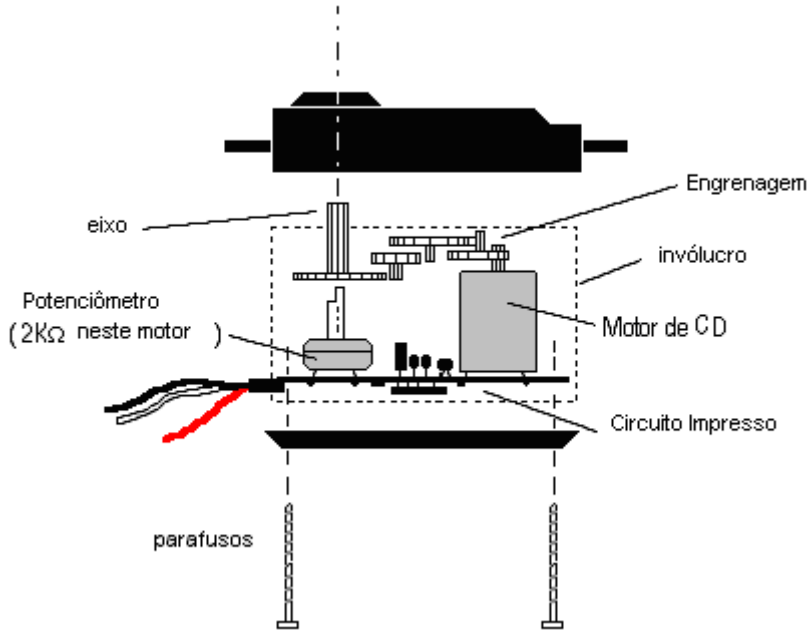


FIGURA 2.9 - INTERIOR DO MOTOR.

O potenciômetro está acoplado ao eixo do servomotor fornecendo um sinal relativo ao deslocamento angular. Um sinal de pulsos deve estar sempre presente no fio branco (ou amarelo). Se por alguma razão, for necessário que o servomotor se mantenha numa posição fixa, e não podendo gerar pulsos, deve-se aterrar este terminal (fio branco ou amarelo).

Pela Figura 2.10 também adaptada de [13] pode-se ver a forma do sinal de pulsos que controla o servo.

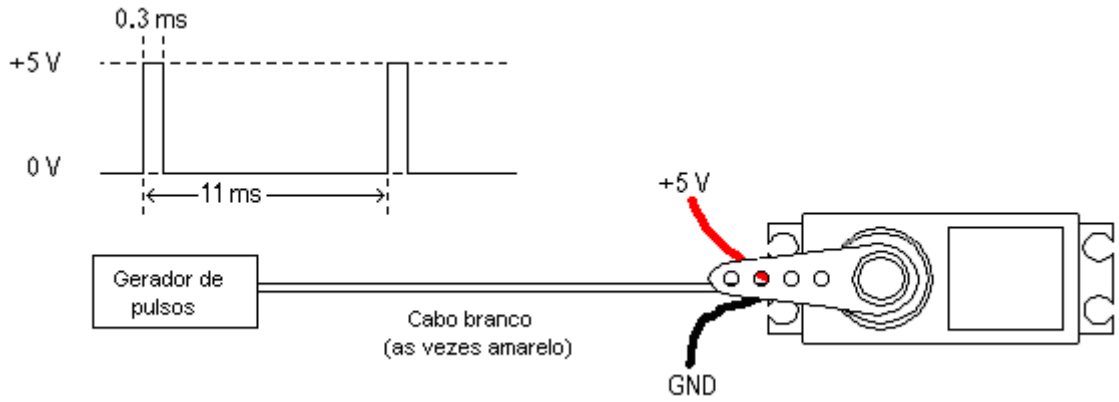


Figura 2.10 - Controle do servomotor.

Nota-se que o intervalo de tempo entre os pulsos se mantém constante, e a variação da largura do pulso indica ao servo a posição deseja. Estes valores de milisegundos têm funcionado muito bem para os servomotores FUTABA FP-S148, FUTABA S3003, Hitec HS-300 e HOBBICO COMMAND CS-51. Estes modelos respondem adequadamente a pulsos desde 50Hz até aproximadamente 100Hz, sendo que uma vez escolhida uma freqüência de operação deve-se procurar mantê-la fixa por todo o tempo.

Para cada tipo de servo deve-se realizar um teste preliminar para encontrar exatamente o período e a duração dos pulsos que melhor funcionem. Um osciloscópio e um gerador de sinais facilitam este trabalho [13].

2.3.7 Modelagem de Robô

Este tópico tem a finalidade de apresentar os conceitos básicos de modelagem de robôs de forma sintética. Para um aprofundamento na matéria aconselha-se consultar as referências [1] e [2].

2.3.7.1 Cinemática direta de manipuladores

Cinemática direta de manipuladores é função das variáveis das juntas, isto é, a determinação da posição e orientação do órgão terminal (no espaço cartesiano) em relação ao sistema coordenado da base do manipulador [2].

No caso da robótica é interessante definir dois termos, quais sejam: espaço das juntas e espaço cartesiano (ou das tarefas). Os elementos do espaço das juntas são as variáveis, ângulos no caso das juntas rotacionais, que indicam o deslocamento das juntas. Os elementos do espaço cartesiano são as posições 3D do órgão terminal.

Pode-se dizer que a cinemática direta serve para encontrar a posição tridimensional, juntamente com os valores de rotação (roll), guinada (yaw) e inclinação (pitch) de cada junta do robô, como se pode ver na Figura 2.10 retirada de [1].

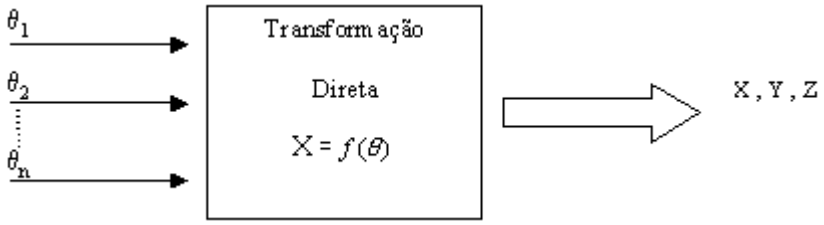


FIGURA 2.10- TRANSFORMAÇÃO DIRETA DE COORDENADAS

Parâmetros de Denavit-Hartenberg

Dentre as várias maneiras existentes para calcular a cinemática direta, escolhe-se aquela que utiliza os parâmetros de Denavit-Hartenberg.

De [2] "A notação de Denavit-Hartenberg é introduzida como um método sistemático para descrever-se um elo separando um par sucessivo de juntas (rotativas ou prismáticas) e o relacionamento entre um par de elos adjacentes. Para isso um número mínimo de parâmetros (de Denavit-Hartenberg) são utilizados... a base do manipulador é considerada como o elo 0, e os outros elos recebem numeração crescente de 1 a n (progressivamente da base para o órgão terminal)".

As juntas são numeradas de 1 a n da mesma forma. A cada elo é afixado um sistema de coordenada. O sistema coordenado {0} permanece imóvel (referência).

Na Figura 2.11 extraída de [2] tem-se a representação gráfica dos parâmetros de denavit-hartenberg.

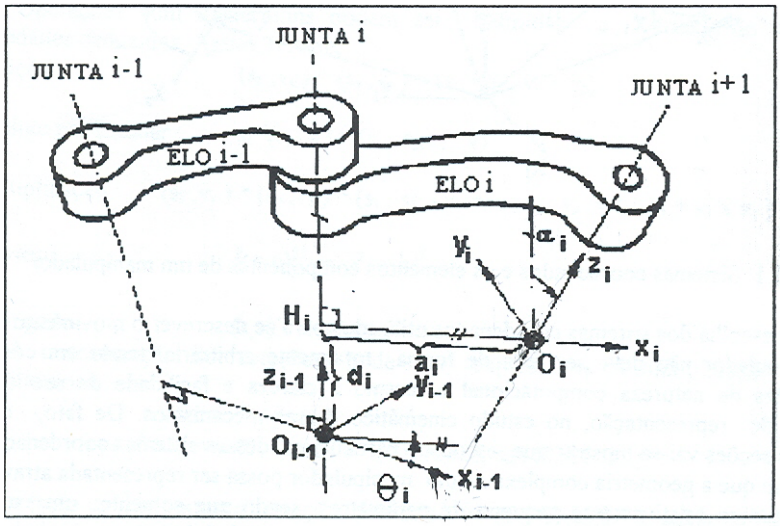


FIGURA 2.11 - NOTAÇÃO DE DENAVIT-HARTENBERG.

Abaixo está descrita a convenção para fixar os sistemas coordenados nos elos do manipulador:

A origem O_i do i-ésimo sistema coordenado (sistema do elo i) está localizada na interseção do eixo da junta i+1 e a normal comum entre os eixos da junta i e i+1 (H_iO_i);

O eixo $\hat{x}_i$ é direcionado ao longo da extensão da normal comum (apontando no sentido da junta i para a junta i + 1);

O eixo $\hat{z}_i$ está ao longo do eixo da junta i + 1;

O eixo $\hat{y}_i$ é escolhido tal que o sistema resultante segue a regra da mão direita.

Dado o exposto acima, o elo i e a situação relativa dos dois sistemas coordenados {i} e {i - 1}, podem ser descritos por quatro parâmetros, denominados parâmetros de Denavit-Hartenberg, quais sejam:

a_i: comprimento da normal comum, ou seja, a distância entre os eixos das juntas em cada terminal do elo i, medida ao longo da linha normal a ambos esses eixos (ao longo de $\hat{x}_i$);

α_i: ângulo entre o eixo da junta i ($\hat{z}_{i-1}$) e o eixo da junta i+1 ($\hat{z}_i$) no sentido da mão direita, medida em torno de $\hat{x}_i$.

d_i: distância entre a origem O_{i-1} e o ponto H_i (distância entre as normais comuns H_{i-1} O_{i-1} e H_iO_i), medida ao longo de $\hat{z}_{i-1}$.

θ_i: ângulo entre o eixo $\hat{x}_{i-1}$ e a normal comum H_iO_i (eixo $\hat{x}_i$), medido em torno do $\hat{z}_{i-1}$ no sentido da mão direita.

2.3.7.2 Cinemática inversa de manipuladores

Cinemática inversa do manipulador consiste em determinar-se o conjunto de ângulos e/ou deslocamentos das juntas que propiciam posição e orientação específicas para o órgão terminal no espaço cartesiano.

A coordenação de movimentos, que consiste na obtenção de um movimento de referência (angular) para cada junta, para um dado movimento de referência do elemento terminal (cartesiano), é expressa matematicamente pela inversão do modelo geométrico, que representa uma função não-linear.

Os métodos analíticos conduzem à obtenção de todas as soluções. Eles não são gerais, isto é, a inversão analítica não é trivial e, além disso, não há garantia de que seja possível fazê-la para um robô qualquer. Os métodos analíticos são adequados para robôs simples, aqueles que possuem um grande número de denavit-hartenberg nulos.

Métodos analíticos interativos são métodos que convergem para uma solução possível entre todas as existentes. Tem caráter geral e, com o atual desenvolvimento dos microcomputadores, sua utilização em tempo real é viável. Diferentemente das soluções analíticas, as soluções numéricas podem ser combinadas com estratégias de anticolisão com objetos cujas posições e dimensões sejam conhecidas [1].

Na Figura 2.12 pode se ver um diagrama da transformação inversa de coordenadas.

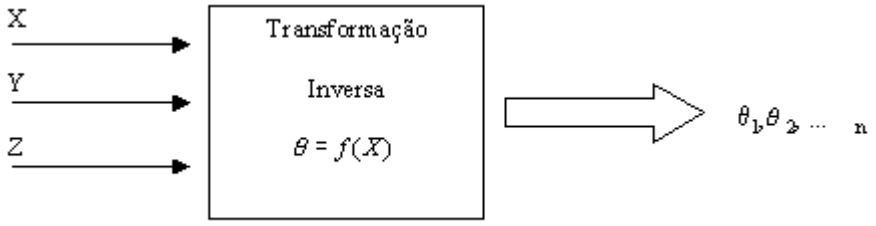


FIGURA 2.12 - TRANSFORMAÇÃO INVERSA DE COORDENADAS

2.3.8 Geração de Trajetórias

"A geração de trajetórias é realizada a partir do modelo geométrico do robô e representa a evolução no tempo da posição, da velocidade e da aceleração das juntas do robô, isto é, a geração de incrementos angulares de juntas necessárias para que o robô realize determinada tarefa" [1].

O planejamento de trajetórias pode ser feito no espaço cartesiano ou no espaço das juntas.

No espaço das juntas: o planejamento da trajetória é realizado junta a junta pelo operador. Neste caso não há necessidade de se obter o seu modelo cinemático inverso, pois o operador leva o órgão terminal (diretamente ou via "teach pendant") à meta desejada. As variáveis das juntas são lidas através de sensores e registradas pelo controlador para servir de referência ao movimento.

No espaço de tarefas: o planejamento da trajetória está relacionado à ferramenta de trabalho, assim precisa-se de um modelo cinemático que relacione as coordenadas de junta com o sistema de coordenadas cartesianas na base do robô.

2.4 Metodologia

Primeiramente, será construída a estrutura física de um manipulador robótico de forma a obter três graus de liberdade. O manipulador será construído com componentes do kit Robix disponível na FCC-UNIVAP. Após a construção do manipulador serão medidas as distâncias entre as juntas e as dimensões dos elos para se determinar os parâmetros de denavit-hartenberg (D-H). A partir da obtenção dos parâmetros de D-H do robô determina-se a cinemática direta, representada por matrizes de transformação homogênea que relacionam os sistemas de coordenadas de elos adjacentes do manipulador. As matrizes obtidas serão utilizadas posteriormente para a determinação da cinemática inversa do manipulador. Dessa forma tem-se o modelo cinemático do manipulador.

Será utilizada uma linguagem visual, no caso, Visual Basic, para a implementação da interface gráfica que gera trajetórias e envia os sinais de comando para o manipulador. O desenvolvimento do *software* de controle será realizado simultaneamente com a modelagem do robô. Será desenvolvido um aplicativo que possui uma entrada para a posição e orientação do órgão terminal, em determinados momentos da trajetória desejada; um banco de dados para armazenar as trajetórias definidas pelo usuário, que o robô deverá seguir; uma rotina de geração de trajetórias; uma rotina de visualização de trajetórias e; uma rotina para envio dos dados, aos atuadores do manipulador, via porta paralela. Para sincronizar o sinal de saída para os atuadores será usado o temporizador do PC.

Após a implementação do *software*, serão realizados os testes finais, integrando todo o sistema. Serão realizados testes a partir das equações das cinemáticas direta e inversa para a validação do modelo cinemático e verificação do funcionamento do sistema. Após análise dos resultados, serão feitos os ajustes necessários, de forma a atender todos os requisitos especificados.

Capítulo 3

Desenvolvimento

3.1 Especificação do sistema

O robô construído é composto por três juntas. Os ângulos de rotação dos eixos das três juntas (1, 2 e 3) têm faixas de 0^o a 180^o. Os limites dos ângulos de rotação determinam o volume do espaço de trabalho do robô. A velocidade do órgão terminal do manipulador que percorre a trajetória gerada deve ser constante. Nesta versão está se analisando a cinemática. O foco do trabalho é o desenvolvimento de uma primeira versão de uma interface gráfica para controle de um manipulador robótico via porta paralela do PC.

O sistema deverá realizar o cálculo dos ângulos, através das coordenadas do espaço cartesiano informadas pelo usuário. Com base nos ângulos obtidos através da cinemática inversa, o *software* mostrará graficamente a trajetória escolhida, e enviará estes valores para o servomotor através de um sinal PWM via porta paralela.

Os sinais que são gerados na porta paralela estão dimensionados para trabalhar com servomotores FUTABA. A freqüência dos pulsos é de 75Hz e o *duty cycle* varia de 0,9ms (0^o) à 2,2ms (180^o). Cada pino da porta paralela fornece corrente suficiente para acionar um servomotor, ou seja, é possível comandar até oito servomotores sem a necessidade de demultiplexadores.

O *software* funcionará no sistema operacional Windows, porém para o usuário otimizar o banco de dados é necessário que tenha instalado o *software Access*.

3.2 Projeto Preliminar

Para um melhor entendimento do projeto é necessário uma análise do diagrama de blocos descrito na Figura 3.1.

FIGURA 3.1 - DIAGRAMA EM BLOCOS DO PROJETO

O projeto foi separado em três blocos, quais sejam: usuário, interface gráfica e robô.

O primeiro bloco é o usuário final, entra com os dados no *software* e visualiza dados e a trajetória percorrida pelo robô.

O segundo bloco, a Interface Gráfica, também pode ser considerado a IHM (Interface homem-máquina) do sistema, ou seja, o *software* que interpreta os comandos do usuário, codifica-os em sinais elétricos para acionar os atuadores do manipulador e fornece ao usuário a visualização dos movimentos que o robô realiza no espaço de trabalho. A codificação, basicamente é a conversão das coordenadas cartesianas em coordenadas de juntas, para a obtenção da posição angular de cada junta. Para o atuador de cada junta é enviado um sinal PWM com largura de pulso correspondente ao deslocamento angular que o eixo do servomotor deve realizar.

O último bloco é o robô, a saída do sistema, e contém os servomotores, elos, juntas e conexões. Ele foi construído para ter três graus de liberdade.

3.2.1 *Software*

Para o desenvolvimento do *software*, poderia ser utilizada qualquer linguagem de programação que fosse capaz de fazer comunicação com o *hardware* através da porta paralela. É aconselhado utilizar uma linguagem visual, pois isto torna a interface mais amigável para o usuário. Escolheu-se o Visual Basic para o desenvolvimento deste projeto, devido ao fato de ser uma linguagem orientada a eventos, servir para gerar aplicações que serão executadas em ambientes gráficos como o Windows, ser mais amigável para o usuário final, ser muito versátil, de aprendizado relativamente fácil, comparado a outras linguagens e possuir uma extensa literatura no mercado.

Para instalar e executar o *software*, são necessários os seguintes requisitos de computador:

- Sistema operacional Windows 95, 98, 2000, XP;
- 128 MB de memória RAM;
- Porta paralela disponível;
- Pentium II (ou processador equivalente);
- CD-ROM (para a instalação do *software*);

A Figura 3.2 mostra o fluxograma resumido do *software*.

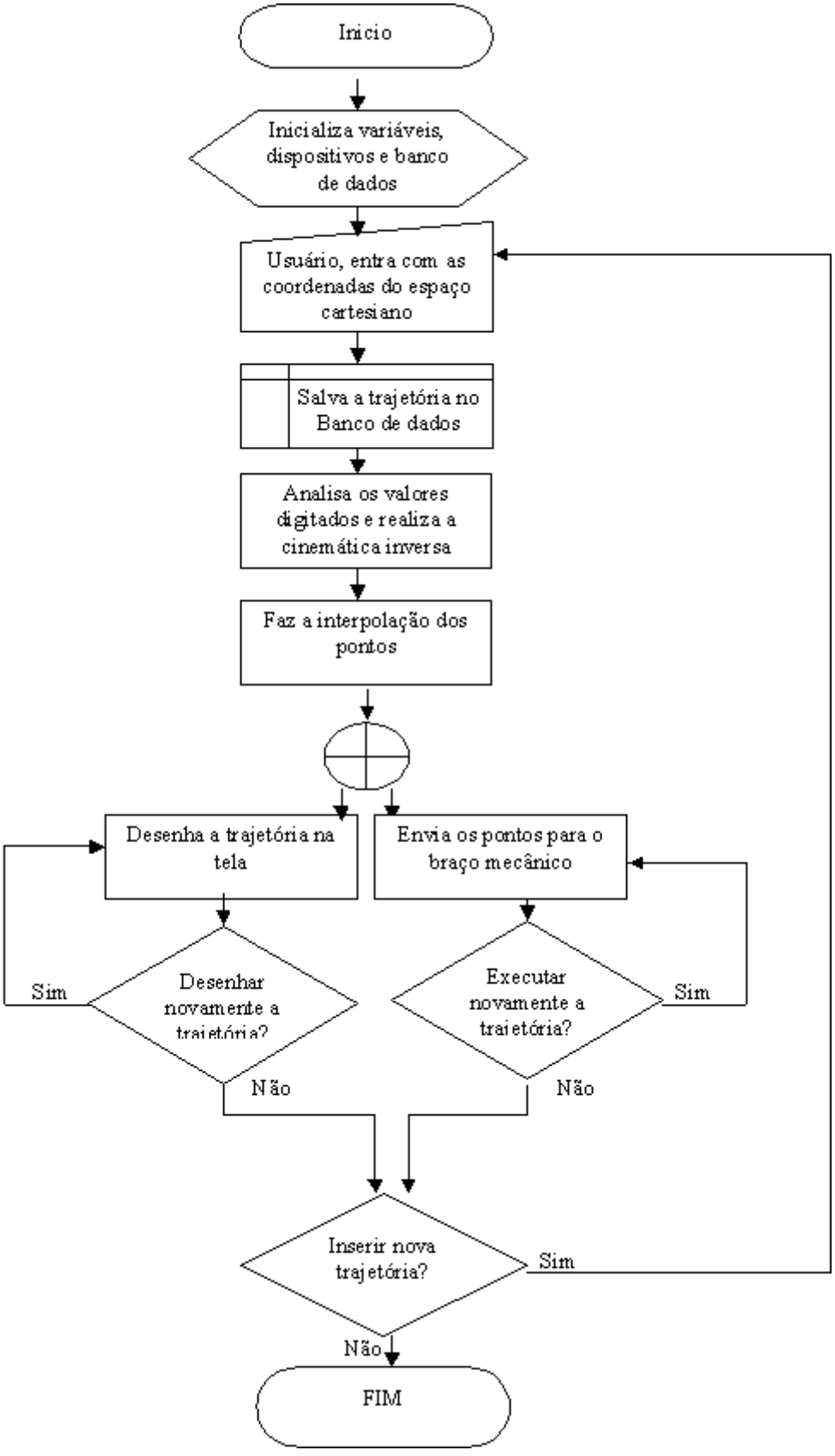


FIGURA 3.2 – FLUXOGRAMA RESUMIDO DO *SOFTWARE*

Explicação do fluxograma

O programa a ser desenvolvido necessita, primeiramente, de uma rotina de iniciação do dispositivo, banco de dados e variáveis. Esse dispositivo inclui a rotina de escrita da porta paralela.

Após a iniciação, caso não ocorra nenhum problema, será executado o programa, que espera a ocorrência do evento ativação do botão “escolher trajetória”. O usuário, entra com as coordenadas do espaço cartesiano. Depois, sob comando do usuário, a trajetória pode ser salva no banco de dados. Com a trajetória já colocada em um vetor interno ao *software*, os dados são analisados, e caso não exista nenhum erro, é calculada a cinemática inversa, obtendo-se assim os ângulos iniciais para se realizar a interpolação. O usuário deverá escolher se quer ver a trajetória na tela ou executá-la primeiramente. Após isto, poderá escolher se deseja repetir a operação. Então, o usuário deverá escolher se deseja criar uma nova trajetória. Caso positivo, o sistema abrirá a tela para que este possa entrar com os valores novamente. Caso negativo, o sistema finaliza as variáveis liberando memória para outros aplicativos e finaliza o *software*.

3.2.2 Banco de Dados

Para que o aplicativo possa armazenar as trajetórias escolhidas pelo usuário, fez-se necessária a criação de uma base de dados.

Mesmo conhecendo a vulnerabilidade do banco de dados *Access*, ele foi escolhido pelo fato de ser o mais fácil SGBDs (Sistemas de Gerenciamento e Bancos de Dados) de ser usado por qualquer pessoa, mesmo não sendo um especialista em administração de banco de dados (DBA – Data Base Administrator).

O *Access* apresenta alguns pontos positivos, conforme pode-se ver abaixo [14]:

- o *Access* exige em torno de 16 MB de memória para um bom desempenho;
- pode-se ter várias tabelas abertas ao mesmo tempo, 254 para ser exato, mas apenas um Banco de Dados por vez (o *Access* pode trabalhar com até 32.768 tabelas em um único banco de dados);
- um único arquivo .MDB pode conter objetos de dados (tabelas, consultas e índices) e objetos da aplicação (formulários, relatórios, macros e módulos);
- uma TABELA do *Access* pode importar dados tipo: texto, Excel, Lotus 1-2-3, FoxPro, Paradox, xBase, SQL etc.

3.2.3 Interpolação e filtragem de pontos de passagem no espaço das juntas

Com o objetivo de traçar uma trajetória a partir de determinados pontos de passagem obtidos pelo operador, no espaço das juntas, torna-se necessária a implementação de algoritmos de interpolação linear. É imprescindível a realização de uma filtragem, uma vez que a realização somente da interpolação acarreta acelerações elevadas no início de uma trajetória, o que não é aconselhável para nenhum dispositivo robótico. Portanto, para que a aceleração seja baixa, o que resultará em alta velocidade da junta, deve-se colocar mais pontos na parte inicial e final da trajetória. Isso é feito pela filtragem da trajetória interpolada.

O objetivo principal da interpolação linear, é a criação de uma seqüência de pontos de passagem que interligam os pontos da trajetória inicial dada [1].

3.2.4 Sinal PWM

Para movimentar o servomotor, temos de gerar um sinal PWM. Dependendo da largura do pulso, o servomotor irá se posicionar diferente. De acordo com o servomotor escolhido (FUTABA), tem-se que obter uma forma de onda com freqüência de pulsos de 75Hz e *duty cycle* variando de 0,9ms (0^o) à 2,2ms (180^o).

3.2.5 Temporização

Verificou-se a possibilidade de gerar o sinal para o robô sem que este ficasse dependente apenas de um determinado computador. Com isso verificou-se que não seria possível utilizar o *delay* do computador, o que implicaria em problemas de atraso e sinais variáveis, ou seja, a cada mudança de máquina, os parâmetros do aplicativo deveriam ser atualizados. Desta forma não seria possível ter um *software* de instalação, uma vez que o *delay* está relacionado com a velocidade de processamento do computador. Para desenvolver um sistema compatível com qualquer computador optou-se pela utilização do temporizador do PC, o CI 8254, que permite gerar sinais de sincronismo de 18,2 Hz até 1.193.810 Hz, possibilitando o controle de eventos em tempo real.

3.2.6 Cabo Paralelo

A montagem do cabo paralelo para utilização no sistema deve seguir algumas recomendações. O cabo deve possuir entre 3 a 5 metros de extensão para interligar um

computador a um periférico. O ideal é que o cabo seja o menor possível, para evitar ruídos e perda de sinal devido ao comprimento.

3.2.7 Servomotor

Optou-se pelo servomotor FUTABA por ele apresentar um *encoder* interno, o que não ocorre para outros tipos de motores. O sistema de *encoder* interno do servomotor garante que a posição angular desejada para o eixo será atingida. Outra vantagem é que não é necessário usar um sensor externo para captar a posição de rotação, o que facilita a validação do modelo cinemático do robô. Utilizando qualquer outro tipo de motor seria necessário construir um circuito *buffer* para proteger a porta paralela.

É importante observar a correta conexão do cabo Centronics com os servomotores. Os sinais de terra da fonte precisam estar conectadas ao sinal de terra da porta paralela para evitar ruídos. É preciso verificar se a tensão de alimentação do motor está com 5V.

3.2.8 Modelagem do Robô

A Figura 3.3 mostra o robô construído. Para obter uma trajetória cartesiana a partir da posição e da orientação do elemento terminal, é necessário, em primeiro lugar, obter o modelo geométrico do robô, como pode ser visto na Figura 3.4.

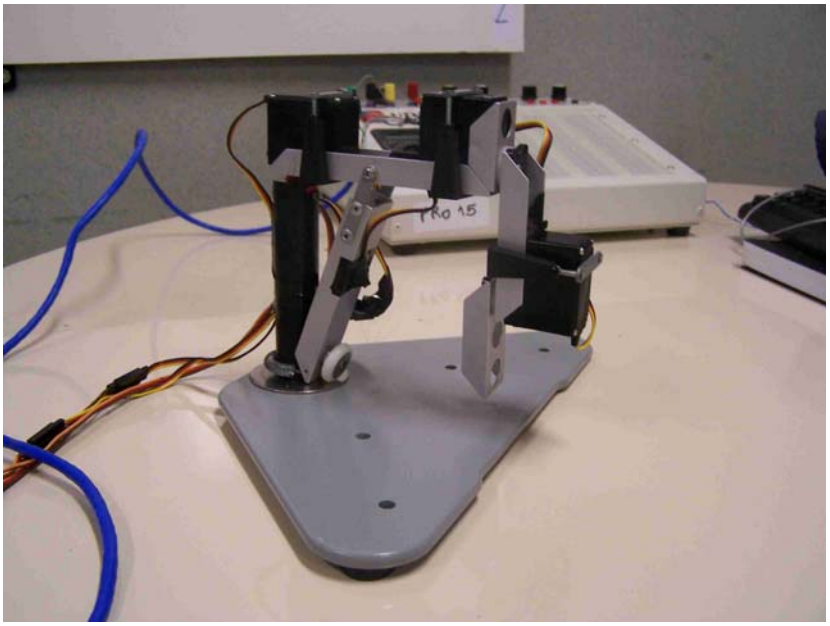


FIGURA 3.3 – ASPECTO DO ROBÔ CONSTRUÍDO

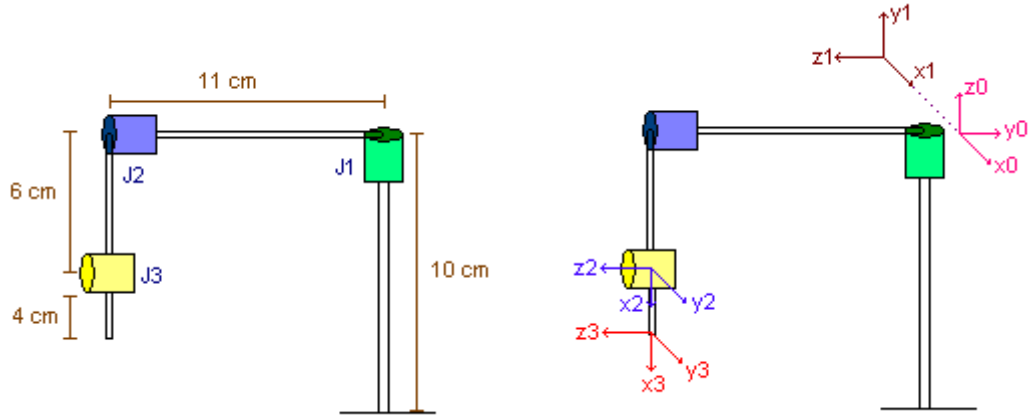


FIGURA 3.4 - MODELO GEOMÉTRICO DO ROBÔ

O modelo geométrico foi obtido através dos parâmetros de Denavit-Hartenberg. A partir do modelo cinemático direto é gerado o modelo geométrico inverso.

Na tabela 3.1 são apresentados os parâmetros de Denavit-Hartenberg, obtidos para o manipulador construído de acordo com as especificações de projeto.

Tabela 3.1 - Parâmetros de denavit-hartenberg

Elos	**θ(graus)**	**d(cm)**	**α(graus)**	**a(cm)**
Elo1	θ1	0	90º	0
Elo2	θ2	11	0º	6
Elo3	θ3	0	0º	4

3.3 Projeto Detalhado

Neste tópico será tratado o projeto detalhado do aplicativo. Ele será dividido em seis partes, quais sejam: (1) metalinguagem do *software*; (2) banco de dados; (3) interpolação; (4) temporizador (8254) utilizado no *software*, para sincronização dos dados enviados; (5) comunicação com o robô; e (6) cálculos das cinemáticas direta e inversa que fornecem os pulsos para os motores relativos ao deslocamento angular solicitado.

3.3.1 *Software*

Para o desenvolvimento de um *software* organizado e de fácil compreensão, dividiu-se o aplicativo em funções. Como o Visual Basic é uma linguagem orientada a eventos, é importante ressaltar que cada botão (ou evento) tem a sua função própria.

O algoritmo do *software* será comentado usando a metalinguagem, sendo que não será explicada cada função, mas sim como funciona cada evento. O código completo está no Apêndice C.

Início do *Software*

Declaração das variáveis;

Iniciação do banco de dados;

Apresentação da tela com informações do *software*;

Se o usuário clicar em escolher trajetória, aparecerá uma nova tela, com algumas opções;

Abrir

Ao clicar em *Abrir*, aparecerá uma tela para que o usuário possa escolher a trajetória;

Ao abrir a janela, o *software* executará a função *Form_Load*, que irá carregar os dados da tabela "trajetória" em uma lista para que usuário possa escolher uma trajetória;

Ao clicar em OK, a janela é fechada, e é executada a função *Form_Activate*. Esta função irá carregar todos os pontos da trajetória escolhida pelo usuário em um novo vetor, carregando esses dados do banco de dados.

Nova

Ao clicar em *Nova*, o vetor com as posições e orientações é zerado, e os campos ficam em branco para que o usuário digite as novas posições, orientações e o nome da nova trajetória.

Salvar

Verifica se a solicitação do usuário é para salvar uma trajetória já existente ou uma nova trajetória.

Trajetória já existente:

Localiza no banco de dados a trajetória corrente;

Apaga esta trajetória;

Inclui a nova trajetória.

Nova trajetória:

Adiciona os dados da nova trajetória nas respectivas tabelas.

Excluir

Apaga a atual trajetória do banco de dados;

Limpa todas as posições dos vetores;

Limpa os textos da tela do usuário.

Next

Mostra os dados da próxima posição do vetor na tela. Caso esta posição não exista, é inserida uma nova posição no vetor e são mostrados os dados em branco.

Back

Retorna para a posição anterior do vetor e esta é mostrada na tela;

Executar Trajetória

Executa a função *executa_trajetória* que tem por finalidade calcular a cinemática inversa e segue para a função interpolar.

É feito um *loop* para que sejam calculados os θ1, θ2 e θ3 para cada posição da trajetória.

Executa a função *interpolar*, que gera 4 novos pontos entre cada posição da trajetória.

Carrega um valor para uma determinada quantidade de pontos. Dentro de um *loop* é possível calcular os ângulos intermediários utilizando-se conceitos da geometria. Através de um triângulo retângulo, encontra-se o ponto intermediário. Estes pontos são inseridos dentro de um novo vetor.

Inicia um laço para enviar todas as posições do vetor para o robô.

Executa a função *aciona_robix*, passando como parâmetros o θ1, θ2 e o θ3.

Prepara o timer para onda com período de 0.1ms (vide ítem 3.3.4 para maiores detalhes).

Inicia um laço.

Monta a onda no formato de onda quadrada com o período em nível lógico um e zero estipulado pelo ângulo a ser enviado para o motor.

Envia a forma de onda para a porta paralela.

A tela é fechada.

Simular Trajetória

Executa a função *executa_trajetória* que tem por finalidade calcular a cinemática inversa e segue para a função interpolar.

É feito um *loop* para que sejam calculados os θ1, θ2 e θ3 para cada posição da trajetória.

Executa a função *interpolar*, que gera 4 novos pontos entre cada posição da trajetória.

Carrega um valor para uma determinada quantidade de pontos.

Dentro de um *loop*, é possível calcular os ângulos (tetas) intermediários, utilizando-se conceitos da geometria. Através de um triângulo retângulo, encontra-se o ponto intermediário.

Estes pontos são inseridos dentro de um novo vetor.

A trajetória é mostrada na tela utilizando um cálculo geométrico.

São desenhadas as linhas para a visualização frontal (plano z e x) e superior do robô (plano x e y).

Caso o usuário deseje escolher ou criar uma nova trajetória, deverá clicar em escolher trajetória, aparecerá novamente a tela como descrito acima e o processo se reinicia. Caso contrário, o usuário solicita encerramento do *software*, as variáveis são descarregadas e o banco de dados finalizado.

Fim do *software*

3.3.2 Banco de Dados

Como uma trajetória pode ter um número indeterminado de pontos e como cada ponto da matriz pode pertencer a mais de uma trajetória, foi adotado para a tabela "MatTraj" o formato n para n. Pode-se ver o relacionamento das tabelas do banco de dados na Figura 3.5.

FIGURA 3.5 - RELACIONAMENTO DAS TABELAS DO BANCO DE DADOS

A tabela “matriz” armazena os dados de cada ponto da trajetória. Pode-se ver a estrutura desta tabela na Tabela 3.2.

Tabela 3.2 - Estrutura da tabela “Matriz”

	Nome do Campo	Tipo de dados
Chave primaria	Code	Inteiro Longo
	Mat_cod	Inteiro Longo
	Px	Duplo
	Py	Duplo
	Pz	Duplo
	Nx	Duplo
	Ny	Duplo
	Nz	Duplo

A tabela “mattraj” faz o relacionamento entre a tabela “Matriz” e a tabela “Trajetória”, por este motivo ela contém as chaves primárias das duas tabelas. Pode-se ver a estrutura desta tabela na Tabela 3.3.

Tabela 3.3 - Estrutura da tabela “Mattraj”

	Nome do Campo	Tipo de dados
	Matriz_code	Inteiro Longo
	Trajetória_code	Inteiro Longo

A tabela “trajetória” armazena os nomes das trajetórias e um código (chave primária para cada uma). Podemos ver a estrutura desta tabela na Tabela 3.4.

Tabela 3.4 - Estrutura da tabela “trajetória”

	Nome do Campo	Tipo de dados
Chave Primaria	Traj_code	Inteiro Longo
	Traj_name	Varchar(50)

3.3.3 Interpolação dos dados

A interpolação dos pontos da trajetória é feita através da geometria.

Para chegar ao cálculo dos pontos intermediários pode-se imaginar um triângulo retângulo, onde se tem o primeiro e o último ponto (Figura 3.6).

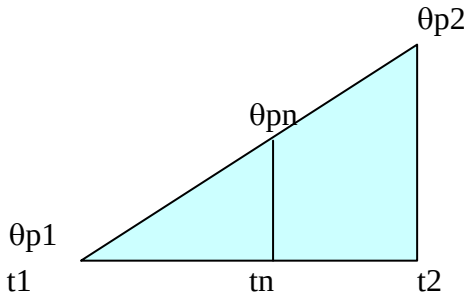


FIGURA 3.6 - TRIÂNGULO RETÂNGULO UTILIZADO PARA O CÁLCULO DA INTERPOLAÇÃO.

Com base no triângulo retângulo desenhado na Figura 3.5 podemos obter a equação abaixo para cada ângulo que se deseja interpolar:

$$\theta pn = \left[\frac{(tn - t1) * (\theta p2 - \theta p1)}{t2 - t1} \right] + \theta p1 \qquad (3.1)$$

Com a equação 3.1, podemos retirar o algoritmo abaixo para fazer a interpolação.

```
t1 = 0
tn=1
Freqüência = 0,5      //determina a freqüência utilizada, neste caso cria um
                      //ponto entre os pontos
For(pontos=1 to 1000)
        t1=t1+1
        t2 = t1+1
        tn = t1+freqüência
        tetap1 = vteta[pontos]
        tetap2 = vteta[pontos+1]
        Do{
```

```
                Tetapn = ((tn-t1)*(tetap2-tetap1)/(t2-t1))+tetap1
                tn = tn+freqüência
        }While (tn<=t2)
    Next pontos
    End
```

3.3.4 Temporizador

Para acionar os servomotores é necessário fornecer, via porta paralela, pulsos com largura variável. O motor tem um curso total ligeiramente maior que 180^o. Com *duty cycle* de 8% o motor gira no sentido anti-horário, buscando o ângulo de 0^o e com *duty cycle* de 16% o motor gira no sentido horário buscando a posição de 180^o.

Para configurar o contador do 8254, para fornecer a forma de onda necessária, deve-se primeiramente enviar uma palavra de controle para o endereço 43h. A Figura 3.7 adaptada de [8] mostra o formato da palavra de controle.

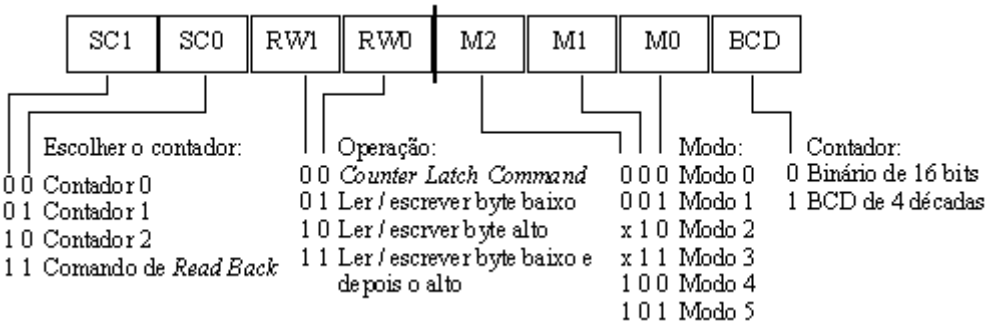


FIGURA 3.7 - PALAVRA DE CONTROLE DO 8254

A palavra de controle para acionar o contador, então é:

96h = 1001 0110b

10 – contador 2

01 – ler/escrever byte baixo

011 – modo 3

0 – binário de 16 bits

e a instrução usada em VB para enviar o comando ao 8254 fica,

```
Out Val("&H43"), Val("&H96")
```

O próximo passo é definir o divisor de freqüência e enviá-lo para o endereço 42h. Neste caso será utilizada uma freqüência de 10kHz. A instrução em VB para este comando ser enviado segue abaixo (1193180/10000=119=77h):

Out Val("&H42"), Val("&H77")

O passo seguinte é definir o comando de *read-back* através do envio de uma nova palavra de controle (endereço 43h). O comando de *read-back* serve para ler o valor do contador. O formato do comando é mostrado na Figura 3.8 adaptada de [8].

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	1	COUNT	STATUS	CNT 2	CNT 1	CNT 0	0

D_5: 0 = *Latch count of select counter (s)*
D_4: 0 = *Latch status of select counter (s)*
D_3: 0 = Seleciona contador 2
D_2: 0 = Seleciona contador 1
D_1: 0 = Seleciona contador 0
D_0: 0 = Reservado para futura expansão, pode ser 0

FIGURA 3.8 - FORMATO DO COMANDO DE READ BACK

A palavra de comando utilizada para acionar o comando de *read-back* é,

E8h = 1110 1000b

11 – Comando de *read back*

1 – Trava a contagem do contador selecionado

0 – Trava o status do contador selecionado

1 – Seleciona o contador 2

0 – Seleciona o contador 1

0 – Seleciona o contador 0

0 – Reservado para futura expansão, pode ser 0

Assim a instrução fica,

Out Val("&H43"), Val("&HE8")

Por último, deve-se desabilitar o som do computador, pelo endereço 61h. Caso o bit 0 da porta 61h não esteja em nível lógico baixo não será possível acionar o contador 2. Só se pode usar o contador 2 para sincronismo de *hardware* externo.

Out Val("&H61"), Val("&H01")

Depois de configurado o 8254, é necessário garantir que a contagem para gerar a largura correta dos pulsos (que serão enviados para os servomotores) comece exatamente no início de algum período da onda quadrada de 10kHz. Isto é feito através da utilização do pedaço de programa a seguir.

‘Monta a forma de onda no formato desejado, neste caso quadrada.

```
inpo = Inp(Val("&H42"))
While inpo > 128
   Out Val("&H43"), Val("&HE8")
   inpo = Inp(Val("&H42")) 'PEGA VALOR DA CONTAGEM
Wend
While inpo <= 128
   Out Val("&H43"), Val("&HE8")
   inpo = Inp(Val("&H42")) 'PEGA VALOR  DA CONTAGEM
Wend
```

3.3.5 Conexão dos servomotores

Para conectar o computador ao robô, é necessário um cabo paralelo de forma que os terminais sejam conectados como descrito abaixo.

O cabo de cada motor constitui-se de três fios: um comum, o Vcc (5V) e o sinal PWM, como mostra a Figura 3.9.

É importante que a alimentação elétrica dos servomotores seja realizada através de uma fonte CC externa de 5V, pois a porta paralela não fornece corrente suficiente para acionar os atuadores. O comum do cabo paralelo e o da fonte de alimentação devem estar conectados para evitar que ruídos afetem o funcionamento correto do sistema.

Porta paralela

Pino 2 – conecta-se ao motor 1 (função do $\theta 1$)

Pino 3 – conecta-se ao motor 2 (função do $\theta 2$)

Pino 4 – conecta-se ao motor 3 (função do $\theta 3$)

Pino 25 – comum, conecta-se com o comum da fonte.

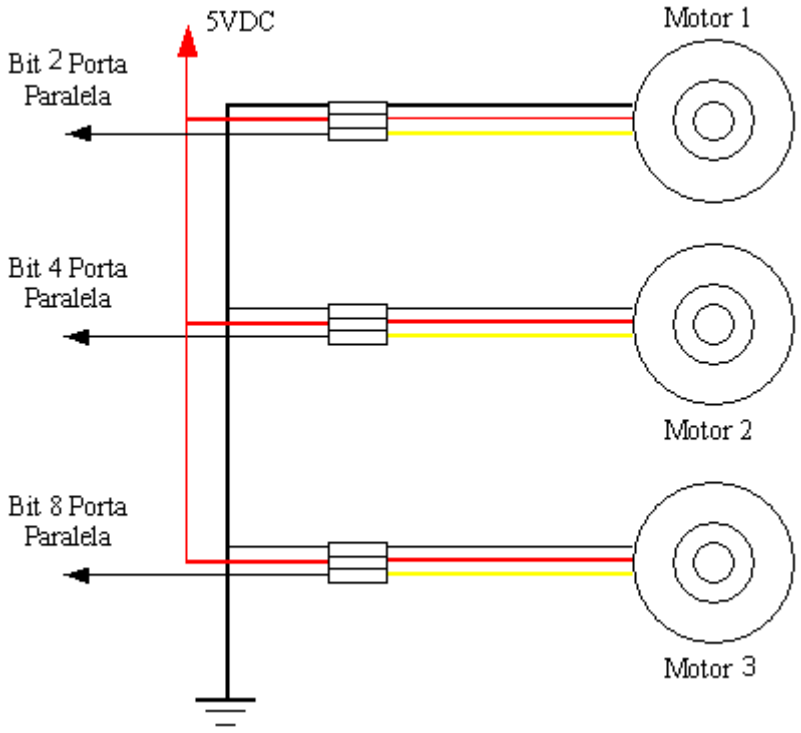


FIGURA 3.9 - CONEXÃO PORTA PARALELA COM OS MOTORES

3.3.6 Cálculo da cinemática inversa e cinemática direta

Para determinar o ângulo de rotação do eixo de cada motor, utiliza-se a cinemática inversa, que pode ser obtida da cinemática direta. Ao entrar com as posições x,y e z, é calculada a cinemática inversa.

Segue abaixo a explicação passo a passo.

1. Primeiramente, desenham-se os sistemas de coordenadas sobre o modelo mecânico do robô (Figura 3.4).
2. Determinam-se os parâmetros de Denavit-Hartenberg (Tabela 3.1).
3. Baseado na Tabela 3.1 e utilizando a matriz de transformação homogênea "padrão" que relaciona os sistemas de coordenadas de dois elos adjacentes de um robô de elos rígidos seriais (Equação 3.2a), determinam-se as matrizes de transformação entre os elos 0 e 1, 1 e 2, 2 e 3; Equações 3.3, 3.4 e 3.5, respectivamente. A Equação 3.2b mostra a matriz de transformação inversa.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)*sen(\theta_i) & sen(\alpha_i)*sen(\theta_i) & a_i*\cos(\theta_i) \\ sen(\theta_i) & \cos(\alpha_i)*\cos(\theta_i) & -sen(\alpha_i)*\cos(\theta_i) & a_i*sen(\theta_i) \\ 0 & sen(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2a) $$

$$ {}^{i}_{i-1}T = {}^{i-1}_{i}T^{-1} = \begin{bmatrix} \cos(\theta_i) & sen(\theta_i) & 0 & -a_i \\ -sen(\theta_i)*\cos(\alpha_i) & \cos(\theta_i)*\cos(\alpha_i) & sen(\alpha_i) & -d_i*sen(\alpha_i) \\ sen(\theta_i)*sen(\alpha_i) & -\cos(\theta_i)*sen(\alpha_i) & \cos(\alpha_i) & -d_i*\cos(\alpha_i) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2b) $$

$$ {}^{0}_{1}T = \begin{bmatrix} \cos(\theta 1) & 0 & sen(\theta 1) & 0 \\ sen(\theta 1) & 0 & -\cos(\theta 1) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3) $$

$$ {}^{1}_{2}T = \begin{bmatrix} \cos(\theta 2) & -sen(\theta 2) & 0 & 6*\cos(\theta 2) \\ sen(\theta 2) & \cos(\theta 2) & 0 & 6*sen(\theta 2) \\ 0 & 0 & 1 & 11 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4) $$

$$ {}^{2}_{3}T = \begin{bmatrix} \cos(\theta 3) & -sen(\theta 3) & 0 & 4*\cos(\theta 3) \\ sen(\theta 3) & \cos(\theta 3) & 0 & 4*sen(\theta 3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5) $$

A Equação 3.6 apresenta o equacionamento para a obtenção do modelo cinemático direto para o manipulador utilizado.

$$ {}^{0}_{3}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T \quad (3.6) $$

A Equação 3.7 mostra cada elemento da matriz de transformação homogênea que representa a cinemática direta do manipulador usado.

$$
\begin{aligned}
{}^{0}_{3}T(1,1) &= \left[\cos(\theta 1) * \cos(\theta 2) * \cos(\theta 3)\right] - \left[\cos(\theta 1) * \operatorname{sen}(\theta 2) * \operatorname{sen}(\theta 3)\right] \\
{}^{0}_{3}T(1,2) &= \left[-\cos(\theta 1) * \cos(\theta 2) * \operatorname{sen}(\theta 3)\right] - \left[\cos(\theta 1) * \operatorname{sen}(\theta 2) * \cos(\theta 3)\right] \\
{}^{0}_{3}T(1,3) &= \operatorname{sen}(\theta 1) \\
{}^{0}_{3}T(1,4) &= \left[4 * \cos(\theta 1) * \cos(\theta 2) * \cos(\theta 3)\right] - \left[4 * \cos(\theta 1) * \sin(\theta 2) * \operatorname{sen}(\theta 3)\right] \\
&\quad + \left[6 * \cos(\theta 1) * \cos(\theta 2)\right] + \left[11 * \operatorname{sen}(\theta 1)\right] \\
{}^{0}_{3}T(2,1) &= \left[\operatorname{sen}(\theta 1) * \cos(\theta 2) * \cos(\theta 3)\right] - \left[\operatorname{sen}(\theta 1) * \operatorname{sen}(\theta 2) * \operatorname{sen}(\theta 3)\right] \\
{}^{0}_{3}T(2,2) &= \left[-\operatorname{sen}(\theta 1) * \cos(\theta 2) * \operatorname{sen}(\theta 3)\right] - \left[\operatorname{sen}(\theta 1) * \operatorname{sen}(\theta 2) * \cos(\theta 3)\right] \\
{}^{0}_{3}T(2,3) &= -\cos(\theta 1) \\
{}^{0}_{3}T(2,4) &= \left[4 * \operatorname{sen}(\theta 1) * \cos(\theta 2) * \cos(\theta 3)\right] - \left[4 * \operatorname{sen}(\theta 1) * \operatorname{sen}(\theta 2) * \operatorname{sen}(\theta 3)\right] \\
&\quad + \left[6 * \operatorname{sen}(\theta 1) * \cos(\theta 2)\right] - \left[11 * \cos(\theta 1)\right] \\
{}^{0}_{3}T(3,1) &= \left[\operatorname{sen}(\theta 2) * \cos(\theta 3)\right] + \left[\cos(\theta 2) * \operatorname{sen}(\theta 3)\right] \\
{}^{0}_{3}T(3,2) &= \left[-\operatorname{sen}(\theta 2) * \operatorname{sen}(\theta 3)\right] + \left[\cos(\theta 2) * \cos(\theta 3)\right] \\
{}^{0}_{3}T(3,3) &= 0 \\
{}^{0}_{3}T(3,4) &= \left[4 * \operatorname{sen}(\theta 2) * \cos(\theta 3)\right] + \left[4 * \cos(\theta 2) * \operatorname{sen}(\theta 3)\right] + \left[6 * \operatorname{sen}(\theta 2)\right] \\
{}^{0}_{3}T(4,1) &= 0 \\
{}^{0}_{3}T(4,2) &= 0 \\
{}^{0}_{3}T(4,3) &= 0 \\
{}^{0}_{3}T(4,4) &= 1
\end{aligned}
\tag{3.7}
$$

4. Para se determinar a cinemática inversa, adota-se uma matriz de transformação qualquer (Equação 3.8), que descreve uma posição do manipulador.

$$
{}^{0}_{3}T = \begin{bmatrix} nx & ox & ax & px \\ ny & oy & ay & py \\ nz & oz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}
$$

Dada ${}^{0}_{3}T$, aplica-se o método das matrizes de transformação inversa [2] para se calcular a cinemática inversa, isto é, os ângulos θ das juntas que levam a extremidade do manipulador para a posição genérica determinada por ${}^{0}_{3}T$.

$$
{}^{0}_{1}T^{-1}\,{}^{0}_{3}T = {}^{1}_{3}T \tag{3.9}
$$

Nas Equações 3.10 (lado esquerdo de 3.9) e 3.11(lado direito de 3.9) pode-se ver os elementos das matrizes a serem igualadas na Equação 3.9.

$$
\begin{aligned}
&{}^{1}_{3}T[1,1] = -\text{sen}(\theta 2) * \text{sen}(\theta 3) + \cos(\theta 2) * \cos(\theta 3)\\
&{}^{1}_{3}T[1,2] = -\cos(\theta 2) * \text{sen}(\theta 3) - \text{sen}(\theta 2) * \cos(\theta 3)\\
&{}^{1}_{3}T[1,3] = 0\\
&{}^{1}_{3}T[1,4] = 4 * \cos(\theta 2) * \cos(\theta 3) - 4 * \text{sen}(\theta 2) * \text{sen}(\theta 3) + 6 * \cos(\theta 2)\\
&{}^{1}_{3}T[2,1] = \text{sen}(\theta 2) * \cos(\theta 3) + \cos(\theta 2) * \text{sen}(\theta 3)\\
&{}^{1}_{3}T[2,2] = -\text{sen}(\theta 2) * \text{sen}(\theta 3) + \cos(\theta 2) * \cos(\theta 3)\\
&{}^{1}_{3}T[2,3] = 0\\
&{}^{1}_{3}T[2,4] = 4 * \text{sen}(\theta 2) * \cos(\theta 3) + 4 * \cos(\theta 2) * \text{sen}(\theta 3) + 6 * \text{sen}(\theta 2)\\
&{}^{1}_{3}T[3,1] = 0\\
&{}^{1}_{3}T[3,2] = 0\\
&{}^{1}_{3}T[3,3] = 1\\
&{}^{1}_{3}T[3,4] = 11\\
&{}^{1}_{3}T[4,1] = 0\\
&{}^{1}_{3}T[4,2] = 0\\
&{}^{1}_{3}T[4,3] = 0\\
&{}^{1}_{3}T[4,4] = 1
\end{aligned}
\qquad (3.10)
$$

$$
\begin{aligned}
&{}^{1}_{3}T[1,1] = \cos(\theta 1) * nx + \text{sen}(\theta 1) * ny \\
&{}^{1}_{3}T[1,2] = \cos(\theta 1) * ox + \text{sen}(\theta 1) * oy \\
&{}^{1}_{3}T[1,3] = \cos(\theta 1) * ax + \text{sen}(\theta 1) * ay \\
&{}^{1}_{3}T[1,4] = \cos(\theta 1) * px + \text{sen}(\theta 1) * py \\
&{}^{1}_{3}T[2,1] = nz \\
&{}^{1}_{3}T[2,2] = oz \\
&{}^{1}_{3}T[2,3] = az \\
&{}^{1}_{3}T[2,4] = pz \\
&{}^{1}_{3}T[3,1] = \text{sen}(\theta 1) * nx - \cos(\theta 1) * ny \\
&{}^{1}_{3}T[3,2] = \text{sen}(\theta 1) * ox - \cos(\theta 1) * oy \\
&{}^{1}_{3}T[3,3] = \text{sen}(\theta 1) * ax - \cos(\theta 1) * ay \\
&{}^{1}_{3}T[3,4] = \text{sen}(\theta 1) * px - \cos(\theta 1) * py \\
&{}^{1}_{3}T[4,1] = 0 \\
&{}^{1}_{3}T[4,2] = 0 \\
&{}^{1}_{3}T[4,3] = 0 \\
&{}^{1}_{3}T[4,4] = 1
\end{aligned}
\tag{3.11}
$$

Igualam-se as equações para se obter os θ. O resultado é mostrado na Equação 3.12.

$$\theta 1 = arctg\left(\frac{ny}{nx}\right) \tag{3.12}$$

Depois de obtido o θ1, verifica-se se é possível extrair θ2 e θ3. Como fica evidenciado a dificuldade em se obter estes dois ângulos, passa-se para o passo posterior. As Equações 3.14 (lado esquerdo de 3.13) e 3.15 (lado direito de 3.13) mostram as matrizes a serem igualadas na Equação 3.13.

$${}^{1}_{2}T^{-1}\,{}^{0}_{1}T^{-1}\,{}^{0}_{3}T = {}^{2}_{3}T \tag{3.13}$$

$$
\begin{aligned}
&{}^{2}_{3}T[1,1] = \cos(\theta 3)\\
&{}^{2}_{3}T[1,2] = \text{-sen}(\theta 3)\\
&{}^{2}_{3}T[1,3] = 0\\
&{}^{2}_{3}T[1,4] = 4 * \cos(\theta 3)\\
&{}^{2}_{3}T[2,1] = \text{sen}(\theta 3)\\
&{}^{2}_{3}T[2,2] = \cos(\theta 3)\\
&{}^{2}_{3}T[2,3] = 0\\
&{}^{2}_{3}T[2,4] = 4 * \text{sen}(\theta 3)\\
&{}^{2}_{3}T[3,1] = 0\\
&{}^{2}_{3}T[3,2] = 0\\
&{}^{2}_{3}T[3,3] = 1\\
&{}^{2}_{3}T[3,4] = 0\\
&{}^{2}_{3}T[4,1] = 0\\
&{}^{2}_{3}T[4,2] = 0\\
&{}^{2}_{3}T[4,3] = 0\\
&{}^{2}_{3}T[4,4] = 1
\end{aligned}
\tag{3.14}
$$

$$
\begin{aligned}
&{}^{2}_{3}T[1,1] = \cos(\theta 1) * \cos(\theta 2) * \text{nx} + \text{sen}(\theta 1) * \cos(\theta 2) * \text{ny} + \text{sen}(\theta 2) * \text{nz} \\
&{}^{2}_{3}T[1,2] = \cos(\theta 1) * \cos(\theta 2) * \text{ox} + \text{sen}(\theta 1) * \cos(\theta 2) * \text{oy} + \text{sen}(\theta 2) * \text{oz} \\
&{}^{2}_{3}T[1,3] = \cos(\theta 1) * \cos(\theta 2) * \text{ax} + \text{sen}(\theta 1) * \cos(\theta 2) * \text{ay} + \text{sen}(\theta 2) * \text{az} \\
&{}^{2}_{3}T[1,4] = \cos(\theta 1) * \cos(\theta 2) * \text{px} + \text{sen}(\theta 1) * \cos(\theta 2) * \text{py} + \text{sen}(\theta 2) * \text{pz} - 6 \\
&{}^{2}_{3}T[2,1] = -\cos(\theta 1) * \text{sen}(\theta 2) * \text{nx} - \text{sen}(\theta 1) * \text{sen}(\theta 2) * \text{ny} + \cos(\theta 2) * \text{nz} \\
&{}^{2}_{3}T[2,2] = -\cos(\theta 1) * \text{sen}(\theta 2) * \text{ox} - \text{sen}(\theta 1) * \text{sen}(\theta 2) * \text{oy} + \cos(\theta 2) * \text{oz} \\
&{}^{2}_{3}T[2,3] = -\cos(\theta 1) * \text{sen}(\theta 2) * \text{ax} - \text{sen}(\theta 1) * \text{sen}(\theta 2) * \text{ay} + \cos(\theta 2) * \text{az} \\
&{}^{2}_{3}T[2,4] = -\cos(\theta 1) * \text{sen}(\theta 2) * \text{px} - \text{sen}(\theta 1) * \text{sen}(\theta 2) * \text{py} + \cos(\theta 2) * \text{pz} \\
&{}^{2}_{3}T[3,1] = \text{sen}(\theta 1) * \text{nx} - \cos(\theta 1) * \text{ny} \\
&{}^{2}_{3}T[3,2] = \text{sen}(\theta 1) * \text{ox} - \cos(\theta 1) * \text{oy} \\
&{}^{2}_{3}T[3,3] = \text{sen}(\theta 1) * \text{ax} - \cos(\theta 1) * \text{ay} \\
&{}^{2}_{3}T[3,4] = \text{sen}(\theta 1) * \text{px} - \cos(\theta 1) * \text{py} - 11 \\
&{}^{2}_{3}T[4,1] = 0 \\
&{}^{2}_{3}T[4,2] = 0 \\
&{}^{2}_{3}T[4,3] = 0 \\
&{}^{2}_{3}T[4,4] = 1
\end{aligned}
\tag{3.15}
$$

Igualando-se novamente as equações, encontra-se o θ2 (Equação 3.16) e o θ3 (Equação 3.17).

$$\theta 2 = \text{arcsen}\left(\frac{-pz - 4 * nz}{6}\right) \tag{3.16}$$

$$\theta 3 = \arccos\left(\frac{\cos(\theta 2) * \left[\cos(\theta 1) *´ px + sen(\theta 1) * py\right] + sen(\theta 2) * pz - 6}{4}\right) \tag{3.17}$$

Com exceção dos cálculos dos θ, todos os outros foram feitos no *software* MatLab. Os procedimentos são mostrados no apêndice A.

3.4 Teste do sistema

Testa-se o robô, fazendo as ligações com cada servomotor conforme ítem 3.3.5. Em relação ao teste dos cálculos, utiliza-se o método de simulação, determinam-se as coordenadas e verifica-se após efetuar a cinemática direta e inversa se os valores são os mesmos. Os cálculos das matrizes foram desenvolvidos no MatLab, também foram

feitos cálculos, manualmente, para determinadas posições do robô, assim foi possível ter certeza que a tabela de Denavit-Hartenberg e as fórmulas encontradas para os ângulos estavam corretas.

Quanto ao *software*, foram realizados testes em massa, ou seja, determinou-se uma tabela de posições para o robô, e foi verificado se ocorria o desejado.

Desenhou-se uma tabela contendo as principais funcionalidades do *software*, e foram feitos testes em cada ítem, verificando assim, a integridade das funções.

Para a funcionalidade e facilidade de uso do sistema, foi colocado um usuário que nunca viu o *software* para testá-lo. Detectadas as dificuldades, foi possível observar os erros e corrigí-los.

Capítulo 4

Resultados

4.1 Sistema Projetado

Este projeto é constituído por um *software* e pelo manipulador (Figura 3.3).

O *software* foi elaborado de maneira simples, para que seja um sistema de fácil utilização.

A Figura 4.1 apresenta a tela inicial do *software*. Ela mostra as informações básicas do projeto, o nome do orientador e das alunas responsáveis pelo seu desenvolvimento.

FIGURA 4.1 - TELA INICIAL DO *SOFTWARE*

Pode-se ver a trajetória executada pelo robô na tela principal do *software* (Figura 4.2).

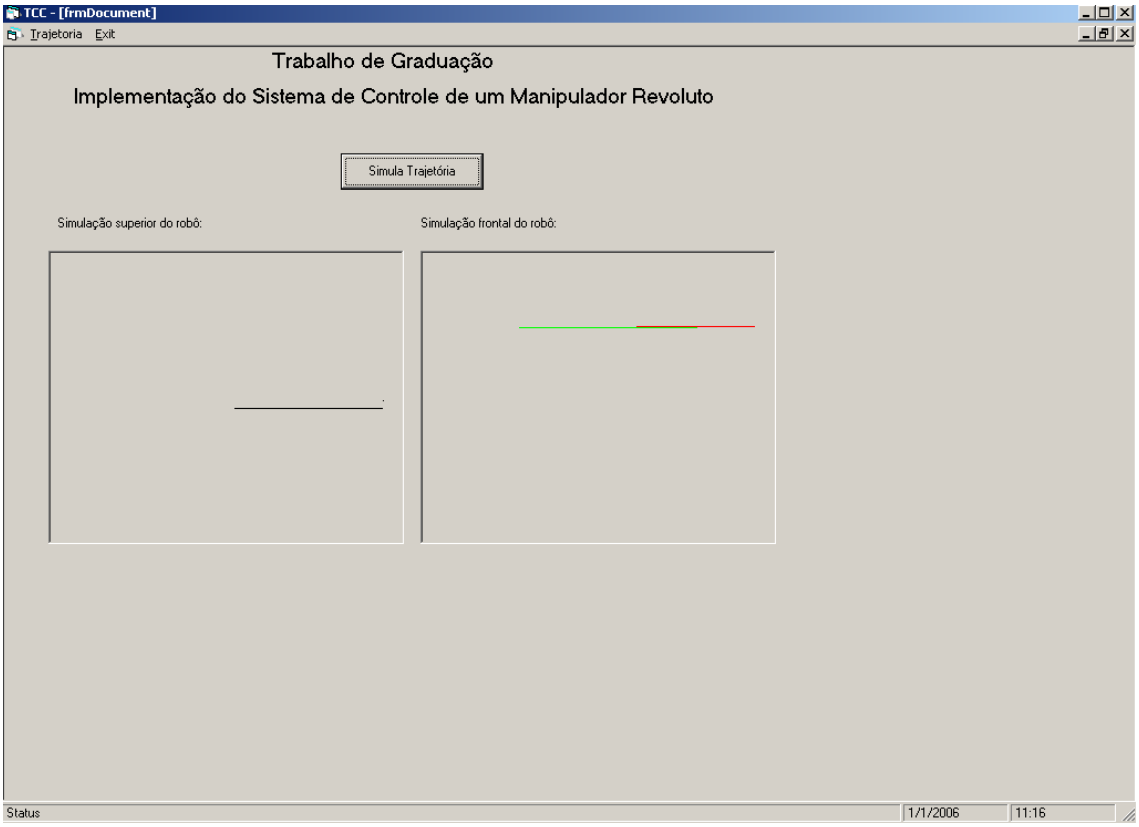


FIGURA 4.2 -TELA PRINCIPAL DO *SOFTWARE*

A Figura 4.3, mostra a tela onde se defini a trajetória a ser executada e onde se pode fazer todo o gerenciamento da trajetória. A Figura 4.4 ilustra a tela onde se escolhe a trajetória, que esta armazenada no banco de dados, a ser executada.

FIGURA 4.3 - TELA DE DEFINIÇÃO DA TRAJETÓRIA

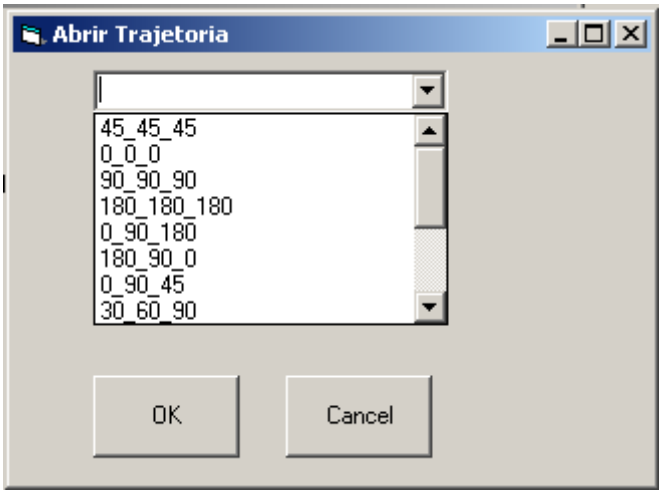


FIGURA 4.4 - TELA UTILIZADA PARA ESCOLHER UMA TRAJETÓRIA.

4.2 Operação do Sistema

Para operar o robô, é necessário primeiramente conectá-lo ao computador, através do cabo paralelo, e instalar o *software* de controle. Após isso, é essencial que se ligue o robô em uma alimentação de 5V e execute o *software* instalado.

O *Software* é formado basicamente por quatro telas. O manual do usuário encontra-se no apêndice D.

A primeira tela, que pode ser vista na Figura 4.1 tem como função mostrar algumas informações básicas do *software*, como nome do sistema, desenvolvedores, versão e assim por diante. Esta tela aparece rapidamente, assim que o sistema é executado e neste momento o usuário não executa nenhuma ação.

A tela principal (Figura 4.2) apresenta o gráfico com a trajetória efetuada pelo robô e o menu com as opções de definir trajetória e sair do *software*. O usuário deve clicar em definir trajetória para prosseguir.

A Figura 4.3 mostra a tela Trajetória, onde o usuário ira manipular as trajetórias do sistema, podendo inserir, apagar, alterar e salvar. Também será por esta tela que o usuário irá definir se deseja que o robô execute a trajetória, ou apenas mostrar a simulação na área do gráfico.

Para abrir a trajetória (Figura 4.4) o usuário deverá selecionar uma das opções da “list” e clicar em OK.

4.3 Validação do Sistema

O sistema foi validado de forma que foram testadas todas as funções do *software* isoladamente e todas as condições foram colocadas a prova.

Para testar a comunicação do computador com o robô enviou-se para cada servomotor um sinal correspondente a um ângulo de rotação do eixo. Este procedimento foi realizado sem utilizar os resultados obtidos no cálculo da cinemática inversa. Verificou-se que o robô se movimentava a contento, descrevendo o ângulo requisitado.

Para validar os cálculos dos ângulos gerados pelo *software*, fez-se os cálculos no MatLab, como mostra o apêndice B. A partir dos ângulos, calculou-se a cinemática direta com o MatLab, obtendo-se a posição e orientação no espaço cartesiano. Os valores do espaço cartesiano obtidos no MatLab foram inseridos no programa para se verificar, se os cálculos da cinemática inversa estavam corretos. O programa gerou os ângulos corretamente.

Após os testes dos componentes em separado, integralizou-se o sistema para novos testes. Repetidas e variadas situações foram executadas, mostrando consistência no funcionamento do sistema (*hardware* e *software*), sem apresentar erros de lógica e falhas mecânicas.

Capítulo 5

Conclusões

O sistema proposto foi totalmente contemplado. A despeito da simplicidade da proposta, o desenvolvimento do *software* é complexo e utiliza muitos recursos avançados de programação, desde a comunicação via *hardware*, utilizando recursos de comunicação com a porta paralela, com o temporizador interno do PC; até a programação utilizando banco de dados.

Trabalhar com o Visual Basic e comunicação paralela sob plataforma Windows 2000, XP e ME necessita a utilização de uma dll específica. No caso usou-se a inpout32 disponível no Anexo C.

Devido à complexidade dos cálculos das cinemáticas direta e inversa, que são obtidas através de equações matriciais, é interessante utilizar um aplicativo como o MatLab para otimizar o trabalho e diminuir a probabilidade de erros nos cáculos.

Montou-se um braço mecânico com três graus de liberdade, onde cada eixo se movimenta com um ângulo máximo de 180º. Fez-se um *software* onde este envia as trajetórias para o robô e também mostra graficamente estas trajetórias. O software tem um sistema de banco de dados, onde é possível armazenar os dados referentes a cada trajetória.

Pode-se apresentar diversas sugestões para implementações futuras para melhorar o sistema. Uma possível melhoria é acrescentar, no *software*, uma rotina para considerar as velocidades e acelerações em determinados pontos da trajetória percorrida pelo órgão terminal do manipulador. Outra melhoria seria permitir que o usuário possa controlar um manipulador com um número maior de graus de liberdade, podendo até considerar vários tipos de manipuladores, usando o mesmo *software*.

Referências Bibliográficas

[1] ROSÁRIO, J.M. **Princípios de Mecatrônica**. São Paulo: Pearson, 2005. 356p.

[2] ADADE FILHO, A. **Fundamentos de Robótica: cinemática, dinâmica e controle de manipuladores robóticos**. São José dos Campos-SP, CTA-ITA-IEMP, 2001. 352p.

[3] LOHSE, G. L. et al. A classification of visual representations. **Communications of the ACM**, New York, v.37, n.12, p.36–49, Dec. 1994.

[4] LICKLIDER, T. R. Ten years of rows and columns (spread sheet programs). **Byte Magazine**, New York, v.14, n.13, p.324–331, Dec. 1989

[5] BURNETT M.; HOSSLI, R.; PULLIAM, T.; VANVOORST, B.; YANG X. Toward Visual Programming Languages for Steering Scientific Computation. **IEEE Computational Science & Engineering**, Los Alamitos, v.1, n.4, p. 44-62, Winter 1994.

[6] ADVANCED VISUAL SYSTEMS. **AVS5**: The Original Visual Programming Data Visualization Computing Environment. Disponível em: http://www.avs.com/products/AVS5/avs5.htm. Acesso em: 02 jan. 2006.

[7] AIKEN, A.; CHEN, J.; STONEBRAKER, M.; WOODRUFF, A. Tioga-2: a direct manipulation database visualization environment. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 12., 1996, New Orleans. **Proceedings**. . . [S.l.: s.n.], 1996. p.208-217.

[8] CELIS, C.G.; **El Temporizador 8253 U 8254**. Disponível em: http://atc.ugr.es/docencia/udigital/1203.html Acesso em: 01 dez 2005

[9] MESSIAS, A.R.; **Porta Paralela**. Disponível em: http://www.rogercom.com/pparalela/introducao.htm Acesso em: 01 dez 2005

[10] WALLES, J.; **Wikipédia**. Disponível em: http://pt.wikipedia.org/wiki/Modula%C3%A7%C3%A3o_por_largura_de_pulso. Acesso em 02 jan. 2006.

[11] VINICIUS; **Modulação por Largura de Pulso**. Disponível em: http://vinicius.brasil.vilabol.uol.com.br/eletronica/controlDCMotor1/controldcmotor1.htm Acesso em: 12 jan 2006

[12] ANGEL; **Modulação por Largura de Pulso**. Disponível em: http://www.grupozug.com.br/ENGEL/Harmonicas/7.htmAcesso em: 12 jan 2006

[13] ZAMORA, C. **Servomotores**. Disponível em: http://elektra.udea.edu.co/~jfelipe/+articulos/SERVOMOTORES.docAcesso em: 03 dez 2005

[14] MIGUEL, F.B. ***Access* 97**. São Paulo: Erica, 1997. 175p.

Apêndice A - Cálculos da cinemática inversa

Cálculos da cinemática inversa obtida pela cinemática direta, efetuada no *software* MatLab

```
>> syms o1 o2 o3
syms nx ny nz ox oy oz ax ay az px py pz

T01 = [cos(o1) 0 sin(o1) 0; sin(o1) 0 -cos(o1) 0; 0 1 0 0; 0 0 0
1]
T12 = [cos(o2) -sin(o2) 0 6*cos(o2); sin(o2) cos(o2) 0
6*sin(o2); 0 0 1 11; 0 0 0 1]
T23 = [cos(o3) -sin(o3) 0 4*cos(o3); sin(o3) cos(o3) 0
4*sin(o3); 0 0 1 0; 0 0 0 1]

TI01 = [cos(o1) sin(o1) 0 0; 0 0 1 0; sin(o1) -cos(o1) 0 0; 0 0
0 1]
TI12 = [cos(o2) sin(o2) 0 -6; -sin(o2) cos(o2) 0 0; 0 0 1 -11; 0
0 0 1]

TE03 = [nx ox ax px; ny oy ay py; nz oz az pz; 0 0 0 1]

T03 = T01 * T12 * T23

T13 = TI01 * T03
TE13 = TI01 * TE03

T23 = TI12 * TI01 * T03
TE23 = TI12 * TI01 * TE03

ST03=simplify(T03)
ST13=simplify(T13)
STE13=simplify(TE13)
ST23=simplify(T23)
STE23=simplify(TE23)


T01 =

[  cos(o1),        0,  sin(o1),        0]
[  sin(o1),        0, -cos(o1),        0]
[        0,        1,        0,        0]
[        0,        0,        0,        1]


T12 =

[   cos(o2),  -sin(o2),         0, 6*cos(o2)]
[   sin(o2),   cos(o2),         0, 6*sin(o2)]
[         0,         0,         1,        11]
[         0,         0,         0,         1]
```

```
T23 =

[   cos(o3),  -sin(o3),         0, 4*cos(o3)]
[   sin(o3),   cos(o3),         0, 4*sin(o3)]
[         0,         0,         1,         0]
[         0,         0,         0,         1]


TI01 =

[  cos(o1),  sin(o1),        0,        0]
[        0,        0,        1,        0]
[  sin(o1), -cos(o1),        0,        0]
[        0,        0,        0,        1]


TI12 =

[  cos(o2),  sin(o2),        0,       -6]
[ -sin(o2),  cos(o2),        0,        0]
[        0,        0,        1,      -11]
[        0,        0,        0,        1]


TE03 =

[ nx, ox, ax, px]
[ ny, oy, ay, py]
[ nz, oz, az, pz]
[  0,  0,  0,  1]


T03 =

[                                  cos(o1)*cos(o2)*cos(o3)-
cos(o1)*sin(o2)*sin(o3),                                  -
cos(o1)*cos(o2)*sin(o3)-cos(o1)*sin(o2)*cos(o3),
sin(o1), 4*cos(o1)*cos(o2)*cos(o3)-
4*cos(o1)*sin(o2)*sin(o3)+6*cos(o1)*cos(o2)+11*sin(o1)]
[                                  sin(o1)*cos(o2)*cos(o3)-
sin(o1)*sin(o2)*sin(o3),                                  -
sin(o1)*cos(o2)*sin(o3)-sin(o1)*sin(o2)*cos(o3),
-cos(o1), 4*sin(o1)*cos(o2)*cos(o3)-
4*sin(o1)*sin(o2)*sin(o3)+6*sin(o1)*cos(o2)-11*cos(o1)]
[
sin(o2)*cos(o3)+cos(o2)*sin(o3),
-sin(o2)*sin(o3)+cos(o2)*cos(o3),
0,
4*sin(o2)*cos(o3)+4*cos(o2)*sin(o3)+6*sin(o2)]
[
0,
0,
```

```
0,
1]


T13 =

[
cos(o1)*(cos(o1)*cos(o2)*cos(o3)-
cos(o1)*sin(o2)*sin(o3))+sin(o1)*(sin(o1)*cos(o2)*cos(o3)-
sin(o1)*sin(o2)*sin(o3)),
cos(o1)*(-cos(o1)*cos(o2)*sin(o3)-
cos(o1)*sin(o2)*cos(o3))+sin(o1)*(-sin(o1)*cos(o2)*sin(o3)-
sin(o1)*sin(o2)*cos(o3)),
0, cos(o1)*(4*cos(o1)*cos(o2)*cos(o3)-
4*cos(o1)*sin(o2)*sin(o3)+6*cos(o1)*cos(o2)+11*sin(o1))+sin(o1)*
(4*sin(o1)*cos(o2)*cos(o3)-
4*sin(o1)*sin(o2)*sin(o3)+6*sin(o1)*cos(o2)-11*cos(o1))]
[
sin(o2)*cos(o3)+cos(o2)*sin(o3),
-sin(o2)*sin(o3)+cos(o2)*cos(o3),
0,
4*sin(o2)*cos(o3)+4*cos(o2)*sin(o3)+6*sin(o2)]
[
sin(o1)*(cos(o1)*cos(o2)*cos(o3)-cos(o1)*sin(o2)*sin(o3))-
cos(o1)*(sin(o1)*cos(o2)*cos(o3)-sin(o1)*sin(o2)*sin(o3)),
sin(o1)*(-cos(o1)*cos(o2)*sin(o3)-cos(o1)*sin(o2)*cos(o3))-
cos(o1)*(-sin(o1)*cos(o2)*sin(o3)-sin(o1)*sin(o2)*cos(o3)),
sin(o1)^2+cos(o1)^2, sin(o1)*(4*cos(o1)*cos(o2)*cos(o3)-
4*cos(o1)*sin(o2)*sin(o3)+6*cos(o1)*cos(o2)+11*sin(o1))-
cos(o1)*(4*sin(o1)*cos(o2)*cos(o3)-
4*sin(o1)*sin(o2)*sin(o3)+6*sin(o1)*cos(o2)-11*cos(o1))]
[
0,
0,
0,
1]




TE13 =

[ cos(o1)*nx+sin(o1)*ny, cos(o1)*ox+sin(o1)*oy,
cos(o1)*ax+sin(o1)*ay, cos(o1)*px+sin(o1)*py]
[                    nz,                   oz,
az,                   pz]
[ sin(o1)*nx-cos(o1)*ny, sin(o1)*ox-cos(o1)*oy, sin(o1)*ax-
cos(o1)*ay, sin(o1)*px-cos(o1)*py]
[                     0,                    0,
0,                    1]
```

```
T23 =

[
cos(o1)*cos(o2)*(cos(o1)*cos(o2)*cos(o3)-
cos(o1)*sin(o2)*sin(o3))+sin(o1)*cos(o2)*(sin(o1)*cos(o2)*cos(o3
)-
sin(o1)*sin(o2)*sin(o3))+sin(o2)*(sin(o2)*cos(o3)+cos(o2)*sin(o3
)),
cos(o1)*cos(o2)*(-cos(o1)*cos(o2)*sin(o3)-
cos(o1)*sin(o2)*cos(o3))+sin(o1)*cos(o2)*(-
sin(o1)*cos(o2)*sin(o3)-sin(o1)*sin(o2)*cos(o3))+sin(o2)*(-
sin(o2)*sin(o3)+cos(o2)*cos(o3)),
0, cos(o1)*cos(o2)*(4*cos(o1)*cos(o2)*cos(o3)-
4*cos(o1)*sin(o2)*sin(o3)+6*cos(o1)*cos(o2)+11*sin(o1))+sin(o1)*
cos(o2)*(4*sin(o1)*cos(o2)*cos(o3)-
4*sin(o1)*sin(o2)*sin(o3)+6*sin(o1)*cos(o2)-
11*cos(o1))+sin(o2)*(4*sin(o2)*cos(o3)+4*cos(o2)*sin(o3)+6*sin(o
2))-6]
[
-cos(o1)*sin(o2)*(cos(o1)*cos(o2)*cos(o3)-
cos(o1)*sin(o2)*sin(o3))-
sin(o1)*sin(o2)*(sin(o1)*cos(o2)*cos(o3)-
sin(o1)*sin(o2)*sin(o3))+cos(o2)*(sin(o2)*cos(o3)+cos(o2)*sin(o3
)),
-cos(o1)*sin(o2)*(-cos(o1)*cos(o2)*sin(o3)-
cos(o1)*sin(o2)*cos(o3))-sin(o1)*sin(o2)*(-
sin(o1)*cos(o2)*sin(o3)-sin(o1)*sin(o2)*cos(o3))+cos(o2)*(-
sin(o2)*sin(o3)+cos(o2)*cos(o3)),
0,  -cos(o1)*sin(o2)*(4*cos(o1)*cos(o2)*cos(o3)-
4*cos(o1)*sin(o2)*sin(o3)+6*cos(o1)*cos(o2)+11*sin(o1))-
sin(o1)*sin(o2)*(4*sin(o1)*cos(o2)*cos(o3)-
4*sin(o1)*sin(o2)*sin(o3)+6*sin(o1)*cos(o2)-
11*cos(o1))+cos(o2)*(4*sin(o2)*cos(o3)+4*cos(o2)*sin(o3)+6*sin(o
2))]
[
sin(o1)*(cos(o1)*cos(o2)*cos(o3)-cos(o1)*sin(o2)*sin(o3))-
cos(o1)*(sin(o1)*cos(o2)*cos(o3)-sin(o1)*sin(o2)*sin(o3)),
sin(o1)*(-cos(o1)*cos(o2)*sin(o3)-cos(o1)*sin(o2)*cos(o3))-
cos(o1)*(-sin(o1)*cos(o2)*sin(o3)-sin(o1)*sin(o2)*cos(o3)),
sin(o1)^2+cos(o1)^2,
sin(o1)*(4*cos(o1)*cos(o2)*cos(o3)-
4*cos(o1)*sin(o2)*sin(o3)+6*cos(o1)*cos(o2)+11*sin(o1))-
cos(o1)*(4*sin(o1)*cos(o2)*cos(o3)-
4*sin(o1)*sin(o2)*sin(o3)+6*sin(o1)*cos(o2)-11*cos(o1))-11]
[
0,
0,
0,
1]


TE23 =
```

```
[   cos(o1)*cos(o2)*nx+sin(o1)*cos(o2)*ny+sin(o2)*nz,
cos(o1)*cos(o2)*ox+sin(o1)*cos(o2)*oy+sin(o2)*oz,
cos(o1)*cos(o2)*ax+sin(o1)*cos(o2)*ay+sin(o2)*az,
cos(o1)*cos(o2)*px+sin(o1)*cos(o2)*py+sin(o2)*pz-6]
[  -cos(o1)*sin(o2)*nx-sin(o1)*sin(o2)*ny+cos(o2)*nz,  -
cos(o1)*sin(o2)*ox-sin(o1)*sin(o2)*oy+cos(o2)*oz,  -
cos(o1)*sin(o2)*ax-sin(o1)*sin(o2)*ay+cos(o2)*az,  -
cos(o1)*sin(o2)*px-sin(o1)*sin(o2)*py+cos(o2)*pz]
[                              sin(o1)*nx-cos(o1)*ny,
sin(o1)*ox-cos(o1)*oy,                              sin(o1)*ax-
cos(o1)*ay,                           sin(o1)*px-cos(o1)*py-11]
[                                                  0,
0,                                                  0,
1]


ST03 =

[                                    cos(o1)*cos(o2)*cos(o3)-
cos(o1)*sin(o2)*sin(o3),                                    -
cos(o1)*cos(o2)*sin(o3)-cos(o1)*sin(o2)*cos(o3),
sin(o1), 4*cos(o1)*cos(o2)*cos(o3)-
4*cos(o1)*sin(o2)*sin(o3)+6*cos(o1)*cos(o2)+11*sin(o1)]
[                                    sin(o1)*cos(o2)*cos(o3)-
sin(o1)*sin(o2)*sin(o3),                                    -
sin(o1)*cos(o2)*sin(o3)-sin(o1)*sin(o2)*cos(o3),
-cos(o1), 4*sin(o1)*cos(o2)*cos(o3)-
4*sin(o1)*sin(o2)*sin(o3)+6*sin(o1)*cos(o2)-11*cos(o1)]
[
sin(o2)*cos(o3)+cos(o2)*sin(o3),
-sin(o2)*sin(o3)+cos(o2)*cos(o3),
0,
4*sin(o2)*cos(o3)+4*cos(o2)*sin(o3)+6*sin(o2)]
[
0,
0,
0,
1]


ST13 =

[              -sin(o2)*sin(o3)+cos(o2)*cos(o3),              -
cos(o2)*sin(o3)-sin(o2)*cos(o3),
0, 4*cos(o2)*cos(o3)-4*sin(o2)*sin(o3)+6*cos(o2)]
[               sin(o2)*cos(o3)+cos(o2)*sin(o3),              -
sin(o2)*sin(o3)+cos(o2)*cos(o3),
0, 4*sin(o2)*cos(o3)+4*cos(o2)*sin(o3)+6*sin(o2)]
[                                                 0,
0,                                                 1,
11]
```

```
[                                                0,
0,                                                0,
1]


STE13 =

[ cos(o1)*nx+sin(o1)*ny, cos(o1)*ox+sin(o1)*oy,
cos(o1)*ax+sin(o1)*ay, cos(o1)*px+sin(o1)*py]
[                    nz,                    oz,
az,                    pz]
[ sin(o1)*nx-cos(o1)*ny, sin(o1)*ox-cos(o1)*oy, sin(o1)*ax-
cos(o1)*ay, sin(o1)*px-cos(o1)*py]
[                     0,                     0,
0,                     1]


ST23 =

[   cos(o3),  -sin(o3),         0, 4*cos(o3)]
[   sin(o3),   cos(o3),         0, 4*sin(o3)]
[         0,         0,         1,         0]
[         0,         0,         0,         1]


STE23 =

[   cos(o1)*cos(o2)*nx+sin(o1)*cos(o2)*ny+sin(o2)*nz,
cos(o1)*cos(o2)*ox+sin(o1)*cos(o2)*oy+sin(o2)*oz,
cos(o1)*cos(o2)*ax+sin(o1)*cos(o2)*ay+sin(o2)*az,
cos(o1)*cos(o2)*px+sin(o1)*cos(o2)*py+sin(o2)*pz-6]
[  -cos(o1)*sin(o2)*nx-sin(o1)*sin(o2)*ny+cos(o2)*nz,  -
cos(o1)*sin(o2)*ox-sin(o1)*sin(o2)*oy+cos(o2)*oz,  -
cos(o1)*sin(o2)*ax-sin(o1)*sin(o2)*ay+cos(o2)*az,  -
cos(o1)*sin(o2)*px-sin(o1)*sin(o2)*py+cos(o2)*pz]
[                             sin(o1)*nx-cos(o1)*ny,
sin(o1)*ox-cos(o1)*oy,                              sin(o1)*ax-
cos(o1)*ay,                          sin(o1)*px-cos(o1)*py-11]
[                                                  0,
0,                                                  0,
1]
```

Apêndice B - Cálculo para validação dos ângulos

Cálculos feitos no MatLab, utilizados para a validação dos cálculos dos ângulos gerados pelo *software*.

```
>> syms nx ny nz ox oy oz ax ay az px py pz

o1=135*pi/180
o2=90*pi/180
o3=45*pi/180

T01 = [cos(o1) 0 sin(o1) 0; sin(o1) 0 -cos(o1) 0; 0 1 0 0; 0 0 0 1]
T12 = [cos(o2) -sin(o2) 0 6*cos(o2); sin(o2) cos(o2) 0 6*sin(o2); 0 0
1 11; 0 0 0 1]
T23 = [cos(o3) -sin(o3) 0 4*cos(o3); sin(o3) cos(o3) 0 4*sin(o3); 0 0
1 0; 0 0 0 1]

TI01 = [cos(o1) sin(o1) 0 0; 0 0 1 0; sin(o1) -cos(o1) 0 0; 0 0 0 1]
TI12 = [cos(o2) sin(o2) 0 -6; -sin(o2) cos(o2) 0 0; 0 0 1 -11; 0 0 0
1]

TE03 = [nx ox ax px; ny oy ay py; nz oz az pz; 0 0 0 1]

T03 = T01 * T12 * T23

o1 =

    2.3562


o2 =

    1.5708


o3 =

    0.7854


T01 =

   -0.7071         0    0.7071         0
    0.7071         0    0.7071         0
         0    1.0000         0         0
         0         0         0    1.0000


T12 =

    0.0000   -1.0000         0    0.0000
    1.0000    0.0000         0    6.0000
         0         0    1.0000   11.0000
         0         0         0    1.0000
```

```
T23 =

    0.7071   -0.7071         0    2.8284
    0.7071    0.7071         0    2.8284
         0         0    1.0000         0
         0         0         0    1.0000


TI01 =

   -0.7071    0.7071         0         0
         0         0    1.0000         0
    0.7071    0.7071         0         0
         0         0         0    1.0000


TI12 =

    0.0000    1.0000         0   -6.0000
   -1.0000    0.0000         0         0
         0         0    1.0000  -11.0000
         0         0         0    1.0000


TE03 =

[ nx, ox, ax, px]
[ ny, oy, ay, py]
[ nz, oz, az, pz]
[  0,  0,  0,  1]


T03 =

    0.5000    0.5000    0.7071    9.7782
   -0.5000   -0.5000    0.7071    5.7782
    0.7071   -0.7071         0    8.8284
         0         0         0    1.0000
```

Apêndice C - Código fonte do *software*

Código fonte do *software*.

```
'*************************************************************************
Public Declare Function Inp Lib "inpout32.dll" _
Alias "Inp32" (ByVal PortAddress As Integer) As Integer
Public Declare Sub Out Lib "inpout32.dll" _
Alias "Out32" (ByVal PortAddress As Integer, ByVal Value As Integer)
'*************************************************************************

Public fMainForm As frmMain


Global mostra_traj As Boolean
Global g_dbtrajetoria As New ADODB.Connection
Global g_rdtrajetoria As New ADODB.Recordset
Global g_rdmatriz As New ADODB.Recordset
Global g_rdmattraj As New ADODB.Recordset
Global g_sOrigemBD As String

Global g_nomeTrajetoria As String
Global g_matriz1, g_matriz2 As String

Global gtrajcode As String
Global gnext As Integer
Global gvposx(100), gvposy(100), gvposz(100), gvpos(100) As Single
Global gvposnx(100), gvposny(100), gvposnz(100) As Single
Global  posicao(10000),  posicaox(10000),  posicaoy(10000),  posicaoz(10000)  As
Single
Global posicaonx(10000), posicaony(10000), posicaonz(10000) As Single
Global teta3(1000), teta2(1000), teta1(1000), gtotal As Integer
Global mat13(5, 5), mat12(5, 5), mat01(5, 5), mat02(5, 5) As Single
Global g_exec_traj, traj_act As Boolean
Global grdatax(), grdatay(), grdataz(), grtotal As Single
Global grdatanx(), grdatany(), grdatanz() As Single

'*************************************************************************
Sub Main()
'*************************************************************************
    mostra_traj = False
    traj_act = False

    frmSplash.Show

   'carrega banco de dados
    DataSource = App.Path & "\trajetoria.mdb"
    g_sOrigemBD    =    "Provider=Microsoft.Jet.OLEDB.4.0;Data    Source='"    &
DataSource & "';Persist Security Info=False"
    Set g_dbtrajetoria = New ADODB.Connection
    g_dbtrajetoria.Open g_sOrigemBD

    Set g_rdtrajetoria = New ADODB.Recordset
    g_rdtrajetoria.CursorType = adOpenKeyset
    g_rdtrajetoria.LockType = adLockOptimistic

    Set g_rdmatriz = New ADODB.Recordset
    g_rdmatriz.CursorType = adOpenKeyset
    g_rdmatriz.LockType = adLockOptimistic

    Set g_rdmattraj = New ADODB.Recordset
    g_rdmattraj.CursorType = adOpenKeyset
    g_rdmattraj.LockType = adLockOptimistic
```

```
    g_nomeTrajetoria = Empty

    frmSplash.Refresh
    Set fMainForm = New frmMain
    Load fMainForm
    Unload frmSplash


    fMainForm.Show
End Sub

'************************************************************************
Function interpolar()
'************************************************************************
Dim gi_vf, freq As Single
Dim gi_v, gi_vi, gi_xi, gi_xf, gi_yi, gi_yf, gi_zi, gi_zf As Single

i = 1
j = 1
freq_timer = 1 / 2
tempo = Timer
gi_v = gvpos(1)
gi_vf = gvpos(i + 1)



        gi_vi = gvpos(i)
        gi_xi = teta1(i)
        gi_vf = gvpos(i + 1)
        gi_xf = teta1(i + 1)
        posicaox(j) = (((gi_v - gi_vi) / (gi_vf - gi_vi)) * (gi_xf - gi_xi)) +
gi_xi

        gi_vi = gvpos(i)
        gi_yi = teta2(i)
        gi_vf = gvpos(i + 1)
        gi_yf = teta2(i + 1)
        posicaoy(j) = (((gi_v - gi_vi) / (gi_vf - gi_vi)) * (gi_yf - gi_yi)) +
gi_yi

        gi_vi = gvpos(i)
        gi_zi = teta3(i)
        gi_vf = gvpos(i + 1)
        gi_zf = teta3(i + 1)
        posicaoz(j) = (((gi_v - gi_vi) / (gi_vf - gi_vi)) * (gi_zf - gi_zi)) +
gi_zi


        j = j + 1

Do

    freq = freq_timer + gi_v
    If freq < gi_vf Then
        gi_v = freq_timer + gi_v

        gi_vi = gvpos(i)
        gi_xi = teta1(i)
        gi_vf = gvpos(i + 1)
        gi_xf = teta1(i + 1)
        posicaox(j) = (((gi_v - gi_vi) / (gi_vf - gi_vi)) * (gi_xf - gi_xi)) +
gi_xi

        gi_vi = gvpos(i)
```

```
        gi_yi = teta2(i)
        gi_vf = gvpos(i + 1)
        gi_yf = teta2(i + 1)
        posicaoy(j) = (((gi_v - gi_vi) / (gi_vf - gi_vi)) * (gi_yf - gi_yi)) +
gi_yi

        gi_vi = gvpos(i)
        gi_zi = teta3(i)
        gi_vf = gvpos(i + 1)
        gi_zf = teta3(i + 1)
        posicaoz(j) = (((gi_v - gi_vi) / (gi_vf - gi_vi)) * (gi_zf - gi_zi)) +
gi_zi


        j = j + 1

   End If
   If freq >= gi_vf Then
        i = i + 1
        gi_vf = gvpos(i + 1)
    End If


     freq = freq_timer + gi_v
Loop While i <> gnext '6
For i = 1 To 13



'MsgBox posicaoz(i)
Next i
End Function

'*************************************************************************
Function executa_trajetoria()
'*************************************************************************
'formulas de cinematica inversa
For i = 1 To gtotal - 1
    px = gvposx(i)
    py = gvposy(i)
    pz = gvposz(i)
    nx = gvposnx(i)
    ny = gvposny(i)
    nz = gvposnz(i)

    'teta1
    angulo1rad = Atn(ny / nx)
    teta1(i) = 180 * angulo1rad / 3.14
    frmTrajetoria.txtposx.Text = angulo1rad
    If angulo1rad < -0.2 Then
        teta1(i) = 180
    End If
    If ny / nx = -0.000001 Then
        teta1(i) = 180
    End If
    teta1rad = 3.14 * teta1(i) / 180
    'teta2
    angulo2eq = (pz - 4 * nz) / 6 'arcseno
    If (angulo2eq > -1) And (angulo2eq < 1) Then
        angulo2rad = Atn(angulo2eq / Sqr(1 - angulo2eq * angulo2eq))
    ElseIf angulo2eq <= -1 Then
        angulo2rad = -3.14 * 0.5
    ElseIf angulo2eq >= 1 Then
        angulo2rad = 3.14 * 0.5
   End If
```

```
   teta2(i) = 180 * angulo2rad / 3.14
   If pz = -0.00001 And nz = -0.00001 Then
    teta2(i) = 180
   End If
   'teta3
    teta2rad = 3.14 * teta2(i) / 180
    ab = Cos(teta1rad) * px + Sin(teta1rad) * py
    cc = Cos(teta2rad) * ab
    cd = Sin(teta2rad) * pz
    angulo3eq = (cc + cd - 6) / 4 'arcoseno
    If (angulo3eq > 0) And (angulo3eq <= 1) Then
        angulo3rad = Atn(Sqr(1 - angulo3eq * angulo3eq) / angulo3eq)
    ElseIf (angulo3eq < 0) And (angulo3eq >= -1) Then
        angulo3rad = Atn(Sqr(1 - angulo3eq * angulo3eq) / angulo3eq) + 3.14
    ElseIf angulo3eq = 0 Then
        angulo3rad = 3.14 / 2
    ElseIf angulo3eq < -1 Then
        angulo3rad = 3.14
    End If
    teta3(i) = 180 * angulo3rad / 3.14
    If teta2rad = 3.14 And teta1rad = 3.14 Then
        teta3(i) = 180
    End If


Next i

End Function

'A passagem relevante do codigo do programa  de geração de onda pela porta
paralela,
'para controle de servomotor, está relatada a seguir. Nessa é colocada como se
deve
'chamar a dll e como programar o timer gerando uma onda quanquer:
'*************************************************************************
Function aciona_robix(angulo1, angulo2, angulo3)
'*************************************************************************
' Preparação do timer para onda de 0.1ms e ativação.
Out Val("&H43"), Val("&H96") 'outportb($43,150);
Out Val("&H42"), Val("&H77") '77'outportb($42,119) frequencia/119
Out Val("&H43"), Val("&HE8") 'outportb($43,232)'READ BACK COMMAND controle
Out  Val("&H61"),  Val("&H01")  'outportb($61,1);'ativa  o  contador  e  desativa  o
som

 For k = 0 To 20000 '8000
 pos1 = Int(0.089 * angulo1 + 9)
 pos2 = Int(0.089 * angulo2 + 8)
 pos3 = Int(0.09 * angulo3 + 9)
        Out Val("&H43"), Val("&HE8")
     inpo = Inp(Val("&H42"))
    While inpo > 128 '128 Montagen da onda no formato desejado.
        Out Val("&H43"), Val("&HE8")
        inpo = Inp(Val("&H42")) 'PEGA VALOR DA CONTAGEM
    Wend
    While inpo <= 128 '128
        Out Val("&H43"), Val("&HE8")
        inpo = Inp(Val("&H42")) 'PEGA VALOR  DA CONTAGEM
    Wend

    If Y < pos3 Then
        saida3 = &H4
    End If
    If Y >= pos3 Then
      saida3 = &H0
    End If
```

```
    If Y < pos2 Then
        saida2 = &H2
    End If
    If Y >= pos2 Then
        saida2 = &H0
    End If

    If Y < pos1 Then
        saida1 = &H1
    End If
    If Y >= pos1 Then
      saida1 = &H0
    End If


      saida = saida1 Or saida2 Or saida3
      Out Val("&H378"), Val(saida)
      If Y > 500 Then 'para mecher na frequencia
        Y = 0
      End If
Y = Y + 1
Next k
 Out Val("&H378"), Val("&Hff")
End Function

'*************************************************************************
Private Sub Form_Load()
'*************************************************************************
    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." &
App.Revision
    lblProductName.Caption = App.Title
End Sub


'*************************************************************************
Private Sub Command1_Click()
'*************************************************************************
Dim t1, t2 As Date
    PictureSuperior.Cls
    PictureFrontal.Cls

    executa_trajetoria
    For i = 1 To gtotal - 1
    For ang = 0 To 180
            On Error Resume Next
    If ang <= teta1(i) Then
        X1 = ang * 3.141593 / 180
        Y1 = X1 + 0.0001
        PictureSuperior.Circle (2500, 2000), 2000, , -X1, Y1
        If teta1(i) = 0 Then
            PictureSuperior.Circle (2500, 2100), 2000, , -0.00001, 0.0001
        End If
     End If
     If ang <= teta2(i) Then
        X2 = (ang + 180) * 3.141593 / 180
        Y2 = X2 + 0.0001
        PictureFrontal.Circle (2500, 1000), 1200, &HFF00&, -X2, Y2
        If teta2(i) = 0 Then
            PictureFrontal.Circle  (2500,  1000),  1200,  &HFF00&,  -0.00001,
0.0001
            Y2 = 0.0001
        End If
    End If
   If ang <= teta3(i) Then
```

```
        endx = 2500 + 1200 * Cos(Y2)
        endy = 1000 - 1200 * Sin(Y2)
        X3 = (ang + 180) * 3.141593 / 180
        Y3 = X3 + 0.001
        PictureFrontal.Circle (endx, endy), 800, &HFF&, -X3, Y3
        If teta3(i) = 0 Then
            PictureFrontal.Circle (endx, endy), 800, &HFF&, -0.00001, 0.0001
            Y2 = 0.0001
        End If
    ElseIf ang > teta3(i) Then
        endx = 2500 + 1200 * Cos(Y2)
        endy = 1000 - 1200 * Sin(Y2)
        X3 = (teta3(i) + 180) * 3.141593 / 180
        Y3 = X3 + 0.001
        PictureFrontal.Circle (endx, endy), 800, &HFF&, -X3, Y3
    End If
    t1 = Time
    While (t2 <= 0.00001)
        t2 = Time - t1
    Wend
    t1 = Time
    While (t2 <= 0.00001)
        t2 = Time - t1
    Wend
    Next ang
    Next i

End Sub

'************************************************************************
Private Sub cmdAbrir_Click()
'************************************************************************
    traj_act = True
    frmAbrir.Show
End Sub
'************************************************************************
Private Sub cmdBack_Click()
'************************************************************************
If gtrajcode = Empty Then

   gvpos(gnext) = txtpos.Text
   gvposx(gnext) = txtposx.Text
   gvposy(gnext) = txtposy.Text
   gvposz(gnext) = txtposz.Text
   gvposnx(gnext) = txtposnx.Text
   gvposny(gnext) = txtposny.Text
   gvposnz(gnext) = txtposnz.Text

   gnext = gnext - 1
   If gnext < 1 Then
    gnext = gnext + 1
   Else
    txtpos.Text = gvpos(gnext)
    txtposx.Text = gvposx(gnext)
    txtposy.Text = gvposy(gnext)
    txtposz.Text = gvposz(gnext)
    txtposnx.Text = gvposnx(gnext)
    txtposny.Text = gvposny(gnext)
    txtposnz.Text = gvposnz(gnext)
   End If
Else
      gvpos(gnext) = txtpos.Text
      gvposx(gnext) = txtposx.Text
      gvposy(gnext) = txtposy.Text
      gvposz(gnext) = txtposz.Text
```

```
      gvposnx(gnext) = txtposnx.Text
      gvposny(gnext) = txtposny.Text
      gvposnz(gnext) = txtposnz.Text
   gnext = gnext - 1

   txtpos.Text = gvpos(gnext)
   txtposx.Text = gvposx(gnext)
   txtposy.Text = gvposy(gnext)
   txtposz.Text = gvposz(gnext)
   txtposnx.Text = gvposnx(gnext)
   txtposny.Text = gvposny(gnext)
   txtposnz.Text = gvposnz(gnext)

   If gvpos(gnext) <> Empty Then
     txtpos.Text = gvpos(gnext)
     txtposx.Text = gvposx(gnext)
     txtposy.Text = gvposy(gnext)
     txtposz.Text = gvposz(gnext)
     txtposnx.Text = gvposnx(gnext)
     txtposny.Text = gvposny(gnext)
     txtposnz.Text = gvposnz(gnext)
   End If
End If

End Sub

'************************************************************************
Private Sub CmdExcluir_Click()
'************************************************************************

 g_dbtrajetoria.Execute "DELETE * FROM trajetoria where traj_code like '" &
gtrajcode & "' "
 g_rdmattraj.Source = "SELECT * FROM mattraj WHERE trajetoria_code like '" &
gtrajcode & "' "
 g_rdmattraj.Open , g_dbtrajetoria, adOpenStatic

 Do While Not g_rdmattraj.EOF
   g_dbtrajetoria.Execute  "DELETE  *  FROM  matriz  where  code  like  '"  &
g_rdmattraj!matriz_code & "' "
   g_rdmattraj.MoveNext

 Loop
 If g_rdmattraj.State = adStateOpen Then
    g_rdmattraj.Close
 End If

 g_dbtrajetoria.Execute "DELETE * FROM mattraj where trajetoria_code like '" &
gtrajcode & "' "

   For i = 0 To gnext
    gvpos(i) = Empty
    gvposx(i) = Empty
    gvposy(i) = Empty
    gvposz(i) = Empty
    gvposnx(i) = Empty
    gvposny(i) = Empty
    gvposnz(i) = Empty
   Next i
   gnext = 0
   txtpos.Text = Empty
   txtposx.Text = Empty
   txtposy.Text = Empty
   txtposz.Text = Empty
   txtposnx.Text = Empty
   txtposny.Text = Empty
```

```
   txtposnz.Text = Empty
   g_nomeTrajetoria = Empty
   txtnometrajetoria.Text = Empty
   gtrajcode = Empty

End Sub
'*************************************************************************
Private Sub CmdExecTrajetoria_Click()
'*************************************************************************
If gvposx(gnext) = Empty Then
   gtotal = genext - 1
End If
'inserir posições na matriz
g_exec_traj = True
' fechar a tela
executa_trajetoria
' movimentar o robô
For cont = 1 To gtotal - 1
aciona_robix teta1(cont), teta2(cont), teta3(cont)
Next cont
End Sub
'*************************************************************************
Private Sub CmdNova_Click()
'*************************************************************************
   For i = 0 To gnext
    gvpos(i) = Empty
    gvposx(i) = Empty
    gvposy(i) = Empty
    gvposz(i) = Empty
    gvposnx(i) = Empty
    gvposny(i) = Empty
    gvposnz(i) = Empty
   Next i
   gnext = 1
   txtpos.Text = 1
   txtposx.Text = Empty
   txtposy.Text = Empty
   txtposz.Text = Empty
   txtposnx.Text = Empty
   txtposny.Text = Empty
   txtposnz.Text = Empty
   g_nomeTrajetoria = Empty
   txtnometrajetoria.Text = Empty
   gtrajcode = Empty
End Sub
'*************************************************************************
Private Sub cmdproximapos_Click()
'*************************************************************************
If gtrajcode = Empty Then

   gvpos(gnext) = txtpos.Text
   gvposx(gnext) = txtposx.Text
   gvposy(gnext) = txtposy.Text
   gvposz(gnext) = txtposz.Text
   gvposnx(gnext) = txtposnx.Text
   gvposny(gnext) = txtposny.Text
   gvposnz(gnext) = txtposnz.Text

   gnext = gnext + 1
   If gvpos(gnext) = Empty Then
      gtotal = gnext
      txtpos.Text = gnext
      txtposx.Text = Empty
      txtposy.Text = Empty
      txtposz.Text = Empty
```

```
      txtposnx.Text = Empty
      txtposny.Text = Empty
      txtposnz.Text = Empty
   Else
     txtpos.Text = gvpos(gnext)
     txtposx.Text = gvposx(gnext)
     txtposy.Text = gvposy(gnext)
     txtposz.Text = gvposz(gnext)
     txtposnx.Text = gvposnx(gnext)
     txtposny.Text = gvposny(gnext)
     txtposnz.Text = gvposnz(gnext)
   End If
Else
      gvpos(gnext) = txtpos.Text
      gvposx(gnext) = txtposx.Text
      gvposy(gnext) = txtposy.Text
      gvposz(gnext) = txtposz.Text
      gvposnx(gnext) = txtposnx.Text
      gvposny(gnext) = txtposny.Text
      gvposnz(gnext) = txtposnz.Text
      gnext = gnext + 1
      gtotal = gnext
      txtpos.Text = gnext
      txtposx.Text = Empty
      txtposy.Text = Empty
      txtposz.Text = Empty
      txtposnx.Text = Empty
      txtposny.Text = Empty
      txtposnz.Text = Empty

   If gvpos(gnext) <> Empty Then
     txtpos.Text = gvpos(gnext)
     txtposx.Text = gvposx(gnext)
     txtposy.Text = gvposy(gnext)
     txtposz.Text = gvposz(gnext)
     txtposnx.Text = gvposnx(gnext)
     txtposny.Text = gvposny(gnext)
     txtposnz.Text = gvposnz(gnext)
   End If
End If

End Sub

'*************************************************************************
Private Sub CmdSimularTraj_Click()
'*************************************************************************
If gvposx(gnext) = Empty Then
   gtotal = genext - 1
End If

frmTrajetoria.Hide


frmPrincipal.Show
frmPrincipal.Enabled = True
mostra_traj = True


End Sub

'*************************************************************************
Private Sub CmdSalvar_Click()
'*************************************************************************
If txtnometrajetoria.Text <> Empty Then
'caso seja uma nova trajetoria
```

```
 If gtrajcode = Empty Then

    If txtposx.Text = Empty Then
       MsgBox ("Por favor, Digite um valor para a posição px")
    Else
     gvpos(gnext) = txtpos.Text
     gvposx(gnext) = txtposx.Text
    End If
    If txtposy.Text = Empty Then
       MsgBox ("Por favor, Digite um valor para a posição py")
    Else
     gvpos(gnext) = txtpos.Text
     gvposy(gnext) = txtposy.Text
    End If
   If txtposz.Text = Empty Then
       MsgBox ("Por favor, Digite um valor para a posição pz")
    Else
     gvpos(gnext) = txtpos.Text
     gvposz(gnext) = txtposz.Text
    End If
  If txtposnx.Text = Empty Then
       MsgBox ("Por favor, Digite um valor para a posição nx")
    Else
     gvpos(gnext) = txtpos.Text
     gvposnx(gnext) = txtposnx.Text
    End If
    If txtposny.Text = Empty Then
       MsgBox ("Por favor, Digite um valor para a posição ny")
    Else
     gvpos(gnext) = txtpos.Text
     gvposny(gnext) = txtposny.Text
    End If
   If txtposnz.Text = Empty Then
       MsgBox ("Por favor, Digite um valor para a posição nz")
    Else
     gvpos(gnext) = txtpos.Text
     gvposnz(gnext) = txtposnz.Text
    End If

'adiciona a trajetoria
    g_rdtrajetoria.Source = "SELECT * FROM trajetoria "
    g_rdtrajetoria.Open , g_dbtrajetoria, adOpenStatic
    If Not g_rdtrajetoria.EOF Then
    g_rdtrajetoria.MoveLast
    traj_code = g_rdtrajetoria!traj_code + 1
    g_rdtrajetoria.AddNew
    g_rdtrajetoria!traj_code = traj_code
    g_rdtrajetoria!traj_name = txtnometrajetoria.Text
    g_rdtrajetoria.Update
    If g_rdtrajetoria.State = adStateOpen Then
       g_rdtrajetoria.Close
    End If
    End If
  'adiciona as posições da matriz
   For i = 1 To gnext
     g_rdmatriz.Source = "SELECT * FROM matriz "
     g_rdmatriz.Open , g_dbtrajetoria, adOpenStatic
     If Not g_rdmatriz.EOF Then
     g_rdmatriz.MoveLast
     code = g_rdmatriz!code + 1
     g_rdmatriz.AddNew
     g_rdmatriz!code = code
     g_rdmatriz!mat_cod = gvpos(i)
     g_rdmatriz!px = gvposx(i)
     g_rdmatriz!py = gvposy(i)
```

```
        g_rdmatriz!pz = gvposz(i)
        g_rdmatriz!nx = gvposnx(i)
        g_rdmatriz!ny = gvposny(i)
        g_rdmatriz!nz = gvposnz(i)
        g_rdmatriz.Update
        End If
        If g_rdmatriz.State = adStateOpen Then
           g_rdmatriz.Close
        End If

        'faz o relacionameto entre matriz e trajetória
        g_rdmattraj.Source = "SELECT * FROM mattraj "
        g_rdmattraj.Open , g_dbtrajetoria, adOpenStatic
        g_rdmattraj.AddNew
        g_rdmattraj!matriz_code = code
        g_rdmattraj!trajetoria_code = traj_code
        g_rdmattraj.Update
        If g_rdmattraj.State = adStateOpen Then
           g_rdmattraj.Close
        End If
      Next i
'caso atualize a trajetoria ja existente
    Else
      If txtposx.Text = Empty Then
          MsgBox ("Por favor, Digite um valor para a posição px")
       Else
        gvpos(gnext) = txtpos.Text
        gvposx(gnext) = txtposx.Text
       End If
       If txtposy.Text = Empty Then
          MsgBox ("Por favor, Digite um valor para a posição py")
       Else
        gvpos(gnext) = txtpos.Text
        gvposy(gnext) = txtposy.Text
       End If
      If txtposz.Text = Empty Then
          MsgBox ("Por favor, Digite um valor para a posição pz")
       Else
        gvpos(gnext) = txtpos.Text
        gvposz(gnext) = txtposz.Text
       End If

      If txtposnx.Text = Empty Then
          MsgBox ("Por favor, Digite um valor para a posição nx")
       Else
        gvpos(gnext) = txtpos.Text
        gvposnx(gnext) = txtposnx.Text
       End If
       If txtposny.Text = Empty Then
          MsgBox ("Por favor, Digite um valor para a posição ny")
       Else
        gvpos(gnext) = txtpos.Text
        gvposny(gnext) = txtposny.Text
       End If
      If txtposnz.Text = Empty Then
          MsgBox ("Por favor, Digite um valor para a posição nz")
       Else
        gvpos(gnext) = txtpos.Text
        gvposnz(gnext) = txtposnz.Text
       End If

   'adiciona a trajetoria
       g_rdtrajetoria.Source = "SELECT * FROM trajetoria  " _
       & "where traj_code like '" & gtrajcode & "' "
       g_rdtrajetoria.Open , g_dbtrajetoria, adOpenStatic
```

```
  'g_rdtrajetoria!traj_code = gtrajcode
  g_rdtrajetoria!traj_name = txtnometrajetoria.Text
  g_rdtrajetoria.Update
  If g_rdtrajetoria.State = adStateOpen Then
     g_rdtrajetoria.Close
  End If
'adiciona as posições da matriz
 For i = 1 To gnext

    g_rdmattraj.Source = "SELECT * FROM mattraj " _
    & "where trajetoria_code like '" & gtrajcode & "' "
    g_rdmattraj.Open , g_dbtrajetoria, adOpenStatic

   'If Not g_rdmattraj.EOF Then
     Do While Not g_rdmattraj.EOF

         g_rdmatriz.Source = "SELECT * FROM matriz " _
         & "where code like '" & g_rdmattraj!matriz_code & "' and " _
         & "mat_cod like '" & i & "' "
         g_rdmatriz.Open , g_dbtrajetoria, adOpenStatic

         'g_rdmatriz!code = code
         'g_rdmatriz!mat_cod = gvpos(i)
         g_rdmatriz!px = gvposx(i)
         g_rdmatriz!py = gvposy(i)
         g_rdmatriz!pz = gvposz(i)
         g_rdmatriz!nx = gvposnx(i)
         g_rdmatriz!ny = gvposny(i)
         g_rdmatriz!nz = gvposnz(i)
         g_rdmatriz.Update
         If g_rdmatriz.State = adStateOpen Then
             g_rdmatriz.Close
         End If
         g_rdmattraj.MoveNext
       i = i + 1
     Loop
'   Else
For j = i To gnext
       If g_rdmattraj.State = adStateOpen Then
           g_rdmattraj.Close
       End If
       g_rdmatriz.Source = "SELECT * FROM matriz "
       g_rdmatriz.Open , g_dbtrajetoria, adOpenStatic
       g_rdmatriz.MoveLast
       code = g_rdmatriz!code + 1
       g_rdmatriz.AddNew
       g_rdmatriz!code = code
       g_rdmatriz!mat_cod = gvpos(i)
       g_rdmatriz!px = gvposx(i)
       g_rdmatriz!py = gvposy(i)
       g_rdmatriz!pz = gvposz(i)
       g_rdmatriz!nx = gvposnx(i)
       g_rdmatriz!ny = gvposny(i)
       g_rdmatriz!nz = gvposnz(i)
       g_rdmatriz.Update
       If g_rdmatriz.State = adStateOpen Then
          g_rdmatriz.Close
       End If
       'faz o relacionameto entre matriz e trajetória
       g_rdmattraj.Source = "SELECT * FROM mattraj "
       g_rdmattraj.Open , g_dbtrajetoria, adOpenStatic
       g_rdmattraj.AddNew
       g_rdmattraj!matriz_code = code
       g_rdmattraj!trajetoria_code = gtrajcode
```

```
            g_rdmattraj.Update
            i = i + 1
         Next j
         If g_rdmattraj.State = adStateOpen Then
            g_rdmattraj.Close
         End If
        Next i



    End If
Else
    MsgBox ("Por favor, Digite um nome para a matriz")
End If

End Sub

'************************************************************************
Private Sub Form_Activate()
'************************************************************************
If traj_act = True Then
i = 1
If g_nomeTrajetoria <> Empty Then
   g_rdtrajetoria.Source = "SELECT * FROM trajetoria " _
   & "where traj_name like '" & g_nomeTrajetoria & "' "
   g_rdtrajetoria.Open , g_dbtrajetoria, adOpenStatic
   Do While Not g_rdtrajetoria.EOF
      txtnometrajetoria.Text = g_rdtrajetoria!traj_name
      gtrajcode = g_rdtrajetoria!traj_code
      g_rdtrajetoria.MoveNext
   Loop
   g_rdtrajetoria.Close
   g_rdmattraj.Source = "SELECT * FROM mattraj " _
   & "where trajetoria_code like '" & gtrajcode & "' "
   g_rdmattraj.Open , g_dbtrajetoria, adOpenStatic
   Do While Not g_rdmattraj.EOF
        g_rdmatriz.Source = "SELECT * FROM matriz " _
        & "where code like '" & g_rdmattraj!matriz_code & "' "
        g_rdmatriz.Open , g_dbtrajetoria, adOpenStatic
        Do While Not g_rdmatriz.EOF
           gvpos(i) = g_rdmatriz!mat_cod
           gvposx(i) = g_rdmatriz!px
           gvposy(i) = g_rdmatriz!py
           gvposz(i) = g_rdmatriz!pz
           gvposnx(i) = g_rdmatriz!nx
           gvposny(i) = g_rdmatriz!ny
           gvposnz(i) = g_rdmatriz!nz
           i = i + 1
           g_rdmatriz.MoveNext
        Loop
         g_rdmattraj.MoveNext
        g_rdmatriz.Close

   Loop
   g_rdmattraj.Close

End If
   gtotal = i
   gnext = 1
   If gvpos(gnext) = 0 Then
      gvpos(gnext) = 1
   End If
   txtpos.Text = gvpos(gnext)
   txtposx.Text = gvposx(gnext)
   txtposy.Text = gvposy(gnext)
```

```
   txtposz.Text = gvposz(gnext)
   txtposnx.Text = gvposnx(gnext)
   txtposny.Text = gvposny(gnext)
   txtposnz.Text = gvposnz(gnext)
    traj_act = False
End If

End Sub
Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal
hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Any)
As Long
Const EM_UNDO = &HC7
Private Declare Function OSWinHelp% Lib "user32" Alias "WinHelpA" (ByVal
hwnd&, ByVal HelpFile$, ByVal wCommand%, dwData As Any)


'************************************************************************
Private Sub MDIForm_Load()
'************************************************************************
    Me.Left = GetSetting(App.Title, "Settings", "MainLeft", 1000)
    Me.Top = GetSetting(App.Title, "Settings", "MainTop", 1000)
    Me.Width = GetSetting(App.Title, "Settings", "MainWidth", 6500)
    Me.Height = GetSetting(App.Title, "Settings", "MainHeight", 6500)
    traj_act = True
    frmPrincipal.Show


End Sub
'************************************************************************
Private Sub MDIForm_Unload(Cancel As Integer)
'************************************************************************
    If Me.WindowState <> vbMinimized Then
        SaveSetting App.Title, "Settings", "MainLeft", Me.Left
        SaveSetting App.Title, "Settings", "MainTop", Me.Top
        SaveSetting App.Title, "Settings", "MainWidth", Me.Width
        SaveSetting App.Title, "Settings", "MainHeight", Me.Height
    End If
End Sub
'************************************************************************
Private Sub mnuFileExit_Click()
'************************************************************************
End

End Sub
'************************************************************************
Private Sub mnuTrajetoria_Click()
'************************************************************************
    frmTrajetoria.Show
End Sub

'************************************************************************
Private Sub cmbTrajetoria_Click()
'************************************************************************
g_nomeTrajetoria = cmbTrajetoria.Text

End Sub

'************************************************************************
Private Sub CmdCancel_Click()
'************************************************************************
frmAbrir.Hide
g_nomeTrajetoria = Empty
End Sub

'************************************************************************
```

```
Private Sub CmdOK_Click()
'************************************************************************
frmAbrir.Hide
End Sub

'************************************************************************
Private Sub Form_Load()
'************************************************************************
   g_rdtrajetoria.Source = "SELECT traj_name, traj_code FROM trajetoria; "
   g_rdtrajetoria.Open , g_dbtrajetoria, adOpenStatic
   Do While Not g_rdtrajetoria.EOF
         cmbTrajetoria.AddItem g_rdtrajetoria!traj_name
         g_rdtrajetoria.MoveNext
   Loop
   g_rdtrajetoria.Close


End Sub
```

Apêndice D – Manual do Usuário

Obs.: Utilize o CD que está junto com a documentação do projeto para instalar o *software*. Insira este CD no *drive* do CD-ROM do computador, e siga as instruções de instalação.

Execute o *software* através do ícone **TCC.exe**.

Primeiramente aparecerá uma tela **Inicial** com algumas informações sobre o *software*, nome dos desenvolvedores e o orientador do trabalho, como pode ser visto na Figura 4.1. Esta tela fechará automaticamente, assim abrindo a tela **Principal** (Figura 4.2).

Na tela **Principal** clique no menu **Trajetória**. Aparecerá a tela **Trajetória** (Figura 4.3).

Na tela **Trajetória**, clique em **Nova** para definir uma nova trajetória. Insira os valores nos campos **px**, **py** e **pz** (posição do manipulador no espaço cartesiano) e nos campos **nx**, **ny** e **nz** (orientação do manipulador no espaço cartesiano), escolha um nome para a nova trajetória e coloque no campo **Nome da Trajetória**, na tela **Trajetória** (Figura D.1). Clique em **Next** para inserir novos pontos na trajetória, e em **Back** para ver os pontos já inseridos.

FIGURA D.1 – DEFININDO UMA NOVA TRAJETÓRIA

Depois de criada a trajetória, clique no botão **Salvar** para armazená-la no banco de dados.

Clique no botão **Excluir** caso deseje excluir do sistema a trajetória criada, ou alguma outra trajetória já armazenada no banco de dados.

Clique em **Abrir** para abrir uma nova trajetória.

Aparecerá a tela **Abrir** (Figura 4.4), então escolha na lista que aparece nesta tela uma trajetória, como pode ser visto na Figura D.2.

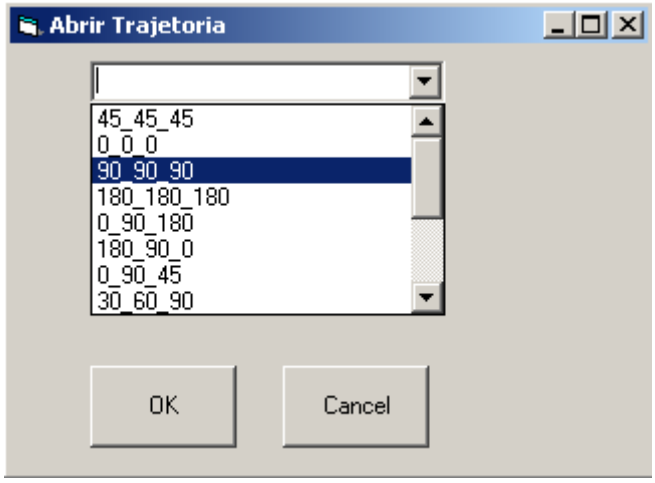


FIGURA D.2 – TELA ABRIR TRAJETÓRIA

Clique em **OK** para confirmar a seleção, ou em **Cancelar** para não aceitar a trajetória escolhida. Independente da opção escolhida, esta tela será fechada, e voltara a tela **Trajetória**.

Depois de ter uma trajetória definida, clique em **Executar Trajetória** para o manipulador poder se movimentar de acordo com os pontos definidos na trajetória.

Clique em **Fechar**, e depois em **Simular Trajetória** (Tela **Principal**, Figura 4.2) para visualizar graficamente a simulação da trajetória escolhida.

Na tela principal existem dois gráficos, um sendo o gráfico com a simulação da trajetória com vista superior do robô, e outra sendo a vista frontal do robô com o gráfico da simulação da trajetória.

Para sair do programa, clique no menu **Exit** na tela **Principal**.

Anexo A – O temporizador 8254

Anexo B – Servomotor FUTABA

http://elektra.udea.edu.co/~jfelipe/+articulos/SERVOMOTORES.doc

CREATUROIDES
Robótica Fantástica

SERVOMOTORES

(Corregí una omisión en el circuito del 555, hay que ponerle un INVERSOR a la salida)

NOTA IMPORTANTE: (Ni modo, es necesario ponerla)

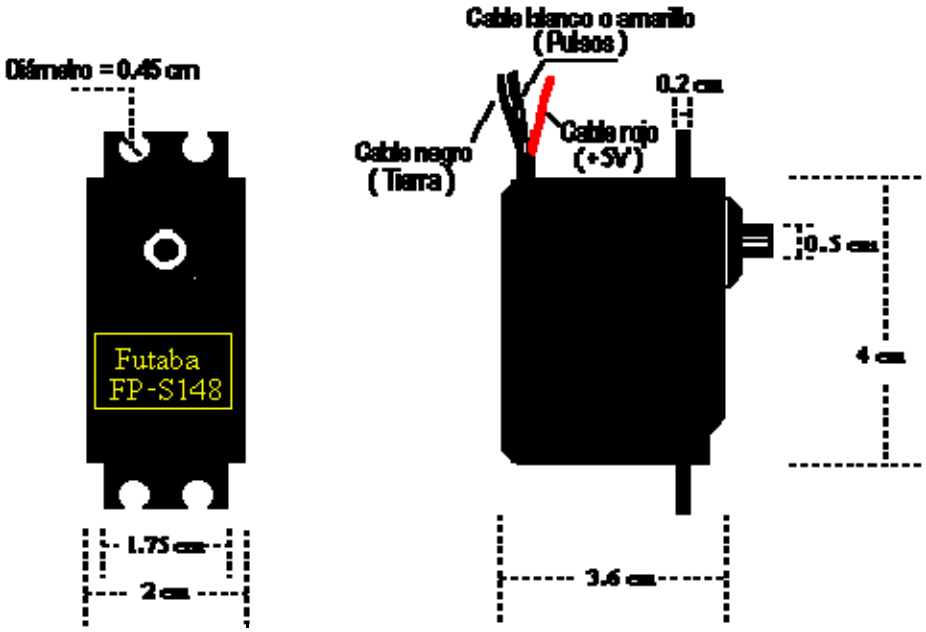
Esta información ha sido tomada de nuestras experiencias personales y se distribuye aquí con fines educativos exclusivamente. Aunque hemos tenido el mayor cuidado posible en la realización de nuestros experimentos y en la redacción de estos documentos, no descartamos que puedan existir errores o deficiencias en el mismo. Por lo cual, no nos hacemos responsables de cualquier perjuicio directo o indirecto que pueda llegar a resultar debido al uso de la información aquí contenida. Gracias.

Empecemos pues:

Este documento se centra en los **servomotores de modelismo**, que son los que se

usan en aviones, coches y vehículos a escala a control remoto. Esos son los motores que más utilizamos nosotros para nuestros robots, sin embargo recuerden que existen MUCHOS tipos más de servomotores todavía.

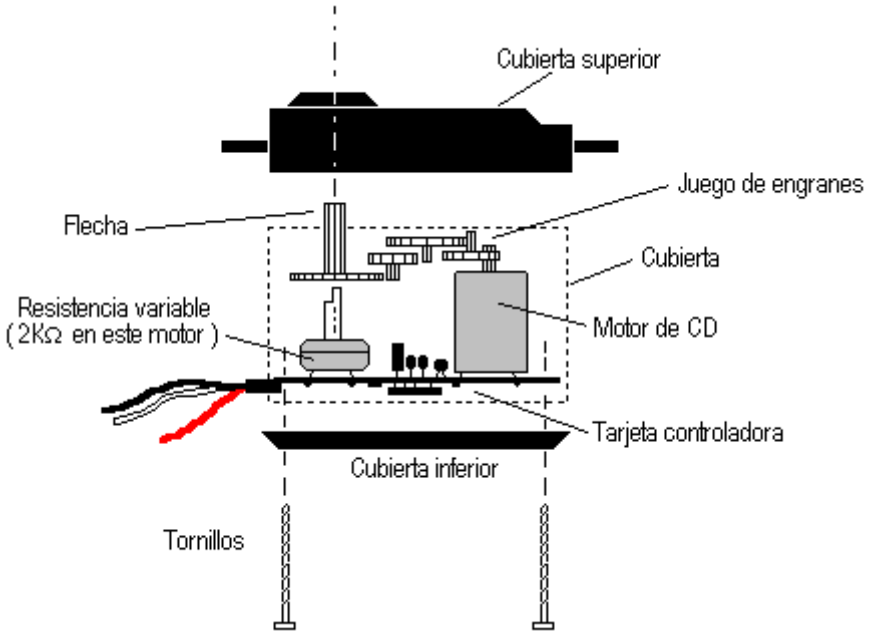
Este es el diagrama de un servomotor típico para modelismo:



Un servomotor de estos es básicamente un motor eléctrico que sólo se puede mover en un ángulo de aproximadamente 180 grados (no dan vueltas completas como los motores normales). Noten que tiene TRES cables que salen de su cajita. El rojo es de alimentación de voltaje (+5V), el negro es de tierra (0V ó GND, creo que en España se le llama "masa"). El cable blanco (a veces amarillo) es el cable por el cuál se le pide al servomotor en qué posición acomodarse (de 0 grados a 180).

Dentro del servomotor, una tarjeta controladora le dice a un pequeño motor de corriente directa cuántas vueltas girar para acomodar la **flecha** (el palito de plástico que sale al exterior) en la posición que se le ha pedido.

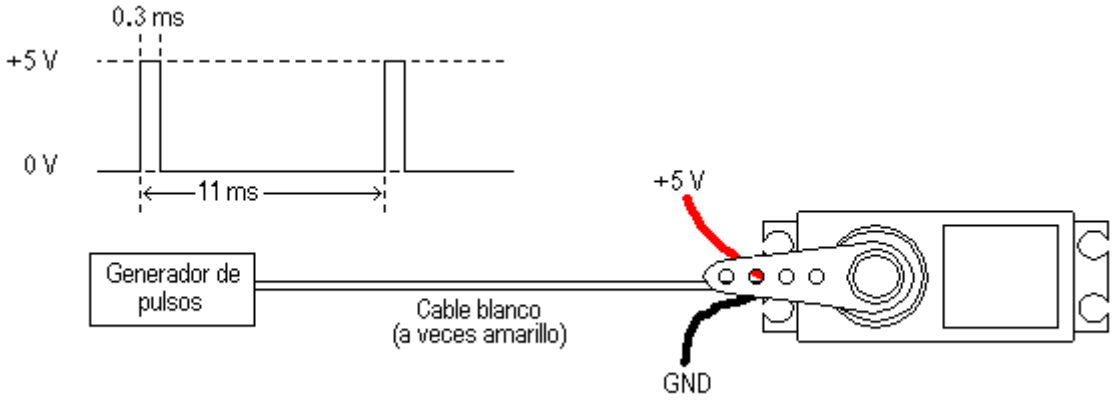
En la siguiente Figura se observa cómo están acomodadas estas piezas dentro del servomotor:



La resistencia variable (también llamada *"potenciómetro"*) está sujeta a la flecha, y mide hacia dónde está rotada en todo momento. Es así como la tarjeta controladora sabe hacia dónde mover al motorcito.

La posición deseada se le da al servomotor por medio de pulsos. Todo el tiempo debe haber una señal de pulsos presente en ese cable. Si por alguna razón necesitan tener el servomotor prendido y no pueden generarle pulsos **entonces aterricen ese cable** (conéctenlo a cero volts). Si no, se arriesgan a que la señal inducida de 60 Hz de las paredes (o de 50Hz, según en qué país vivan) haga que el servo se mueva como loco.

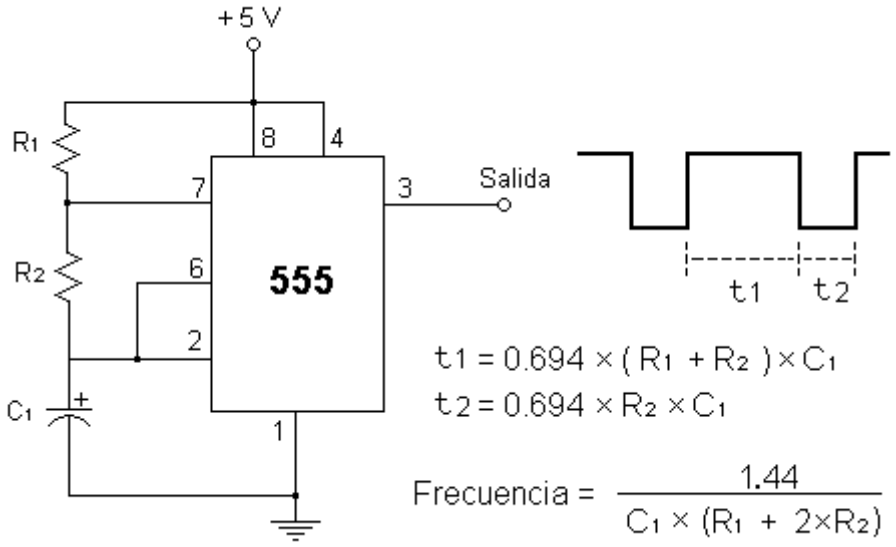
La señal de pulsos controla al servo de la siguiente forma:



Nótese que el intervalo de tiempo entre pulsos se mantiene constante, y la

variación del ancho de los mismos es lo que le indica al servo la posición que se desea. Estos valores de milisegundos nos han funcionado bastante bien para los servomotores FUTABA FP-S148, FUTABA S3003, Hitec HS-300 y HOBBICO COMMAND CS-51, y hemos encontrado también que son bastante tolerables en cuanto al período de los pulsos de control. Responden adecuadamente a pulsos desde 50 Hz hasta aproximadamente 100 Hz, pero una vez escogida una frecuencia de operación debe procurarse mantener la misma frecuencia todo el tiempo.

Para cada tipo de servo que se desee controlar, se deberá realizar una prueba preliminar para encontrar exactamente el período y la duración de los pulsos que mejor le funcionen. Un osciloscopio y un generador de señales facilitan mucho las cosas. Sin embargo, si no se cuenta con estas herramientas, se puede construir un generador de pulsos barato y sencillo con un circuito integrado 555 de la siguiente forma:



Puedes comenzar con estos valores: C1=4.7 microFarads, R1=7.5K y poner en R2 una resistencia variable de 1K. **ADEMÁS** necesitas pasar la salida de este oscilador por un **INVERSOR** antes de conectarla al servo. Un 7404 o un 7405 pueden servirte.

DETALLES IMPORTANTES CUANDO TRABAJAS CON SERVOS:

1.- Pon mucha atención en las **TIERRAS**. La tierra (cable negro) del servo tiene que ir

conectada a la tierra de su fuente de alimentación (es decir, a la salida de cero volts), **y también tiene que ir conectado a la tierra de tu microcontrolador** (o de tu computadora o de lo que sea con que lo estés controlando).

2.- Si usas cables demasiado largos para controlar tus servos, es probable que tengas ruido (tartamudeo) en los servos, esto ocurre porque mientras más largo es el cable resulta más vulnerable a ruido electromagnético (efecto antena) e incluso es perturbado por señales de otros servos. Esto se soluciona utilizando cable blindado (coaxial), solo recuerda aterrizar el blindaje.

3.- Trata de no cargarles demasiado peso a los servos. Un servo en operación normal **NO** se debe de calentar. Si se calienta es que le estás pidiendo que sostenga más peso del que es capaz (y entonces puede echarse a perder pronto). Recuerda que básicamente un servo es para **mover** algo, no para **cargarlo**. Si tu servo tiene que soportar mucho peso, rediseña tu brazo de palanca o coloca resortes (o ligas) para ayudarle.

4.- Siempre que sea posible utiliza fuentes de voltaje separadas para tus servomotores y para tu electrónica digital. Cuando controlas tus servos con una PC no tienes que preocuparte por eso (la PC ya tiene su fuente propia), pero si quieres manejar los servos con un microcontrolador es muy recomendable que tengas dos fuentes de voltaje separadas (claro, con las tierras unidas también) porque los servomotores generan bastante ruido hacia su línea de alimentación.

5.- Los servos también envejecen con el uso. Si los tratas bien pueden durarte mucho tiempo funcionando (tengo algunos que me han aguantado hasta un año y medio), pero otros se me han hecho tartamudos incorregibles después de haberlos hecho trabajar forzados durate 85 horas. Si tu servo comienza a tartamudear y estás seguro de que la causa no es ninguna de las anteriores, todavía puedes tratar de recalibrarlo. Esto significa cambiar el intervalo de tiempo entre los pulsos de control hasta encontrar el nuevo más óptimo. Otra opción es desarmarlo y limpiar el potenciómetro que tienen dentro con algun spray limpiador. Si todo falla y tu servo definitivamente ya no quiere funcionar bien, no lo tires. Todavía puedes desarmarlo y utilizar el motor con la caja de

engranes, y (a veces) puedes aprovechar incluso parte de la electrónica de control de su tarjetita para convertirlo en un motor bidireccional. Pero eso se verá más adelante en otro documento.

Anexo C – DLL inpout32

Parallel port interfacing in Win32

In the newer versions of Windows(R) I was left wondering how, if at all, I could read and write my parallel port. Even Borland had dropped support for direct I/O to the *hardware*.

But now there's inpout32.dll -- which works on Win98SE, Win2K and WinXP. This DLL provides *hardware* I/O to your computer's parallel port.

Using Borland's Free C++ Command-line Compiler, I wrote a test program which demostrates I/O to a printer port using inpout32.DLL -- Now you can do port I/O directly from a C program, and compile everything in a DOS console.

- The test program builds with Borland's Command-line C Compiler, which is free for public download from Borland.com, at http://www.borland.com/bcppbuilder/freecompiler
- Download the Inpout32 package at http://www.logix4u.net/inpout32.htm
- Download the source code for TEST.C, and you'll be up-and-running quickly.
- Also, get the book, "Parallel Port Complete," all about parallel port interfacing, and program examples that work with Inpout32.dll from Jan Axelson's Lakeview Research.

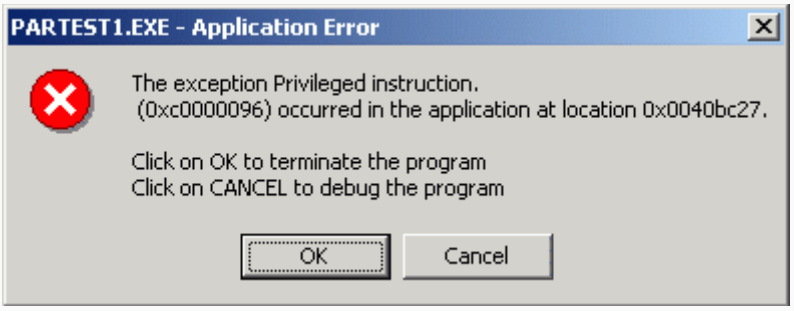
Please let me know what you discover, and if it also works with other I/O devices too (I haven't checked if it does yet...)

http://www.logix4u.net/inpout32.htm

Inpout32.dll for WIN NT/2000/XP

The Problem

Writing programs to talk with parallel port was pretty easy in old DOS days and in Win95/98 too. We can use Inporb and outportb or _inp() or _Outp functions in our program without any problem if we are running the program on Dos or WIN95/98. But entering to the new era of NT clone operating systems like WIN NT4, WIN2000, WINXP, all this simplicity goes away. Being interested in Parallel port interfacing and programming you might have experienced the problems in writing a program that can talk to parallel port successfully in NT based operating systems. When we are trying to run a program which is written using the the conventional *software* functions like Inporb, outportb, _inp() or _Outp on a WINNT or WIN2000 system, it will show an error message that "The exception privileged instruction occurred in the application at location ....". The picture of such a messagebox is given below.



Staring to this messagebox, you might have been thinking that "did i make a mistake in my program ?" it is working fine on WIN98 ... Who is guilty here. 'Nobody' that is the answer. Then why it is happening like this ..? The answer is in the next paragraph

Being a very secure operating system, Windows NT assigns some privileges and restrictions to different types of programs running on it.It classifies all the programs in to two categories , User mode and Kernel mode ie; running in ring3 and ring0 modes. user mode programs are running in ring3 mode and Kernel mode programs are running in ring0 mode. The programs you generally write falls in the user mode category. The user mode programs are restricted to use certain instructions like IN, OUT etc.. Whenever the operating system find that a user mode program is trying to execute such instructions , the operating system stops execution of those programs and will display an error message. Eventually our interfacing programs stops where they are executing IN or OUT instructions to read or write data to parallel port. But in the same time Kernel mode programs are in no way restricted in executing such instructions. Device drivers are capable of running in kernel mode. So the workaround for the above stated problem is to write a kernel mode driver capable of reading and writing data to parallel port and let the user mode program to communicate with it. Writing a driver is not an easy job for even experienced programmers. But writing a simple driver for communicating with parallel port is a simple task when drivers like USB, sound card etc.. are concerned. Even though you get a working driver from somewhere else, installing and configuring it can be very cumbersome task.

The Solution

Introducing Inpout32.dll for WIN 98/NT/2000/XP. This dll have the following features

1) Works seamless with all versions of windows (WIN 98, NT, 200 and XP)

2) Using a kernel mode driver embedded in the dll

3) No special *software* or driver installation required

4) Driver will be automatically installed and configured automatically when the dll is loaded

5) No special APIs required only two functions Inp32 and Out32

6) Can be easily used with VC++ and VB

7) Functions are compatible with Jan Axelsons Inpout32.dll (available at http://www.lvr.com/parport.htm). So this dll can be used with the sample programs available with the book Parallel Port Complete without any modification. You can find some code samples.

Click here for a sample program written for Borland C++ command line compiler, by Douglas Beattie Jr.

http://www.hytherion.com/beattidp/comput/pport/test.c

```
/*************************************************/
/***                                           ***/
/*** TEST.c  -- test interface to inpout32.dll  ***/
/***  ( http://www.logix4u.net/inpout32.htm )   ***/
/***                                           ***/
/*** Copyright (C) 2003, Douglas Beattie Jr.    ***/
/***                                           ***/
/***    <beattidp@ieee.org>                    ***/
/***    http://www.hytherion.com/beattidp/     ***/
/***                                           ***/
/*************************************************/


/*******************************************************/
/*                                                     */
/*  Builds with Borland's Command-line C Compiler      */
/*    (free for public download from Borland.com, at   */
/*  http://www.borland.com/bcppbuilder/freecompiler )  */
/*                                                     */
/*   Compile with:                                     */
/*                                                     */
/*   BCC32 -IC:\BORLAND\BCC55\INCLUDE  TEST.C          */
/*                                                     */
/*                                                     */
/*  Be sure to change the Port addresses               */
/*  accordingly if your LPT port is addressed          */
/*  elsewhere.                                         */
/*                                                     */
/*******************************************************/



#include <stdio.h>
#include <conio.h>
#include <windows.h>


/* Definitions in the build of inpout32.dll are:           */
/*   short _stdcall Inp32(short PortAddress);              */
/*   void _stdcall Out32(short PortAddress, short data);   */


/* prototype (function typedef) for DLL function Inp32: */

     typedef short _stdcall (*inpfuncPtr)(short portaddr);
     typedef void _stdcall (*oupfuncPtr)(short portaddr, short datum);

int main(void)
{
     HINSTANCE hLib;
     inpfuncPtr inp32;
     oupfuncPtr oup32;

     short x;
     int i;

     /* Load the library */
     hLib = LoadLibrary("inpout32.dll");

     if (hLib == NULL) {
          printf("LoadLibrary Failed.\n");
          return -1;
```

```
     }

     /* get the address of the function */

     inp32 = (inpfuncPtr) GetProcAddress(hLib, "Inp32");

     if (inp32 == NULL) {
          printf("GetProcAddress for Inp32 Failed.\n");
          return -1;
     }


     oup32 = (oupfuncPtr) GetProcAddress(hLib, "Out32");

     if (oup32 == NULL) {
          printf("GetProcAddress for Oup32 Failed.\n");
          return -1;
     }


/*************************************************************/
/* now test the functions */

     /* Try to read 0x378..0x37F, LPT1:  */

     for (i=0x378; (i<0x380); i++) {

          x = (inp32)(i);

          printf("port read (%04X)= %04X\n",i,x);
     }



     /*****  Write the data register */

     i=0x378;
     x=0x77;

     (oup32)(i,x);

     printf("port write to 0x%X, datum=0x%2X\n" ,i ,x);

     /***** And read back to verify  */
     x = (inp32)(i);
     printf("port read (%04X)= %04X\n",i,x);



     /*****  One more time, different value */

     i=0x378;
     x=0xAA;

     (oup32)(i,x);

     printf("port write to 0x%X, datum=0x%2X\n" ,i ,x);

     /***** And read back to verify  */
     x = (inp32)(i);
     printf("port read (%04X)= %04X\n",i,x);
```

```
    FreeLibrary(hLib);
    return 0;
}
```

Anexo D – DataSheet 82C54