10 (dez)

HUGO LEONARDO PAIVA RODRIGUES
MARCOS ALEXANDRE SCHOLTZ

PROJETO E CONSTRUÇÃO DE UM ROBÔ MANIPULADOR DE ENDOSCÓPIO CIRÚRGICO

Trabalho de formatura

~~Dissertação~~ apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro

São Paulo
2001

HUGO LEONARDO PAIVA RODRIGUES
MARCOS ALEXANDRE SCHOLTZ

PROJETO E CONSTRUÇÃO DE UM ROBÔ MANIPULADOR DE ENDOSCÓPIO CIRÚRGICO

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro

Área de Concentração :
Engenharia Mecânica – Automação e Sistemas

Orientador :
Prof. Dr. Lucas Moscato

São Paulo
2001

AGRADECIMENTOS

Ao orientador Prof. Dr. Lucas Moscato, pelo amplo apoio prestado em todos os momentos do trabalho.

A Walter de Britto, pelo seu contínuo envolvimento, suporte e orientação, de importância imprescindível para a realização deste trabalho.

A Fidel Vicente de Paula e Gilberto Garcia, por sua constante disposição em auxiliar com a usinagem de peças e componentes.

A Antônio Fernando Maiorquim, pelo seu inestimável auxílio nos projetos eletrônicos contidos neste trabalho.

A todos aqueles que, direta ou indiretamente, colaboraram para a execução deste trabalho.

RESUMO

O presente trabalho reúne o projeto completo de um robô manipulador de endoscópio cirúrgico, desde a concepção e análise do problema a ser resolvido, a apresentação de diferentes soluções, o desenvolvimento da solução escolhida até a execução da solução em si. O problema consiste em se criar uma forma automatizada, que possa futuramente ser utilizada à distância (via Internet), para a manipulação de um endoscópio cirúrgico a ser utilizado em cirurgias torácicas. A solução adotada consiste em um mecanismo com três graus de liberdade, composto por um carro que se movimenta sobre um trilho curvo giratório. O acionamento é realizado por três motores de passo, controlados via PC através de um sistema de mouse de cabeça (que detecta inclinações da cabeça do médico) com confirmação de comandos por pedais, oferecendo segurança à entrada de dados e livrando as mãos do médico para a operação do módulo principal do robô. O desenvolvimento do software se deu em linguagem C operando em sistema operacional DOS. O módulo manipulador de endoscópio é parte de um projeto mais amplo de um robô cirúrgico completo para cirurgias torácicas.

ABSTRACT

The present report assembles the complete project of a surgical endoscope manipulating robot, starting at the conception and analysis of the problem to be solved, the presentation of different solutions, the development of the chosen solution and finally it's execution. The problem consists in creating an automated form, which may be used at great distances (through the Internet) in the future, of manipulating a surgical endoscope to be used in thoracic surgeries. The adopted solution consists in a mechanism with three degrees of freedom, which consists of a rolling car mounted on a round, rotatory track. The motion is provided by three stepper motors controlled by a PC through a head mouse system (which detects the surgeon's head inclinations) with commands confirmation being provided by pedals, thus offering security to data input and freeing the surgeon's hands to operate the robot's main module. The software was developed using C language operating under DOS operational system. The endoscope manipulator is part of a wider project of a complete surgical robot to be used in thoracic surgeries.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

1. INTRODUÇÃO **1**

2. REVISÃO DA LITERATURA **2**

3. ESTUDO DE VIABILIDADE **6**

3.1 – ESTABELECIMENTO DA NECESSIDADE 7
3.2 – ESPECIFICAÇÃO TÉCNICA DA NECESSIDADE 9
3.3 – DESENVOLVIMENTO DE ALTERNATIVAS 11
3.3.1 – Mecanismos para a Movimentação do Sistema *11*
3.3.2 – Motores para o Acionamento do Mecanismo *20*
3.3.3 – Linguagens de programação *24*
3.3.4 – Método de controle do mecanismo (Interface homem-máquina) *26*

4. SELEÇÃO DA MELHOR SOLUÇÃO **38**

4.1 - MECANISMOS PARA A MOVIMENTAÇÃO DO SISTEMA 39
4.2 - MOTORES DE ACIONAMENTO 41
4.3 - LINGUAGEM DE PROGRAMAÇÃO 42
4.4 - MÉTODO DE CONTROLE DO MECANISMO (INTERFACE HOMEM-MÁQUINA) 43

5. PROJETO DO MECANISMO **45**

5.1 - DEFINIÇÕES 46
5.1.1 - Trilhos *46*
5.1.2 - Espaçamento *47*
5.1.3 - Carro *49*
5.1.4 - Fixação ao Eixo *52*
5.1.5 - Caixa de Motores *53*
5.1.6 - Fixação à Mesa *54*
5.2 - ANÁLISE ESTÁTICA 55
5.2.1 - Flexão *55*
5.2.2 - Torção *58*
5.3 - ANÁLISE DINÂMICA 61
5.3.1 - Variação do Comprimento do Cabo *61*
5.4 - DESENHOS DE CONJUNTO E FABRICAÇÃO 64

6. SELEÇÃO DE MOTORES **65**

6.1 – PRIMEIRO MOTOR : ROTAÇÃO DA ESTRUTURA 69
6.2 - SEGUNDO MOTOR : MOVIMENTO DE *ZOOM* 72
6.3 – TERCEIRO MOTOR : MOVIMENTO DO CARRO SOBRE O TRILHO 74
6.4 – CÁLCULO DA CORREIA PARA REDUÇÃO (MOTOR PRINCIPAL) 77

7. PROJETO DO DRIVER DE MOTOR DE PASSO **79**

7.1 – ACIONAMENTO DO MOTOR 80
7.1.1 – Lógica de Acionamento *81*
7.1.2 – Circuito de Potência *83*
7.1.3 – Driver Final de Acionamento de Motor de Passo *85*
7.2 – COMUNICAÇÃO COM O MICROCOMPUTADOR 87

8. PROJETO DO DISPOSITIVO DE ENTRADA DE DADOS 92

8.1 – MOUSE DE CABEÇA 93
8.1.1 – Modelo do Mouse de Cabeça *94*
8.1.2 - Análise de Sistemas de 2° Ordem *96*
8.1.3 – Determinação das Dimensões do Pêndulo *99*
8.2 – SISTEMA DE PEDAIS 104

9. DESENVOLVIMENTO DO SOFTWARE DE CONTROLE 108

9.1 – ESTRUTURAÇÃO 109
9.2 – INTERFACE 112
9.3 – COMUNICAÇÃO COM O MOUSE DE CABEÇA 113
9.4 – COMUNICAÇÃO COM O DRIVER DOS MOTORES DE PASSO 114
9.4.1 – Desenvolvimento de Timer com Alta Precisão *115*
9.4.2 – Composição dos Sinais de Saída *120*
9.5 - PROGRAMAÇÃO DOS MÓDULOS 121

10. CONCLUSÕES 126

11. REFERÊNCIAS BIBLIOGRÁFICAS 128

ANEXO I – DESENHOS DE CONJUNTO

ANEXO II – DESENHOS DE FABRICAÇÃO

ANEXO III – LISTAGEM DOS ARQUIVOS FONTE

APÊNDICE I – FOTOS DOS COMPONENTES E SISTEMAS DO ROBÔ

LISTA DE FIGURAS

Figura 1 – Exemplos de alguns endoscópios comerciais 3
Figura 2 – Campo de movimentação de um endoscópio 9
Figura 3 – Mecanismo de barras articuladas 14
Figura 4 – Mecanismo de trilhos curvos 16
Figura 5 – Mecanismo de junta esférica multi-link 20
Figura 6 – Alguns exemplos de motores DC 22
Figura 7 – Exemplo de um servomotor 22
Figura 8 – Alguns exemplos de motores de passo 23
Figura 9 – Controle por mouse convencional 28
Figura 10 – Controle por Joystick 30
Figura 11 – Controle por comando de voz 32
Figura 12 – Controle por sistema de pedais 34
Figura 13 – Controle por mouse de cabeça associado a pedais 37
Figura 14 - Esquema construtivo do mouse de cabeça 37
Figura 15 - Modelo inicial da peça do trilho 47
Figura 16 - Separadores cilíndricos para os trilhos 48
Figura 17 - Separação final dos trilhos, com polia para o cabo 48
Figura 18 - Diagrama esquemático dos roletes do carro 49
Figura 19 - Sistema de fixação e movimentação vertical do endoscópio 50
Figura 20 - Sistema de molas de fixação dos cabos ao carro 51
Figura 21 - Retorno do cabo através do carro (parte inferior) 51
Figura 22 - Peça de fixação dos trilhos ao eixo 52
Figura 23 - Modelo da caixa dos motores aberta 54
Figura 24 - Modelo para o cálculo da deformação por flexão 56
Figura 25 - Vista esquemática frontal para o cálculo da torção 58
Figura 26 - Modelo para o cálculo do comprimento do cabo 62
Figura 27 – Exemplo de curvas de um motor de passo 67
Figura 28 – Vista do manipulador do endoscópio completo 68
Figura 29 – Posição crítica para o primeiro motor 69

Figura 30 – Mecanismo de molas responsável pelo zoom 72
Figura 31 – Movimento acionado pelo motor 3 74
Figura 32 – Detalhe da ranhura no carro para passagem de cabo 75
Figura 33 – Ilustração do acoplamento por correia 77
Figura 34 – Esquema simplificado de motor de passo unipolar 80
Figura 35 – Acionamento utilizando o UCN4202A 82
Figura 36 – Circuito de potência para uma bobina do motor 83
Figura 37 – Projeto final de driver de motor de passo 86
Figura 38 – Ligação comum com optoacoplador 87
Figura 39 – Ligação na saída de buffer 'open collector' 88
Figura 40 – Projeto final de comunicação com porta paralela 90
Figura 41 – Modelo do mecanismo do mouse de cabeça construído 93
Figura 42 – Modelo físico do pêndulo do mouse 94
Figura 43 – Resposta a uma entrada degrau de um sistema de 2ª ordem 97
Figura 44 – Resposta a uma entrada impulso de um sistema de 2ª ordem 98
Figura 45 – Resolução angular do encoder do mouse 101
Figura 46 – Resposta a uma entrada degrau unitário 103
Figura 47 – Configuração de um pedal individual 104
Figura 48 – Suporte para terceiro botão 105
Figura 49 – Disposição em ângulo dos pedais 106
Figura 50 – Sistema completo dos pedais 107
Figura 51 – Estrutura do software de controle 111
Figura 52 – Envio de trem de pulsos pelo software 115
Figura 53 – Saída do programa de testes do timer 118
Figura 54 – Exemplo de tela – Tela principal 121
Figura 55 – Exemplo de tela – Calibração do mouse de cabeça 122
Figura 56 – Exemplo de tela – Configuração dos parâmetros do robô 123
Figura 57 – Exemplo de tela – Operação (indicação de opções de movimento) ... 124
Figura 58 – Exemplo de tela – Operação (efetuando movimentação) 124

LISTA DE TABELAS

Tabela I – Matriz de decisão para mecanismos 39
Tabela II – Matriz de decisão para motores 41
Tabela III – Matriz de decisão para linguagens de programação 42
Tabela IV – Matriz de decisão para método de controle 43
Tabela V – Cálculos de balanceamento da estrutura 70
Tabela VI – Tabela com dados ensaiados para o pêndulo 99
Tabela VII – Freqüências de sinal possíveis para T_{timer} = 1 ms 116
Tabela VIII – Freqüências de sinal possíveis para T_{timer} = 0,25 ms 117

LISTA DE SÍMBOLOS

δ	Deformação vertical sofrida pelos trilhos em balanço
$M_{(x)}$	Momento fletor em função da coordenada de comprimento
E	Módulo de elasticidade
I_z	Momento de inércia em torno do eixo Z
b_s	Base da secção transversal de um trilho
h	Altura da secção transversal de um trilho
T_{or}	Momento de torção
P	Força peso
ϕ	Deformação angular por unidade de comprimento
γ	Deformação angular
G	Módulo de elasticidade torcional
ψ	Ângulo de posição do carro sobre os trilhos em relação à horizontal
ψ_0	Ângulo de posição do carro em relação à horizontal nas posições limite
φ	Ângulo determinado pelo comprimento do carro e o centro dos trilhos
D	Comprimento da seção de cabo passante no carro
R	Raio da seção curva dos trilhos
h_0	Altura do sistema em relação ao paciente
L_1	Comprimento do cabo na seção anterior dos trilhos
L_2	Comprimento do cabo na seção posterior dos trilhos
D_{sup}	Diferença de comprimento no cabo na parte superior
D_{inf}	Diferença de comprimento no cabo na parte inferior
K	Constante de mola (força aplicada por unidade de deformação)
τ	Torque aplicado
F	Carga nominal de resistência
d	Diâmetro da polia
D_p	Diâmetro primitivo do pinhão
D_c	Diâmetro primitivo da coroa
n	Número de dentes do pinhão
N	Número de dentes da coroa

p	Passo dos dentes da polia
A	Distância nominal entre centros das polias da transmissão
A'	Distância calculada entre centros das polias da transmissão
L	Comprimento da correia de transmissão
V_{cc}	Tensão de alimentação do circuito de controle (+5V)
$V_{Alim.}$	Tensão de alimentação das bobinas do motor (circuito de potência)
R_L	Valor da resistência do resistor limitador de corrente
I_b	Corrente de acionamento na base do transistor de potência
U	Tensão elétrica aplicada
I	Corrente elétrica percorrendo o componente
Dir	Sinal binário que indica a direção a ser seguida pelo motor de passo
Stp	Sinal binário contendo o trem de pulsos de acionamento do motor de passo
T	Energia cinética do sistema
V	Energia potencial do sistema
Q	Forças de natureza não conservativa
θ	Ângulo de inclinação do pêndulo em relação à vertical
l	Comprimento do pêndulo
m	Massa concentrada na ponta do pêndulo
g	Aceleração da gravidade
ω_n	Freqüência natural do sistema de segunda ordem
ω_d	Freqüência natural amortecida do sistema de segunda ordem
ζ	Coeficiente de amortecimento do sistema de segunda ordem
b	Constante de amortecimento angular do pêndulo
t_r	Tempo de subida do sinal em sistema de segunda ordem
T_d	Período do sinal amortecido no sistema de segunda ordem
M_p	Sobre-sinal máximo do sistema de segunda ordem
α	Deslocamento angular do encoder do mouse
$T_{mín}$	Menor período de sinal possível de implementar com o timer
T_{timer}	Período do timer de software (tempo para um ciclo)
$F_{máx}$	Freqüência máxima do sinal de alimentação

1. INTRODUÇÃO

A automação das atividades humanas é um processo que está cada vez mais inserido e presente no cotidiano das pessoas. Desde uma simples máquina de café até complexas máquinas de usinagem CNC, a automação de processos vem conhecendo um crescente incremento, a ponto de existirem atividades que antes eram impossíveis de serem desenvolvidas sem o auxílio de processos automáticos de uma forma produtiva e eficiente como hoje são conhecidas. Processos de inspeção metrológica constituem um bom exemplo do avanço proporcionado pela automação de processos.

A área da bioengenharia é bastante promissora. Além de conhecer um espantoso crescimento durante os últimos anos, a utilização de robôs, seja para a obtenção de diagnósticos, seja para a realização de intervenções cirúrgicas, é algo muito novo e que ainda encontra-se em franco desenvolvimento.

No caso de intervenções cirúrgicas, não existe ainda um sistema desenvolvido capaz de realizar todo um procedimento cirúrgico sem nenhuma intervenção humana. O desenvolvimento de novos acionamentos e novos processos de aquisição e processamento de dados e a utilização de inteligência artificial juntamente com o desenvolvimento da microeletrônica promete dar condições para o surgimento, num futuro muito próximo, de máquinas capazes de realizar complexos procedimentos cirúrgicos de forma praticamente independente da ação humana.

O presente projeto apresenta o desenvolvimento de um robô manipulador de endoscópio para a realização de cirurgias torácicas minimamente invasivas. Uma vez identificada a necessidade de desenvolver-se um sistema capaz de posicionar uma câmera durante uma intervenção cirúrgica de uma forma que comprometa minimamente a atuação do médico durante a operação, este projeto mostra uma possível solução para tal questão. É válido ressaltar que este robô é um dispositivo que deverá atuar em conjunto com um outro robô, desenvolvido no mesmo projeto, capaz de realizar cirurgias comandado por um médico. O projeto como um todo representa um primeiro passo rumo à automação total dos processos cirúrgicos, os quais, num futuro não muito distante, poderão ser realizados sem a intervenção do cirurgião.

2. REVISÃO DA LITERATURA

Existem vários tipos de endoscópios disponíveis comercialmente, para diversas finalidades diferentes, desde endoscópios para manutenção de grandes máquinas operatrizes até endoscópios para cirurgias veterinárias.

Em relação aos equipamentos de uso humano, os endoscópios são divididos conforme o tipo de cirurgia ou diagnósticos para que foram destinados. Podem ser citadas como ramos da medicina onde o endoscópio é bastante utilizado ou quase indispensável:
urologia, gastroenterologia, cirurgia geral, ginecologia e obstetrícia, diagnósticos torácicos, dentre outros.

Existem modelos que oferecem altíssima resolução, com processadores digitais de imagem que destacam e realçam detalhes em tempos de processamento razoavelmente curtos.

Deseja-se sempre um endoscópio de pequeno diâmetro, estável e que apresente boa flexibilidade, sem no entanto perder resolução das imagens. A captura das imagens deve ser bastante eficiente, pois o campo de visão da lente do equipamento é limitado. O que se faz comercialmente é utilizar uma lente esférica, mais indicada para captar imagens de regiões mais amplas. Os modelos mais modernos incluem ainda facilidades como desembaçador da lente esférica, eliminando os problemas de visualização dentro do paciente.

Quanto às dimensões, os diâmetros mais comuns são de 5mm e 10 mm, dependendo da aplicação. Os comprimentos e a rigidez também podem variar conforme a finalidade do equipamento. Em geral, têm-se endoscópios que medem de 800mm até 1700mm. Os maiores endoscópios são os que apresentam maiores dificuldades de posicionamento, pois os maiores endoscópios são também os mais flexíveis, dificultando a correta montagem do *site* cirúrgico pelo médico.

Figura 1 – Exemplos de alguns endoscópios comerciais

Comercialmente, existem algumas empresas que fornecem mecanismos para o posicionamento de endoscópio para cirurgias. Em grande parte dos casos, os posicionadores são manuais e de difícil operação, o que representa um empecilho na atuação do cirurgião.

Um sistema manipulador de câmera endoscópica comercialmente encontrado é o EndoSista, desenvolvido e construído em 1992 pela Armstrong Projects Ltd. Este sistema é formado por um manipulador de cabeça, uma unidade de controle e um subsistema de sensoreamento. A primeira versão tinha três graus de liberdade , correspondentes aos movimentos de *tilt*, *pan*, além do *zoom*. Conseguia-se um ângulo de operação de aproximadamente um quarto de esfera. A versão atualmente comercializada, após diversas melhoria, apresenta quatro graus de liberdade, um dos quais já permite a rotação do equipamento em torno de seu próprio eixo. A grande vantagem deste sistema é que o médico consegue posicionar o endoscópio sem a utilização das mão, apenas com movimentos de cabeça, captados por um dispositivo inercial.

Existem outras montagens, como da Andronic Devices que só conseguem prender o osciloscópio, sem dotá-lo de nenhum movimento, ou ainda a montagem da AESOP Comercial System by Computer Motion Inc., que consegue movimentar-se para posições pré-definidas pelo usuário.

Muitos estudos vêm sendo desenvolvidos na área com o objetivo de se desenvolver um equipamento de fácil operação e que possa ser telecomandado pelo médico, de preferência sem a utilização das mãos.

Novos mecanismos, como o apresentado por Faraz, A.; Payandeh, S. [1], tentam otimizar os ângulos de movimentação do osciloscópio através de estruturas com bastante mobilidade. Nestes casos, a rigidez e a mobilidade são geralmente fatores conflitantes que devem ser corretamente gerenciados pelo projetista, de modo que não sujam equipamentos com pouca rigidez mecânica ou pouca estabilidade no momento da cirurgia.

No Japão, mais precisamente na Universidade de Tóquio, foi desenvolvido um sistema diferente de manipulação de endoscópio cirúrgico. O trabalho pode ser melhor visto na publicação [4].

O sistema consiste em um mecanismo composto por cinco barras articuladas. Em balanço entre elas há um ponto pivô que contém o duto do endoscópio. Nas extremidades há dois motores DC, com encoders acoplados, que fornecem movimento às barras. Combinando-se os movimentos de ambos os motores pode-se obter o movimento no plano do ponto pivô e, com isso, a rotação do endoscópio em torno do ponto de inserção.

A preocupação principal que levou os japoneses a este sistema foi a segurança e a facilidade de esterilização. Os sistemas elétricos ficam separados, e com isso é fácil isolá-los durante a cirurgia e durante a esterilização do mecanismo.

Para o controle do zoom eles não utilizaram o movimento do endoscópio em si, mas uma câmera de vídeo no endoscópio com um zoom óptico, ou seja, o zoom é feito pela movimentação das lentes da câmera.

Para a interface homem-máquina foi utilizado um sistema de "mouse de cabeça", que consiste em um sensor detector de inclinações na cabeça do cirurgião (modelo comercial) ligado ao microcomputador. A movimentação da cabeça do médico é a responsável pela movimentação do mecanismo em si.

Foi realizado um experimento no Hospital Metropolitano de Tóquio em que um porco foi submetido a uma cirurgia utilizando-se o manipulador de endoscópio. Os resultados foram satisfatórios, e os tempos dos procedimentos cirúrgicos foram

menores do que os normais médios. Foi detectada vibração do mecanismo durante os movimentos, no entanto.

3. ESTUDO DE VIABILIDADE

Antes de se poder iniciar qualquer projeto ou desenvolvimento, é necessário se conhecer a fundo o problema a ser resolvido. Para isso se faz o estabelecimento da necessidade, e sua conseqüente análise e quantificação.

Após se ter avaliado de forma completa o problema, passa-se então ao desenvolvimento de soluções. Soluções devem ser geradas e as melhores devem ter um pré-desenvolvimento, tratando-se de um desenvolvimento simplificado que visa a obtenção de parâmetros que permitam sua comparação com outras alternativas e a análise de sua viabilidade.

Neste capítulo será realizada a análise do problema e serão também colocadas as melhores soluções.

3.1 – Estabelecimento da Necessidade

A cirurgia com endoscópio é um tipo de cirurgia que vêm se tornando muito popular pelo seu caráter pouco invasivo. A anatomia interna é observada por um endoscópio, que é dotado de uma câmera de vídeo de pequenas dimensões, e através das imagens o médico pode operar utilizando instrumentos inseridos no corpo do paciente através de uma cânula.

No entanto, o cirurgião fica geralmente limitado ao campo de visão limitado do endoscópio. Adicionalmente, como ele necessita também manipular os instrumentos, é necessário que haja um auxiliar para manipular o endoscópio, e em geral a movimentação do mesmo é precária e é difícil mantê-lo estável e apontado para onde deseja o cirurgião.

Um mecanismo robótico para a manipulação do endoscópio resolveria estes problemas. No entanto, em se tratando de um sistema a ser utilizado em cirurgias, há alguns aspectos que devem ser levados em consideração.

Um deles, e talvez o principal, é a segurança. Em se tratando de uma cirurgia em um paciente humano, nenhum erro ou falha pode ser admitido. Por isso, deve-se considerar aspectos como a segurança da própria estrutura, do funcionamento correto dos acionadores e limitações físicas movimento. Além disso, deve-se garantir a segurança do software de controle.

Para isso, um dos principais pontos do projeto é buscar um manipulador simples, com um software de controle simples. Também a interface homem-máquina deve ser a mais simples e intuitiva possível, e deve buscar eliminar entradas indesejadas.

Sabe-se que o endoscópio é introduzido no paciente através de uma pequena incisão. Assim, a movimentação é limitada a três graus de liberdade. O endoscópio pode rotacionar livremente em torno do ponto de penetração, e pode ser transladado mais para dentro ou para fora do corpo do paciente, proporcionando um efeito de zoom. Um maior número de graus de liberdade pode vir a causar o risco de movimentos inesperados, logo três graus de liberdade deveriam ser suficientes.

Outro ponto a ser considerado é a necessidade de se remover o endoscópio com facilidade. Durante o procedimento cirúrgico, muitas vezes o endoscópio pode

sujar-se com sangue ou outros fluidos e deve ser limpo. Assim, deve-se permitir a fácil desmontagem da peça do endoscópio.

Além disso, após uma cirurgia todos os instrumentos e equipamentos utilizados devem ser esterilizados. No entanto, componentes elétricos perdem grande parte de sua vida útil com este processo. Por isso, deve-se buscar uma separação dos componentes mecânicos dos componentes elétricos (que ficariam envoltos por alguma proteção durante a esterilização).

Sendo assim, as necessidades principais percebidas podem ser listadas como:

- Proporcionar uma forma automática de manipulação de um endoscópio;
- A movimentação deve ser estável e abrangente;
- Todo o sistema deve ser seguro para o paciente;
- O sistema deve evitar entradas indesejadas (movimentos bruscos);
- A utilização pelo cirurgião deve ser simples e evitar o uso das mãos;
- O sistema deve ser confiável;
- Deve-se buscar o menor número de graus de liberdade possível (apenas três são necessários);
- As peças componentes do endoscópio deve ser removível para limpeza e esterilização;
- Deve-se buscar separação das partes elétricas e mecânicas, a fim de facilitar a esterilização;

O presente trabalho visa o estudo de algumas soluções capazes de satisfazer as necessidades acima levantadas do modo mais completo e confiável possível. É bom lembrar que as soluções de um problema de um modo geral representam sempre um compromisso entre fatores conflitantes, o que leva a adoção de um compromisso entre os aspectos mais críticos levantados.

3.2 – Especificação Técnica da Necessidade

Quando se fala em controle de um osciloscópio, deseja-se o maior ângulo de visão possível, visto que cirurgias minimamente invasivas exigem uma grande perícia do profissional devido ao limitado campo de visão oferecido pelos endoscópios, bem como pela dificuldade de se manipular instrumentos cirúrgicos por cânulas. Além disso, não pode ser deixado de lado o fato de que o cirurgião não pode utilizar as mão para o posicionamento da câmera.

Os instrumentos utilizados para a visualização do paciente são geralmente dotados de lentes esféricas que propiciam um maior campo visual. Assim, com um pequeno ângulo de movimentação consegue-se varrer de forma satisfatória todo o interior do paciente.

Através de estudos realizados junto a médicos que já utilizam o sistema, conclui-se que o ideal para se obter uma adequada visualização é a variação do ângulo de visão do equipamento por uma região correspondente a um ângulo sólido que vai de aproximadamente 60° até 140° . Um ângulo sólido corresponde à movimentação do endoscópio em todas as direções possíveis, a partir da posição perpendicular ao paciente, de um valor correspondente à metade do ângulo sólido.

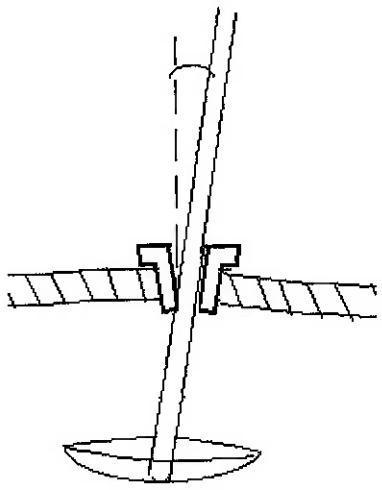


Figura 2 – Campo de movimentação de um endoscópio

Adicionalmente, o médico necessita de um movimento de zoom que permita aproximar e afastar a câmera do local que está sofrendo a intervenção cirúrgica. Este

movimento representa um movimento perpendicular ao paciente e não necessita ser muito grande, até porque existem limitações devido aos órgão internos. Foi levantado que um adequado valor para o zoom seria de aproximadamente 100mm.

3.3 – Desenvolvimento de Alternativas

Como foi constatado por experiências anteriores, é pouco eficiente um sistema posicionador que não possa ser controlado pelo médico ou por algum de seus assistentes de uma forma eficaz e prática. Assim, as alternativas foram desenvolvidas pensando-se em três aspectos principais do desenvolvimento das soluções: o tipo de acionador (motor) a ser utilizado, o mecanismo a ser acionado pelo atuador, o método utilizado para o controle do mecanismo e o *software* que implementa o controle. Estes itens foram considerados os mais cruciais e decisivos na elaboração de um eficiente sistema de visualização através de um endoscópio. Abaixo, serão discutidas alternativas para cada um destes elementos em particular.

3.3.1 – Mecanismos para a Movimentação do Sistema

Em se tratando de um mecanismo a ser utilizado para a movimentação de um endoscópio cirúrgico, alguns pontos importantes devem ser observados.

- Deve-se dar ênfase à segurança do equipamento, pois trata-se de um paciente humano. Assim, limites mecânicos devem ser impostos ao movimento. O caso de falha dos motores deve ser previsto.
- O mecanismo deve ser estável, evitando vibração ou deflexões exageradas.
- Um endoscópio comercial deve ser de fácil acoplamento ao mecanismo. Além disso, a peça que contém o endoscópio deve ser facilmente removível para limpeza.
- Para evitar movimentos indesejados, o mecanismo deve se mover apenas com o número de graus de liberdade necessários.
- Deve-se buscar uma separação dos motores e controle elétricos dos componentes mecânicos do sistema, a fim de facilitar a esterilização.

Algumas propostas foram apresentadas, tendo sido as melhores as três selecionadas abaixo.

A) Mecanismo de Barras Articuladas

- Descrição :

Neste mecanismo, a idéia é promover a movimentação de rotação do endoscópio através da movimentação em um plano de um ponto pivô. Esta movimentação no plano se dá através de quatro barras, ligadas e articuladas, e dois motores. Considerando o ponto de inserção fixo, movendo-se o ponto pivô em um plano faz com que o endoscópio gire em torno do ponto de inserção.

O zoom deve ser obtido através de um terceiro motor que ocasiona o movimento de translação do tubo que contém o endoscópio. Assim, o mesmo pode penetrar ou sair do paciente.

- Acionamento :

Os dois motores primários fazem com que o ponto pivô se mova em um plano. Sendo os pontos C, D e E articulados, com o movimento combinado de ambos os motores é possível mover-se em qualquer direção. Por exemplo, movendo o motor da esquerda no sentido horário e o da direita no sentido anti-horário força-se o ponto pivô (E) para cima.

Colocando-se limites ao tamanho das barras e limitando-se a movimentação das articulações pode-se ter a limitação mecânica do movimento.

O terceiro motor, responsável pelo zoom, é ligado simplesmente a um cabo na ponta mais interior do tubo que contém o endoscópio. Ao rotacionar em um certo sentido, ele puxa o cabo forçando o endoscópio para dentro do paciente. A mola de compensação puxa o endoscópio de volta, e é a favor da segurança, caso haja falha no motor ou rompimento do cabo. Imaginou-se a opção de envolver este cabo em um tubo plástico (como nos freios de bicicleta).

- Vantagens e Desvantagens :

O sistema é razoavelmente simples, e construtivamente trata-se apenas de barras e articulações. Outra grande vantagem é a separação completa dos motores e partes elétricas do mecanismo mecânico em si. Pode-se citar também o fato de que o

sistema como um todo permanece a uma distância razoavelmente grande do paciente, não sobrecarregando a área da cirurgia com equipamentos.

Há uma grave falha, no entanto. Supõe-se que o ponto de inserção seja fixo, ou seja, que o endoscópio não pode transladar no plano do corpo do paciente. Isto é uma verdade, visto que o endoscópio é introduzido através de um orifício, mas implica em esforços sobre o corpo do paciente. Estes esforços são extremamente indesejáveis.

Pela leveza e simplicidade do mecanismo pode-se concluir que estes esforços no paciente não inviabilizam o projeto. Há um agravante, no entanto, que é o movimento de zoom. Quando se dá o zoom, um esforço é transmitido pelo motor ao cabo e, se não muito bem direcionado pode fazer com que o tubo do endoscópio seja puxado para baixo. Isto acarretaria em mais esforços no paciente.

Outro problema é a movimentação composta. Não é possível executar de forma simples uma determinada movimentação, e o controle dos motores é mais complexo. Além disso, o controle do zoom é necessário sempre, pois mantendo o ponto pivô em um plano acaba-se movendo o endoscópio para dentro e para fora do paciente toda vez que se efetua um movimento rotatório. Assim, uma compensação no sistema do zoom seria sempre necessária.

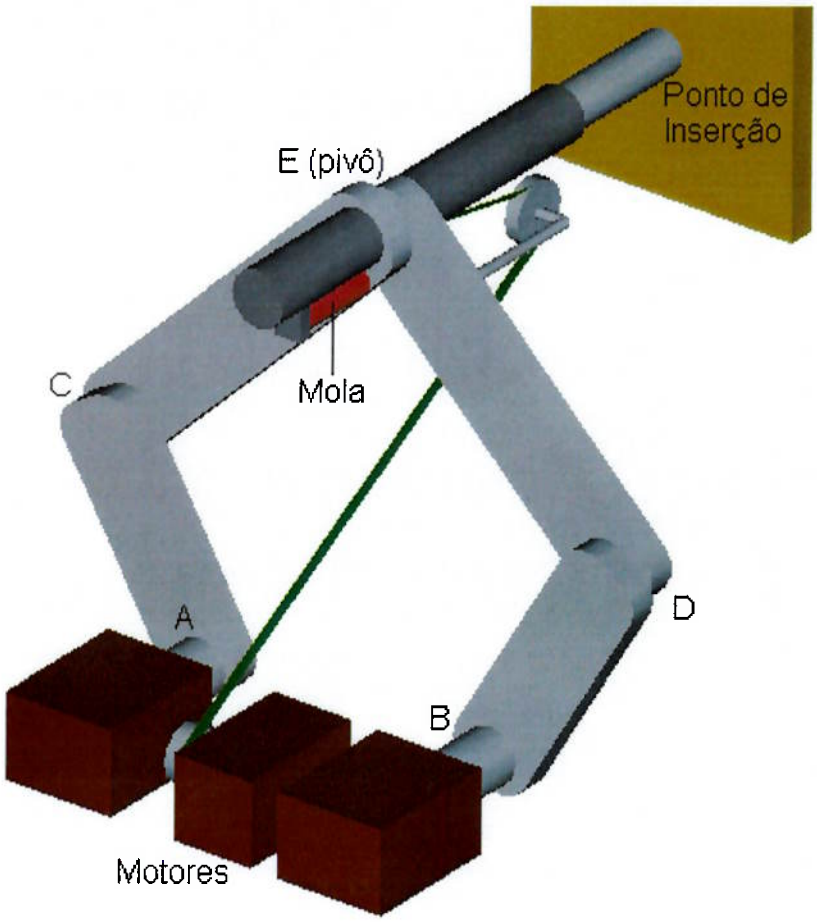


Figura 3 – Mecanismo de barras articuladas

B) Mecanismo de Trilhos Curvos

- Descrição :

Neste mecanismo buscou-se uma separação dos três movimentos do endoscópio. Há montado sobre o paciente um trilho curvo, cujo centro da circunferência encontra-se no ponto de inserção. Um pequeno carro ao qual o tubo que contém o endoscópio está preso corre sobre este trilho. Desta forma obtém-se um dos movimentos de rotação. Há um motor específico para este movimento. Pode-se estabelecer sobre os trilhos limites mecânicos variáveis ao movimento do carro.

O outro movimento de rotação obtém-se através de um outro motor ligado através de um eixo diretamente aos trilhos. Desta forma, pode-se rotacionar os trilhos em torno do eixo que passa pelo ponto de inserção, obtendo-se o segundo movimento desejado.

O movimento de zoom é obtido, assim como no mecanismo anterior, através da movimentação do tubo que contém o endoscópio. É necessário um motor apenas para este movimento.

- Acionamento :

O acionamento neste caso é mais simples, havendo um motor para cada movimento e não havendo movimentos compostos. A movimentação mais simples seria aquela dada pela rotação dos trilhos. Estando já afastado do paciente, basta acoplar um motor a um eixo diretamente conectado aos trilhos na altura do ponto de inserção (centro da circunferência) e desta forma controlar esta rotação. É necessário estabelecer limites mecânicos para este movimento, visando a segurança do paciente.

Outro motor fica responsável pelo movimento de zoom. Devido a aspectos de tamanho e aspectos construtivos, além de ser desejável a separação dos motores do mecanismo em si, chegou-se à conclusão de que seria inviável a instalação deste motor no carro ou no tubo do endoscópio. Assim, optou-se por uma solução idêntica à do mecanismo de barras articuladas, transmitindo o movimento através de um cabo. Este cabo poderia ser contido em um tubo plástico (como em um freio de bicicleta). Bastaria ligar o cabo ao eixo do motor de forma que este o puxe. O retorno também se daria por cabo, passando este através de uma base superior.

Constatou-se ser o movimento do carro o mais complexo. Não se pôde definir nesta etapa do projeto a melhor solução para esta movimentação, ficando duas soluções possíveis para posterior análise :

1) Nesta solução, o motor ficaria montado sobre o carro do endoscópio, e seria ligado diretamente ao trilho. Sobre os trilhos seria fixada uma correia dentada aberta.
2) Nesta segunda solução, o motor ficaria montado afastado do mecanismo em si, e o movimento do mesmo seria transmitido através de um fio ou cabo para o carro. Com a movimentação do carro, no entanto, o comprimento total do fio varia. Para resolver este problema adiciona-se ao carro, no ponto em que o fio é fixo, uma mola, que compensa a variação no comprimento e tenciona o fio como um todo.

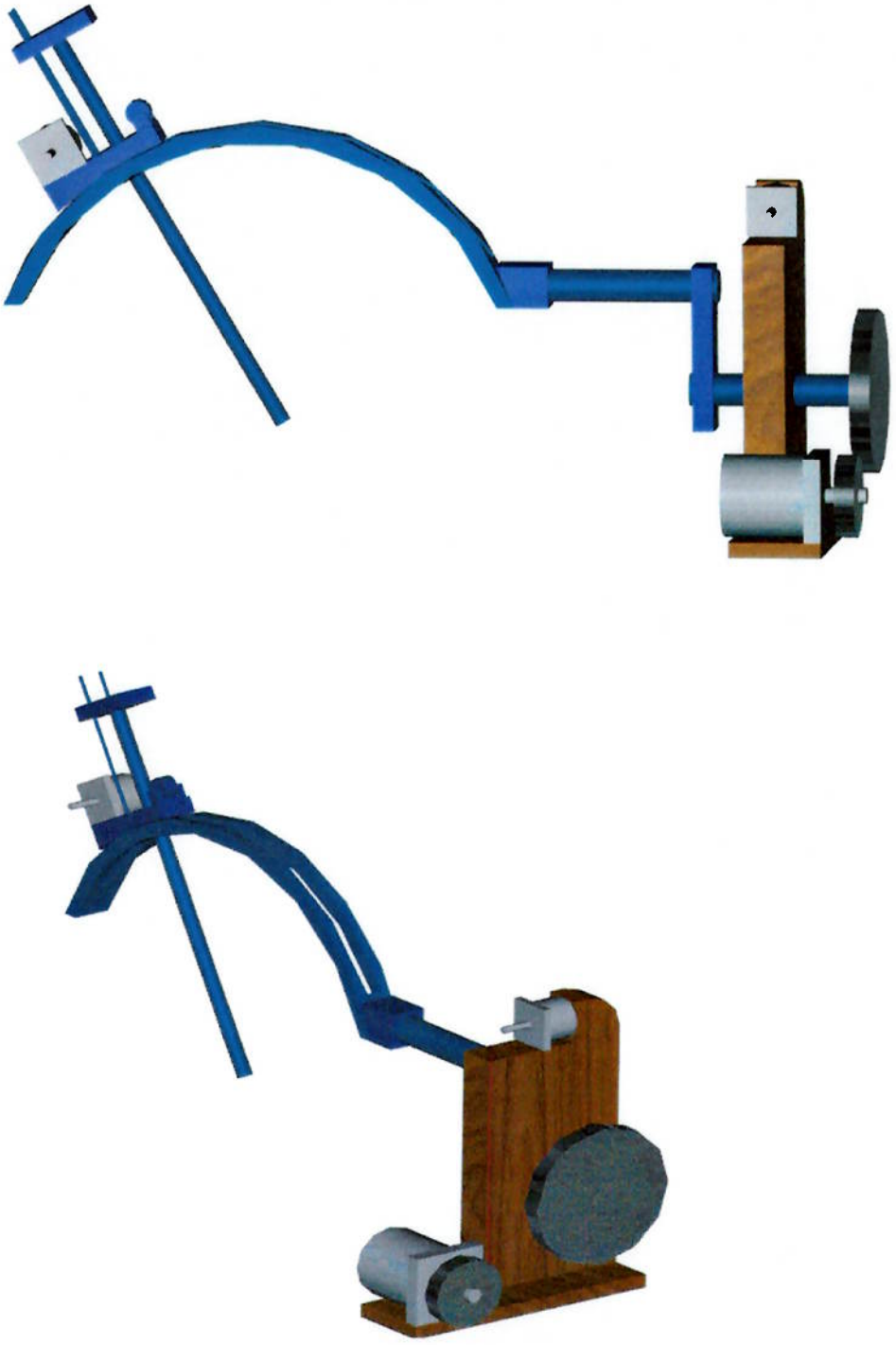


Figura 4 – Mecanismo de trilhos curvos

- Vantagens e Desvantagens :

Pode-se citar como vantagem deste sistema a separação dos movimentos. Não há composição de movimentos, sendo que cada um dos três motores controla um dos três graus de liberdade do sistema. Desta forma, o controle é mais simples e a movimentação mais fácil.

Outra vantagem é que, após a correta instalação do sistema, não há de forma alguma esforço sobre o corpo do paciente, ficando todo o esforço sobre os componentes do mecanismo. Isto porque o mecanismo 'em aberto', ou seja, sem paciente nenhum, já movimenta-se em torno do ponto de inserção naturalmente.

Outra vantagem é a forma construtiva relativamente simples, embora mais complexa do que a do mecanismo de barras articuladas. A maior dificuldade fica no sistema do carro e nas transmissões de movimentos.

Uma desvantagem de razoável gravidade é de que o sistema ocupa excessivo espaço sobre a área da cirurgia. No entanto, se considerar-se que os trilhos ficarão durante a cirurgia inclinados na direção contrária à do orifício de inserção dos instrumentos cirúrgicos, o mecanismo permanece viável.

Quanto às soluções apresentadas para a movimentação do carro :

1) É certamente a solução que construtivamente é mais simples, dispensando elaborados sistemas de transmissão de movimento. No entanto, há o problema do aumento de espaço necessário no carro. Além disso, outro grave problema é de que não há a separação das partes elétricas das mecânicas para esterilização, e há os problemas convencionais de motores móveis no mecanismo. Talvez ainda mais grave seja o aumento do peso que deve ser suportado nos trilhos, aumentando assim o tamanho dos trilhos, e por conseqüência sobrecarregando o motor que rotaciona os trilhos.

2) A inserção do fio ou cabo metálico acarreta em mais problemas construtivos para o sistema. No entanto, não deixa de resolver problemas como o do tamanho e peso do carro, além de garantir a separação dos sistemas elétricos do mecanismo.

C) Junta de movimentação esférica concêntrica *multi-link*

- Descrição :

O mecanismo busca simular o funcionamento de uma junta esférica montada junto ao ponto de inserção do endoscópio. Para se obter uma boa amplitude de movimento utiliza-se um sistema de barras articuladas, que pode ser visto na figura mais abaixo.

O ponto chave deste mecanismo é o ponto D. O controle da rotação é feito através deste ponto, e rotacionando-o move-se todo o sistema no sentido de girar o endoscópio em torno do eixo Y da figura. Para se obter uma movimentação em torno do eixo X é simples, basta acoplar um motor à extremidade (ponto H) e com isso rotacionar todo o mecanismo. O mecanismo das barras é articulado apenas no eixo Y-Z.

O zoom seria obtido através da adição de um terceiro grau de liberdade, que seria a translação do eixo primário (eixo H-B) na direção do eixo Y. Assim, compondo-se esta movimentação com a rotação do ponto D pode-se obter o movimento vertical do endoscópio.

- Acionamento :

Para efetuar o acionamento deste sistema três motores são necessários. Um deles faria o controle da rotação do ponto D do mecanismo, assim fornecendo a rotação em torno do eixo Y. Para efeito de manter os motores em separado do mecanismo, ao invés de incorporar o motor ao mesmo imaginou-se uma transmissão de movimento simples, através de polias dentadas (com correia dentada). A polia ficaria fixa a uma das articulações, forçando-a a girar em relação à outra. O comprimento necessário de correia é variável com a movimentação do mecanismo, por isso é preciso uma polia esticadora com mola para mantê-la esticada.

O acionamento da rotação em torno de X é simples e pode ser feito meramente através de um motor conectado ao eixo primário, no ponto H.

Já para a movimentação do eixo no sentido Y basta fazer com que haja uma cremalheira no apoio do motor anterior. Um terceiro motor movimentaria através da

cremalheira o apoio, e movendo-se este apoio com seu motor move-se o eixo, e logo todo o sistema.

É preciso observar que há neste mecanismo uma forte composição de movimentos. Apenas a rotação em torno do eixo X é independente, sendo os movimentos de rotação em torno de Y e o movimento de zoom do endoscópio movimentos compostos, e assim o controle para estes movimentos deve ser implementado.

- Vantagens e Desvantagens :

O sistema apresenta a vantagem de ser razoavelmente simples manter os motores longe do mecanismo em si, facilitando a esterilização. Além disso, a flexibilidade é alta devido ao mecanismo com as barras.

No entanto, sua forma construtiva é de razoável complexidade. Além disso, há o fato do movimento vertical ser composto com a rotação em torno de Y, fazendo com que o controle seja também mais complexo. Também pode-se citar o fato de que o motor que executa o movimento vertical do endoscópio (translação do mecanismo em Y) precisa mover todo o sistema, inclusive os outros motores, de forma que este motor terá que ter elevada potência.

O sistema também ocupa razoável espaço sobre o local da cirurgia, mas também pode-se citar que ficará inclinado na direção oposta do ponto de inserção dos instrumentos cirúrgicos, permanecendo portanto viável.

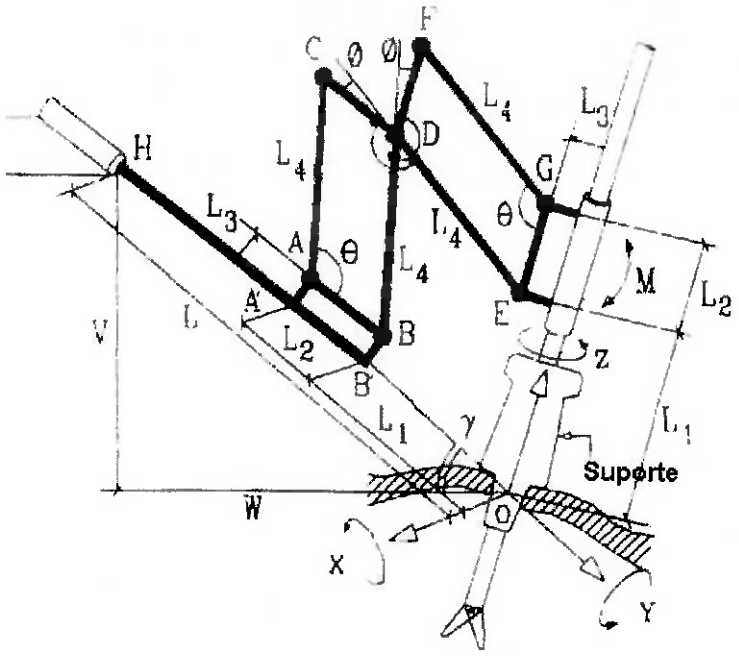


Figura 5 – Mecanismo de junta esférica multi-link

3.3.2 – Motores para o Acionamento do Mecanismo

Para a movimentação de um endoscópio deseja-se que os motores a serem acoplados no mecanismo sejam pequenos e leves, de modo a não tomar muito espaço físico do cirurgião e nem exigir uma estrutura mecânica muito reforçada para esse peso extra. Os motores devem também apresentar aceitáveis níveis de vibração, de modo a não comprometer a estrutura do mecanismo. Deseja-se também que sejam de fácil controle, de modo a se despender pouco esforço com a programação do *software* de controle. Adicionalmente, deseja-se baixos níveis de ruído bem como motores que sejam isolados do meio externo devido à própria natureza de sua aplicação, seja por questões de higiene ou por questões de restrições operacionais.

De acordo com as especificações apresentadas, foram estudados os principais tipos de motores encontrados comercialmente e foram encontradas três soluções que poderiam ser mais diretamente implementadas: servomotores, motores DC e motores de passo.

A) Motores DC

Motores DC são motores que trabalham através de uma alimentação de corrente contínua. Eles funcionam a partir da força resultante que aparece em uma espira condutora imersa num campo magnético. Quando o condutor é atravessado por uma corrente, a espira tende a girar devido à variação do fluxo magnético que vai ocorrendo. Esta variação continua até que ela fique em uma posição neutra no campo magnético, quando torna-se necessário a comutação da corrente na espira, possibilitando a continuação do ciclo.

Motores DC apresentam um fácil controle de velocidade através do controle da tensão de entrada. As variações do torque em função da corrente e da velocidade angular em função da corrente são lineares, o que contribui para uma maior facilidade do controle de posição. São motores que apresentam uma boa precisão e um excelente rendimento.

Suas principais desvantagens são: a utilização de escovas para a comutação da corrente, ocasionando desgaste contínuos das peças que sofrem fricção (cabe ressaltar que existem motores DC que não utilizam escovas para a comutação – *brushless DC motors* – os quais são bastante caros e não serão analisados), além de um contínuo faiscamento que pode representar um risco em ambientes potencialmente inflamáveis, controle via computador mais complicado, maior preço de aquisição quando comparado ao motor de passo.

Os motores *brushless*, apesar de resolverem o problema do desgaste das escovas e do faiscamento, introduzem conceitos novos e acabam por ser bem mais elaborados e de maior complexidade, incompatível com as necessidades estabelecidas. O tamanho do motor é determinado basicamente pela potência útil necessária e pela velocidade de trabalho.

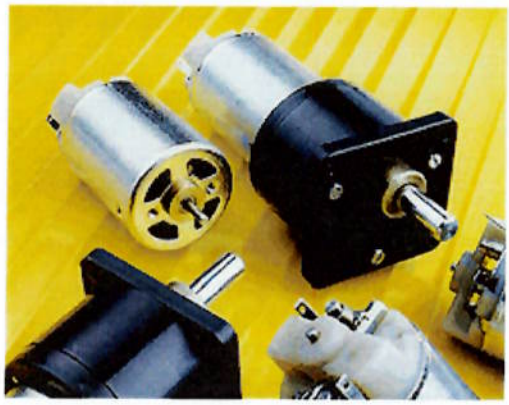


Figura 6 – Alguns exemplos de motores DC

B) Servomotores DC

A segunda possibilidade para a movimentação do endoscópio seria a utilização de vários servomotores DC. O servomotor DC consiste num motor de corrente contínua dotado de um mecanismo bastante preciso de posicionamento por contagem de voltas (*encoder*), o qual permite uma precisa operação em relação à posição angular do eixo. É um motor que demanda um elevado e desnecessário investimento em precisão de posicionamento. Possui todas as vantagens do acionamento de um motor DC, além do fato de permitir um ajuste mais fino na velocidade de posicionamento. É uma solução mais adequada a posicionamentos que requerem elevados níveis de precisão.

Figura 7 – Exemplo de um servomotor

C) Motores de passo

Por último, apresenta-se a solução referente a motores de passo. Motores de passo, ao contrário dos motores DC, funcionam através da injeção contínua de um trem de pulsos em cada fase do enrolamento do motor. É um motor cujo posicionamento pode ser feito sem a elaboração de um controle mais apurado, sem a utilização de um *feedback* exclusivo (observa-se que a taxa de deslocamento do motor de passo por fase é fixada construtivamente, bastando então alterar a freqüência dos pulsos para se obter maiores ou menores velocidades).

O princípio de operação baseia-se no aparecimento de um campo girante induzido no estator. Este campo é gerado a partir de uma lógica que consegue acionar de modo eficiente as fases de uma maneira seqüencial. Este motor possui como características principais um torque de saída pulsante devido à comutação contínua das fases e um fácil manuseio via computador. Ele disponibiliza um aproveitamento menor que os dos motores DC, porém é muito mais facilmente encontrado devido à sua facilidade de controle. A curva torque *versus* rotação do motor é decrescente, o que limita a utilização do motor para altas rotações.

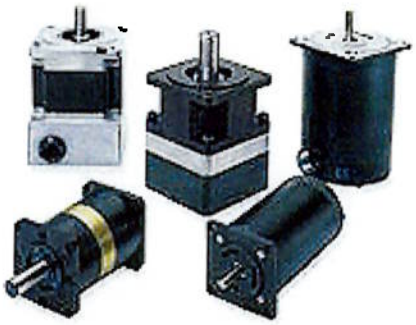


Figura 8 – Alguns exemplos de motores de passo

3.3.3 – Linguagens de programação

Para a seleção de uma linguagem de programação para o software de controle do endoscópio (comunicação entre a interface homem-máquina e os motores) deve-se considerar os seguintes aspectos :

- Aspectos de segurança. Em um procedimento cirúrgico, não se pode correr o risco de falhas no sistema ou no software. Não podem haver bugs no software, e a linguagem de programação deve oferecer boa confiabilidade.
- Simplicidade de programação. Algoritmos muito complexos ou linguagens que exijam detalhamento a nível mais baixo de funções e procedimentos simples acarretam em maior risco de ocorrência de bugs e falhas.
- Deve-se considerar a facilidade de acesso aos softwares compiladores, dando-se preferência às linguagens de programação de acesso livre (freeware).

A partir disto, selecionou-se para análise as três linguagens de programação a seguir :

A) Linguagem C

- Descrição :

Linguagem de programação mais conhecida atualmente, existe em várias versões, incluindo C++ (orientada a objeto) e Visual C++ para programação em ambiente Windows.

- Vantagens e Desvantagens :

É extensamente conhecida, e sendo assim o software possui mais fácil manutenção e a pesquisa torna-se também mais simples. A programação em si, no entanto, não é tão simples, exigindo um nível mais baixo de programação. É

extremamente confiável, no entanto, se bem programada e utilizada em sistema operacional confiável. É preciso compilar no sistema operacional em que se irá utilizar. Os compiladores e bibliotecas podem ser obtidos gratuitamente.

B) Linguagem Java

- Descrição :

É uma linguagem de programação mais nova, criada pela Sun, e é baseada na linguagem C. Possui muitas características voltadas à internet, e é baseada em classes e objetos, sendo assim toda orientada a objeto. Existe versão visual da linguagem (Visual Café) para ambiente Windows.

- Vantagens e Desvantagens :

Uma das vantagens é a semelhança com a linguagem C (no que se trata da sintaxe). As bibliotecas de Java são mais completas e todo o software também pode ser adquirido gratuitamente. Assim como em C, a programação não é tão simples, exigindo um nível mais baixo de programação. A confiabilidade é menor do que no C, mas a principal desvantagem é a complexidade do executável final. Uma vez compilado, o programa pode ser utilizado em qualquer sistema operacional, tornando-o flexível, mas para isso o Java torna-se uma linguagem interpretada, o que torna o software maior e consumidor de mais recursos.

C) Visual Basic

- Descrição :

Uma das linguagens de programação que mais vêm conquistando mercado, é baseada na linguagem Basic e é toda visual, sendo aplicável apenas a ambientes Windows. Comercializada pela Microsoft, muitos softwares são hoje produzidos nesta linguagem, que é orientada a objeto.

- Vantagens e Desvantagens :

A grande vantagem do Visual Basic e que o torna tão popular é a extrema facilidade de programação. A programação é muito simples e intuitiva, e não é necessário programar em nível mais baixo. Praticamente qualquer coisa pode ser realizada apenas baixando da internet os controles específicos, sendo que é rápido e simples programar desde janelas, algoritmos e cálculos, gráficos, até interface serial, entre outros. No entanto, é um software comercial e sua distribuição não é gratuita, é aplicável apenas ao ambiente Windows e peca principalmente pela sua baixa confiabilidade. O executável final também torna-se muito grande e consumidor de muitos recursos.

3.3.4 – Método de controle do mecanismo (Interface homem-máquina)

Em se tratando de um mecanismo utilizado em cirurgias médicas, é necessário avaliar alguns pontos relativos à interface homem-máquina.

- Deve-se considerar aspectos de segurança. Isto é, deve-se evitar que o endoscópio execute movimentos indesejados (que podem eventualmente causar danos ao paciente) devido a comandos errados fornecidos pela interface.
- Deve-se buscar simplicidade na maneira de introdução dos comandos, de forma que esta se dê o mais facilmente e intuitivamente possível, não contribuindo para a complexidade de uma cirurgia.
- Deve-se buscar, na medida do possível, formas de introdução de comandos que não se utilizem das mãos do cirurgião, desta forma permitindo que este movimente o endoscópio enquanto opera os instrumentos de cirurgia, e eliminando assim a necessidade de um auxiliar apenas para esta movimentação.
- Deve-se considerar a simplicidade da construção do sistema de entrada de dados.
- Deve-se considerar a simplicidade da implementação do software que efetuará a interface.

Desta forma, analisou-se as seguintes soluções para a interface homem-máquina :

A) Sistema de Mouse Convencional

- Descrição do Sistema :

Consiste na entrada de dados mais comum, que é simplesmente um mouse convencional. O cirurgião movimentaria o mouse sobre uma plataforma horizontal, e de acordo com sua movimentação (em duas dimensões) o endoscópio responderia na direção selecionada.

Para evitar movimentos inesperados do endoscópio, os botões seriam utilizados para confirmar as movimentações. Além disso, distinguindo-se os botões, pode-se também controlar dois tipos de movimentação. Com isso, o médico controlaria também o zoom do endoscópio (que consiste na movimentação vertical do duto de sustentação do endoscópio).

- Forma de Funcionamento :

Assim, caso queira movimentar o endoscópio nas direções X ou Y (rotações), o cirurgião pressionaria o botão esquerdo e, segurando-o pressionado, moveria o mouse, assim enviando os comandos ao endoscópio. A direção de movimentação seria a mesma da tela (se mover para cima, o movimento do endoscópio resulta em uma movimentação da imagem no vídeo também para cima), desta forma tornando a entrada de dados mais intuitiva.

Analogamente, caso queria alterar o zoom (translação na direção Z), bastaria que pressionasse o botão direito. Neste caso, seria considerada apenas a movimentação vertical do mouse. Movê-lo para cima aumentaria o zoom (o endoscópio adentra mais o corpo do paciente), movê-lo para baixo faria o inverso.

- Vantagens e Desvantagens :

As vantagens é que não há necessidade de nenhuma construção adicional (o sistema está pronto) e de que a implementação do software é bem simples neste caso. Talvez a única coisa que devesse ser construída seria a plataforma sobre a qual o mouse ficaria. Seria construída também uma 'borda' nesta plataforma, apenas para evitar que o mouse caia da mesma com facilidade.

Pode-se citar também a familiarização que todos têm hoje com a operação de um mouse, de forma que seria uma forma fácil de entrada dos dados.

Como desvantagens pode-se citar o problema da necessidade de uma plataforma horizontal sobre a qual o médico deve mover o mouse, não havendo assim liberdade na movimentação. Além disso, como principal e grave desvantagem, este sistema não livra as mãos do médico, e sendo assim, para mover o endoscópio, o mesmo terá que soltar o controle dos instrumentos ou então dispor de um auxiliar que movimente o mouse.

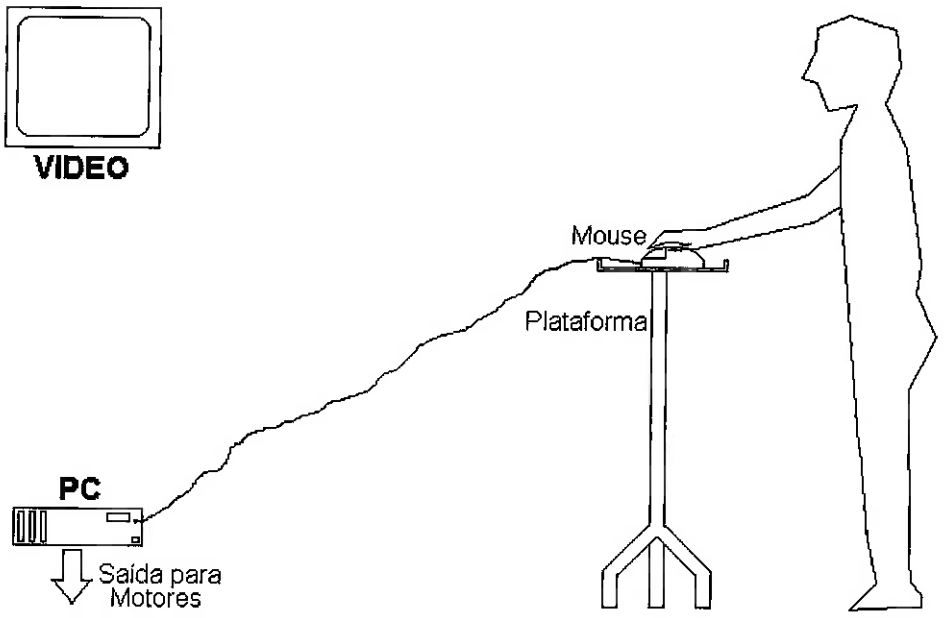


Figura 9 – Controle por mouse convencional

B) Sistema de Joystick convencional

- Descrição do Sistema :

Este sistema consiste de um joystick comum (composto por uma haste com rotação livre no plano e, normalmente, dois botões sobre a haste). A movimentação do endoscópio se daria pela leitura da movimentação da haste.

De uma maneira similar ao sistema do mouse, os comandos teriam que ser confirmados pelos botões, desta forma evitando movimentações indesejadas do endoscópio.

- Forma de Funcionamento :

Similar ao sistema de mouse convencional. Para rotacionar o endoscópio (direções X e Y), o cirurgião pressionaria um dos botões (botão 1) e, mantendo-o pressionado, moveria a haste na direção desejada. Novamente, o endoscópio move-se de forma que a imagem formada no vídeo mova-se na mesma direção que foi dada à haste.

Da mesma forma, para alterar o zoom o cirurgião pressionaria o outro botão (botão 2) e, mantendo-o pressionado, moveria a haste para cima (frente) para aumentar o zoom ou para baixo (trás) para diminuir o zoom. Neste caso, a movimentação horizontal seria ignorada.

- Vantagens e Desvantagens :

É um sistema muito similar ao sistema de mouse convencional. Assim, possui a vantagem de não necessitar de nenhuma construção adicional (o sistema está pronto). No entanto, desta vez nem mesmo a plataforma precisa ser construída. Os joysticks funcionam em qualquer posição, e são de mais fácil movimentação, não requerendo espaço para o movimento como é o caso do mouse. Podem ser colocados ou colados em qualquer posição, em superfícies diversas (muitos hoje já vêm com ventosas para aderir a superfícies).

Sua desvantagem está na leitura dos dados. Diferentemente do mouse, a posição do joystick é dada por potenciômetros, e por isso eles requerem portas especiais para

a leitura. Para isso, é necessária uma placa específica para efetuar tal leitura, que possua esta porta específica para joysticks. A implementação do software também fica mais complexa com isso.

Além, é claro, da grave desvantagem de que o sistema também necessita das mãos do operador para a entrada dos dados, e desta forma não satisfaz esta condição.

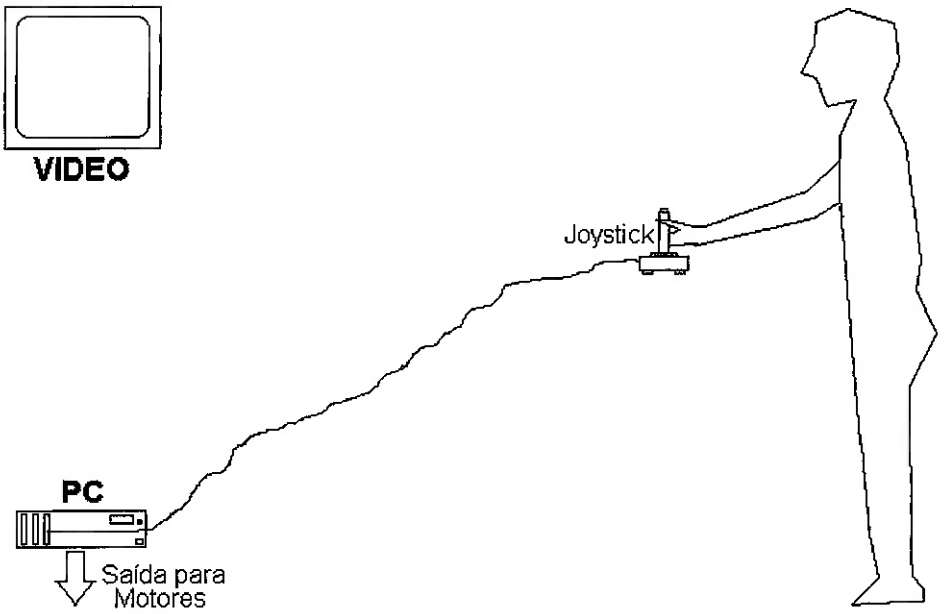


Figura 10 – Controle por Joystick

C) Sistema de Comando de Voz

- Descrição do Sistema

O sistema como um todo consiste simplesmente em um microfone que faz a aquisição dos dados através de comandos sonoros do operador. Este microfone envia seus dados a uma placa de som instalada no computador, que os interpreta.

- Forma de Funcionamento

Antes da cirurgia, é necessário gravar com a voz do próprio médico quais são os comandos e o que eles fazem. Por exemplo, programa-se que ao dizer a palavra

"esquerda" o computador deve entender que o médico deseja que o endoscópio movimente-se para a esquerda (novamente, fala-se do efeito da imagem no vídeo). Deve também ser gravado um comando de acionamento e desligamento do sistema, para que o médico possa conversar durante a cirurgia com o sistema desligado, procurando assim evitar entradas incorretas de dados.

Os comandos de movimentação funcionam de forma discreta. Assim, quando o médico diz a palavra "esquerda" o endoscópio move-se de forma a mover a imagem para a esquerda, mas apenas uma pequena distância (alguns milímetros). O controle discretizado não prejudica a visualização, visto que o ângulo de captação da câmera é suficiente para fornecer visão. Se o médico desejasse mover o endoscópio mais para a esquerda, teria que dizer a palavra "esquerda" novamente. Uma outra solução seria a movimentação contínua, com um movimento de parada. No entanto, esta foi descartada pois não se pode correr o risco do computador não entender o comando de parada, causando assim movimentos indesejados do endoscópio.

Haveriam assim oito comandos, um para acionar o sistema, um para desligá-lo, quatro para a movimentação e dois para o controle do zoom.

- Vantagens e Desvantagens :

Gravando-se comandos que de alguma forma identifiquem o movimento desejado, como a palavra "esquerda" por exemplo para movimentações para a esquerda, os comandos tornam-se fáceis e intuitivos. Não há complicações extras para o médico.

Da mesma forma que os sistemas anteriores, neste também não há a necessidade de se construir nada, o sistema já está pronto.

Outra grande vantagem é de que o sistema dispensa qualquer tipo de comando físico do médico, deixando suas mãos livres para atuar com os instrumentos da cirurgia. Desta forma, economiza-se tempo e não há a necessidade de auxiliares para a cirurgia.

Como desvantagem pode-se citar a necessidade da placa de som para adquirir os dados. Além disso, o tipo de microfone utilizado é crucial para o bom funcionamento do sistema. Um microfone muito sensível é necessário.

Outra desvantagem é que o movimento através deste sistema ocorre de forma discretizada. Assim, o médico possui menos controle real de até onde o endoscópio se moverá.

No entanto, a principal desvantagem, e que torna o sistema inviável do ponto de vista do escopo deste projeto, é a complexidade do software que fará a leitura dos sons. A tecnologia de leitura de voz é uma tecnologia que hoje ainda se encontra em desenvolvimento, e envolve o reconhecimento dos padrões de freqüência e timbre das ondas sonoras recebidas pelo microfone. Além da complexidade, é de pouca confiabilidade, ao menos no estado em que a tecnologia se encontra hoje, não sendo assim aplicável a uma cirurgia, que é uma situação em que erros não podem ser cometidos.

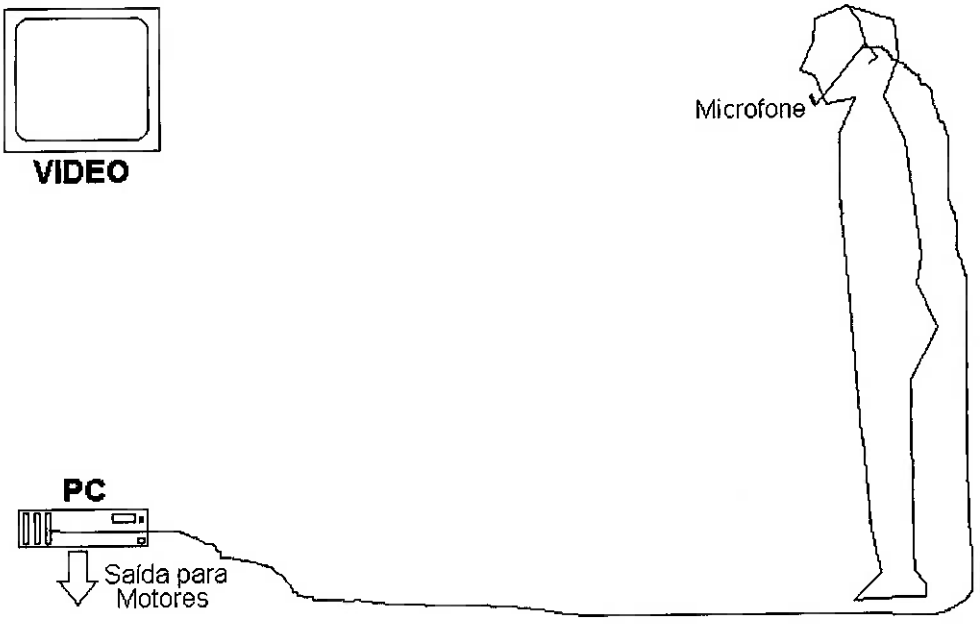


Figura 11 – Controle por comando de voz

D) Sistema de Pedais

- Descrição do Sistema :

O objetivo deste sistema é utilizar os pés do cirurgião como fonte de entrada dos dados de movimentação. Para tanto utilizaria-se um sistema baseado em dois pedais,

do tipo que possuem movimentação tanto para a frente quanto para trás. Através de sensores que detectariam a movimentação destes pedais poderia-se passar ao computador a leitura dos comandos. Para a utilização deste sistema, o médico teria que permanecer sentado.

De uma forma similar à discutida anteriormente, os comandos teriam que ser confirmados por botões. Desta vez, no entanto, no intuito de não utilizar as mãos do médico, um sistema seria adaptado para funcionar através de comandos passados pelo joelho do mesmo. Isto seria feito com a movimentação do joelho esquerdo ou do direito para dentro (parte interna da cadeira), fechando assim um contato que atuaria como um botão.

Para a aquisição dos dados pelo computador, imaginou-se duas soluções : Uma delas seria a própria utilização de uma placa de aquisição apenas para a coleta de todos os dados. Outra seria a adaptação de um mouse convencional, utilizando-se todos os seus circuitos, ligando os pedais no lugar dos encoders e os contatos na cadeira no lugar dos botões.

- Forma de Funcionamento :

Sendo assim, caso o médico deseje mover o endoscópio nas direções X ou Y, ele deve utilizar o joelho esquerdo e pressionar o contato do lado esquerdo (interno) da cadeira. Mantendo-o pressionado, ele pode então movimentar os pedais. O pedal esquerdo controlaria a movimentação na direção horizontal (para a frente = direita / para trás = esquerda), e o pedal direito na direção vertical (para a frente = cima / para trás = baixo).

Para o controle de zoom, o cirurgião deve utilizar o joelho direito, fechando o contato direito. Mantendo-o pressionado, e movendo o pedal direito, ele controla o zoom (translação) do endoscópio (para a frente = mais zoom / para trás = menos zoom).

- Vantagens e Desvantagens :

A mais evidente vantagem deste sistema é que ele, assim como o comando de voz, não necessita da utilização das mãos do cirurgião para a movimentação do

endoscópio. Além disso, sua implementação é bem mais simples e exeqüível do que o controle de voz.

Há várias desvantagens, no entanto. A primeira delas é que há a necessidade de toda a construção do sistema, desde os pedais, contatos, sensores, até a própria cadeira em si. Além disso, há uma maior complexidade para o envio dos dados ao computador.

Outra séria desvantagem é que, uma vez que obriga o cirurgião a permanecer sentado, elimina muito de sua liberdade e restringe em muito seus movimentos. Além disso, costuma-se permanecer em pé durante cirurgias, para que o médico possa mover-se e agir com maior rapidez em situações de emergência.

Pode-se citar ainda o fato da movimentação através dos pedais ser difícil e muito pouco intuitiva, especialmente para o pedal esquerdo, em que movimentos de frente/trás dos pés controlam movimentos esquerda/direita da imagem.

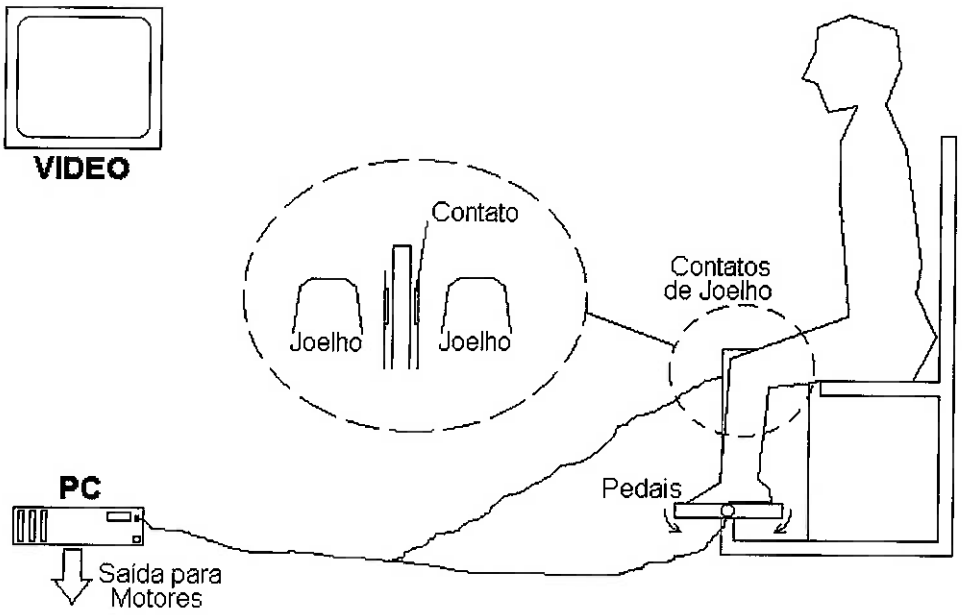


Figura 12 – Controle por sistema de pedais

E) Sistema de Mouse de Cabeça

- Descrição do Sistema :

Neste sistema, o objetivo é obter os comandos de movimentação do endoscópio através da movimentação da cabeça do cirurgião. Esta solução pode ser imaginada levando em consideração o fato de que o médico passa a maior parte da cirurgia olhando diretamente para o vídeo. Assim, com movimentos de sua cabeça, ele poderia indicar a direção em que pretende que o endoscópio se movimente.

Evidentemente, este sistema é suscetível a entradas indesejadas. Por isso, um sistema de confirmação deve também ser aqui implementado. Este consistiria em dois contatos que deveriam ser acionados pelo médico com os pés, através de pedais no chão.

A configuração mais simples imaginada para este sistema seria a adaptação de um mouse convencional. Para tanto, bastaria anexar à esfera do mesmo uma haste com uma massa, formando assim um pêndulo. O próprio deslocamento da esfera passaria a informação ao computador da angulação do pêndulo. Os dois botões seriam facilmente adaptados aos contatos do sistema de pedais.

Para a confirmação da movimentação a ser executada, surge no vídeo, por sobre a imagem, uma seta indicando em que direção o endoscópio irá se mover.

- Forma de Funcionamento :

Imaginemos, assim, que o cirurgião pretenda mover a imagem do vídeo para a esquerda. Ele inclina lateralmente sua cabeça para a esquerda, como se tentasse encostar seu ouvido em seu ombro. Após um certo ângulo, surge a seta no canto esquerdo do vídeo sugerindo que a movimentação se dará para a esquerda. O médico então pressiona o pedal esquerdo, fechando o contato esquerdo e confirmando o movimento. Este movimento se mantém enquanto o pedal estiver pressionado e a cabeça do médico inclinada.

Digamos, agora, que o cirurgião deseja aumentar o zoom. Ele inclina sua cabeça para a frente, como se tentasse encostar o queixo em seu peito. Após um certo ângulo, surgem no canto inferior do vídeo dois sinais. Um é uma seta que indica

movimentação para baixo. O outro é um sinal que indica mais zoom. Pressionando o pedal direito, o movimento de zoom se inicia, e mantém-se enquanto o pedal estiver pressionado e a cabeça inclinada.

- Vantagens e Desvantagens :

Ao contrário do sistema de pedais, a adaptação deste sistema a um mouse comum é mais simples. A leitura dos encoders do mesmo é feita exatamente da mesma forma, simples. A única complexidade construtiva fica por conta do sistema dos pedais que deve ser construído, mas que ainda assim não apresenta complexidade estrutural.

Por ser lido como um mouse pelo computador, não há a exigência de placas de aquisição ou nada parecido, e a implementação do software é mais simples.

Além disso, o sistema permite que o médico fique de pé, e inclusive que se movimente livremente. A única restrição é de que para controlar o endoscópio o cirurgião deverá estar em uma posição de onde possa utilizar os pedais. O sistema dos pedais pode ser movimentado facilmente, no entanto.

Outra vantagem é a movimentação mais intuitiva do endoscópio. Os movimentos com a cabeça observando o vídeo ficam simples e intuitivos, e o sistema de indicação com as setas e sinais, com posterior confirmação, evitam movimentações indesejadas do endoscópio.

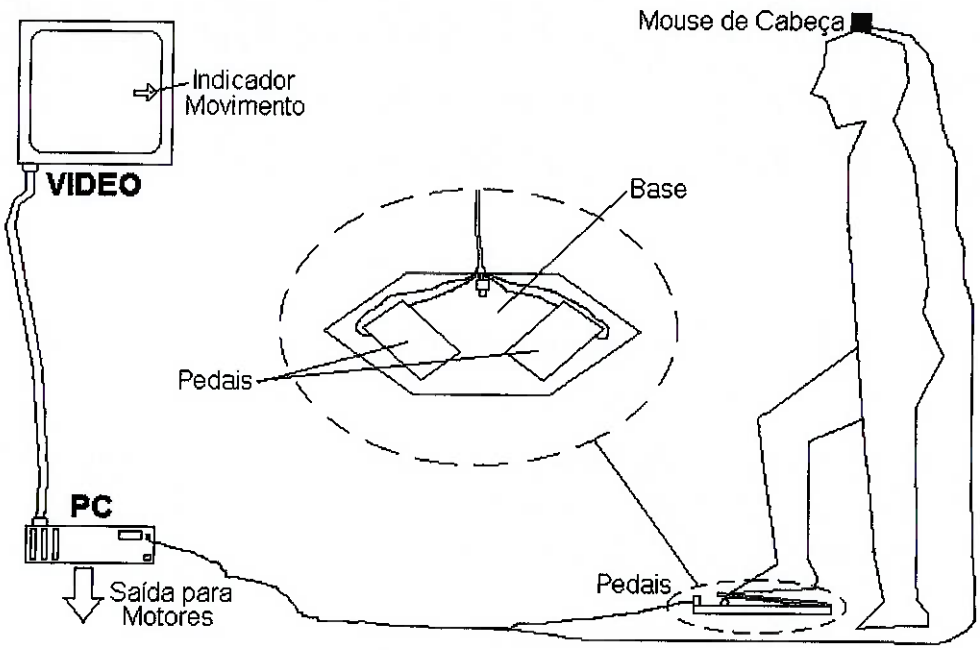


Figura 13 – Controle por mouse de cabeça associado a pedais

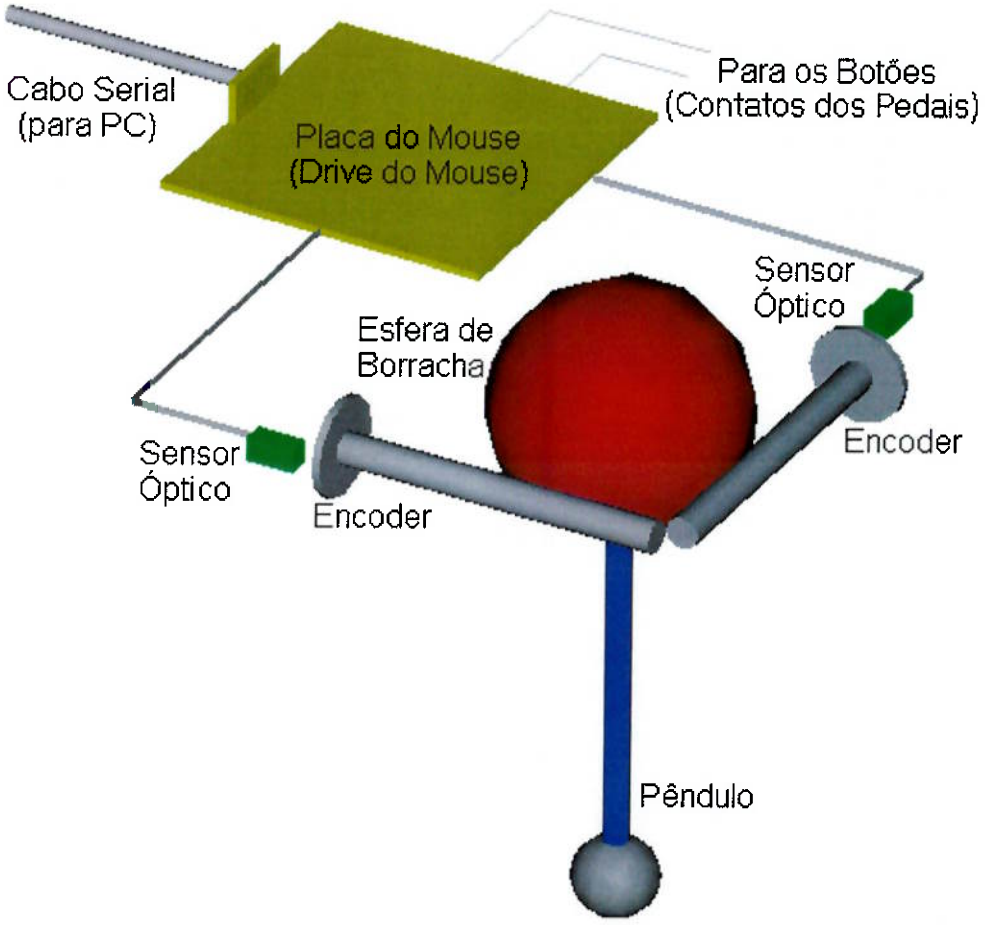


Figura 14 - Esquema construtivo do mouse de cabeça

4. SELEÇÃO DA MELHOR SOLUÇÃO

Uma vez apresentadas as soluções selecionadas, deve-se proceder à seleção da melhor entre elas, aquela que será efetivamente implementada. Para isso, utilizou-se o método da matriz de decisão, que nada mais é do que uma análise quantitativa das vantagens e desvantagens de cada solução.

Da mesma forma que foi feito no estudo de viabilidade, aqui também se dividirá o sistema em :

- Mecanismos para a Movimentação do Sistema
- Motores para o Acionamento do Mecanismo
- Linguagens de Programação
- Método de Controle do Mecanismo (Interface homem-máquina)

4.1 - Mecanismos para a Movimentação do Sistema

Para o mecanismo, considerou-se como muito importantes três fatores, que são : segurança (principalmente), menor número de graus de liberdade possível e separação dos sistemas mecânico e elétrico. Além disso, analisou-se também a facilidade de construção, de implementação do software de controle (mais simples sem movimentos acoplados) e estabilidade do mecanismo.

Vale notar que fez-se aqui uma separação da solução do mecanismo de trilhos curvos, sendo que uma solução prevê a instalação de um motor sobre o carro que corre levando o endoscópio, enquanto que outra prevê uma transmissão por cabo, desta forma separando a parte elétrica da mecânica.

Atribuiu-se notas a cada mecanismo, de acordo com as características de cada um. As principais vantagens e desvantagens encontram-se descritas no estudo de viabilidade.

		Notas			
Critério	**Peso**	**I**	**II**	**III**	**IV**
Segurança Oferecida ao Paciente	4	3	4	4	4
Estabilidade do Mecanismo	2	3	4	5	4
Número de Graus de Liberdade (Menor = Melhor)	3	4	5	5	5
Separação de Sistemas Elétricos e Mecânicos	3	5	2	5	4
Facilidade de Construção do Mecanismo	2	4	4	3	2
Simplicidade da Implementação do Controle	2	2	5	4	3
Total (Notas Ponderadas)	**16**	**57**	**63**	**70**	**61**

I - Mecanismo de Barras Articuladas

II - Mecanismo de Trilhos Curvos, motor junto ao mecanismo

III - Mecanismo de Trilhos Curvos, motor separado

IV - Junta de Movimentação Esférica Concêntrica Multi-link

Critério para as notas :

(1 - Péssimo / 2 - Ruim / 3 - Regular / 4 - Bom / 5 - Ótimo)

Tabela I – Matriz de decisão para mecanismos

Vê-se, pela soma das notas (multiplicadas pelos pesos de cada item avaliado), que a melhor solução é a III, ou seja, o mecanismo de trilhos curvos, na versão com o motor separado. Embora acrescente dificuldades construtivas, esta solução é a melhor por oferecer todos os requisitos mais importantes ao projeto.

Pode-se ver que o mecanismo de barras articuladas peca principalmente pela sua dificuldade de controle e construção, além de não oferecer segurança compatível ao paciente devido a utilizar o próprio corpo do paciente para apoio do endoscópio.

O mecanismo de trilhos curvos com o motor no carro possui praticamente todas as vantagens do mecanismo vencedor, mas péssima separação das partes elétrica e mecânica do sistema.

Por fim, a junta esférica concêntrica têm o problema de dificuldade de construção e de controle, devido aos movimentos acoplados.

4.2 - Motores de Acionamento

Para a seleção dos motores que acionam o mecanismo considerou-se como fatores principais o tamanho e peso (deve ser compacto e leve o quanto possível) e a facilidade de controle do motor. Também analisou-se os níveis de ruído e vibração, menos importantes, no entanto.

Seguiu-se a mesma metodologia utilizada para os mecanismos. A matriz de decisão ficou :

		Notas		
Critério	**Peso**	**I**	**II**	**III**
Tamanho / Peso (Menor = Melhor)	3	2	2	3
Nível de Vibração	2	4	4	2
Nível de Ruído	1	3	3	2
Facilidade de Controle	3	2	3	4
Total (Notas Ponderadas)	**9**	**23**	**26**	27

I - Motores DC (Corrente Contínua)

II - Servomotores DC

III - Motores de Passo

Critério para as notas :

(1 - Péssimo / 2 - Ruim / 3 - Regular / 4 - Bom / 5 - Ótimo)

Tabela II – Matriz de decisão para motores

Pode-se ver que, embora o motor de passo apresente alguns problemas com vibração (em alguns modos de operação) e ruído, ele é mais leve e certamente o mais simples de controlar. Nota-se, no entanto, uma quase equivalência com servomotores, que sacrificam facilidade de controle e peso para obter menor ruído e vibração.

Decidiu-se, no entanto, utilizar realmente os motores de passo, devido à sua facilidade de controle e fácil obtenção.

4.3 - Linguagem de Programação

Novamente utilizando a metodologia da matriz de decisão, considerou-se neste caso a segurança contra falhas ou 'bugs' a característica principal e necessária para o software a ser implementado. Analisou-se também a facilidade de programação e a facilidade de obtenção da linguagem ou dos compiladores.

		Notas		
Critério	**Peso**	**I**	**II**	**III**
Segurança contra Bugs ou Falhas	5	3	2	1
Simplicidade de Programação	3	2	3	4
Facilidade de Acesso (FreeWare)	3	5	4	2
Total (Notas Ponderadas)	**11**	**36**	**31**	**23**

I - Linguagem C

II - Linguagem Java

III - Linguagem Visual Basic

Critério para as notas :

(1 - Péssimo / 2 - Ruim / 3 - Regular / 4 - Bom / 5 - Ótimo)

Tabela III – Matriz de decisão para linguagens de programação

A linguagem C mostrou ser a mais adequada, pois é mais segura e de muito fácil acesso, não só aos compiladores, mas também a referências bibliográficas. Assim, embora seja a mais complexa do ponto de vista de programação, é a melhor opção.

A linguagem Java é mais simples do que o C, mas menos confiável. Já o Visual Basic peca pela sua completa falta de confiabilidade, além de ser um software comercial.

4.4 - Método de Controle do Mecanismo (Interface homem-máquina)

Para a inserção de comandos, ou seja, a interface homem-máquina, os principais quesitos necessários considerados foram : Segurança contra entradas indesejadas (o principal e mais importante) e a liberdade das mãos do médico na utilização.

Foram analisadas ainda a simplicidade de utilização (inserção de comandos) e a simplicidade de construção mecânica e de implementação do software.

A tabela resultante pode ser vista abaixo.

		Notas				
Critério	**Peso**	**I**	**II**	**III**	**IV**	**V**
Segurança Contra Entradas Acidentais	5	4	4	3	3	4
Simplicidade da Inserção de Comandos	3	4	5	4	3	4
Liberdade das Mãos do Médico	4	1	2	5	4	5
Simplicidade de Construção	3	4	5	5	2	3
Simplicidade do Software de Controle	3	4	3	1	4	4
Total (Notas Ponderadas)	**18**	**60**	**67**	**65**	**58**	73

I - Sistema de Mouse Convencional

II - Sistema de Joystick Convencional

III - Sistema de Comando de Voz

IV - Sistema de Pedais

V - Sistema de Mouse de Cabeça

Critério para as notas :

(1 - Péssimo / 2 - Ruim / 3 - Regular / 4 - Bom / 5 - Ótimo)

Tabela IV – Matriz de decisão para método de controle

Pode-se notar a maior nota atribuída ao sistema com mouse de cabeça basicamente por ser a opção que apresenta boas notas em praticamente todos os quesitos. Apenas sua construção mecânica é de razoável complexidade.

Todas as outras soluções apresentam alguma espécie de falha, como a não liberação das mãos do médico, risco de entradas acidentais, complexidade de construção ou implementação do software, etc.

Sendo assim, deve-se passar a trabalhar no projeto do mouse de cabeça e de seu sistema como um todo.

5. PROJETO DO MECANISMO

Definiu-se que o primeiro passo a partir da determinação da solução seria o projeto puramente mecânico do mecanismo. Isto porque é este o maior limitante do projeto. O mecanismo pode mostrar ser inviável o projeto como um todo. Mas uma vez projetado e montado, basta proceder ao seu controle.

Também decidiu-se projetar inicialmente o mecanismo devido ao fato de que é o componente a ser construído / implementado primeiro, seguindo assim uma ordem lógica para o trabalho.

Neste capítulo são feitas as considerações necessárias, assim como os cálculos relevantes, para o projeto do mecanismo. Ao final do capítulo obtém-se os desenhos finais do mecanismo como um todo.

5.1 - Definições

Devido às limitações do material disponível para a construção do mecanismo, decidiu-se efetuar um projeto inicial do mesmo para depois constatar se tal projeto atinge os requisitos do projeto.

Sendo assim, fazendo uso do material disponível, faz-se as considerações necessárias e define-se assim as peças e o funcionamento do mecanismo como um todo.

5.1.1 - Trilhos

O primeiro passo é definir a forma construtiva dos trilhos. Os trilhos devem ter um tamanho correto, de forma a poderem atender à metade da largura de uma mesa cirúrgica (na média). Também há uma grave preocupação com não atingir de forma alguma o paciente. Por isso decidiu-se que a extensão do trilho teria um trecho inclinado, de forma que a peça toda atinja o eixo de rotação (na altura do ponto de inserção) mas também evite o contato com o paciente.

O raio da parte curva do trilho é outro fator de elevada importância. Se muito grande, o mecanismo tende a ficar muito pesado e espaçoso. Se muito pequeno, perde-se precisão de movimento e também, devido à dificuldade de se construir um carro pequeno, perde-se amplitude de movimento.

Desta forma decidiu-se por um trecho de circunferência com diâmetro interno de 150 mm e diâmetro externo de 190 mm, um diâmetro razoável para a amplitude do movimento. A diferença de 20 mm visa manter a resistência mecânica da peça.

A espessura do trilho imaginada seria de 5 mm, de acordo com o material disponível. Esta espessura é provavelmente suficiente para resistir aos esforços deste projeto.

Por simplicidade de construção, decidiu-se por fazer com que os trilhos curvos e sua extensão até o eixo de rotação sejam uma única peça na verdade. Com isso têm-se também um vão livre para a passagem dos cabos que conduzirão o movimento até o carro no meio das peças.

Um desenho de um dos trilhos, já com as dimensões imaginadas (fora de escala), pode ser visto abaixo.

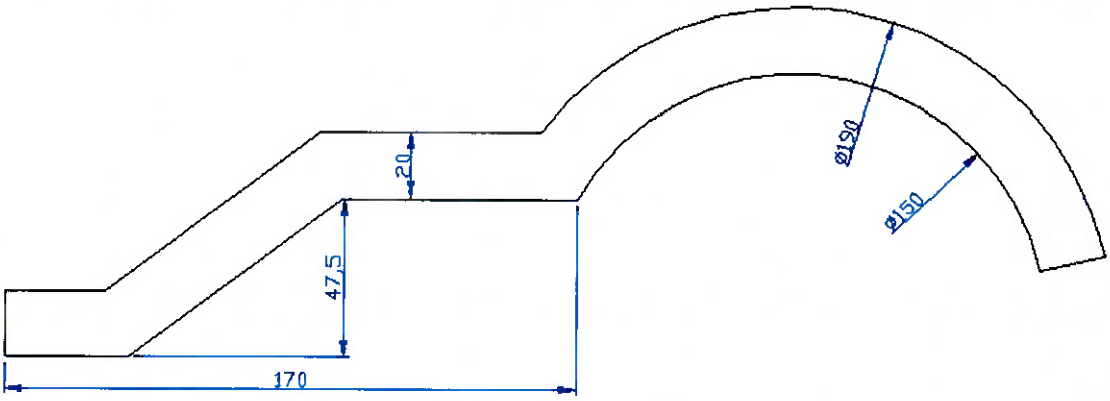


Figura 15 - Modelo inicial da peça do trilho

5.1.2 - Espaçamento

O espaçamento entre os trilhos deve ser pensado, e deve ser o mínimo possível. Definiu-se que, uma vez que o endoscópio (assim como seu suporte) possui em torno de 10 mm de diâmetro, um espaçamento interno de 20 mm entre os trilhos seria mais do que o suficiente para que o sistema possa ser montado corretamente. Menos do que isso acarretaria em mais dificuldades construtivas (no momento da ligação ao eixo de rotação).

Espaçadores devem então ser montados ao longo das duas peças que compõem os trilhos. Estes espaçadores serão também utilizados para guiar os cabos que efetuam a transmissão do movimento através do mecanismo. Por esse motivo, adotou-se pequenos cilindros metálicos, sendo que em cada 'dobra' devem ser instalados dois destes espaçadores, um para o cabo que vai e outro para o cabo que retorna.

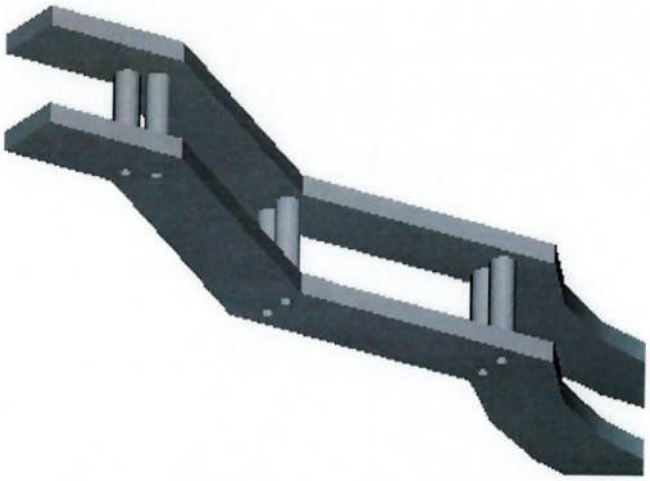


Figura 16 - Separadores cilíndricos para os trilhos

Ao final dos trilhos têm-se o ponto mais crítico do percurso do cabo. Neste ponto, o mesmo é forçado a fazer um retorno em uma curva próxima de 180° . Considerou-se que este é sem dúvida o ponto mais crítico para o cabo, e também para a fixação dos dois trilhos.

Desta forma, neste ponto, uma separação maior é requerida. Também se utilizará o separador como um eixo para uma polia, que terá a liberdade de girar livremente, proporcionando maior facilidade ao cabo para executar esta curva.

Figura 17 - Separação final dos trilhos, com polia para o cabo

5.1.3 - Carro

A parte mais difícil do projeto é, sem dúvida, o carro. Deve-se buscar manter as dimensões do mesmo as menores possíveis, de forma a possibilitar maior ângulo de movimentação e menor peso.

A forma construtiva mais simples para o carro é a de uma chapa dobrada em forma de U. De simples construção, esta chapa envolveria os trilhos, estando suportada pelo sistema de rodas do carro.

É importante, no entanto, que o atrito entre o carro e os trilhos seja minimizado da melhor forma possível. Por isso, considerou-se essencial a utilização de rolamentos para que o carro deslize livremente.

Os menores rolamentos encontrados foram rolamentos com diâmetro externo total de 8 mm, e com diâmetro interno de 3 mm. Estes rolamentos serão empregados dentro de um rolete de PVC, que girará. Um eixo fino de 3 mm passará pelo centro deste rolete (fixo ao rolamento), e será este eixo que ficará preso à carcaça do carro.

O desenho esquemático abaixo ilustra melhor a idéia para a construção destes roletes.

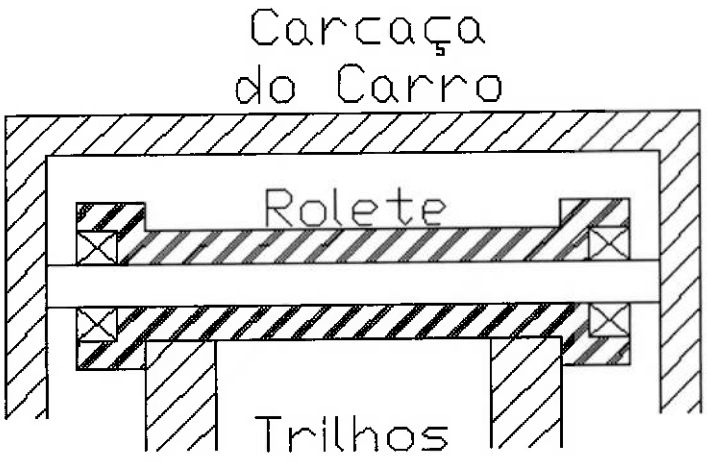


Figura 18 - Diagrama esquemático dos roletes do carro

Para o eixo que atravessa o rolete imaginou-se utilizar um simples parafuso. O rolete rodaria em torno do parafuso, que seria fixo por porca na carcaça do carro.

Seria necessária então a confecção de uma pequena bucha para ser fixada no parafuso e segurar a parte menor (diâmetro interno) do rolamento.

Também é necessário definir a forma de fixação do suporte do endoscópio no carro. Este será feito prendendo-se uma peça separada, com as guias, através de parafusos. Sobre as guias correrá a peça fixa ao endoscópio. Uma peça em L conduz o conduto com o cabo de aço. Nas guias é encaixada e presa uma peça em alumínio, à qual o suporte do endoscópio é realmente fixo, através de uma borboleta, facilitando assim a retirada da peça.

Girando para um lado, a peça, juntamente com o endoscópio, é puxada para baixo. Para o outro lado obtém-se o movimento inverso, levantando o endoscópio. A figura abaixo ilustra este sistema.

Figura 19 - Sistema de fixação e movimentação vertical do endoscópio

Já foi discutido que serão necessárias molas para absorver a variação no comprimento do cabo. O cabo de aço será fixo nestas molas, que por sua vez serão fixas ao carro.

Desta forma, imaginou-se uma chapa em L com um furo, pelo qual apenas o cabo passe. Assim, há uma limitação mecânica ao deslocamento da mola em si.

A figura abaixo fornece uma idéia básica do sistema imaginado.

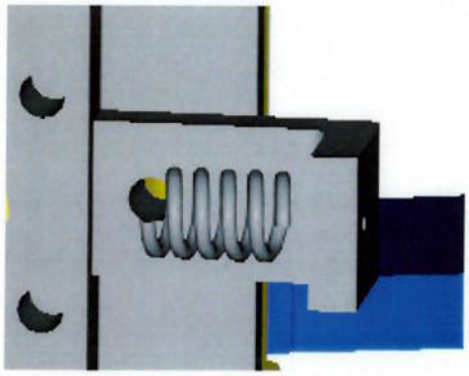


Figura 20 - Sistema de molas de fixação dos cabos ao carro

Por fim, ainda em relação ao carro, é preciso que haja um local para a volta do cabo. Este deve correr livremente pela parte de baixo do mesmo. Para isso, projetou-se uma nova dobra para dentro da carcaça do carro (uma vez que o cabo deve passar por dentro dos trilhos). Próximo à extremidade desta dobra deve ser fixado um trecho de conduite (o mesmo utilizado para o cabo que aciona o movimento vertical do endoscópio). Assim, o cabo é guiado a passar por ali.

Figura 21 - Retorno do cabo através do carro (parte inferior)

5.1.4 - Fixação ao Eixo

De posse de todo o projeto dos trilhos e dos carros, segue-se então à fixação dos trilhos ao eixo de rotação. Para o eixo selecionou-se um tubo de 16 mm de diâmetro (14 mm interno). A fixação das peças dos trilhos ao tubo se dá através de uma peça intermediária. Para evitar flexão, imaginou-se uma peça com rasgos laterais onde os trilhos se encaixem. A fixação em si se dá através de parafusos.

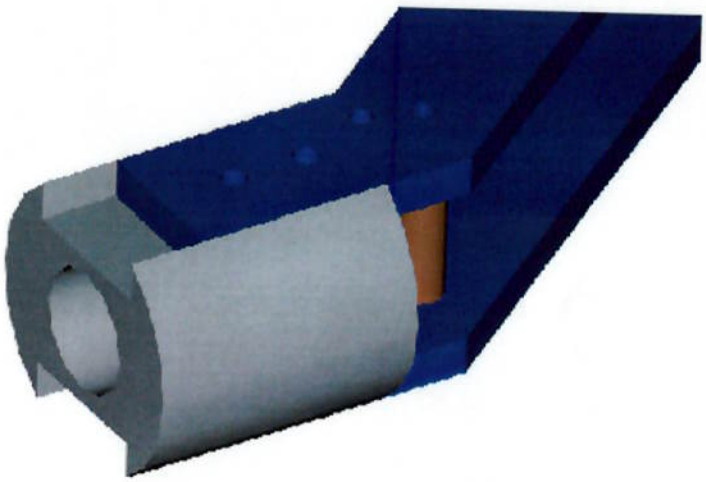


Figura 22 - Peça de fixação dos trilhos ao eixo

Aqui vale observar que os espaçadores que guiam os cabos os trazem exatamente para a boca do tubo. Isto porque imaginou-se passar os cabos (inclusive os conduites de acionamento vertical do endoscópio) dentro do tubo, com o objetivo de oferecer uma completa separação dos sistemas elétrico e mecânico por fatores de esterilização do equipamento.

5.1.5 - Caixa de Motores

Os motores elétricos de acionamento deverão ficar todos agrupados na outra ponta do eixo. Isso para garantir a separação dos sistemas elétrico e mecânico.

Este grupo de elementos deverá ficar dentro de uma caixa fechada, que pode ser construída com chapas finas de alumínio. Estas chapas podem ser muito finas pois seu objetivo não é sustentação, mas apenas garantir a separação dos motores no momento da esterilização.

O motor que fica logo abaixo do eixo é aquele que movimentará todo o sistema do eixo. A transmissão se dará através de correia dentada. Pode-se pensar que o peso do sistema e do motor superior podem causar problemas. De fato, este é o motor mais solicitado, mas ainda assim possui potência suficiente para movimentar o mecanismo.

O motor que está acoplado ao eixo é o motor que movimenta os cabos. Ele possui uma polia na qual as duas pontas do cabo são presas, após enroladas. Assim, rodando para um lado o motor efetua tração de um lado do cabo, e vice versa.

O motivo deste motor estar acoplado ao eixo é de que ele deve obrigatoriamente girar junto com o sistema. Isso porque sua polia deve girar juntamente com os trilhos, para manter a orientação e o percurso dos cabos constante.

O terceiro e último motor, abaixo do anterior, é o motor que têm como função única acionar o cabo que movimenta o endoscópio verticalmente. O cabo é preso a ele da mesma forma que no motor anterior, e conduzido por dois conduites que saem da parte superior de seu suporte e atravessam o eixo por dentro, levando o cabo até o local de acionamento.

O modelo abaixo fornece uma idéia da caixa dos motores como seria aberta.

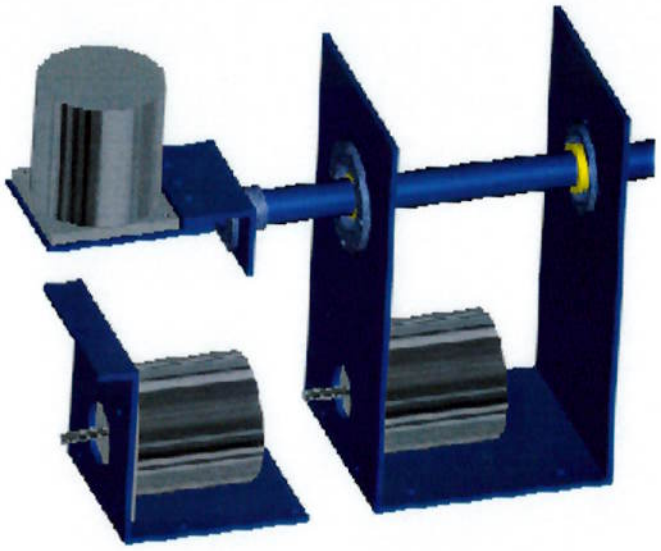


Figura 23 - Modelo da caixa dos motores aberta

5.1.6 - Fixação à Mesa

Ainda é preciso discutir a fixação do sistema como um todo à mesa de cirurgia. Imaginou-se que a melhor solução seria fixar o sistema à mesa, que é um objeto suficientemente rígido e o ponto de apoio mais próximo possível.

Sabe-se que existem comercialmente grampos (mecanismos parecidos com morsas) genéricos que são utilizados para fins como este. Assim, a solução mais simples seria a obtenção de um grampo de tamanho adequado e uma adaptação da caixa de motores para ser fixada ao grampo.

Outra solução é a construção de uma base totalmente independente para o manipulador, que conteria os motores, parte elétrica, entre outros. Esta base poderia ser posicionada ao lado da mesa cirúrgica no momento da operação.

5.2 - Análise Estática

Uma vez definido o mecanismo a ser utilizado, passou-se então a uma análise da satisfação deste mecanismo aos requisitos de projeto. O primeiro passo é analisar se o mecanismo suporta bem os esforços envolvidos.

Nos cálculos abaixo considerou-se que a peça dos trilhos é o grande limitante do sistema como um todo, uma vez que é a peça mais frágil, que suporta os esforços diretamente e encontra-se sem apoio na sua extremidade oposta ao eixo.

Sendo que o eixo de suporte, assim como o restante do sistema, é muito mais resistente, tanto à flexão quanto à torção, considerou-se para efeito de simplificação que as peças dos trilhos encontram-se engastadas em sua extremidade.

Assim, procedeu-se à análise das duas principais fraquezas dos trilhos, que são a flexão (devido a encontrar-se sem apoio na extremidade dos trilhos) e a torção (devido à inclinação que ocorre em torno do eixo).

5.2.1 - Flexão

Considerou-se que um dos pontos críticos do mecanismo seja a flexão no sentido de translação do carro, uma vez que todo o sistema se encontra em balanço.

Sendo assim, procedeu-se ao cálculo da deformação vertical, ocorrida devido a uma força vertical. Esta força representa o peso do sistema do carro, mais o endoscópio.

Sabe-se que o endoscópio pesa cerca de 500 gramas, no máximo. O sistema do carro, todo de alumínio e plástico, não ultrapassa também 500 g. Para máxima segurança, admitiu-se nos cálculos uma massa total de 2 kg, aproximando esta força para 20 N. Sendo que esta força será suportada por dois trilhos, fez-se o cálculo para um trilho apenas, com 10 N.

A força não foi colocada na extremidade extrema do trilho, mas sim a 30° da horizontal, pois é este o ponto máximo que o carro atingirá. Isto porque há um tamanho do carro a ser considerado, e ainda há o espaço no final do trilho para a instalação de uma polia (pela qual correrá o cabo de acionamento).

Espera-se que a deformação não ultrapasse 1 mm , mantendo perfeito controle do endoscópio.

Utilizou-se a forma média do trilho, de acordo com a figura.

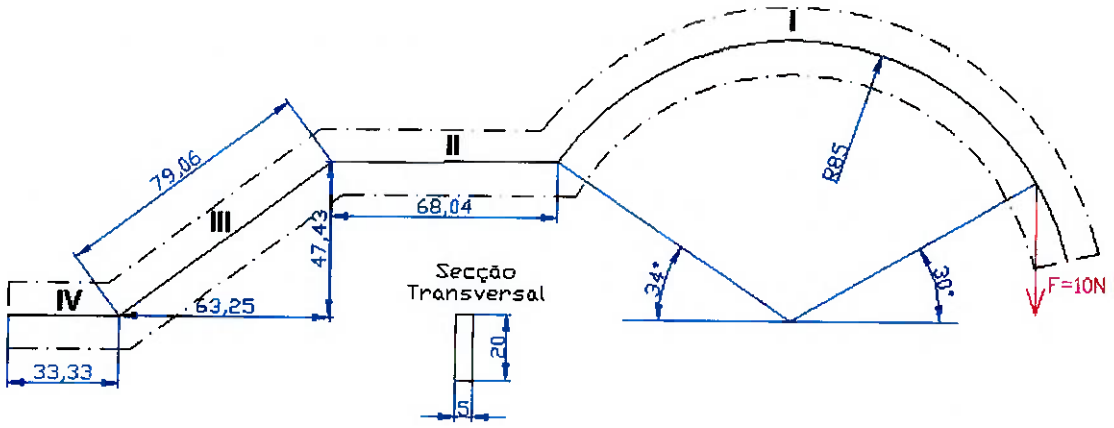


Figura 24 - Modelo para o cálculo da deformação por flexão

O cálculo da deformação pode ser feito através da expressão :

$$\delta = \int \frac{x \cdot M(x)}{E \cdot I_z} dx \qquad (1)$$

Por simplicidade, pode-se calcular a integral em separado para cada segmento do modelo e depois soma-las. Os segmentos estão indicados na figura.

$$\delta = \sum \int_0^{Xf} \frac{x \cdot M(x)}{E \cdot I_z} dx \qquad (2)$$

No segmento I, um cuidado especial deve ser tomado ao se integrar, pois a forma é circular. Assim, deve-se transformar a equação e trabalhar com o ângulo. O momento na seção circular pode ser calculado por :

$$M(\theta) = F \cdot R \cdot (1 - \cos(\theta)) \qquad (3)$$

Basta então transformar a variável X em θ e integrar a equação (1) com limites de integração de 30° a 146° .

Os segmentos II e IV são simples retas, e a equação do momento aqui é simplesmente a força (10 N) multiplicada pelo comprimento (a variável X, no caso). Isto acrescido, é claro, do momento acumulado até o segmento anterior.

No segmento III cabe a observação de que a força que ali causa momento não é 10 N, mas sim sua componente perpendicular ao segmento. Assim, deve-se calcular baseado nesta força, que vale 8 N, ao longo do comprimento total.

Os resultados dos cálculos são :

Segmento I - $(2{,}236 \cdot 10^{-2}) / E.I_Z$

Segmento II - $(4{,}647 \cdot 10^{-3}) / E.I_Z$

Segmento III - $(8{,}287 \cdot 10^{-3}) / E.I_Z$

Segmento IV - $(1{,}787 \cdot 10^{-3}) / E.I_Z$

Total - $0{,}037081 / E.I_Z$

O momento de inércia em torno de Z pode ser calculado através da seção transversal (representada na figura).

$$I_Z = \frac{b_s \cdot h^3}{12} \qquad (4)$$

Assim, chega-se ao valor :

$I_Z = 3{,}3333 \cdot 10^{-9}\ m^4$

O valor utilizado para E foi extraído de tabela, considerando alumínio.

$E = 70\ Gpa = 70 \cdot 10^9\ Pa$

A partir destes valores, chega-se finalmente à deformação vertical, no caso crítico em que todo o peso está sobre o final dos trilhos.

$\underline{\delta \cong 0{,}16\ mm}$

Nota-se que o valor é muito baixo, e de maneira nenhuma influi no funcionamento do mecanismo. Desta forma, o mecanismo projetado suporta bem a flexão.

5.2.2 - Torção

Quando o mecanismo todo se inclina de forma a oferecer a rotação em torno de seu próprio eixo, nasce uma força de torção nos trilhos resultante do peso do sistema do carro e do endoscópio.

Para avaliar esta torção se utilizará um trilho apenas. Isto foi adotado pois, no caso do mecanismo estar inclinado, o trilho que ficar abaixo suportará uma fração maior do peso do que o de cima. O de cima ainda suportará algum peso, pois o carro é preso a ele também, mas por simplicidade de cálculos e segurança adotou-se um trilho apenas suportando todo o peso.

Este peso já foi estimado para o caso da flexão. O endoscópio pesa, no máximo, 500 g, enquanto que o carro dificilmente chegará a este valor. Por segurança, adota-se 2 kg, ou 20 N.

Esquematicamente, têm-se a situação abaixo.

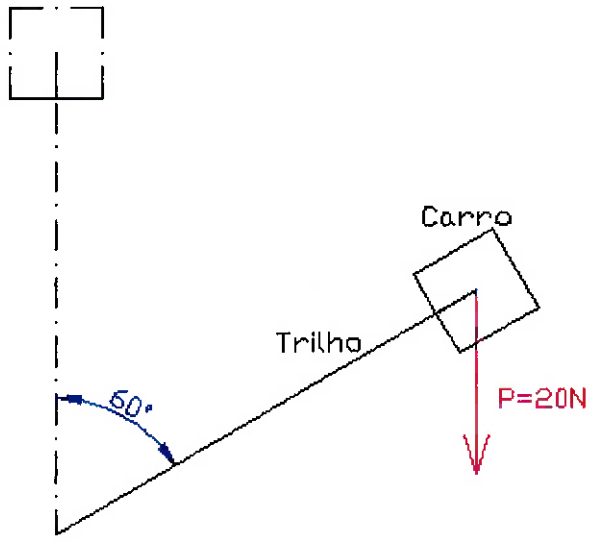


Figura 25 - Vista esquemática frontal para o cálculo da torção

Pode-se notar também que considerou-se, como inclinação máxima do sistema como um todo um ângulo de 60^{o} , satisfazendo os requisitos de projeto de inclinação máxima.

A torção pode ser calculada da forma :

$T_{or} = P.R.sen(60^o) = 2\ kgf\ .\ 8,5\ cm\ .\ 0,866 = 14,72\ kgf.cm$

As unidades utilizadas acima foram adotadas devido aos dados e fórmulas encontrados na literatura utilizarem as mesmas.

Vale dizer também que esta torção é a máxima torção (calculada com a carga no meio do trilho, ou seja, à maior distância do centro de rotação, que é a distância do raio médio).

O maior problema com o cálculo de torção neste caso é que a seção do trilho não é circular. Desta forma, torna-se impossível utilizar os métodos convencionais.

Isto porque os métodos convencionais admitem uma seção constante, com deformações em um plano. No caso de barras prismáticas, a seção não permanece plana, mas deforma-se, ocasionando tensões adicionais em alguns pontos e menos tensões em outros do que o esperado.

Em 1853, St. Venant criou um método para o cálculo de torção em barras prismáticas. Em seu método, ele determinou inicialmente os pontos de maior solicitação para cada tipo de seção por ele analisada, determinou também que nas pontas, de forma geral, a tensão deve ser nula (para haver continuidade), e então aproximou a distribuição das tensões por uma parábola. Desta forma, obteve uma equação para a torção em barras prismáticas.

No caso de barras com seção retangular, St. Venant chegou experimentalmente à seguinte relação :

$$\phi = \frac{T_{or}}{\beta \cdot b_s \cdot h^3 \cdot G} \qquad \phi = \gamma / l \tag{5}$$

Onde b_s e h são as dimensões da secção transversal e β é um fator tabelado, dependente de b_s e h. Esta fórmula é válida para $b_s/h > 1$. Para o caso do trilho analisado, com $b_s/h = 4$ (considerando $b_s = 4$ cm e $h = 0,5$ cm) têm-se da tabela elaborada por St. Venant :

$\beta = 0,281$

Têm-se também o G do alumínio :

$G = 0{,}28 \cdot 10^6 \text{ kgf/cm}^2$

Finalmente, é preciso informar o comprimento da barra prismática. No caso, aproximou-se pelo comprimento total da peça do trilho.

$l = 35{,}25 \text{ cm}$

Sendo assim, pode-se efetuar o cálculo da deformação angular por torção. O resultado da deformação angular em seu ponto máximo é :

$\underline{\gamma \cong 1{,}5^o}$

Pode-se dizer que este valor não é desprezível. O ângulo é aceitável, no entanto, e não compromete o funcionamento do sistema. Deve-se lembrar também que a carga será menor do que a utilizada nos cálculos, e dificilmente se utilizará ângulos de inclinação tão elevada. O comprimento estimado também é muito elevado, pois considera todo o trilho, mas isto em uma situação de torção máxima (que ocorre no meio do trilho).

5.3 - Análise Dinâmica

Há um ponto importante a analisar na movimentação do sistema. Mais especificamente, aqui cabe analisar o movimento do carro, que é realmente o único movimento com um maior grau de complexidade.

O problema no caso é a alteração do comprimento total do cabo de acordo com a posição do carro. Isto ocorre porque o cabo não percorre uma circunferência, mas sim um 'quase triângulo', que varia de acordo com a posição.

Tentar fazer com que o cabo percorresse uma circunferência foi considerado inviável, uma vez que implicaria em complexidade elevada do sistema dos trilhos. Assim, pensou-se em instalar molas para absorver esta diferença de comprimento. É preciso estimar esta diferença, no entanto.

5.3.1 - Variação do Comprimento do Cabo

Conhecer a diferença no comprimento do cabo, de acordo com a posição do carro, é essencial para o controle de posição e para a correta determinação das molas. Além disso, se esta variação de comprimento for muito alta, todo o sistema de transmissão de movimento por cabo pode ser descartado, ao menos da forma como foi imaginado.

A figura abaixo apresenta o modelo utilizado para estimar esta diferença de comprimento.

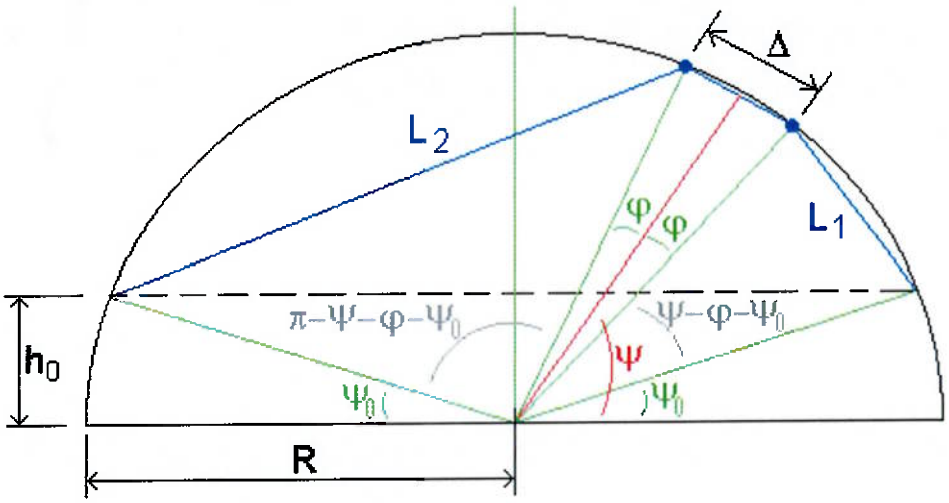


Figura 26 - Modelo para o cálculo do comprimento do cabo

O cabo está representado por uma linha azul, e possui três segmentos. Um antes do carro (1), um na passagem pelo carro e outro depois do carro (2).

Sendo assim, o comprimento do cabo nesta situação pode ser dado por :

$L_{TOT} = L_1 + \Delta + L_2$

Os comprimentos L_1 e L_2 podem ser obtidos geometricamente, em função de ψ. Para isso, por geometria, encontra-se :

$$\varphi = \arccos\left(1 - \frac{\Delta^2}{8 \cdot R^2}\right) \tag{6}$$

$$\psi_0 = arcsen\left(\frac{h_0}{R}\right) \tag{7}$$

A partir destes valores, pode-se obter os comprimentos desejados.

$$L_1 = R \cdot \sqrt{2 - 2 \cdot \cos(\psi - \varphi - \psi_0)} \tag{8}$$

$$L_2 = R \cdot \sqrt{2 - 2 \cdot \cos(\pi - \psi - \varphi - \psi_0)} \tag{9}$$

Os valores de R, h_0 e Δ são parâmetros do projeto e, a partir deles, pode-se obter assim o comprimento total.

Pelas expressões acima, pode-se provar que o máximo do comprimento é atingido a 90° e o mínimo é na parte mais inferior (próximo de 0° ou de 180°). Sendo assim, a diferença máxima no comprimento do cabo pode ser obtida subtraindo-se o comprimento a 90° do comprimento no ângulo limite de operação do carro.

Como o cabo segue um percurso total de ida e volta, deve-se considerar a diferença para ambos. Os raios percorridos pelos pontos de apoio, no entanto, são diferentes.

As diferenças calculadas estão abaixo.

D_{SUP} = 5,29 mm

D_{INF} = 3,88 mm

A diferença é maior na parte de cima pelo fato do cabo percorrer um caminho mais longo na parte superior (onde está fixo ao carro) do que na parte inferior. No cálculo para ambos os casos muda o valor de R, que não é o raio do trilho mas sim da circunferência que engloba o percurso do cabo.

O que se pode notar dos valores obtidos é que esta diferença de comprimento não é desprezível ou tão pequena quanto se pensava. As molas terão de absorver um total de 10 mm de diferença no comprimento. O projeto ainda é viável, no entanto.

Há outros problemas envolvidos nessa montagem. Embora, com o carro parado, as molas (caso idênticas, como se planeja) se estabilizem com uma igual deformação, há problemas na aceleração do carro. Neste caso, a deformação será desigual. E pior, em um primeiro instante de aplicação de torque pelo motor, a mola correspondente será muito solicitada, deformando muito e prejudicando o bom funcionamento do controle.

Para amenizar isso, a solução imaginada foi colocar limites mecânicos para as molas, de forma que as mesmas não possam sofrer grande deflexão (apenas a necessária para compensar a diferença no comprimento do cabo). Desta forma, com um limite mecânico de 5 mm para cada mola, obteria-se ainda uma precisão razoável no movimento.

Basta lembrar que 5 mm de comprimento na direção do trilho representam um ângulo de apenas 3° na posição do endoscópio. Assim, o controle por parte do médico não fica prejudicado no momento da movimentação do sistema.

5.4 - Desenhos de Conjunto e Fabricação

Após todas as considerações apresentadas, passou-se ao projeto final do mecanismo a ser implementado. O resultado final de tal projeto são os desenhos de conjunto, dando a idéia da montagem final do mecanismo, e os desenhos de fabricação, que fornecem os dados para que se fabrique as peças.

O Anexo I traz os desenhos de conjunto do mecanismo, onde se pode avaliar o sistema como um todo, compreender melhor seu funcionamento e sua montagem.

O Anexo II traz os desenhos de fabricação das peças e componentes do mecanismo, que possibilitam a execução mecânica do mesmo.

Finalmente, o Apêndice I traz as fotos do mecanismo de protótipo já montado e de seus componentes.

6. SELEÇÃO DE MOTORES

Após a definição da estrutura mecânica, deve-se dimensionar os motores responsáveis pelo acionamento da estrutura. Seguindo-se as matrizes de decisão apresentadas anteriormente, concluiu-se que os motores de passo seriam utilizados para a implementação do acionamento. Cabe ressaltar que os motores DC poderiam até, sob certas circunstâncias, apresentar um melhor resultado final. Porém, os motores de passo facilitam a implementação do *software* controle. Além do mais, podem ser citadas como vantagens:

- Alta precisão de movimentação e repetibilidade, com erro não acumulativo de um passo para o seguinte (acurácia de 3% a 5%);
- Ausência de contatos mecânicos deslizantes (escovas), prolongando a vida útil do motor e não produzindo faíscas, o que poderia representar um perigo em um ambiente inflamável como uma sala de cirurgias;
- Facilidade no controle de velocidade através do controle da freqüência dos pulsos de entrada.

Os maiores cuidados a serem tomados para a escolha dos motores de passo referem-se à possibilidade de se perder passos durante os movimentos de aceleração e desaceleração dos motores. A perda de passos poderia acarretar problemas relacionados ao controle de posição do sistema (perda de referência), uma vez que o sistema de controle adotado será em malha aberta. A escolha de uma malha fechada de controle de posição esbarraria no critério de facilidade de esterilização do conjunto, visto que para um controle em malha fechada é necessária a adoção de um sistema de realimentação da posição do endoscópio, o que implicaria na utilização de sensores de posição a serem distribuídos pela estrutura, bem como de uma complexa fiação e demais sistemas de fixação que dificultariam a esterilização. Logo, o controle em malha aberta simplifica o processo de construção e garante facilidade na higienização do mecanismo. Além do mais, uma escolha correta dos motores de passo para o controle em malha aberta permite a obtenção de uma precisão maior do que a que realmente será necessária de este projeto.

Para a adoção de uma malha de controle aberta, deve-se garantir que os motores de acionamento não perderão o sincronismo (perda de passos). Para tanto, deve-se estudar como ocorre a movimentação de um motor de passos quando ele é acionado por um trem, de pulsos.

Pode-se dividir o regime de operação dos motores em duas fases distintas:

- Fase de acelerações e desacelerações instantâneas, quando ocorrem grandes variações de velocidade;
- Fase de movimentação em regime constante , na qual os motores já venceram as inércias do sistema e as variações de velocidade são menores ou feitas de modo mais suave.

Para cada um destes regimes de funcionamento, os motores experimentam diferentes limites de rotação para um dado torque.

Existem duas classes de curvas de torque x rotação para um dado motor, uma para cada um dos regimes de funcionamento descritos acima. Para o caso de regimes de acelerações e desacelerações instantâneas, considera-se a curva de *pull in* do motor para se determinar a relação torque/freqüência aceitável, ao passo que para um regime de funcionamento em velocidades mais constantes ou para regimes de acelerações/desacelerações através de rampas, utiliza-se a curva de *pull out*. De uma forma geral, estas curvas representam os limites máximos de rotação que cada motor pode suportar sem perder passos, para cada um dos regimes de funcionamento, ou seja, sem perder o sincronismo. Ambas as curvas são decrescentes à medida que se aumentam as rotações dos motores. Os limites máximos para um motor operando em regime de *pull in* são menores que aqueles das curvas de *pull out*.

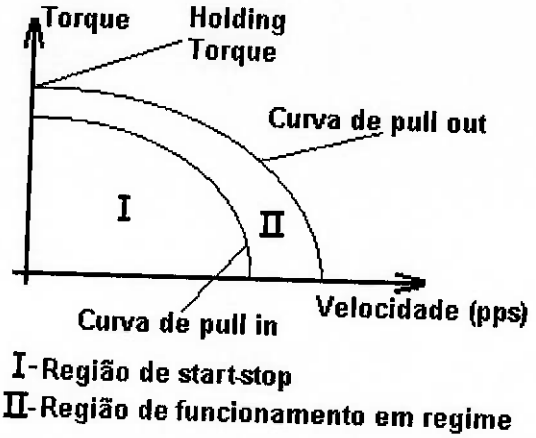


Figura 27 – Exemplo de curvas de um motor de passo

A determinação de tais curvas é dependente do motor que se está utilizando, do tipo de acionamento a ser adotado (*full step, half step* ou *micro step*) e do *driver* a ser utilizado para a produção dos pulsos. Logo, as curvas de *pull in* e *pull out* devem ser determinadas especificamente para uma aplicação particular.

No caso do mecanismo do endoscópio, deseja-se utilizar acelerações e desacelerações instantâneas, visto que o movimento não pode ser previsto (apenas o médico, com o auxílio do monitor, saberá onde o endoscópio deve parar), de forma que rampas de aceleração e desaceleração ficam inviáveis ou muito difíceis de serem adotadas. O controle deverá, portanto, ser capaz de absorver movimentos instantâneos. Desta forma, os motores devem ser capazes de operar dentro dos limites da curva de *pull in*. Para tanto, é preciso determinar o máximo torque a ser acionado pelo motor e, para tal torque, verificar qual o máximo valor de rotação permissível para o funcionamento dentro da curva de *pull in*.

A figura a seguir mostra como estão dispostos os motores ao longo do mecanismo. Cada um dos motores é responsável pela movimentação de um grau de liberdade do endoscópio.

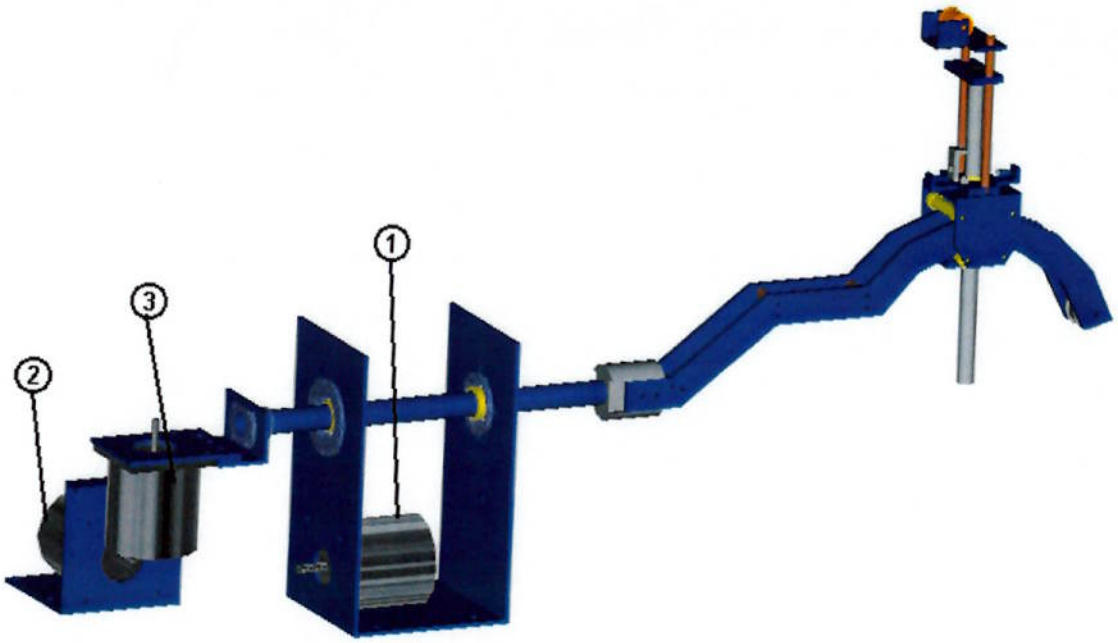


Figura 28 – Vista do manipulador do endoscópio completo

Cada um dos três motores do mecanismo acima estará sujeito a uma determinada carga. A análise será feita para os dois motores cujo grau de solicitação é mais grave, que são o motor responsável pela rotação do mecanismo como um todo (motor 1) e o motor responsável pelo movimento de *zoom* da lente (motor 2).

6.1 – Primeiro Motor : Rotação da Estrutura

Para o motor responsável pela rotação da estrutura, o torque a ser resistido corresponde ao torque necessário para manter o mecanismo parado numa determinada posição, sem contar o torque que deve ser fornecido par aceleração do sistema. Na situação mais crítica, o eixo de sustentação dos trilhos estará rotacionado de 90° em relação à posição de repouso, com os trilhos paralelos à base de sustentação. O carro de movimentação, na posição crítica, deve estar na posição mais distante do eixo de rotação, contribuindo com o maior torque correspondente à massa dos componentes que o constituem. Cabe ressaltar que esta posição jamais será alcançada na prática, pois a especificação de rotação ao redor deste eixo é de apenas 60° para cada direção. Logo os esforços resultantes serão menores que aqueles aqui calculados. A figura abaixo mostra a posição crítica adotada para o motor de rotação.

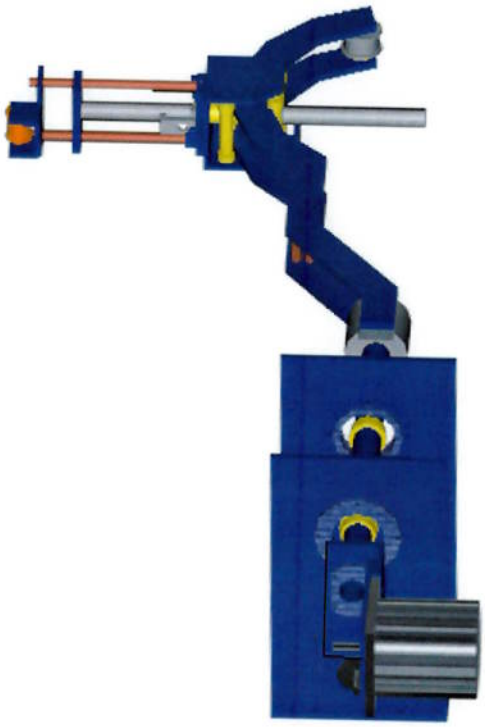


Figura 29 – Posição crítica para o primeiro motor

Observa-se que nesta configuração existe um balanceamento do torque causado pela massa do motor 3 e o torque causado pela estrutura 'carro+trilhos'. Desta forma, o torque a ser fornecido pelo motor 1, desconsiderando-se perdas nos

rolamentos, será apenas o necessário para equilibrar um possível resíduo não compensado e para colocar a estrutura em movimento.

A tabela abaixo mostra o cálculo dos torques que as peças exercem no eixo do motor.

Peça	Material	Volume (mm^3)	Massa (Kg)	Distância do eixo de rotação (mm)	Torque (N.m)
Carro	Al	18077,8585	0,048810	85	0,041488685
Trilhos	Al	74856,9917	0,202114	85	0,171796796
Roletes	Al	5051,6810	0,013640	85	0,011593608
Peças diversas	Al	9256,0346	0,024991	145	0,036237375
Guias	Cu	2582,2233	0,023137	145	0,033548245
Suporte Sup.	Al	6128,9929	0,016548	200	0,033096562
Bucha do endo	PVC	1553,1249	0,002190	85	0,001861420
Polia Superior	Latão	153,9335	0,001379	200	0,002758488
Polia	PVC	3233,0915	0,004559	26	0,001185251
				Total das peças	0,333566431
MOTOR	-	-	0,612000	55	0,3366
				Residual	-0,003033569

Tabela V - Cálculos de balanceamento da estrutura

Para o cálculo das massas, foram utilizados os volumes das peças fornecidas pelo AutoCAD. As distâncias em relação ao eixo de rotação representam as distâncias dos centros de massa das respectivas peças em relação ao eixo. Para as peças menores, foi utilizado o centro de massa de todas elas. O erro introduzido por este tipo de cálculo para estas peças é compensado pela pequena massa que possuem (pequena contribuição no torque). A análise mostra que a estrutura está praticamente completamente balanceada. O torque residual é muito pequeno (0,003 N.m). Além disso, as rotações de acionamento devem ser bem pequenas, o que garante que o torque a ser utilizado deve se aproximar bastante do *holding torque*.

Dispõe-se de três motores idênticos. As curvas de torque não estão disponíveis. Sabe-se, porém, através de alguns ensaios, que o torque médio na curva de *pull out* é de aproximadamente 0,3 N.m. Desta forma, o torque disponível é de pelo menos 100 vezes superior ao necessário.

Para assegurar que o torque realmente será suficiente e que a rotação máxima não ultrapassará 1 Hz no eixo que aciona os trilhos (valor adotado como referência máxima), foi adotada uma redução através de uma correia dentada (correia sincronizadora), a fim de se evitar perdas com deslizamentos. A redução escolhida foi de 3:1, o que garante que o torque fornecido pelo motor será amplificado em três vezes e que a rotação será reduzida em três vezes. Desta forma, o motor escolhido para a rotação da estrutura será plenamente capaz de ser acionado de forma instantânea sem deixar que aconteça perda de sincronia, ou seja, trabalhará dentro da curva de *pull in*.

6.2 - Segundo Motor : Movimento de *Zoom*

O movimento de *zoom* foi idealizado através de um acionamento por cabo de aço guiado por um duto condutor que direciona o cabo desde o eixo do motor até o carro onde o endoscópio vai ser fixado. No carro, existe uma estrutura de guias sobre as quais corre o suporte do endoscópio, e um suporte superior fixo para o retorno do cabo (*zoom out*). A figura abaixo mostra o mecanismo do movimento de *zoom*.

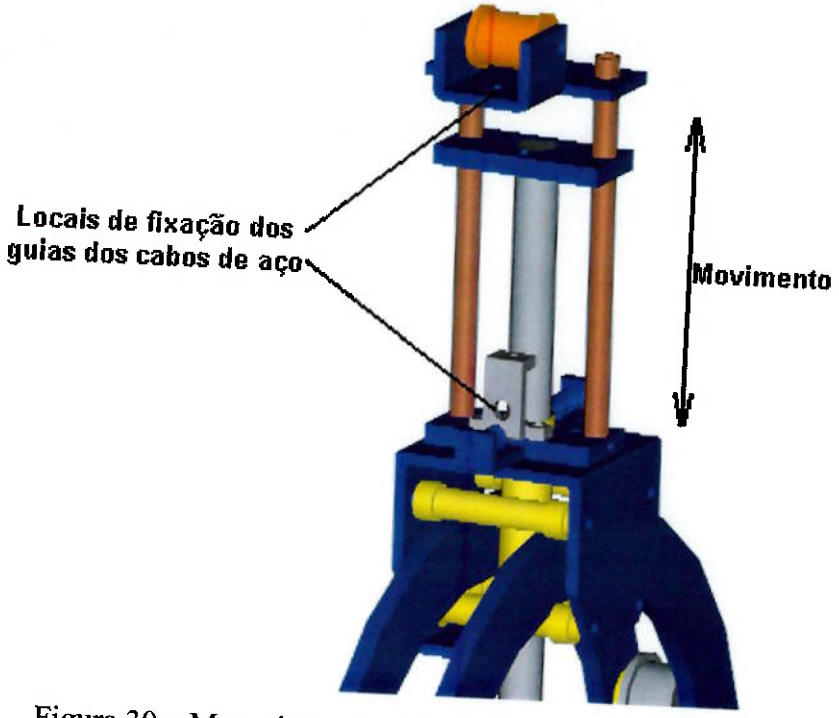


Figura 30 – Mecanismo de molas responsável pelo *zoom*

O movimento de *zoom in* será feito pelo motor 2, o qual está separado da estrutura principal. Ao ser acionado, este motor enrolará o cabo de aço em uma polia acoplada em seu eixo, fazendo com que a estrutura onde o endoscópio está preso se mova no sentido desejado.

Neste movimento, há a introdução de uma razoável resistência ao movimento devido aos tubos condutores do cabo. Isto porque o contato do cabo com as paredes do duto, em toda a sua extensão, provoca um atrito razoável que amortece o movimento. O motor deve ser capaz de vencer esta resistência.

O atrito entre o cabo e o conduto que serve de guia é proporcional ao tamanho do conduto. O conduto deve ter um comprimento total próximo de 1m. Será utilizado um tipo de lubrificante entre o cabo e o conduto (grafite por exemplo) capaz de minimizar ainda mais o atrito entre ambos. Experimentalmente estimou-se que o atrito máximo resulta em uma carga total de aproximadamente 72 N. Como segurança, será adotada uma carga nominal de 80 N.

Supondo que o motor forneça um torque útil de 0,3 N.m, o diâmetro da polia a ser instalada no motor deve ser:

$$\tau = F \cdot \frac{d}{2} \tag{10}$$

Onde :

F = Carga nominal

d = Diâmetro da polia

τ = Torque do motor

Para F = 80 N e τ = 0,3 N.m, deve-se ter um diâmetro de 4 mm. A fixação dos cabos nesta polia poderá ser feita por parafusos ou por algum tipo de cola, por exemplo.

6.3 – Terceiro Motor : Movimento do Carro Sobre o Trilho

O terceiro motor experimenta a menor solicitação dos três motores utilizados. Ele é responsável por movimentar o carro sobe os trilhos. Uma vez que o carro está apoiado sobre roletes com rolamentos e sua massa é bastante reduzida, o movimento experimenta um carregamento bem menor que os demais. A figura abaixo ilustra o arranjo a ser movimentado melo motor.

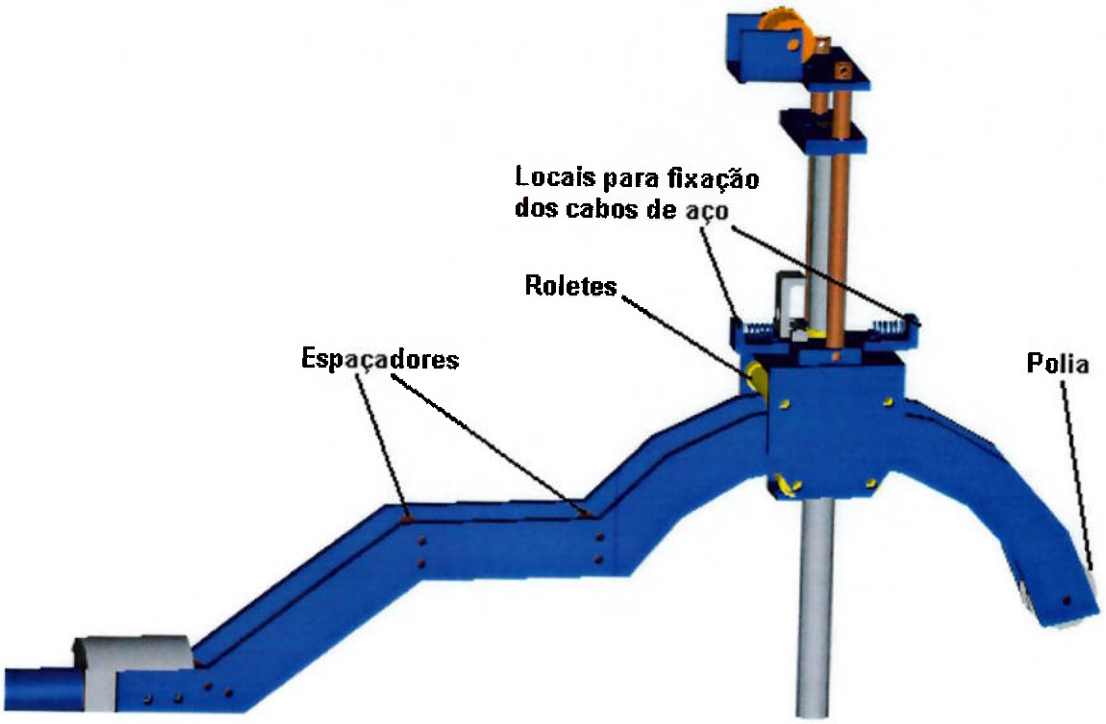


Figura 31 – Movimento acionado pelo motor 3

A transmissão do movimento do motor para o carro vai acontecer por um cabo de aço que será fixado nos locais indicados na figura acima. O cabo ficará exposto, ou seja, sem os condutos utilizados anteriormente. Neste caso, ele será guiado pelos espaçadores dos trilhos distribuídos ao longo da estrutura.

A montagem prevê a fixação de uma extremidade do cabo no fixador anterior do carro. Daí, ele passará pelos espaçadores superiores que servirão como guia até a entrada do tubo onde os trilhos são fixados. Daí ele será esticado por dentro do tubo e encontrará uma polia fixada ao eixo do terceiro motor na outra extremidade do tubo.

O cabo será conectado ao motor através de duas ou três voltas em torno da polia, com o objetivo de evitar escorregamentos no momento do acionamento. Em seguida, ele voltará por dentro do tubo e fará o caminho inverso pelos espaçadores inferiores até o carro. No carro, o cabo passará por uma ranhura feita em sua parte inferior, conforme mostra o detalhe abaixo.

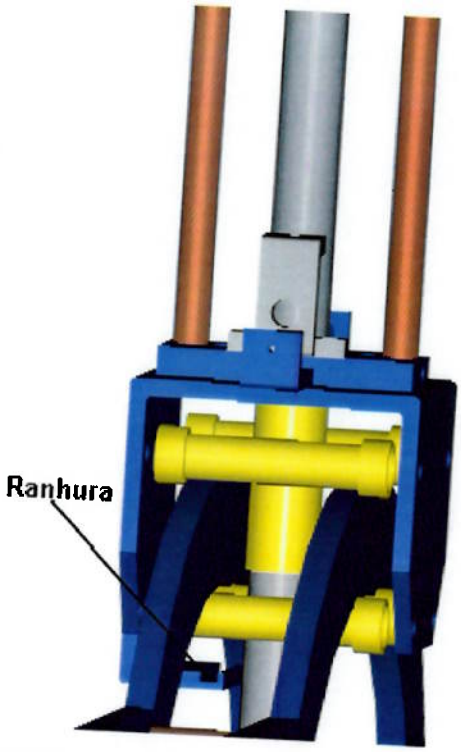


Figura 32 – Detalhe da ranhura no carro para passagem de cabo

Sobre esta ranhura haverá um pedaço do conduto por onde o cabo passará, a fim de minimizar o atrito no local. Ao deixar o carro, o cabo passará pela polia localizada na extremidade dos trilhos, revertendo sua direção e indo finalmente fixar-se ao fixador posterior do carro.

Os fixadores são formados por uma mola e uma peça em "L". A mola tem a função de compensar a diferença de comprimento que deve haver no cabo à medida que o carro desloca-se o longo dos trilhos. As peças em "L" servem para limitar a ação da mola, de modo que a extremidade oposta do carro àquela que está sendo solicitada por um acionamento não sofra uma queda de tensão no cabo, provocando nele uma folga.

Percebe-se, nesta montagem, que os pontos de atrito ocorrem principalmente nos pontos onde o cabo muda de direção (espaçadores e polia no fim do trilho) e na polia de acionamento. Este atrito, porém, deve ser bem menor que o atrito que ocorre, por exemplo, no mecanismo de *zoom*. Logo, as cargas e solicitações devem ser menores.

6.4 – Cálculo da Correia para Redução (Motor Principal)

Na seção anterior, adotou-se uma redução de 3:1 na transmissão de movimento do motor para a estrutura. Esta transmissão deverá ser feita através de uma correia dentada (correia sincronizadora). Por tratar-se de um serviço extra-leve para aplicações pequenas, a classe escolhida para trabalho foi a classe MXL (micro serviço leve).

O motor a ser utilizado para o acionamento possui um polia sincronizadora na ponta do eixo com n=26 dentes (pinhão). O passo desta polia é de p=0,8" (2,032mm). Daí conclui-se que o diâmetro primitivo desta polia deve ser de:

$$D_p = \frac{n \cdot p}{\pi} \tag{11}$$

Ou seja, $D_p = 16{,}825$ mm.

Como a redução adotada é de 3:1, a coroa deve ter N = 78 dentes com o mesmo passo p = 2,032mm, o que resulta num diâmetro primitivo $D_c = 50{,}45$mm. Polias sincronizadoras também são tabeladas e vendidas comercialmente. Porém, no catálogo utilizado como referência encontrou-se, para a classe MXL, polias de 72 dentes e de 80 dentes. Logo, escolheu-se uma polia de 80 dentes ($D_c = 51{,}74$mm), o que resulta numa redução de aproximadamente 3,077:1; que é muito próxima da redução calculada.

O dimensionamento da correia deve seguir o esquema abaixo:

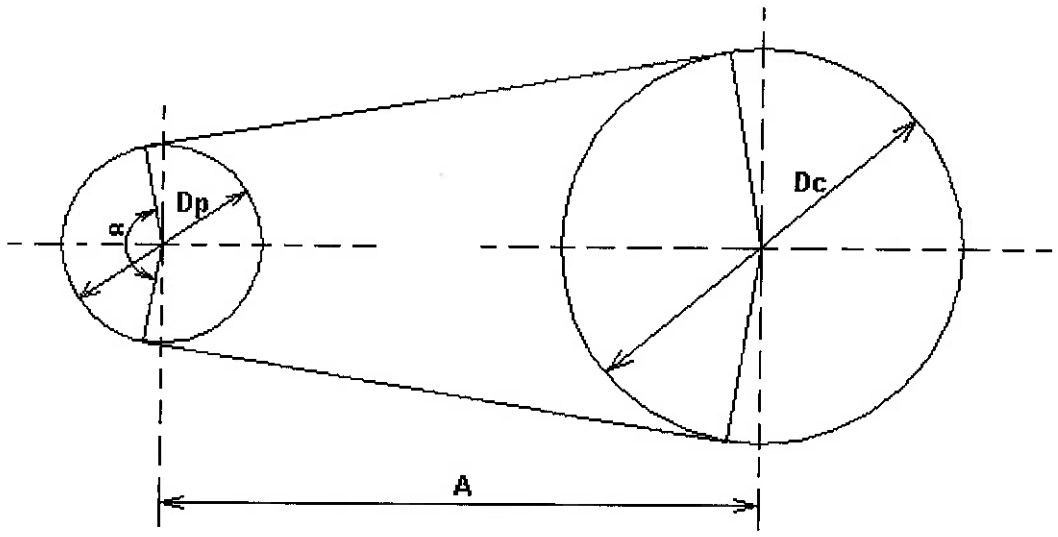


Figura 33 – Ilustração do acoplamento por correia

A distância entre os centros das polias A foi definida no momento do desenho como A = 96mm. A partir desta distância e dos diâmetros das polias anteriormente calculados ou definidos pode-se calcular o tamanho da correia L necessária. Para tanto, utiliza-se a relação:

$$L = 2 \cdot A + \frac{\pi}{2} \cdot (D_C + D_P) + \frac{(D_C - D_P)^2}{4 \cdot A} \tag{12}$$

Utilizando-se D_p = 16,825mm; D_c = 51,74mm e A = 96 mm, chega-se a L=302,859mm. Como o passo da correia deve ser de p = 2,032 mm, tem-se uma correia com N_c=149 dentes. Pelo catálogo, encontrou-se uma correia MXL com 150 dentes (1200MXL, L = 304,80mm). Com o tamanho desta correia definida, recalculamos a distância entre centros com os diâmetros já apresentados, e obtemos A' = 96,978 mm. Esta diferença de 0,978 mm pode ser compensada com a adoção de rasgos (oblongos) nos furos de fixação deste motor, permitindo ainda que a correia possa ser apertada, eliminando folgas.

7. PROJETO DO DRIVER DE MOTOR DE PASSO

Para que os sinais fornecidos pela porta paralela do microcomputador sejam transformados em real movimento dos motores, um driver de motor de passo é necessário.

Um driver de motor de passo nada mais é do que um circuito eletrônico que aciona um motor de passo. De uma forma geral, o circuito interpreta um sinal enviado (+5V ou 0V , sinal de controle) e aciona as bobinas (enrolamentos) dos motores.

Existem drivers comerciais disponíveis no mercado. Decidiu-se, no entanto, pela utilização de driver próprio a ser projetado, pela simplicidade do acionamento desejado e pelo alto custo e difícil aquisição de drivers comerciais.

7.1 – Acionamento do Motor

Sabe-se que o motor que se deseja acionar é um motor de passo unipolar de duas fases. Este é um tipo comum de motor de passo, sendo que cada uma de suas duas fases possui dois enrolamentos ligados entre si em série. Existem três conexões para cada fase, sendo duas das extremidades dos enrolamentos e uma central, no ponto de junção. Para acionar o motor alimenta-se o centro de uma fase com a tensão nominal do motor e alterna-se a posição do terra (destino da corrente) entre as duas extremidades. Quando se quer inverter a corrente, desliga-se uma extremidade e liga-se a outra ao circuito, trocando assim o enrolamento ativo e o sentido da corrente e do campo magnético resultante.

A figura abaixo ilustra de forma simplificada esta configuração.

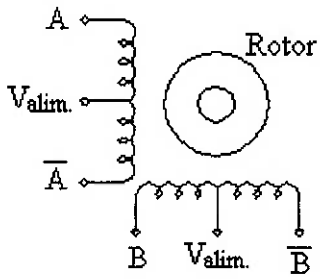


Figura 34 – Esquema simplificado de motor de passo unipolar

Um driver de motor de passo unipolar deve, assim, ser capaz de alimentar as quatro fases de forma ordenada de acordo com a entrada fornecida. O primeiro passo para isso é efetuar uma lógica de acionamento que receba os pulsos do sinal de entrada e os transforme em sinais de chaveamento de cada bobina, indicando se a mesma encontra-se acionada ou não. Assim, há quatro saídas, para as quatro partes dos enrolamentos (A, A^-, B e B^- da figura).

Uma segunda parte a ser implementada diz respeito ao circuito de potência em si. Comumente se utiliza um transistor de potência para cada saída, sendo chaveado pela lógica de acionamento. Quando fechado, o transistor permite a passagem de corrente para a bobina desejada. Uma ressalva importante é a necessidade de diodos colocados entre cada bobina e a alimentação das mesmas, uma

vez que ao se desligar esta alimentação a corrente tenderia a retornar pelo circuito, ficando restrita, no entanto, pelo diodo.

7.1.1 – Lógica de Acionamento

Por simplicidade, escolheu-se pela utilização de um circuito integrado pronto para efetuar a lógica de acionamento das bobinas do motor de passo. Isto porque a implementação de tal lógica adicionaria complexidade ao projeto, e porque se têm acesso a CI's que já a efetuam

O UCN4202A, fabricado pela 'Sprague' (atual 'Allegro'), é um exemplo de CI especialmente desenvolvido para o acionamento de motores de passo. Seu uso primário, para o qual foi desenvolvido, é o acionamento de motores de passo presentes em antigos discos rígidos utilizados em microcomputadores.

Vale notar que o UCN4202A já possui internamente, além da lógica de acionamento, os transistores e diodos necessários para o acionamento, sendo que se poderia simplesmente efetuar a ligação do CI diretamente às bobinas do motor.

Além de alimentação de controle (+5V) e de acionamento, este CI recebe dois sinais. Um deles é um sinal binário que indica a direção do movimento do motor. Invertendo-se este bit inverte-se a direção de rotação. O segundo sinal é a entrada "Step". É uma entrada sensível à borda de subida, sendo que sempre que uma borda ativá-la, o motor executa um passo. Normalmente alimenta-se esta entrada com uma onda quadrada com a freqüência necessária para se atingir a rotação desejada.

A figura abaixo mostra uma recomendação do fabricante para o acionamento utilizando-se este circuito integrado.

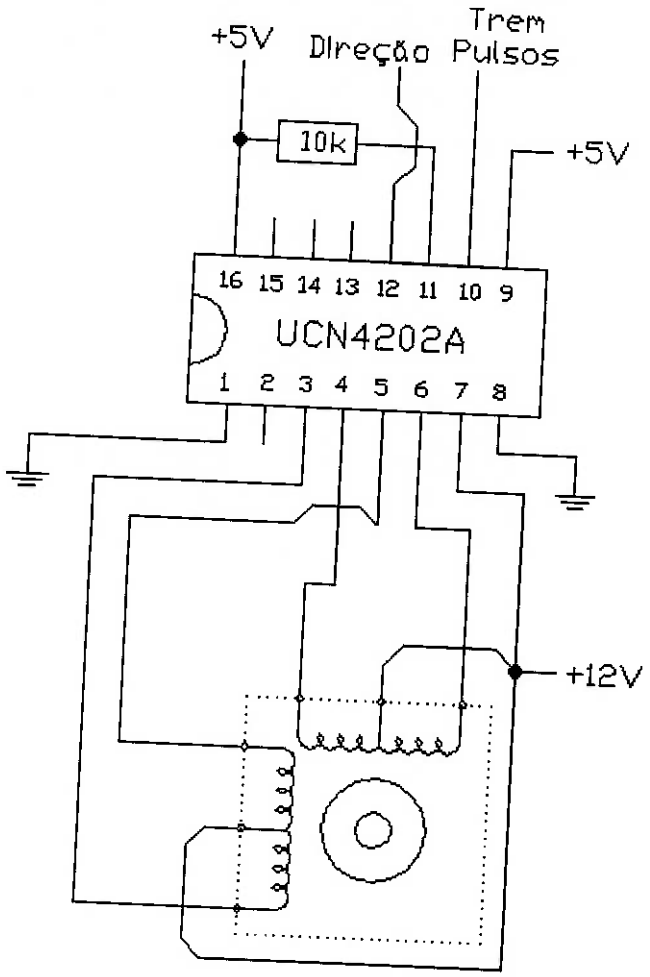


Figura 35 – Acionamento utilizando o UCN4202A

A alimentação do CI se dá nos pinos 8 (terra) e 16 (+Vcc). O pino 1 é o bit de habilitação da saída. A saída fica ativada quando este se encontra aterrado. O pino 9 é o bit de habilitação do disparo (step), e encontra-se ativado quando em nível alto (+5V). As duas entradas são nos pinos 10 (step) e 12 (direção), como indicado, e as saídas para as bobinas nos pinos de 3 a 6, sendo que os pinos 3 e 5 alimentam uma fase e os pinos 4 e 6 alimentam outra. O pino 7 é o pino onde se injeta a tensão de alimentação do circuito de potência interno do CI, ligado aos coletores dos transistores.

O pino 11, que encontra-se ligado à alimentação (+Vcc) através de um resistor de 10 KΩ, corresponde ao controle do monoestável, parte da lógica de controle, e sua ligação desta forma indica a utilização de 'Full Step' no acionamento

do motor. Neste modo de operação aciona-se sempre duas bobinas, de forma a obter mais torque.

Nota-se, no entanto, que o CI é dimensionado para trabalhar com uma tensão de potência de +12V, que é a fornecida a um disco rígido. Além disso, o fabricante recomenda se trabalhar sempre abaixo de 500 mA de corrente máxima nos transistores, que é o suficiente para o acionamento de um disco rígido.

O motor que se deseja acionar, no entanto, opera com condições nominais de + 4,2 V na alimentação das bobinas e 1,5 A de corrente nominal. Pode-se notar a diferença e a elevada corrente necessária, de forma que a utilização direta do UCN4202A não é viável. É preciso construir um circuito de potência para acionar os enrolamentos. Pode-se, no entanto, utilizar a lógica do CI para gerar os sinais de chaveamento deste circuito.

7.1.2 – Circuito de Potência

O circuito de potência para o acionamento de cada bobina é simples, e seu esquema geral pode ser visto na figura abaixo.

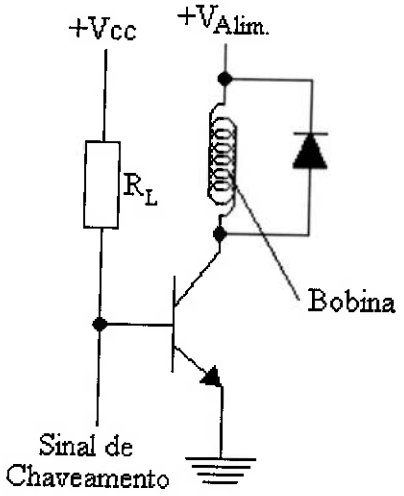


Figura 36 – Circuito de potência para uma bobina do motor

O sinal é ligado através de uma resistência ao nível lógico 1, de forma que o transistor chaveará quando o nível lógico do chaveamento for zero. O resistor deve ser dimensionado para atender à necessidade de corrente de chaveamento do transistor de potência.

Ao ter seu circuito fechado, o transistor permite a passagem de corrente através da bobina, ou seja, do ponto de alimentação (que fica entre as duas metades do enrolamento) até o terra.

O diodo presente em paralelo com a bobina é necessário para impedir o retorno de corrente para o circuito quando do desligamento da bobina. A corrente fica presa no circuito formado pela bobina e pelo diodo, dissipando-se ali.

Para efetuar a ponte entre os circuitos de controle e de potência escolheu-se o transistor de potência TIP41C, fabricado pela 'Fairchild Semiconductors'. Este transistor pode suportar correntes de até 6 A em tensões no coletor de até 100 V, sendo mais do que suficiente para a operação desejada.

Observando as especificações do transistor pode-se obter a corrente de chaveamento do mesmo. Esta corrente depende do ganho do transistor (o ganho entre a corrente na base e a corrente que vai do coletor ao emissor), e este depende, por sua vez, da tensão aplicada ao coletor.

No caso do motor a ser acionado, a tensão de alimentação será de 4,2 V. De acordo com a especificação, para 4 V se possui um ganho variando de um mínimo de 15 vezes até no máximo 75 vezes. Experimentalmente pôde-se notar que o ganho do transistor é alto, e de forma a não correr o risco de prejudicar o motor a resistência será dimensionada para um ganho de 70 vezes.

Sendo assim, procede-se ao cálculo da resistência R_L. Sabe-se que a corrente na bobina, ou seja, no coletor, deve ser de 1,5 A. Sabendo disso, com o ganho 70, obtém-se a corrente na base.

$I_b = 1,5\ A / 70 = 0,0214\ A = 21,4\ mA$

Assim, deseja-se aproximadamente 21,4 mA de acionamento na base do transistor. Sabendo que a tensão sobre a resistência é de +5V, e que a corrente nela incidente deve ser de 21,4 mA, pode-se calcular o valor da resistência.

$U = R_L.I$

$5\ V = R_L \cdot 21{,}4\ mA$

$R_L = 233{,}33\ \Omega$

Pode-se aproximar esta resistência por uma resistência de 220 Ω, disponível no mercado, deixando assim uma pequena sobra de corrente para o acionamento do transistor. Assim :

$R_L = 220\ \Omega$

Utilizando quatro acionamentos idênticos ao analisado obtém-se o acionamento de todos os quatro pontos de alimentação das bobinas do motor.

7.1.3 – Driver Final de Acionamento de Motor de Passo

De acordo com o que foi visto acima, pode-se construir assim um driver de motor de passo completo. Este driver será criado combinando um circuito integrado do tipo UCN4202A, com suas devidas ligações, a quatro circuitos de potência do tipo apresentado na seção anterior, com quatro transistores de potência do tipo TIP41C, quatro diodos e quatro resistores de 220 Ω.

O driver deverá ser montado em uma placa perfurada. Os componentes do circuito são colocados na parte superior da placa, sendo que as ligações são efetuadas por fios colocados na parte inferior da placa.

A placa disponível possui 16 x 35 perfurações, sendo mais do que o suficiente para a montagem do circuito. Uma placa menor não pode ser utilizada, no entanto, pois é necessário um razoável espaço físico para os dissipadores de calor dos transistores, que devem, assim, ficar bem espaçados.

O diagrama a seguir mostra o projeto completo de um driver de motor de passo unipolar do tipo a ser utilizado, já no formato com o qual deverá ser montado na placa.

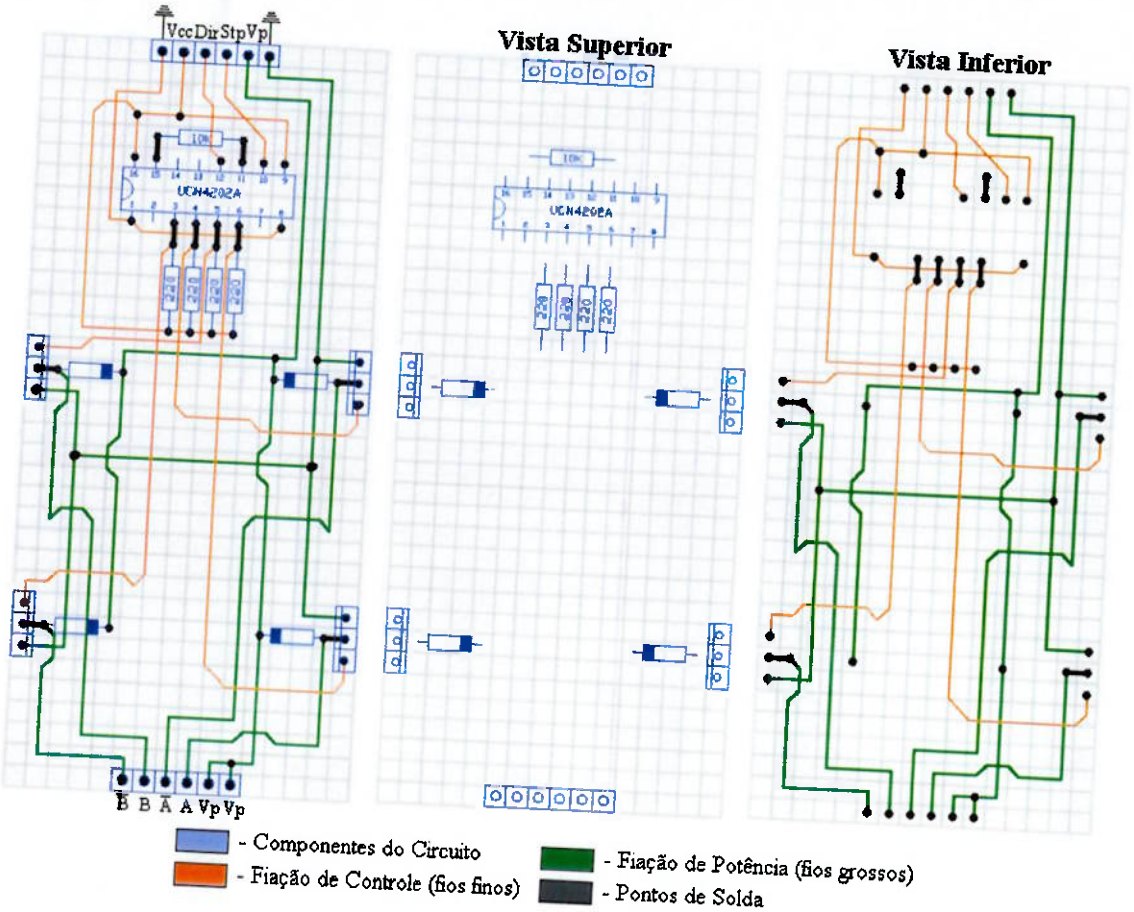


Figura 37 – Projeto final de driver de motor de passo

Na figura é apresentado o diagrama total do driver à esquerda, seguido das partes superior e inferior da placa mostradas separadamente. O objetivo desta separação é despoluir o diagrama e permitir uma melhor visualização do esquema construtivo.

Observa-se que há dois circuitos. O primeiro, indicado em laranja, é o circuito de controle, composto pela alimentação de controle e por fios finos e simples. A partir do acionamento dos transistores, o circuito torna-se um circuito de potência, indicado na figura em verde. Neste caso têm-se correntes maiores passando pelo circuito, e a fiação do mesmo deve ser mais robusta.

Este driver foi construído e testado com sucesso, sendo que as fotos correspondentes ao protótipo e sua montagem encontram-se no Apêndice I.

7.2 – Comunicação com o Microcomputador

A comunicação com o microcomputador (PC) se dará através da porta paralela. Isto porque a porta paralela é capaz de enviar simultaneamente 8 bits de dados, e o projeto irá precisar de seis (um bit de direção e um de step para cada um dos três motores).

O sinal da porta paralela não pode, no entanto, ser conectado diretamente ao CI controlador de motor de passo (UCN4202A). Isso porque o sinal enviado pelo computador é, em geral, razoavelmente inferior a 5V, prejudicando o funcionamento do sistema, e também porque um problema no circuito com a alimentação de potência do motor pode prejudicar enormemente a porta paralela e outros componentes do computador.

Para atender a estes dois quesitos, utiliza-se comumente no mercado dispositivos optoacopladores. Um dispositivo optoacoplador é um dispositivo que transmite o sinal (0 ou 1) de sua entrada para a sua saída através de luz, de forma a não existir ligação elétrica entre as duas partes. O funcionamento pode ser melhor compreendido com a ajuda da figura abaixo.

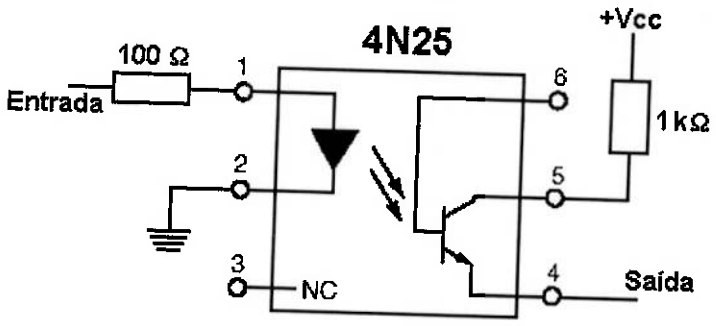


Figura 38 – Ligação comum com optoacoplador

O 4N25 é um optoacoplador comum no mercado. A ligação mostrada acima é a mais simples, onde o optoacoplador apenas transfere o sinal da entrada para a saída.

De forma básica, existe um diodo emissor de infravermelho que é ativado pelo sinal da entrada. Quando ativado, ele ativa, por luminosidade infravermelha, o foto-transistor à direita, permitindo a passagem de tensão para a saída.

As duas resistências colocadas no circuito servem para limitar corrente no CI.

Testes experimentais realizados com optoacopladores do tipo 4N25 mostraram, no entanto, que os mesmos não são aplicáveis ao projeto. Isto porque um dos sinais enviados pela porta paralela é um sinal em freqüência (um trem de pulsos), e os optoacopladores não transmitem de forma eficiente este tipo de sinal para freqüências mais elevadas.

A saída observada no osciloscópio não apresentava uma boa queda de tensão, permanecendo assim em +5V por todo o tempo. Isto ocorre provavelmente devido ao diodo fotoemissor passar a gerar luz infravermelha continuamente com freqüências mais altas, ou devido ao transistor não conseguir detectar as piscadas extremamente rápidas do diodo.

Desta forma, decidiu-se pela utilização de um buffer convencional para fazer o tratamento inicial dos sinais da porta paralela e oferecer proteção à mesma. O circuito integrado SN7407 é um buffer com seis canais de dados, podendo atender a seis sinais simultaneamente, e dessa forma se encaixa perfeitamente no projeto.

Por ser um buffer do tipo 'open-colector' é necessária a ligação da saída à alimentação através de um resistor limitador de corrente. A lógica desta ligação pode ser melhor compreendida com o auxílio da figura abaixo.

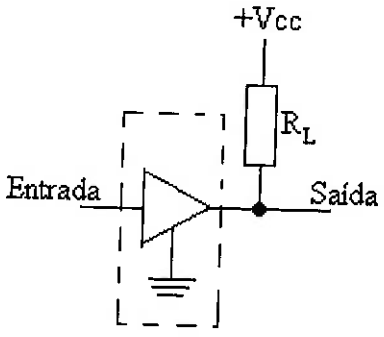


Figura 39 – Ligação na saída de buffer 'open-colector'

Caso a entrada não esteja habilitada, a corrente segue através da resistência para o terra, por dentro do CI. A saída fica portanto desabilitada. No caso da entrada estar habilitada, há um corte de forma que o terra do CI fica desconectado do circuito.

Assim, a corrente é obrigada a passar pela saída, e a saída fica habilitada. A lógica de um buffer como este é positiva, portanto.

De acordo com o fabricante, a corrente máxima suportada nas entradas do UCN4202A é da ordem de 40 mA. Desta forma, a resistência do circuito de saída do buffer deve garantir uma corrente mais baixa.

Adotando-se 20 mA, têm-se :

$U = R_L.I$

$5\ V = R_L\ .\ 20\ mA$

$R_L = 250\ \Omega$

Para garantir uma corrente um pouco superior a este valor, serão utilizadas resistências de 220 Ω neste circuito.

$R_L = 220\ \Omega$

O buffer SN7407 possui 14 pinos, sendo os pinos 7 (terra) e 14 (+Vcc) os pinos de alimentação do CI e os outros pinos correspondentes às entradas (pinos ímpares) e saídas (pinos pares).

A porta paralela de um PC convencional possui 25 pinos numerados, e seu endereçamento vai do endereço 378h até 37Ah (endereço hexadecimal da porta). Para o caso desejado, são importantes os pinos de 2 a 9, que compõem 8 bits exclusivamente de saída endereçados em 378h, e os pinos de 18 a 25, que são todos ligados ao terra do PC.

Escolheu-se arbitrariamente pelo uso dos pinos de 2 a 7 para a saída (seis sinais de saída, direção e 'step' de três motores), por comporem os seis bits menos significativos da palavra (byte) localizada em 378h.

Desta forma, pode-se montar o circuito da figura a seguir, também planejado para uma placa de 16 x 35 perfurações.

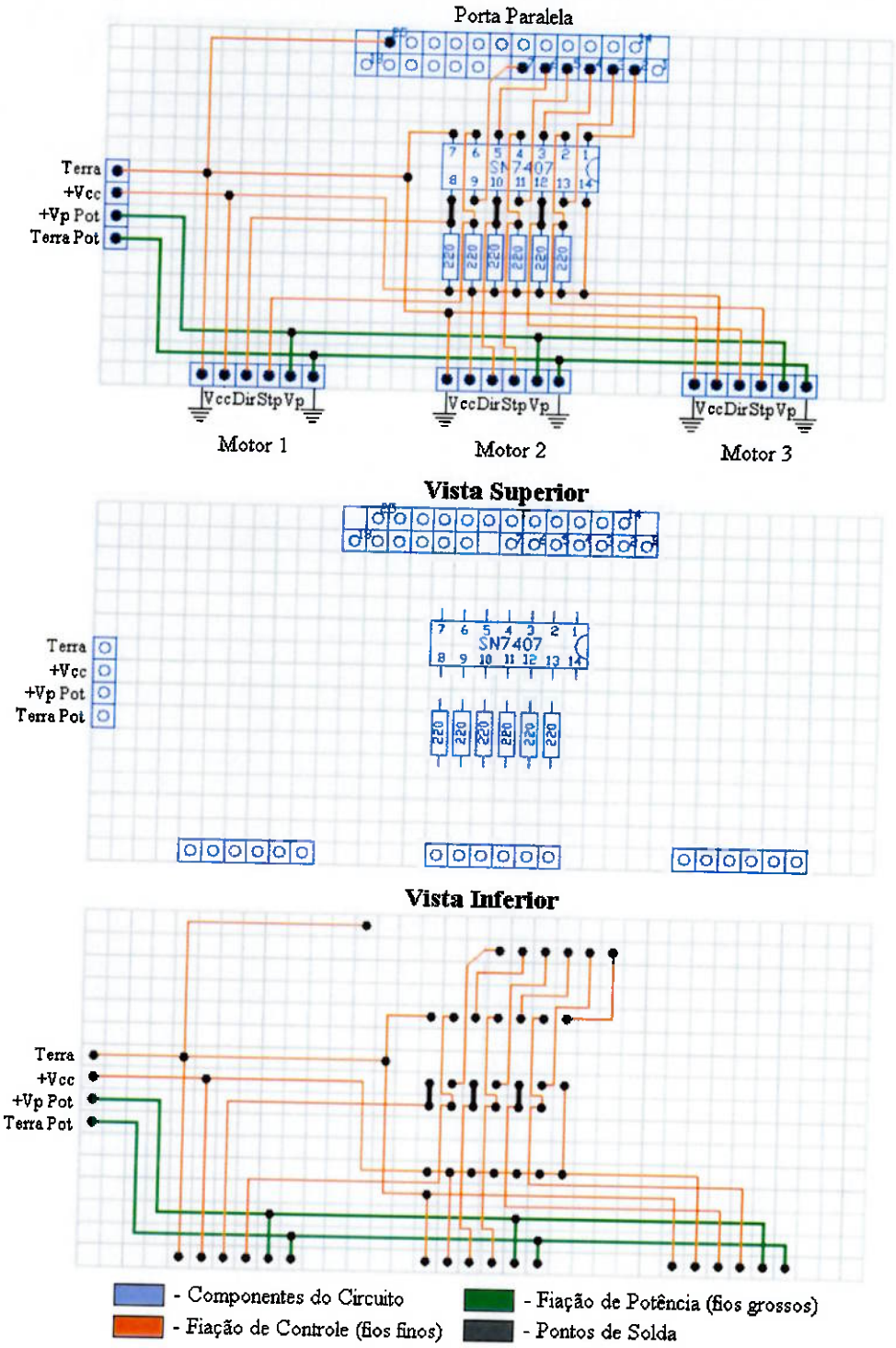


Figura 40 – Projeto final de comunicação com porta paralela

Mais uma vez há um circuito separado com fios grossos para o circuito de potência. Na verdade, não há potência envolvida na lógica desta placa, sendo que a alimentação de potência é simplesmente repassada aos drivers de motor de passo.

A alimentação é feita através de quatro pinos à esquerda da placa. O número de pinos é devido à separação da alimentação de potência da alimentação de controle. Esta separação visa, primeiramente, a utilização de cabos mais robustos em circuitos de potência e a proteção dos circuitos de controle, no caso de haver problemas no circuito de potência. Separando-se os circuitos atinge-se um grau maior de segurança.

Esta placa também foi construída em laboratório, tendo sido testada com sucesso na composição dos movimentos dos três motores. As fotos correspondentes encontram-se no Apêndice I.

8. PROJETO DO DISPOSITIVO DE ENTRADA DE DADOS

O dispositivo aqui chamado como dispositivo de entrada de dados é o conjunto de todos os dispositivos utilizados pelo médico durante a cirurgia para introduzir comandos ao robô. Neste caso, pode-se separar o dispositivo de entrada de dados em duas partes : o mouse de cabeça e o sistema de pedais.

O mouse de cabeça é a parte do dispositivo que será utilizada para que o cirurgião, com movimentos da cabeça, possa indicar sentidos de movimentação para o robô. Consiste em um mouse convencional adaptado (através de sistema de pêndulo) acoplado à cabeça do usuário.

O sistema de pedais é simplesmente uma base suportando dois pedais, que correspondem aos botões do mouse. A cada pedal é acoplado um botão de contato simples para detectar seu pressionamento. Existe um terceiro botão que não é acoplado a nenhum pedal (corresponde ao botão central do mouse de três botões) que serve exclusivamente para a centralização do mouse.

O projeto destes dois componentes é discutido abaixo.

8.1 – Mouse de Cabeça

O dispositivo de mouse de cabeça a ser implementado consiste basicamente na adaptação de um pêndulo à esfera localizada na parte inferior do mouse, cuja inclinação é transmitida para os encoders do mouse, os quais captam esta inclinação de acordo com a rotação da esfera na qual o pêndulo é fixado.

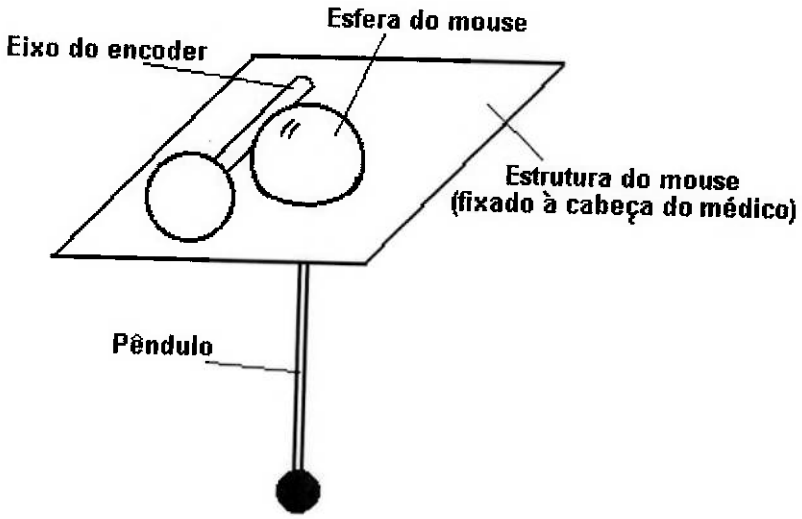


Figura 41 – Modelo do mecanismo do mouse de cabeça construído

O princípio de funcionamento é simples: à medida que o médico inclinar a cabeça para a esquerda ou para a direita, o pêndulo tenderá a permanecer na vertical, girando a esfera, a qual transmitirá esta rotação para um dos encoders. Caso a inclinação da cabeça do médico seja para frente ou para trás, o princípio de funcionamento é o mesmo, porém acionando o outro encoder.

A sensibilidade do encoder do mouse, ou seja, a relação entre o movimento de rotação da esfera e o deslocamento do cursor na tela, pode ser ajustada através do software de controle, conforme as necessidades do médico. O que deve ser projetado, portanto, são as características construtivas do pêndulo: sua massa e o comprimento da haste de fixação.

8.1.1 – Modelo do Mouse de Cabeça

Foi visto que o mouse de cabeça em questão deve ser modelado como um pêndulo. Para tanto, adotar-se-á um modelo semelhante ao da figura abaixo.

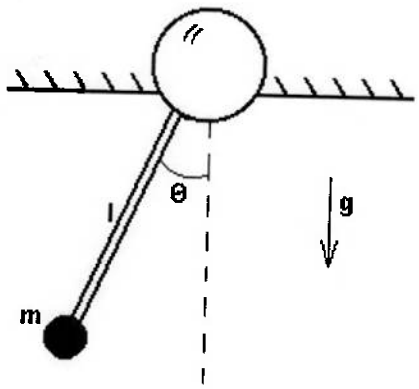


Figura 42 – Modelo físico do pêndulo do mouse

Observa-se neste modelo que existem duas forças de naturezas diferentes atuando no modelo: o peso da massa m do pêndulo e as forças de atrito que atuam na esfera do mouse (atrito com a estrutura, com os eixos dos encoders e com os dispositivos de fixação). A primeira força possui natureza conservativa, ao passo que a segunda é de natureza puramente dissipativa.

Deseja-se conhecer a relação entre estas duas forças e o comportamento do posicionamento relativo do mouse e da estrutura representado pelo ângulo θ. Para tanto, utilizaremos a mecânica Lagrangeana e as equações que ela fornece.

A primeira equação de Lagrange mostra que:

$$\frac{d}{dt}\left(\frac{\delta T}{\delta \dot{p}}\right) - \frac{\delta(T-V)}{\delta p} = -Q \qquad (13)$$

Onde:

T = Energia cinética do pêndulo

V = Energia potencial do pêndulo

p = coordenada generalizada do mecanismo (no caso em estudo, p = θ)

Q = Forças de natureza não-conservativa (atrito)

De acordo com a equação acima, consegue-se estabelecer uma relação entre a resposta dinâmica do sistema e a posição angular do pêndulo. No sistema real, poderão ocorrer inclinações em duas direções simultaneamente. A modelagem levará em conta apenas os movimento do pêndulo num único plano perpendicular ao eixo de um dos encoders. Essa consideração não causa perda de generalidade, pois os movimentos em ambas as direções são independentes. O valor da força não conservativa pode ser modelada como sendo exclusivamente dependente da velocidade angular do pêndulo, de forma que, sem perda de generalidade, pode-se escrever :

$$Q = \dot{\theta} \cdot b \tag{14}$$

Onde b é a constante de amortecimento angular de atrito presente no sistema. Esta constante, como dito anteriormente, depende exclusivamente do arranjo do mecanismo.

Primeiramente deve-se, então, escrever as relações de energia cinética e potencial para o sistema apresentado. Tem-se então:

$$T = \frac{m \cdot l^2 \cdot \theta^2}{2} \tag{15}$$

$$V = m \cdot g \cdot l \cdot [1 - \cos(\theta)] \tag{16}$$

Onde :

m = Massa do pêndulo (na modelagem, desprezar-se-á a massa da haste)

g = Aceleração da gravidade (será adotada $10 m/s^2$);

l = Comprimento da haste do pêndulo;

θ = Inclinação do pêndulo em relação à vertical;

Substituindo as relações (15) e (16) em (13) e fazendo-se as operações necessárias chega-se à seguinte relação:

$$\ddot{\theta} + \frac{g}{l} \cdot \text{sen}(\theta) + \frac{\dot{\theta} \cdot b}{m \cdot l^2} = 0 \tag{17}$$

Observa-se que o sistema obtido é um sistema de segunda ordem não linear. A resolução de tal sistema é bastante complexa. Como uma primeira aproximação, pode-se linearizá-lo de modo que sen(θ) = θ. Sabe-se que esta aproximação é válida para ângulos pequenos, o que não é o caso do presente sistema. Porém, esta

abordagem serve como uma primeira aproximação. Adotando esta simplificação, o sistema fica:

$$\ddot{\theta} + \frac{b}{m \cdot l^2} \cdot \dot{\theta} + \frac{g}{l} \cdot \theta = 0 \qquad (18)$$

As respostas de um sistema linear de 2ª ordem são conhecidas para diferentes tipos de entradas. O caso de interesse para o presente projeto corresponde a uma entrada degrau relativa à inclinação da cabeça do médico. O degrau de interesse para o projeto será adotado como uma força exercida pela cabeça do médico capaz de provocar uma inclinação de 25° no pêndulo. Essa força atuará sobre a base, que estará fixa a um boné ou capacete, de modo que a inclinação da cabeça do médico seja repassada ao sistema.

8.1.2 - Análise de Sistemas de 2ª Ordem

Um método bastante utilizado para análise de sistemas de 2ª ordem é a Transformada de Laplace. Utilizando esta ferramenta para este tipo de sistema, pode-se escrever a relação entre a saída Y e a entrada U através da seguinte função de transferência:

$$\frac{Y}{U} = \frac{\omega_n^2}{s^2 + 2 \cdot \xi \cdot \omega_n \cdot s + \omega_n^2} \qquad (19)$$

Onde:

$$\omega_n = \sqrt{\frac{g}{l}} \ ; \text{ freqüência natural de oscilação do pêndulo} \qquad (20)$$

$$\xi = \frac{b}{2 \cdot m \cdot l \cdot \sqrt{g \cdot l}} \ ; \text{ coeficiente de amortecimento do sistema} \qquad (21)$$

No caso do sistema em estudo, a saída Y é representada pelo ângulo com a horizontal e a entrada U é uma força externa que excita o sistema.

O gráfico seguinte ilustra como é a saída deste sistema para uma entrada degrau unitário.

Figura 43 – Resposta a uma entrada degrau de um sistema de 2ª ordem

É necessário conhecer as constantes do sistema para sua completa determinação. Dentre elas, a constante de amortecimento angular do pêndulo b é a de mais difícil obtenção.

Para obtê-lo, será utilizado o método do decaimento logarítmico. Este método baseia-se na resposta de um sistema de segunda ordem quando há amortecimento, resposta esta que pode ser vista na figura a seguir.

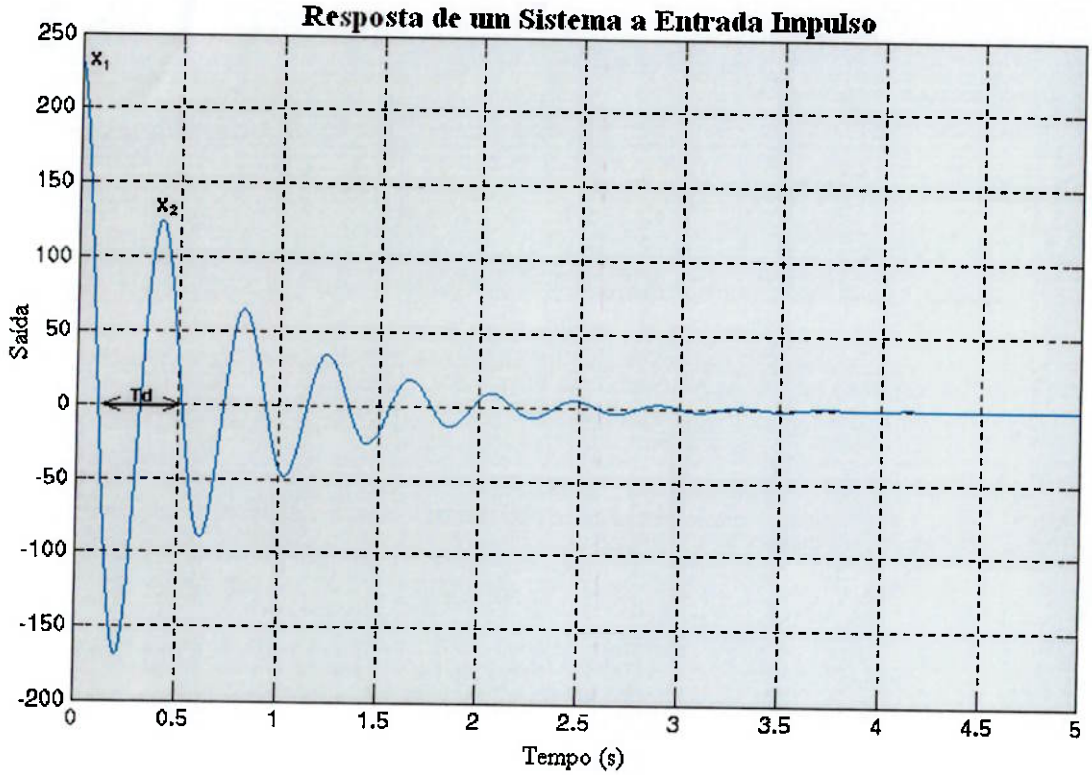


Figura 44 – Resposta a uma entrada impulso de um sistema de 2ª ordem

Sabe-se que os valores de pico fazem parte de uma envoltória que é descrita por uma exponencial do tipo :

$$X_p = K \cdot e^{-\xi \cdot \omega_n \cdot t} \tag{22}$$

Sabe-se também que a freqüência natural amortecida pode ser calculada por :

$$\omega_d = \omega_n \cdot \sqrt{1-\xi^2} = \frac{2 \cdot \pi}{T_d} \tag{23}$$

Através destas duas expressões, chega-se à expressão do decaimento logarítmico para o sistema em estudo:

$$\ln\left(\frac{x_2}{x_1}\right) = \frac{-2 \cdot \pi \cdot \xi}{\sqrt{1-\xi^2}} \tag{24}$$

Onde x_1 é o primeiro pico de amplitude e x_2 é o segundo.

8.1.3 – Determinação das Dimensões do Pêndulo

Através de um ensaio, utilizando-se então o método do decremento logarítmico, pode-se determinar experimentalmente a constante de amortecimento do sistema a ser utilizado. O ensaio consiste em montar-se um pêndulo com parâmetros conhecidos e, através destes parâmetros, analisar como se comporta o atrito na interface entre a esfera e os encoders por meio da avaliação de seu comportamento frente a oscilações livres (solta-se o pêndulo a partir de um ângulo θ conhecido e deixa-se o mesmo oscilar de forma unidimensional). É necessário observar que, uma vez que esta interface não mudará, para qualquer que seja o pêndulo escolhido, a relação de atrito deve permanecer aproximadamente constante. Contudo, podem ocorrer mudanças no coeficiente b devido às deformações da esfera, ou seja, o coeficiente b acabaria dependendo também da massa do pêndulo. Esta relação, porém, não será considerada com o intuito de simplificar a obtenção dos parâmetros desejados.

Para o ensaio, adotou-se m = 0,120Kg e l = 0,16m. Com estes parâmetros, montou-se a seguinte tabela:

	Primeira Amplitude de pico (Graus)	**Segunda Amplitude de pico (Graus)**	**Período (s)**
Medição 1	30	25	0,63
Medição 2	30	25	0,56
Medição 3	30	25	0,76
Medição 4	30	25	0,69
Média	**30**	**25**	**0,66**

Tabela VI – Tabela com dados ensaiados para o pêndulo

A primeira amplitude corresponde de onde o pêndulo foi solto para iniciar o movimento. A segunda amplitude corresponde ao máximo valor angular que o pêndulo atingiu após a realização de um ciclo de oscilação, e o período indicado é o tempo gasto para que um ciclo de oscilação seja realizado.

Aplicando-se o método do decremento logarítmico aos dados da tabela obtém-se :

$\xi = 0{,}029$

Com os valores do comprimento l e da massa m adotados, pode-se, através da equação (21) encontrar o valor de b, obtendo-se :

$$b = 1{,}409 \cdot 10^{-3} \frac{N \cdot s}{rad}$$

Em seguida, passa-se à determinação dos valores de l e m a serem implementados no mecanismo.

Existem várias condições de contorno utilizadas para a caracterização deste tipo de resposta envolvendo as constantes do sistema.. Dentre eles estão: sobre-sinal máximo (M_p); tempo de subida (t_r); instante de pico; tempo de acomodação e tempo de atraso. Para o projeto, os dois primeiros parâmetros são os de maior interesse e, através deles, os parâmetros que identificam o sistema serão especificados.

Para o máximo sobre-sinal M_p, tem-se a seguinte relação:

$$M_p = e^{\frac{-\pi \cdot \xi}{\sqrt{1-\xi^2}}} \qquad (25)$$

O máximo sobressinal mostra quanto é o máximo desvio da saída sobre o valor em regime. A especificação deste valor depende do encoder do mouse a ser utilizado. Existem limitações físicas construtivas que impedem o aparecimento de um sobressinal muito grande, o que facilita a modelagem.

O mouse adotado é um mouse de 520 dpi (divisões por polegada). O encoder deste mouse é dividido em 35 'dentes', de forma que o menor deslocamento angular possível de ser monitorado é de 10°. Adotou-se um máximo sobressinal correspondente a 90°, de forma que a regulagem da sensibilidade via software consegue acomodar este valor.

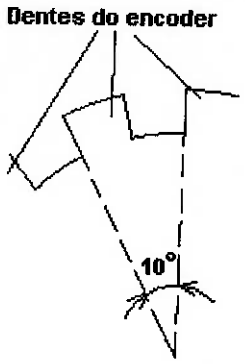


Figura 45 – Resolução angular do encoder do mouse

Para saber o quanto é necessário girar o pêndulo para se conseguir esta resolução, basta considerarmos a transmissão de movimento que há entre a esfera (onde o pêndulo está engastado) e o eixo do encoder. Supondo não haver deslizamento no contato entre estes dois elementos, a relação entre o deslocamento angular do pêndulo θ e o deslocamento angular do encoder α é dada pela relação:

$$\theta = \frac{d}{D} \cdot \alpha \tag{26}$$

Onde d é o diâmetro do eixo do encoder e D é o diâmetro da esfera.

Medindo-se estas duas dimensões no dispositivo obtém-se d=2,7 mm e D=27 mm. Desta forma, a resolução a ser adotada de 90° no encoder corresponde a uma rotação de 9° do pêndulo. Como o degrau unitário do projeto foi definido como sendo 25°, este valor de rotação corresponde a 36% do valor em regime, ou 0,36. Portanto, utilizando-se como sobressinal máximo um valor de 0,36 e aplicando a relação (25), chega-se a um coeficiente de amortecimento de ξ=0,30926.

A definição da freqüência natural ω_n é obtida através da expressão do tempo de subida, que é dada por :

$$t_r = \frac{\pi - \beta}{\omega_n \cdot \sqrt{1-\xi^2}} \tag{27}$$

Onde :

$$tg\ \beta = \frac{\sqrt{1-\xi^2}}{\xi} \tag{28}$$

A especificação do tempo de subida é mais subjetiva, e deve ser feita levando-se em conta o conforto do médico ao lidar com o sistema. O software de controle não possui limitações em relação a este parâmetro e, portanto, o que deve ser levado em consideração é mesmo o tempo percebido pelo médico para a resposta que ele deseja. Observa-se que, pelas equações (20) e (27), para um dado amortecimento fixo, o tempo de subida é diretamente proporcional ao comprimento da haste do pêndulo. Logo, tempos de subida muito curtos exigem curtos comprimentos desta haste. Por outro lado, uma vez que o amortecimento está definido, percebe-se pela relação (21) que menores comprimentos de haste requerem uma maior massa do pêndulo. Isto é ruim para o médico, pois torna o controle mais difícil, aumentando a carga sobre seu pescoço e causando desconforto, visto que existem cirurgias que podem durar várias horas.

Pode-se concluir por esta modelagem, portanto, que a definição das grandezas que caracterizam este sistema depende basicamente de um compromisso entre a massa do pêndulo e o comprimento da haste. As especificações adotadas devem então possuir soluções possíveis de serem implementadas para o modelo, de uma forma que a execução do projeto torne-se possível.

Para a especificação do tempo de subida, considerou-se suficiente que o pêndulo fosse capaz de subir de um patamar de 10% do sinal em regime até 90% deste sinal em 0,13 s. Este tempo foi obtido através de simulações com o dispositivo real e algumas considerações funcionais. Com o auxílio da relação (27) e do coeficiente de amortecimento anteriormente obtido, chega-se a um valor de freqüência natural de ω_n = 15,2492 rad/s. Observa-se que se trata de um valor bastante alto, o que assegura que o sistema em funcionamento jamais chegará perto desta ordem de grandeza em uma situação normal de funcionamento, livrando o sistema de possíveis problemas de batimento.

Portanto, os dois parâmetros que caracterizam este sistema de segundo grau e que atendem às especificações de projeto são:

$\omega_n = 15,2492$ rad/s

$\xi = 0,30926$

Simulando-se a resposta deste sistema a uma entrada em degrau, obtém-se o seguinte gráfico de resposta em função do tempo:

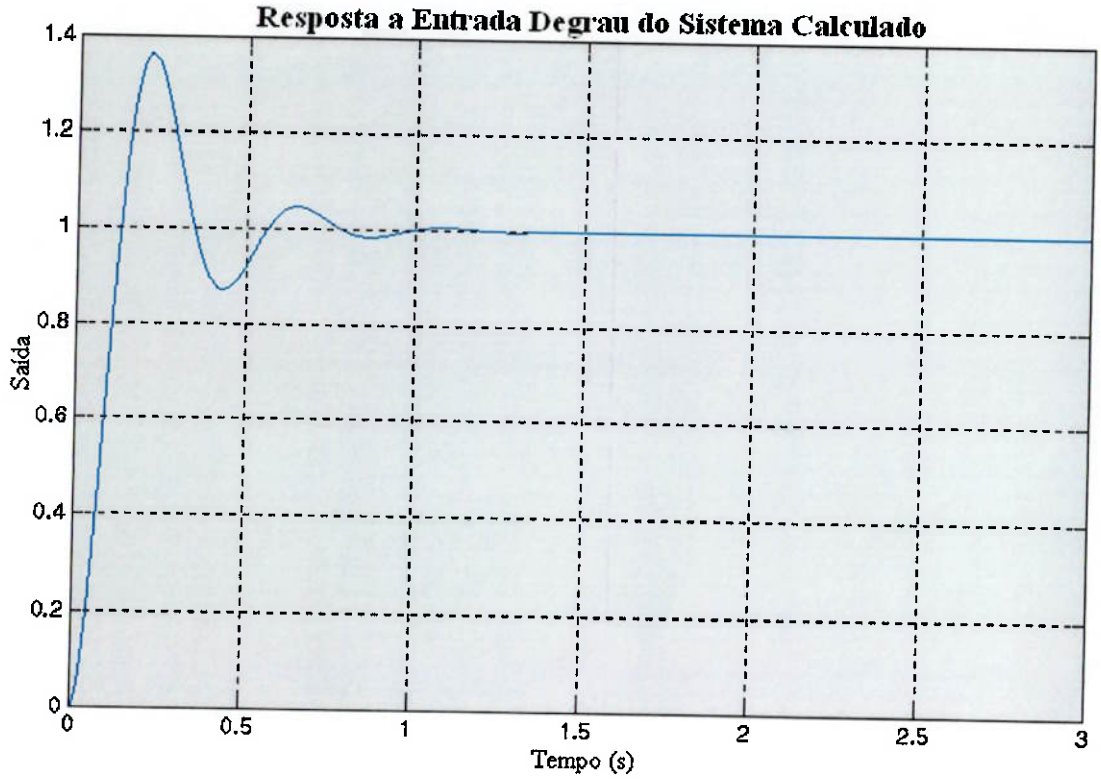


Figura 46 – Resposta a uma entrada degrau unitário

Para encontrar as grandezas físicas que garantem a obtenção da freqüência de ressonância e do coeficiente de amortecimento especificados, basta utilizar-se as relações (20) e (21), juntamente com os parâmetros físicos já apresentados (constante de amortecimento por atrito e aceleração da gravidade). Encontra-se então :

l = 43 mm

m = 80 g

Portanto, o detector de inclinação da cabeça do médico será um mouse no qual será acoplado um pêndulo com 43 mm de comprimento de haste e uma massa de 80 g na ponta (esta abordagem desconsidera a massa da haste).

Na construção prática, o mouse de cabeça assim montado mostrou um bom funcionamento, mas uma necessidade de constantes centralizações por perda de rotação por parte dos encoders. As fotos do mesmo encontram-se no Apêndice I.

8.2 – Sistema de Pedais

O sistema de pedais é, na verdade, um sistema simples que visa apenas fornecer ao usuário a confirmação dos comandos via pressionamento de um pedal.

De uma forma simplificada, o sistema consiste apenas de uma tira retangular apoiada sobre uma base. De um lado ela está presa à base por uma dobradiça, e de outro lado ela está simplesmente apoiada sobre um botão com retorno por mola.

A posição deste botão deve ser o mais longe possível da dobradiça, de forma a haver o maior braço de alavanca possível. Isto é necessário para que a força da mola do botão consiga sustentar o pedal, e também para que a inclinação do pedal não seja excessiva.

Devido ao tamanho pequeno e conseqüente fragilidade do botão, e considerando que sobre o mesmo poderá incidir uma razoável parte do peso do usuário, decidiu-se pela colocação de um apoio extra para o pedal ao final do mesmo.

A figura abaixo ilustra a configuração final imaginada para cada pedal individual.

Figura 47 – Configuração de um pedal individual

Vale observar que será necessária a confecção de três peças diferentes para cada pedal. A primeira é o pedal em si, a tira de metal. A segunda é o suporte do botão, cuja única função é prender o botão na posição desejada. A terceira é o apoio de fim de curso do pedal.

O terceiro botão, que têm como único objetivo a centralização do mouse, não terá pedal, ficando localizado em algum ponto estratégico da base para ser

simplesmente pressionado pelo usuário. Não há a necessidade do pedal uma vez que o botão será muito pouco utilizado.

Sendo assim, para este botão, um simples suporte é suficiente. A configuração final imaginada para este suporte pode ser vista abaixo.

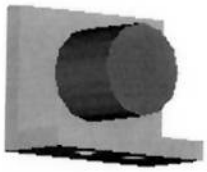


Figura 48 – Suporte para terceiro botão

Basta agora unir todo o conjunto. Para tanto, é importante considerar a ergonomia, ou seja, o conforto do usuário. Deve-se ter em mente o fato de que o mesmo estará olhando sempre para a tela, e não para os pedais, quando estiver utilizando o sistema. Deve-se também ter em mente o fato de que o usuário estará em pé ao efetuar a cirurgia.

Sendo assim, definiu-se que os pedais seriam dispostos em ângulo em torno de um centro, deixando um espaço em seu meio que funcionaria como descanso para o pé do usuário. O ângulo que o pé deve girar deve ser o menor possível, no entanto, para que seja confortável e simples ao usuário o pressionamento dos pedais. Compondo-se uma rotação em torno da parte de trás do pé do usuário, considerando o comprimento médio de um pé e a largura que deve sobrar entre os pedais para que haja espaço para o descanso, chegou-se no valor de ângulo de 65°, o que é razoável pois significa apenas 30° aproximadamente de rotação do pé para cada lado para ativar os comandos.

A figura abaixo ilustra este conceito.

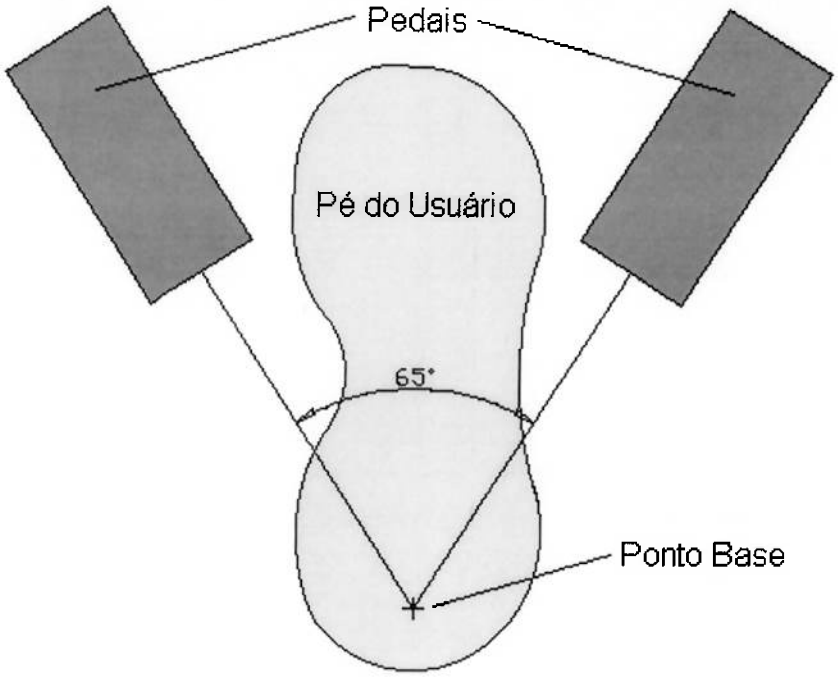


Figura 49 – Disposição em ângulo dos pedais

Finalmente, basta desenhar uma base para conter todo o sistema. O formato mais simples seria um retângulo simples englobando todo o sistema, mas perderia-se muito espaço. Optou-se por um retângulo com as pontas recortadas, de forma a acompanhar o contorno dos pedais.

Por fim, optou-se por colocar o terceiro botão no centro, entre os dois pedais, na parte superior da base. Assim, quando quiser centralizar o mouse, o usuário precisa apenas mover o pé para a frente para pressionar o terceiro botão.

A figura a seguir ilustra o conceito final do sistema dos pedais.

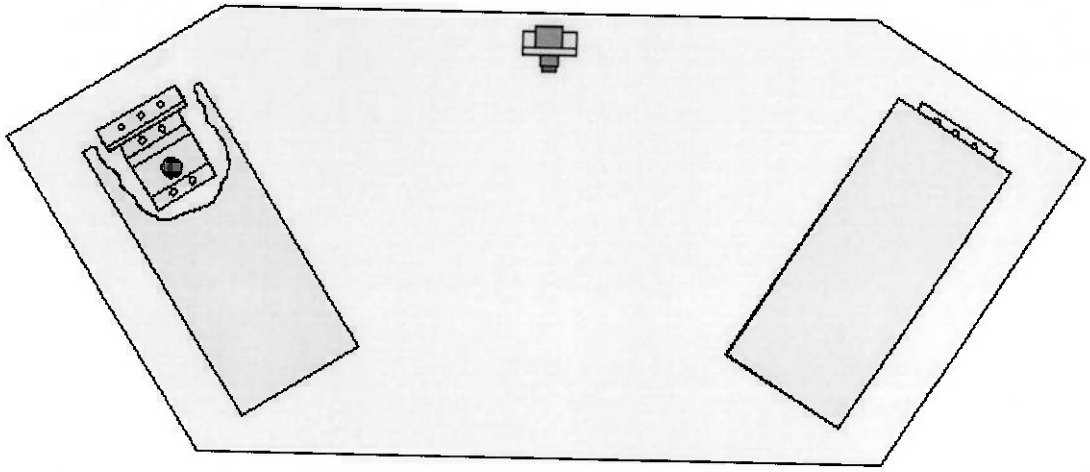


Figura 50 – Sistema completo dos pedais

O sistema todo ficou com dimensões 480 mm x 200 mm . Os desenhos das peças independentes podem ser vistos no Anexo II, onde constam todas as suas dimensões.

O material escolhido a ser utilizado em todas as peças do sistema foi aço ABNT 1020, pois chapas deste tipo de aço são baratas e de fácil obtenção. Todos os furos projetados nas peças foram definidos como furos de 3,2 mm , de forma que se utilizará parafusos M3 passantes, com porca, para a fixação das partes.

O sistema de pedais foi construído com extremo sucesso, sendo o resultado bastante ergonômico e os pedais perfeitamente funcionais. As fotos do sistema encontram-se no Apêndice I.

9. DESENVOLVIMENTO DO SOFTWARE DE CONTROLE

O software de controle do robô é o elemento que fará a ligação entre o input dado pelo médico através do mouse de cabeça e o acionamento do robô em si. Assim, é a função do software receber os inputs do mouse, reconhecê-los, interpretá-los e enviar uma saída que possa ser interpretada corretamente pelo driver dos motores de passo.

Foi decidido na fase de avaliação de alternativas que o software deveria ser criado em C, utilizando plataforma DOS. O software foi compilado em sistema MSDOS v.6.22 , utilizando o compilador QuickC v.2.5 , da Microsoft.

Todas as listagens dos arquivos de fonte do software podem ser encontradas no Anexo III, intitulado "Listagem dos Arquivos Fonte".

9.1 – Estruturação

O programa será dividido em quatro módulos principais, que serão acessados através de um menu principal.

O primeiro módulo têm como função a calibração do dispositivo de entrada, ou mouse de cabeça. Possui quatro etapas a serem seguidas, sendo :

- Centralização do Mouse – Antes de começar qualquer operação envolvendo o mouse, é preciso centralizar o cursor do mesmo na tela na posição mais confortável para o médico (posição 0). O médico apenas se posiciona e pressiona um contato.
- Definição das Sensibilidades do Mouse – Normalmente, inclinando a cabeça até um determinado limite, espera-se que o cursor atinja o final da tela. A sensibilidade horizontal e a vertical devem ser configuradas para garantir isso (regulam quanto o cursor se move de acordo com a quantidade de impulsos do encoder do mouse). Os contatos aumentam ou diminuem a sensibilidade, qualquer tecla aceita a sensibilidade.
- Centralização do Mouse – Novamente, faz-se necessária a centralização do cursor na tela.
- Definição do Retângulo Limite de Comandos – Espera-se que os comandos só sejam válidos após uma certa inclinação da cabeça, e não antes disso. É preciso configurar qual a inclinação desejada que ative os comandos, nas quatro direções, definindo um retângulo. Comandos só poderão ser passados com inclinações maiores (cursor fora do retângulo). O usuário apenas inclina a cabeça até a posição desejada e pressiona um contato qualquer para confirmar.

O segundo módulo é o responsável por configurar os parâmetros de operação do robô, o que se traduz na configuração dos motores que acionam o robô. São três etapas :

- Definição das Posições Limite – Os motores só podem operar dentro de uma certa faixa de espaço. Aqui, utilizando comandos do teclado, o usuário posiciona um a um os motores nas posições limite desejadas,

confirmando pelo próprio teclado. O posicionamento limite é contado por passos executados pelo motor.

- Definição das Velocidades – Para cada motor, enquanto o mesmo se movimenta entre as posições limite estipuladas, o usuário seleciona através do teclado a velocidade desejada para a movimentação, confirmando pelo próprio teclado.
- Posicionamento do Robô – Ao final da configuração, deve-se posicionar o robô para que o mesmo possa ser instalado para a cirurgia. O posicionamento ocorre por comandos do teclado. O procedimento serve também para posicionamentos do robô via teclado, caso se prefira, ao invés de se utilizar o sistema de mouse de cabeça e pedais.

O terceiro módulo corresponde ao procedimento de posicionamento manual do robô (via teclado) já explanado no módulo anterior.

Por fim, o quarto módulo é o responsável pela operação do robô em si, utilizando as calibrações e configurações definidas anteriormente e colocando em prática o uso do robô. Possui apenas duas etapas :

- Centralização do Mouse – Ao se posicionar para a cirurgia, o médico deve centralizar o cursor na tela para iniciar a correta operação. Qualquer contato confirma a posição desejada.
- Operação – Consiste na operação em si, como discutida anteriormente no texto. Através de inclinações da cabeça, o usuário observa indicadores na tela de comandos a serem executados. O usuário confirma um comando pressionando um contato, de acordo com indicações na tela. Qualquer tecla encerra a operação.

A figura a seguir ilustra de forma simplificada a simples estrutura do software.

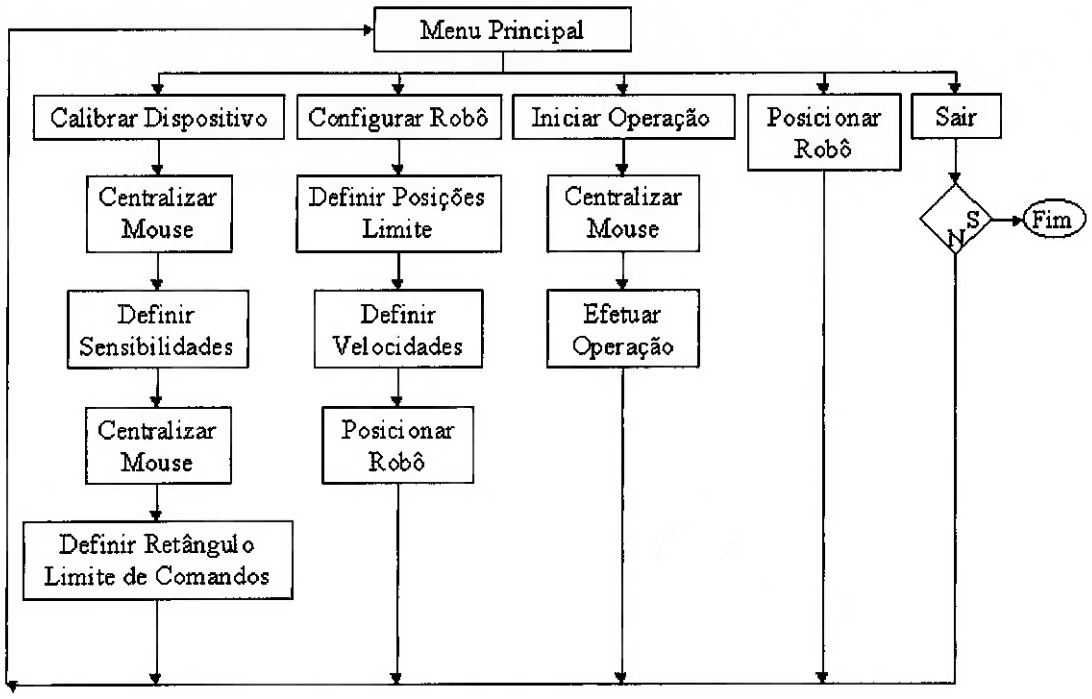


Figura 51 – Estrutura do software de controle

9.2 – Interface

Para a correta utilização do mouse e para um posicionamento mais preciso de seu cursor na tela, modos gráficos de interface são mais recomendáveis. Desta forma, optou-se por realizar toda a interface do software em modo gráfico.

O modo de exibição a ser utilizado possui resolução de 640 x 480 pixels (sendo esta a precisão do posicionamento do cursor na tela), em 16 cores. Os menus e opções ativados por teclado serão ativados por simples pressionamento de teclas, e todos serão contidos em janelas gráficas.

Para o desenho das janelas, linhas e gráficos informativos, várias funções foram escritas constando do arquivo "VIDEO.H", contido no Anexo III. Este arquivo também possui procedimentos de inicialização do vídeo.

Já o controle do buffer do teclado (recuperação das teclas digitadas) encontra-se no arquivo "TECLADO.H" no Anexo III , com funções específicas de recuperação de teclas e limpeza do buffer.

9.3 – Comunicação com o Mouse de Cabeça

O primeiro desafio é a comunicação com o mouse de cabeça. Como decidido na análise de alternativas, o mouse de cabeça é na verdade um mouse convencional adaptado através da adoção de um pêndulo, de forma que o sinal enviado pelo mesmo ao PC é exatamente o mesmo.

Desta forma, optou-se pela utilização do driver de mouse "CTMOUSE", v.1.8. Este driver cria uma interrupção de software para o mouse, que possui inúmeras funções, facilitando a configuração e leitura do mouse. A leitura direta da porta serial não é mais necessária. Este driver foi selecionado por se tratar de um software de livre distribuição e com excelente otimização do uso de memória do computador.

O mouse utilizado foi um mouse de três botões, sendo que dois botões (acoplados aos pedais) indicam comandos de movimentação do robô e o terceiro botão (colocado na ponta da estrutura dos pedais) serve apenas para forçar a centralização do mouse de cabeça durante a operação.

Os dados do mouse são alocados em uma estrutura global chamada "dispositivo", de forma a ficarem sempre acessíveis em todo o programa.

Foram desenvolvidas inúmeras funções, com os propósitos de inicialização, configuração de sensibilidade, posicionamento, leitura e gerenciamento do cursor.

A criação de um cursor específico ao invés da utilização do próprio cursor do driver do mouse é necessária porque os movimentos dos encoders não significam necessariamente o mesmo movimento que o convencional. Um movimento que equivaleria a mover o mouse para cima, por exemplo, pode corresponder a um movimento de inclinação para baixo da cabeça do usuário.

A listagem com as funções e procedimentos relativos ao mouse se encontra no Anexo III (arquivo "MOUSE.H") .

9.4 – Comunicação com o Driver dos Motores de Passo

Como visto na seção de projeto do driver dos motores de passo, para cada motor de passo deve haver o envio de dois sinais : um bit indicando a direção do movimento, e um bit que carrega um trem de pulsos (ou onda quadrada) para indicar a execução de passos pelo motor.

O sinal de direção é gerado facilmente, mas o trem de pulsos exige um procedimento bem mais complexo. A primeira opção considerada para a geração da onda quadrada seria uma inversão no sinal de saída que ocorreria a cada execução de uma interrupção, e esta interrupção seria programada para atuar no tempo correto.

A segunda opção seria utilizar um loop no próprio software, temporizado no tempo desejado, onde a cada loop se efetuaria a inversão deste sinal.

Decidiu-se pela segunda opção, uma vez que o controle de interrupções é bem mais complexo, dado o fato da mesma ocorrer em qualquer momento da execução do programa, o que pode gerar erros. O que se faz comumente é desativar a interrupção durante a execução de tarefas que não podem ser interrompidas, mas isso implica em se perder uma contagem de tempo e, logo, uma inversão de sinal, causando descontinuidade no movimento do motor.

Mas ainda há um terceiro problema. A saída oferecida na porta paralela é na verdade uma composição das saídas de três motores, logo haverão três ondas quadradas, todas com freqüências diferentes. A solução encontrada foi utilizar um loop com período fixo baixo dentro do software, de forma que todos os períodos utilizados na geração do trem de pulsos sejam múltiplos deste período, que indicará a freqüência máxima de um motor.

A seguir será feita a abordagem dos problemas de criar um timer capaz de fornecer períodos muito curtos e da composição dos sinais para a geração do sinal de saída para a porta paralela.

9.4.1 – Desenvolvimento de Timer com Alta Precisão

Uma vez que se deseja temporizar a execução de determinado código com baixos períodos, um timer de alta precisão é necessário. Em primeiro lugar, portanto, deve-se definir qual é a precisão desejada para o timer, e depois deve-se prosseguir para o desenvolvimento do mesmo.

A) Definição da Precisão do Timer

Sabe-se que os motores todos atuarão a baixas velocidades, uma vez que não pode haver movimentação brusca do robô. No entanto, quanto maior for a freqüência que o software possa atingir, maior será a precisão na seleção de velocidades a baixas freqüências. Isto por causa do método que estará sendo utilizado para criar os trens de pulsos, em que o período do pulso deverá ser múltiplo do período do loop de software (ou máximo do timer).

A figura abaixo ajuda a compreender o que foi dito.

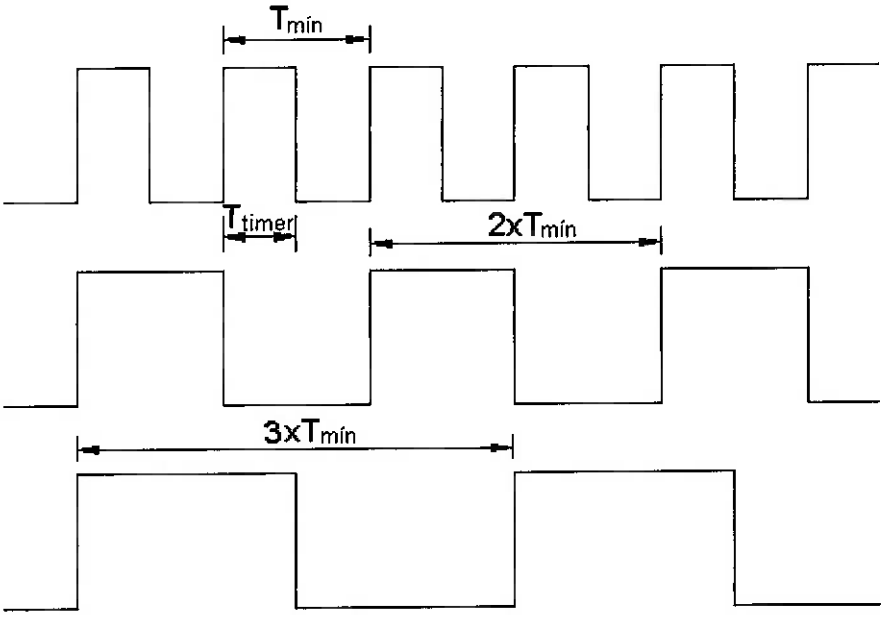


Figura 52 – Envio de trem de pulsos pelo software

O período mostrado como T_{timer} é o período mínimo (freqüência máxima) que o timer consegue criar. Desta forma, a cada T_{timer} um loop é executado no software.

Assim, o mínimo período que se consegue no sinal é de $2xT_{timer}$, pois há uma inversão de sinal a cada loop neste caso. Este é $T_{mín}$. Com uma inversão de sinal a cada dois loops têm-se $2xT_{mín}$, com uma inversão de sinal a cada três loops têm-se $3xT_{mín}$, e assim sucessivamente.

Assim, se T_{timer} for 1 ms, por exemplo, as freqüências de sinal obtidas podem ser de :

$T_{mín} = 2 \times T_{timer} = 2$ ms

Período do Sinal (ms)	Freqüência do Sinal (Hz)
$T_{mín} = 2$	500 Hz
$2xT_{mín} = 4$	250 Hz
$3xT_{mín} = 6$	166,67 Hz
$4xT_{min} = 8$	125 Hz

Tabela VII – Freqüências de sinal possíveis para $T_{timer} = 1$ ms

Já se T_{timer} for 0,25 ms, quatro vezes mais preciso, pode-se notar que as freqüências possíveis serão bem mais próximas (e mais numerosas) na faixa de 125 a 250 Hz, por exemplo.

$T_{mín} = 2 \times T_{timer} = 0{,}5$ ms

Período Sinal (ms)	Freqüência (Hz)	Período Sinal (ms)	Freqüência (Hz)
$T_{mín} = 0.5$	2000	$9xT_{mín} = 4.5$	222,22
$2xT_{mín} = 1.0$	1000	$10xT_{mín} = 5.0$	200
$3xT_{mín} = 1.5$	666,67	$11xT_{mín} = 5.5$	181,82
$4xT_{mín} = 2.0$	500	$12xT_{mín} = 6.0$	166,67
$5xT_{mín} = 2.5$	400	$13xT_{mín} = 6.5$	153,85
$6xT_{mín} = 3.0$	333,33	$14xT_{mín} = 7.0$	142,86
$7xT_{mín} = 3.5$	285,71	$15xT_{mín} = 7.5$	133,33
$8xT_{mín} = 4.0$	250	$16xT_{mín} = 8.0$	125

Tabela VIII – Freqüências de sinal possíveis para $T_{timer} = 0{,}25$ ms

Assim, nota-se que o timer deve atingir períodos bem menores do que o necessário para a correta execução do acionamento, de forma a manter menor discretização das freqüências na faixa de operação.

Sabe-se que dificilmente algum motor passará da freqüência de rotação de 1 Hz na ponta do eixo. Sabe-se também que os motores possuem 200 passos por revolução, logo a freqüência máxima do sinal deverá ser de 200 Hz, aproximadamente.

Por questão de segurança, e para garantir uma menor discretização das freqüências nesta faixa de operação (e abaixo), adota-se que a freqüência máxima fornecida do sinal (freqüência em $T_{mín}$) deve ser cinco vezes maior, ou seja, 1000 Hz.

$F_{máx} = 1000$ Hz

$T_{mín} = 1/F_{máx} = 1$ ms

$T_{timer} = T_{mín} / 2 = 0{,}5$ ms

Assim, deseja-se um timer com precisão de 0,5 ms (ou 500 microssegundos).

B) Programação do Timer

A maneira mais convencional de se temporizar eventos em um PC é a utilização da interrupção 0 (IRQ 0), que é a interrupção do sistema que fornece um clock de período determinado ao computador. Esta interrupção é chamada de "System Timer", ou "Temporizador do Sistema".

O timer do sistema DOS têm como padrão um ciclo de 18,6 Hz, ou seja, a interrupção é ativada a cada 53,76 ms , aproximadamente. Evidentemente, este período é alto demais para se conseguir precisões de timer abaixo de 100 ms, de forma que é inviável a utilização simples do IRQ0.

Desta forma, é preciso reprogramar a interrupção de hardware para que seja possível ler tempos com períodos menores. Para tanto, utilizou-se como base o timer desenvolvido por Brian Foley e Kim Kokkonen da TurboPower Software (o software é de domínio público), adaptando-o à aplicação desejada.

A listagem das funções do timer se encontra no Anexo III (arquivo "TIMER.H").

Para se realizar um teste do timer e saber se é viável a utilização do mesmo, foi criado o programa de testes localizado no Anexo III (arquivo "TESTIMER.C"). A saída do programa se encontra abaixo, tendo sido o programa testado em um microcomputador com processador AMD K5 , com 32 Mb de memória RAM.

```
Tempo entre chamadas sucessivas : 0.010895 ms (10.895239 micros)

Tempo para a chamada de 'espera(5000)' : 5.041982 ms (5041.981591
micros)
Tempo para a chamada de 'espera(1000)' : 1.036724 ms (1036.723941
micros)
Tempo para a chamada de 'espera(500)'  : 0.527162 ms (527.161972
micros)
Tempo para a chamada de 'espera(100)'  : 0.139124 ms (139.123827
micros)

Tempo para contagem em vazio, 5000 microseg. : 5.003429 ms
(5003.429205 micros)
Tempo para contagem em vazio, 1000 microseg. : 1.002362 ms
(1002.362032 micros)
Tempo para contagem em vazio, 500 microseg.  : 0.509562 ms
(509.561969 micros)
Tempo para contagem em vazio, 100 microseg.  : 0.102248 ms
(102.247632 micros)
```

```
Calculando repetibilidade (1000 amostras, 500 microsegundos)...
Tempo medio para 500 microseg.      : 0.504795 ms (504.794883 micros)
Tempo maximo encontrado na amostra : 0.514590 ms (514.590541 micros)
Tempo minimo encontrado na amostra : 0.500343 ms (500.342921 micros)

Tempo estimado p/ calculo dos num. aleatorios : 0.489448 ms
(489.447681 micros)
Tempo contado com calculos, 5000 microseg. : 5.006782 ms
(5006.781587 micros)
Tempo contado com calculos, 1000 microseg. : 1.008229 ms
(1008.228699 micros)
Tempo contado com calculos, 500 microseg.  : 0.502857 ms (502.857207
micros)
Tempo contado com calculos, 100 microseg.  : 0.487771 ms (487.771490
micros)

Testando tempo necessario para saida no video...
Tempo estimado p/ saida de 1 linha no video : 4.760382 ms
(4760.381556 micros)

Calculando tempo maximo para execucao de ciclo real no software...
Tempo estimado p/ executar 1 ciclo real : 0.046095 ms (46.095244
micros)
```

Figura 53 – Saída do programa de testes do timer

Pode-se notar um erro maior no tempo registrado para a função "espera" do que para a função composta (que considera o tempo utilizado pela execução de outros códigos). Na verdade isso apenas decorre do fato que para a contagem do tempo desta função, que não utiliza variáveis globais, faz-se chamadas adicionais de leitura do timer, que levam aproximadamente 10 microssegundos cada. Na verdade, a precisão de ambos os procedimentos é a mesma.

Outro ponto importante é que o erro máximo observado em uma amostra de 1000 medições (500 microssegundos cada) não ultrapassou os 3%, o que é mais do que satisfatório para a aplicação pretendida.

Vale notar, no entanto, que no caso do tempo levado para a execução do loop ultrapassar o tempo desejado para o mesmo a precisão do timer é inútil, pois a função apenas retorna, e assim o período do sinal será inconstante e randômico.

Pode-se notar, por exemplo, que a simples impressão na tela de uma linha de texto leva um tempo total de aproximadamente 5 ms. Logo, para se ter precisão de microssegundos, não se pode incluir no loop nenhum código que altere o status do vídeo.

Sendo assim, o loop deverá ser composto por atualização e verificação dos dados do mouse, além da execução de um ciclo para a composição dos sinais de

saída. O tempo máximo estimado para executar este loop foi calculado em aproximadamente 50 microssegundos (mesmo com o acréscimo dos 10 microssegundos da chamada da função), de forma que o loop poderia ter até 100 microssegundos de precisão sem problemas.

Sendo assim, o timer é preciso o suficiente e satisfatório para o ciclo desejado.

9.4.2 – Composição dos Sinais de Saída

Os dados de cada motor serão armazenados em estruturas globais, de forma a estarem acessíveis a todo o software. Um exemplo de dado são as posições atual e limite do motor, a direção de movimentação e um indicativo da movimentação.

Um dos dados é a velocidade do motor. Esta velocidade é na verdade um contador, no sentido de que indica quantas vezes um loop de software deverá ser executado para que se tenha uma inversão do sinal de saída. Desta forma, quanto maior este número, mais lento o motor irá girar. Utilizando um loop com 500 microssegundos, a velocidade máxima é 1, que indica uma freqüência máxima de rotação do motor de 5 Hz ($1/200x1x10^{-3}$).

Desta forma, a cada loop, se executa o acréscimo de 1 na contagem de velocidade do motor. Ao atingir o valor definido de velocidade, o sinal do bit correspondente para o motor é invertido (de 0 para 1 ou vice-versa).

Como todos os sinais para o acionamento dos motores possuem períodos que são múltiplos do período do loop x 2, é possível enviar os sinais para os três motores simultaneamente sem problemas (cria-se um byte que é a composição dos sinais de todos e envia-se este byte à porta paralela).

As funções utilizadas para o controle e composição dos sinais para os motores encontram-se listadas no Anexo III, no arquivo "MOTORES.H".

9.5 - Programação dos Módulos

Tendo prontas todas as funções e procedimentos que interagem com o driver de motor de passo, com o mouse de cabeça e também que controlam o timer de alta precisão necessário, pode-se avançar para a execução do programa em si.

Para acessar cada uma das funções do software, há uma tela principal, contida no programa principal. Este programa está contido no arquivo "ENDO.C", que pode ser visto no Anexo III. O programa principal apenas monta um menu de opções e funciona como uma ponte para a chamada de todos os outros procedimentos.

A figura abaixo mostra a tela principal do software.

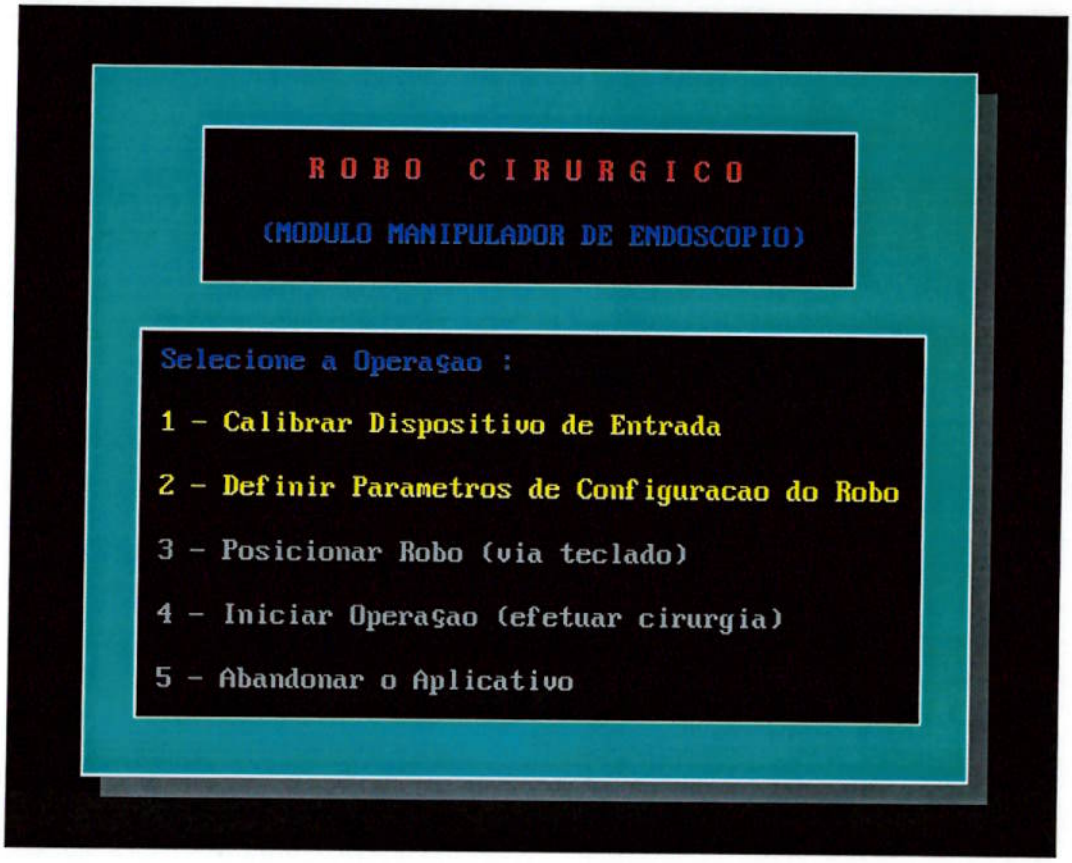


Figura 54 – Exemplo de tela – Tela principal

O restante do programa encontra-se dividido em três módulos distintos.

O primeiro módulo diz respeito à calibração do dispositivo de entrada, ou mouse de cabeça. Os procedimentos que efetuam esta calibração encontram-se no arquivo "CLBRMOUS.C" (que pode ser encontrado no Anexo III). De uma forma

geral, apenas se faz uso das funções desenvolvidas anteriormente (presentes no arquivo "MOUSE.H").

A figura abaixo apresenta um exemplo de tela presente na calibração do dispositivo de entrada, que é a tela onde se configura o retângulo limite de comandos (a partir de qual inclinação da cabeça se entende como comando).

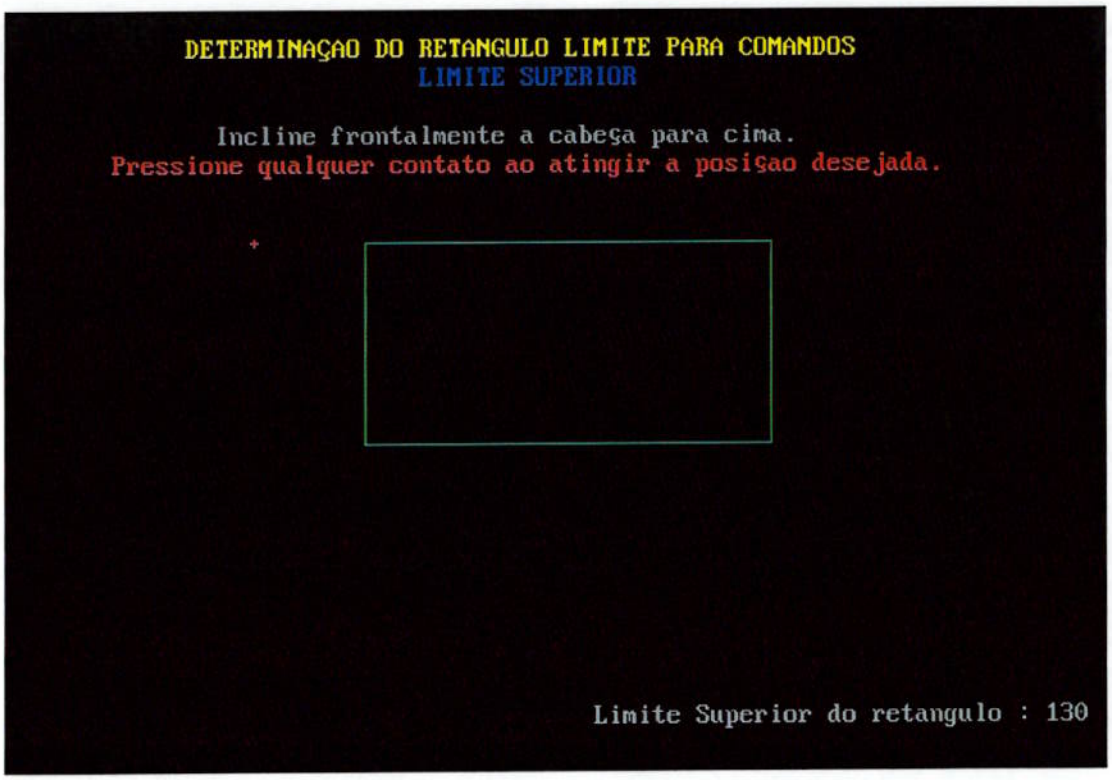


Figura 55 – Exemplo de tela – Calibração do mouse de cabeça

O segundo módulo diz respeito aos procedimentos que configuram parâmetros do robô (ou seja, dos motores). O arquivo "CNFGMOT.C" (que pode ser encontrado no Anexo III) possui os procedimentos e funções que efetuam a configuração dos motores, assim como o procedimento que efetua o posicionamento do robô manualmente (via teclado).

As funções e procedimentos aqui utilizados utilizam-se dos procedimentos definidos em "MOTORES.H" (também presente no Anexo III).

A figura abaixo apresenta um exemplo de tela presente nas configurações dos parâmetros do robô, no caso a tela na qual se efetua o posicionamento do robô.

Figura 56 – Exemplo de tela – Configuração dos parâmetros do robô

O terceiro e último módulo possui apenas o procedimento utilizado na cirurgia em si. É o procedimento mais complexo, pois envolve interface, comunicação com o mouse de cabeça e comunicação com o driver dos motores.

O procedimento em si encontra-se no arquivo "OPERACAO.C" (presente no Anexo III) e utiliza todos os arquivos com as funções primárias ("MOTORES.H", "MOUSE.H", "TIMER.H").

As figuras abaixo mostram dois exemplos de tela durante a operação. Na primeira figura, o usuário inclinou a cabeça tanto frontalmente quanto lateralmente, sendo que o software indica através de setas e símbolos os comandos possíveis de serem executados.

A segunda figura mostra a mesma situação, mas após o usuário pressionar e manter pressionado o pedal direito. Neste caso, o software entende que deve efetuar uma operação de 'Zoom Out' e indica o movimento do motor ao usuário.

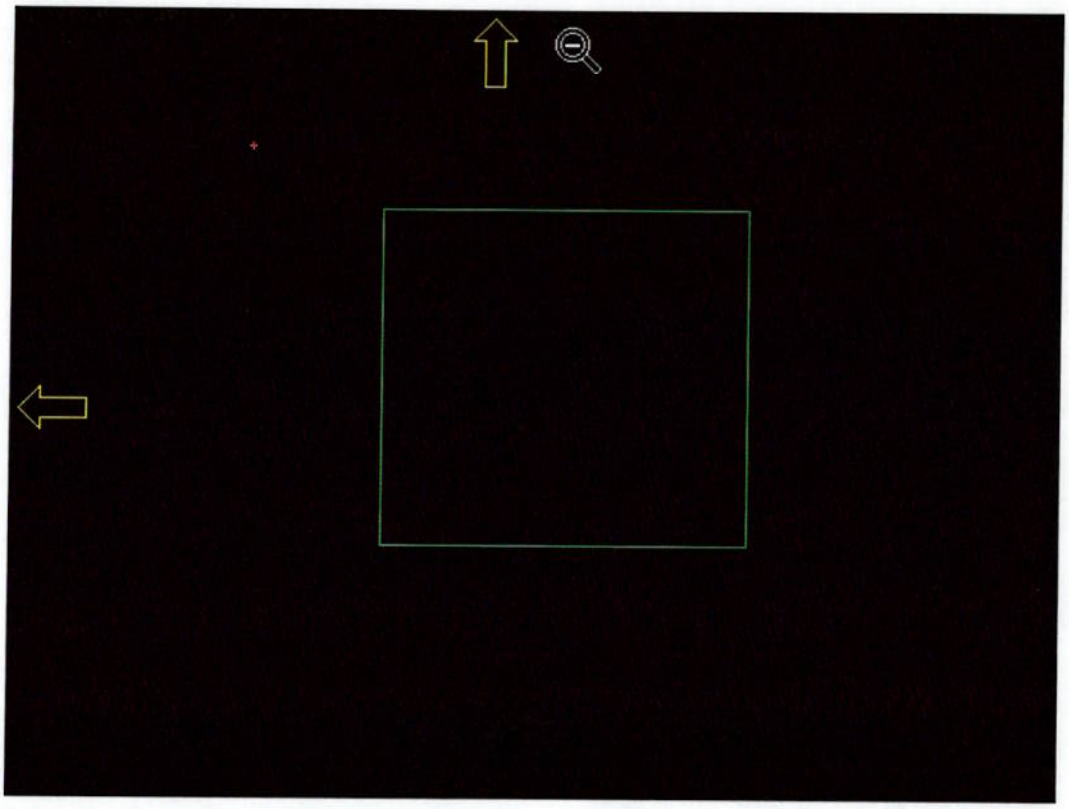


Figura 57 – Exemplo de tela – Operação (indicação de opções de movimento)

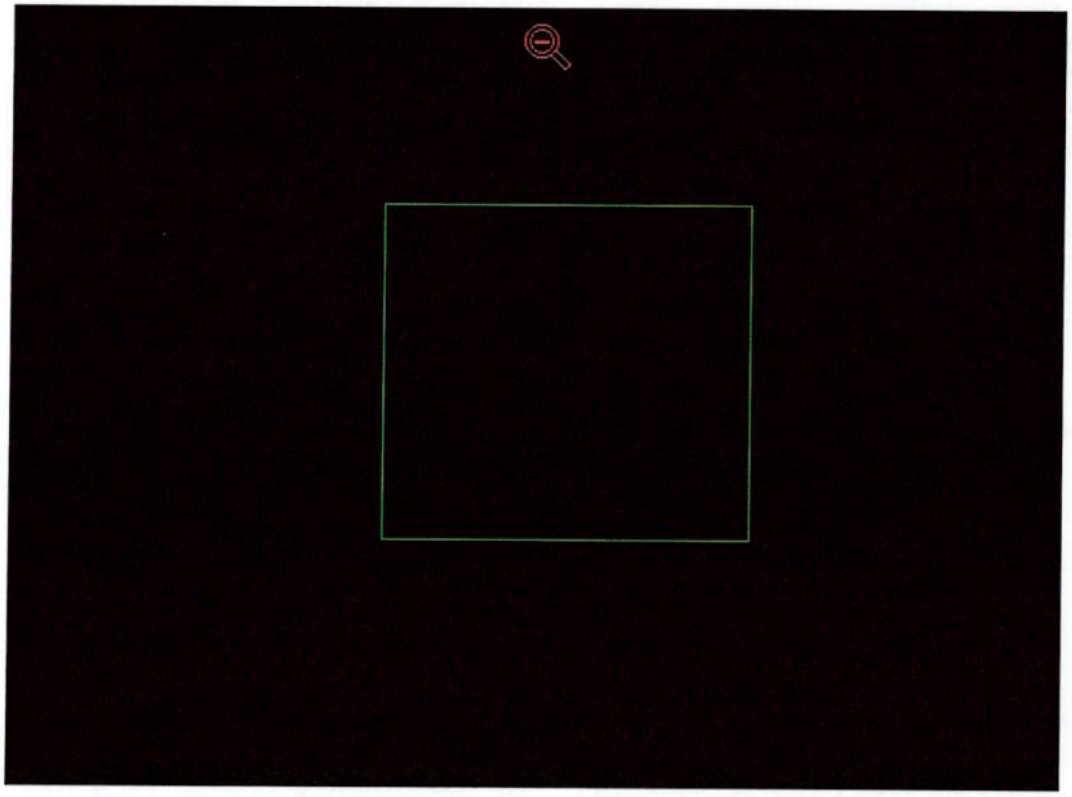


Figura 58 – Exemplo de tela – Operação (efetuando movimentação)

No funcionamento real do robô, o fundo da tela (que nas duas figuras acima se encontra em preto) seria na verdade preenchido com a imagem obtida pela câmera do endoscópio. Pensando nisto, tanto o cursor (em vermelho) quanto o retângulo limite de comandos (em verde) podem ser ativados/desativados à vontade, bastando pressionar o pedal direito dentro da área do retângulo.

A aquisição e o processamento de imagens provenientes da câmera não fazem parte do escopo deste trabalho, ficando assim para um desenvolvimento futuro.

10. CONCLUSÕES

O projeto, de uma forma geral, permitiu a aplicação de vários conhecimentos, que variam desde a área da mecânica estrutural até microeletrônica. Percebe-se nitidamente que a solução proposta não é única e que existem ainda vários tópicos que merecem um melhor estudo e a proposição de melhores soluções.

Cabe ressaltar que o robô desenvolvido, assim como o software implementado e os demais componentes constituintes do sistema proposto, são parte de uma solução que, apesar de não ser única, apresentam tamanhas restrições que possivelmente forçariam outros projetistas a reverem os mesmos pontos aqui apresentados.

O projeto, como um todo, satisfez muito bem as necessidades estabelecidas, principalmente levando-se em conta fatores altamente limitantes que atuaram integralmente no desenvolvimento: custo e tempo.

Pode-se perceber que, comparando-se os resultados obtidos e os recursos aplicados, a solução obtida foi bastante satisfatória, apesar de merecer uma melhor revisão em alguns aspectos relacionados principalmente à execução mecânica. A partir do que hoje já está desenvolvido, novas soluções podem ser ainda propostas de modo a ter-se um desempenho mais otimizado do sistema desenvolvido.

Um ponto que necessita claramente de melhorias é o movimento principal da estrutura (a rotação de todo o mecanismo em torno do eixo principal). Neste movimento detectou-se elevada vibração, ocasionada pela elevada massa em balanço que há na extremidade oposta ao acionamento. O motor de passo possui naturalmente uma oscilação em torno de sua nova posição de equilíbrio a cada vez que executa um passo, e esta é amplificada pela massa em balanço.

Uma possível solução seria a adição de amortecimento viscoso ao sistema, por meio de cilindros viscosos existentes no mercado, assim evitando esta vibração. Outra solução seria a adoção de um outro tipo de motor, como um servomotor DC, de forma a evitar estas oscilações naturais do motor de passo.

A solução mais simples, no entanto, consistiria em se alterar o driver de motor de passo para que se fizesse possível o acionamento do mesmo em "half-step"

ou até "micro-step". Com uma melhor resolução dos passos reduz-se drasticamente a oscilação observada a cada passo, o que viria a reduzir a vibração

Já nos outros dois graus de liberdade, observou-se excelente continuidade e suavidade do movimento. Isto porque o próprio atrito existente nos condutos dos cabos contribuiu para eliminar a oscilação. Além disso, por ser bem maior a redução, os motores neste caso giram mais rapidamente, com movimento contínuo, ao passo que no motor principal observa-se um movimento discreto (passo a passo). Com isso, a vibração nestes dois movimentos foi eliminada.

Outra melhoria necessária diz respeito ao mouse de cabeça. Construído com recursos limitados e aproveitando um mouse convencional, verificou-se que não há grande confiabilidade nos encoders do mouse, de forma que se verifica constante perda de pulsos pelos mesmos, isto é, movimentos não detectados pelos encoders incrementais.

O ideal neste caso seria a utilização de inclinômetros comerciais, baseados em acelerômetros, ou então a confecção de um mouse de cabeça mais preciso, de preferência utilizando-se encoders absolutos já ligados diretamente à massa do pêndulo, assim sem a transmissão de movimento entre a esfera e os eixos dos encoders.

Considerando-se, no entanto, os limitados recursos empregados e o limitado tempo disponível para a execução do projeto, percebe-se ser o resultado obtido extremamente satisfatório, dependendo o projeto como um todo de um desenvolvimento futuro nos pontos onde isto se fizer necessário.

11. REFERÊNCIAS BIBLIOGRÁFICAS

[1] - FARAZ, A.; PAYANDEH, S. **A Robotic Case Study :** Optimal Design for Laparoscopic Positioning Stands.

[2] - MOCZALA, H. **Small electrical motors.** Power and energy series 26, IEE, Redwood Books, 1998.

[3] - KENJO, T.; SUGAWARA, A. **Stepping Motors and Their Microprocessor Controls.** Oxford University Press, 1994.

[4] - KOBAYASHI, E. **A New Safe Laparoscopic Manipulator System with a Five-Bar Linkage Mechanism and an Optical Zoom.** Wiley-Liss, 1999.

[5] - KAMINSKI, P. **Desenvolvendo produtos com planejamento, criatividade e qualidade.** Livros Técnicos e Científicos Editora, 2000.

[6] - TIMOSHENKO, S.P. **Mecânica dos Sólidos.** Livros Técnicos e Científicos Editora, 1992.

[7] - SEELY, F.B.; SMITH, J.O. **Advanced Mechanics of Materials.** John Wiley & Sons, 1963.

[8] - GOODWIN, M. **User Interfaces in C and C++.** MIS Press, 1992.

[9] - GOODWIN, M. **Serial Communications Programming in C and C++.** MIS Press, 1992

[10] - MICROSOFT CORPORATION **Programming the Microsoft Mouse Under MS-DOS.** Microsoft Product Support Services, 1994.

[11] - PLAUGER, P.J. **The Standard C Library.** Prentice Hall, 1992.

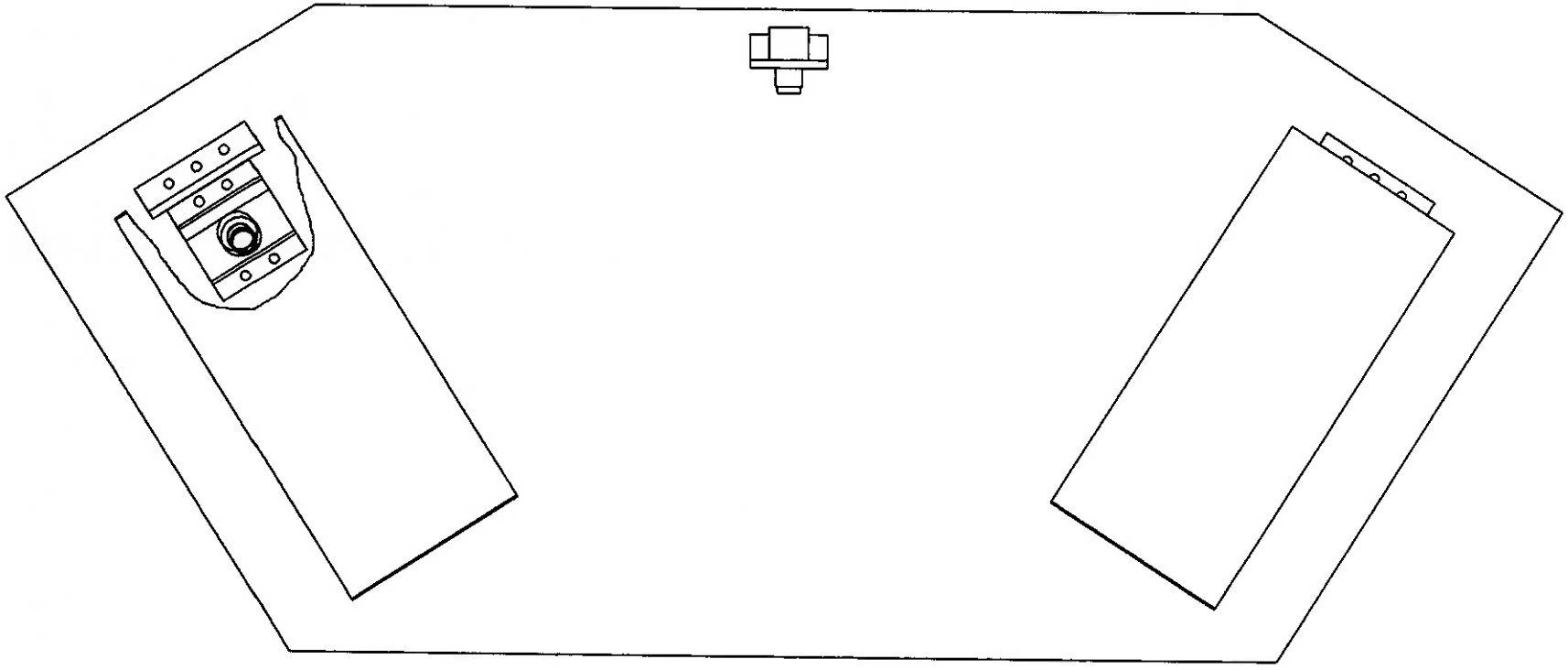
[12] - MENDONÇA, A. **PC e Periféricos : Um Guia Completo de Programação.** Ciência Moderna, 1996.

[13] - BOYLESTAD, R.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos.** Prentice Hall do Brasil, 1994.

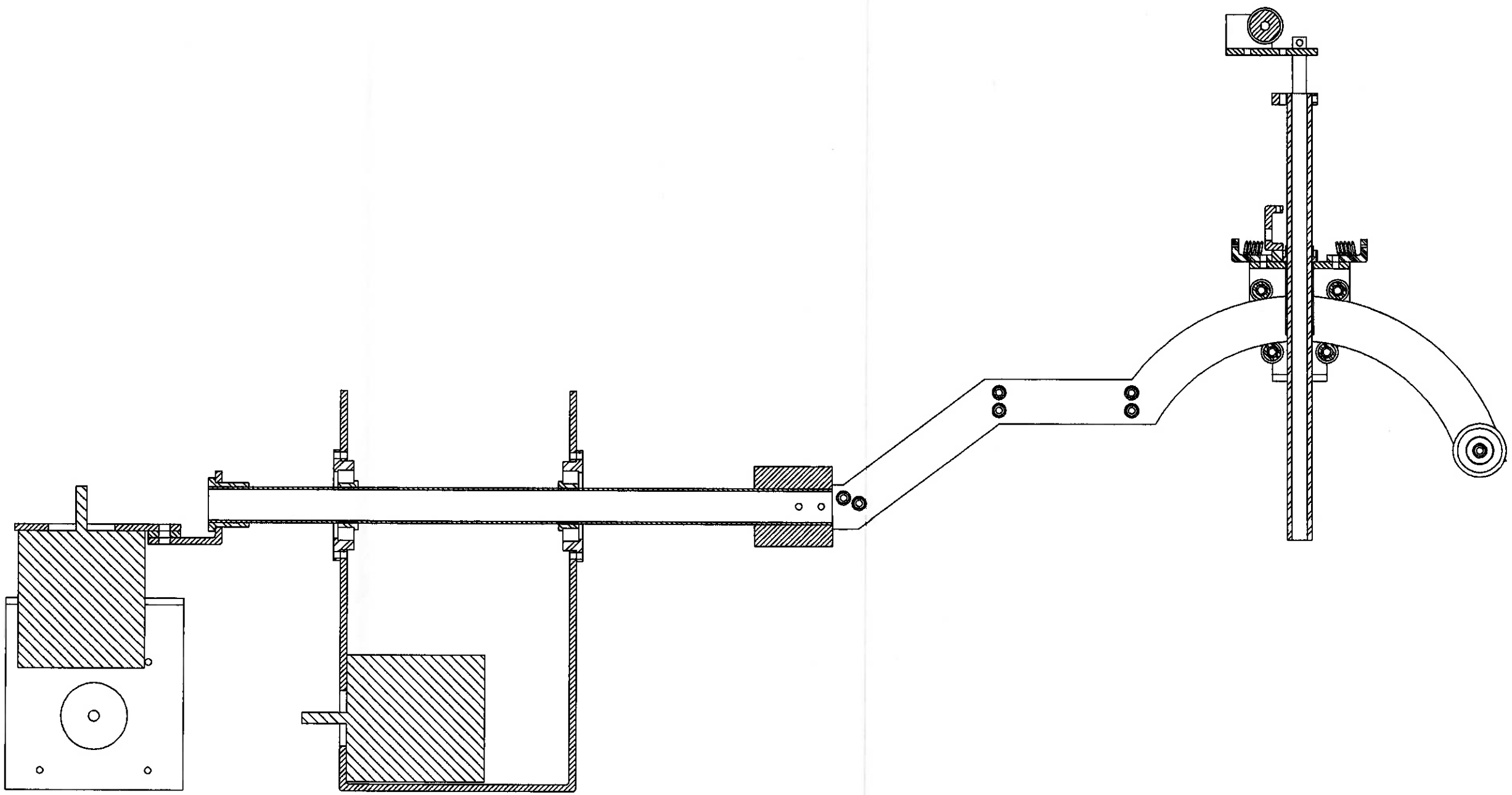
[14] - SHIGLEY, J. E.; MITCHELL, L. D. **Mechanical Engineering Design.** McGraw Hill, 1983.

[15] - SPIEGEL, M. R. **Theory and Problems of Theoretical Mechanics.** Schaum Publishing, 1967.

ANEXO I - DESENHOS DE CONJUNTO



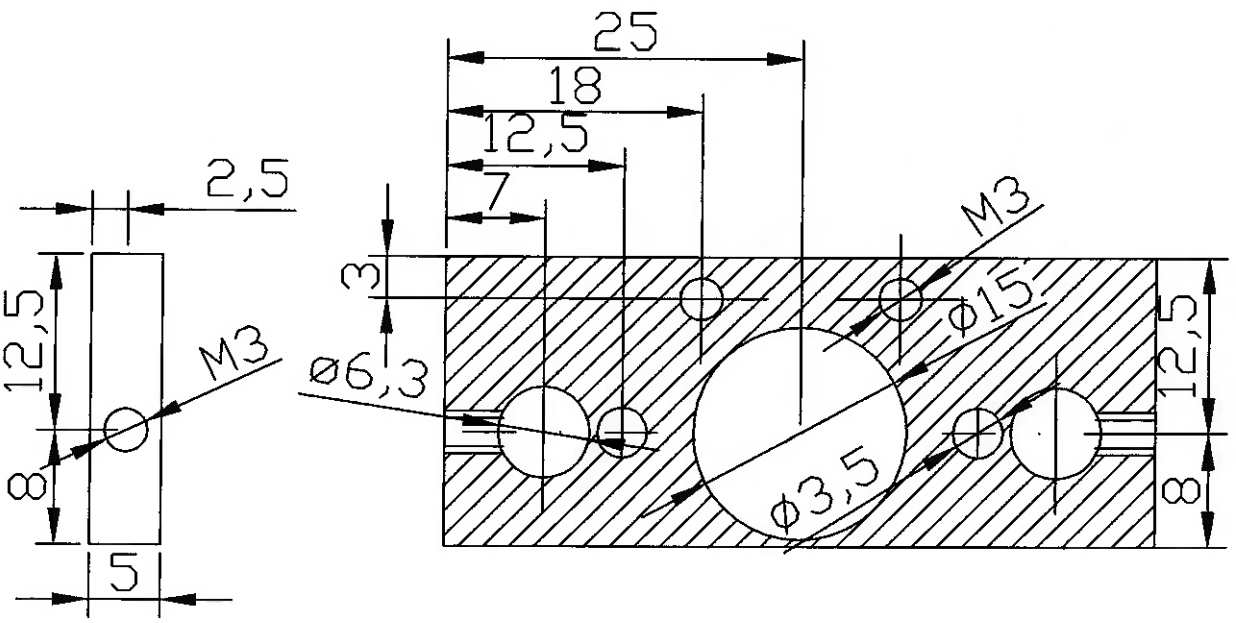
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 09/12/2001	Escala 1:4
Título Desenho de conjunto dos pedais			



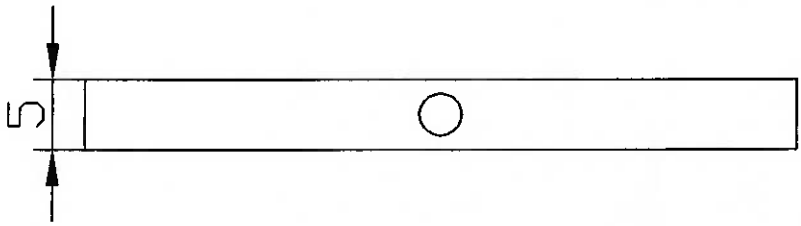
Responsável	Data	Escala
Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452	04/12/2001	1:2

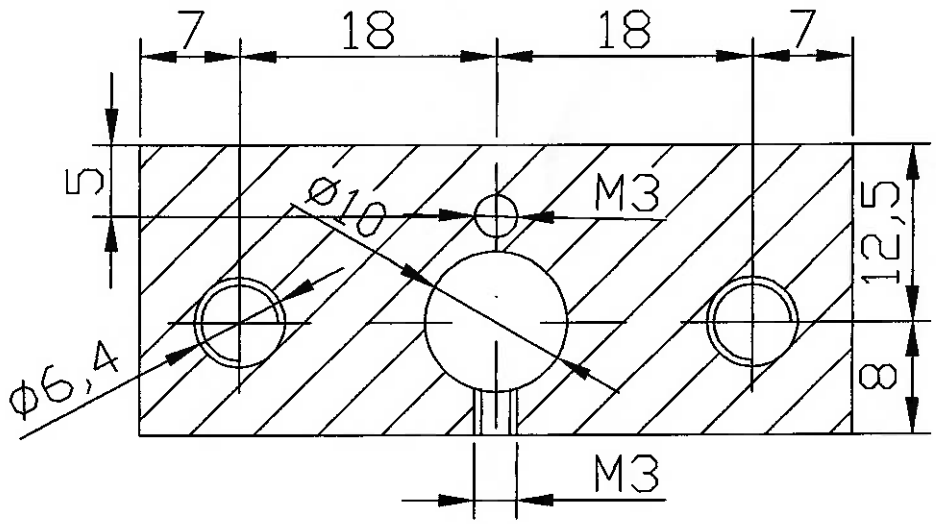
Título

Desenho de conjunto do robô

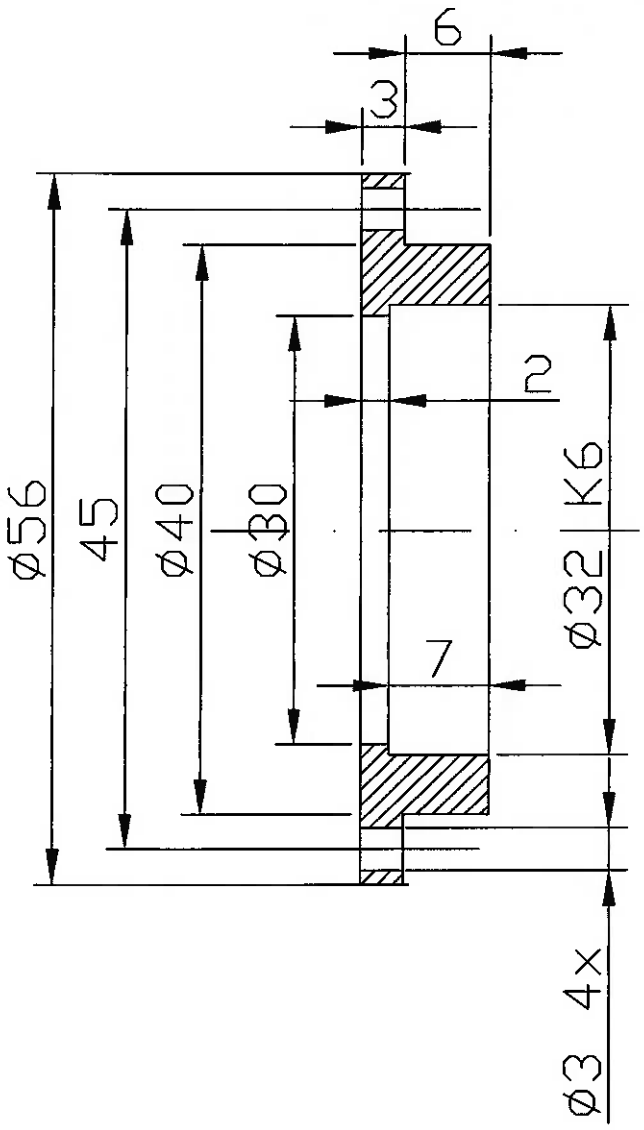


Item: 02	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 26/11/01	Escala 2:1
Título Movimentador do endo. - Peça 2			

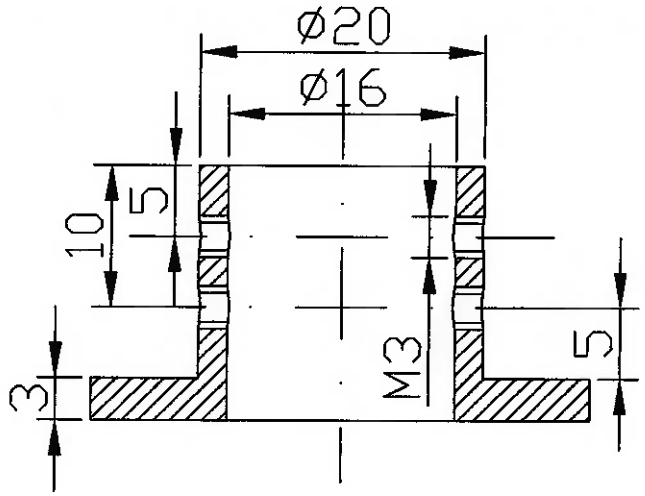


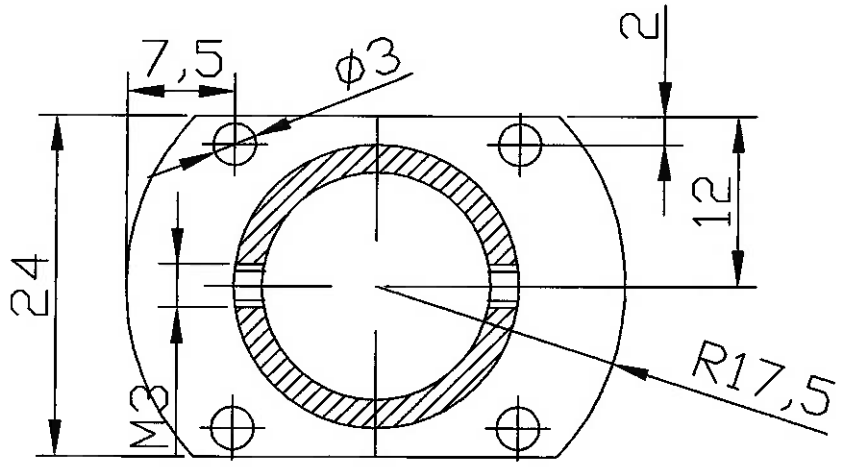


Item: 03	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1
Título Movimentador do endo. - Peça 1			



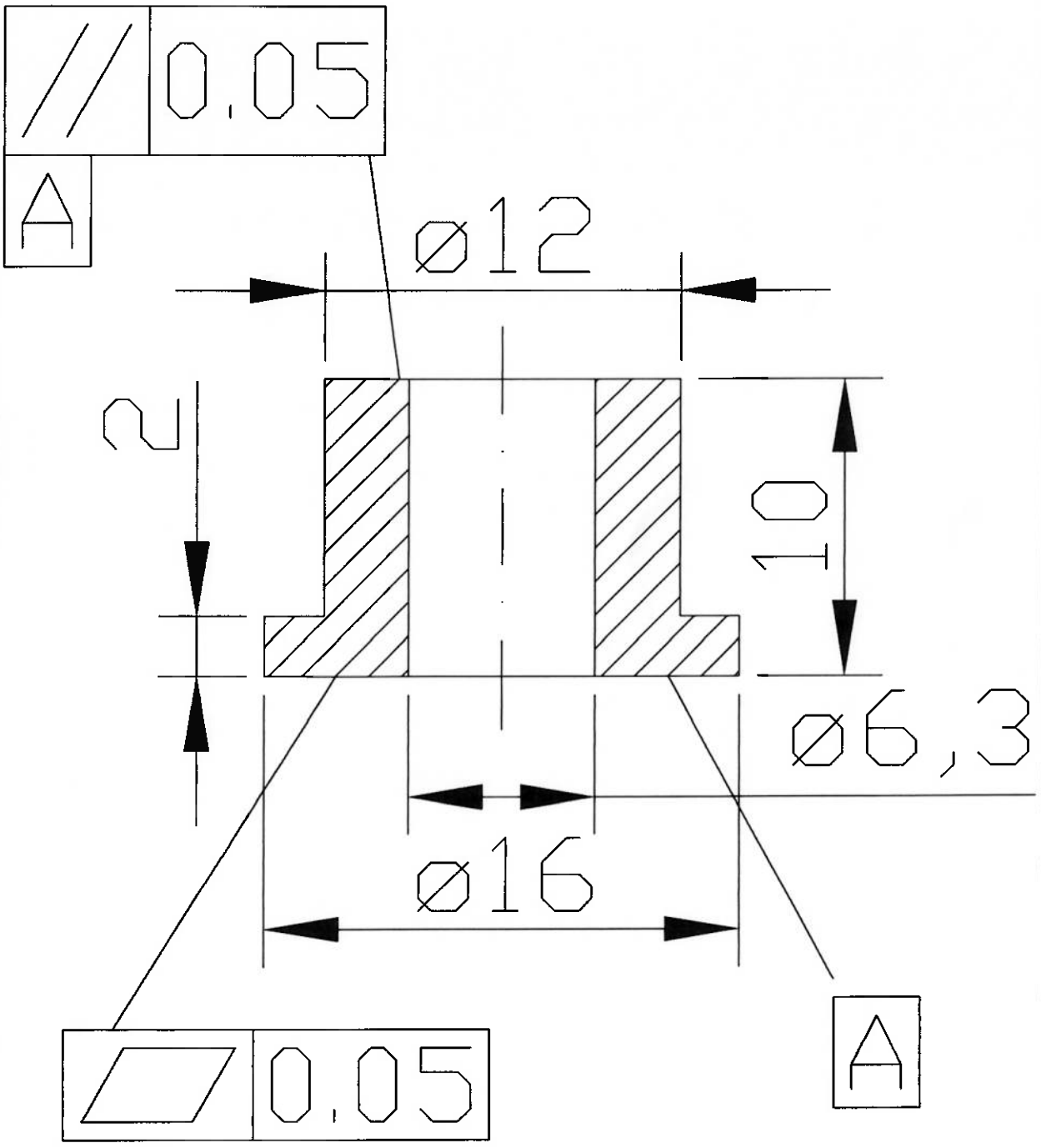
Item: 04	Quantidade: 2	Material: Alumínio		
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452			Data 23/11/01	Escala 1:1
Título Mancal dos rolamentos da base				





Item: 05	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:1

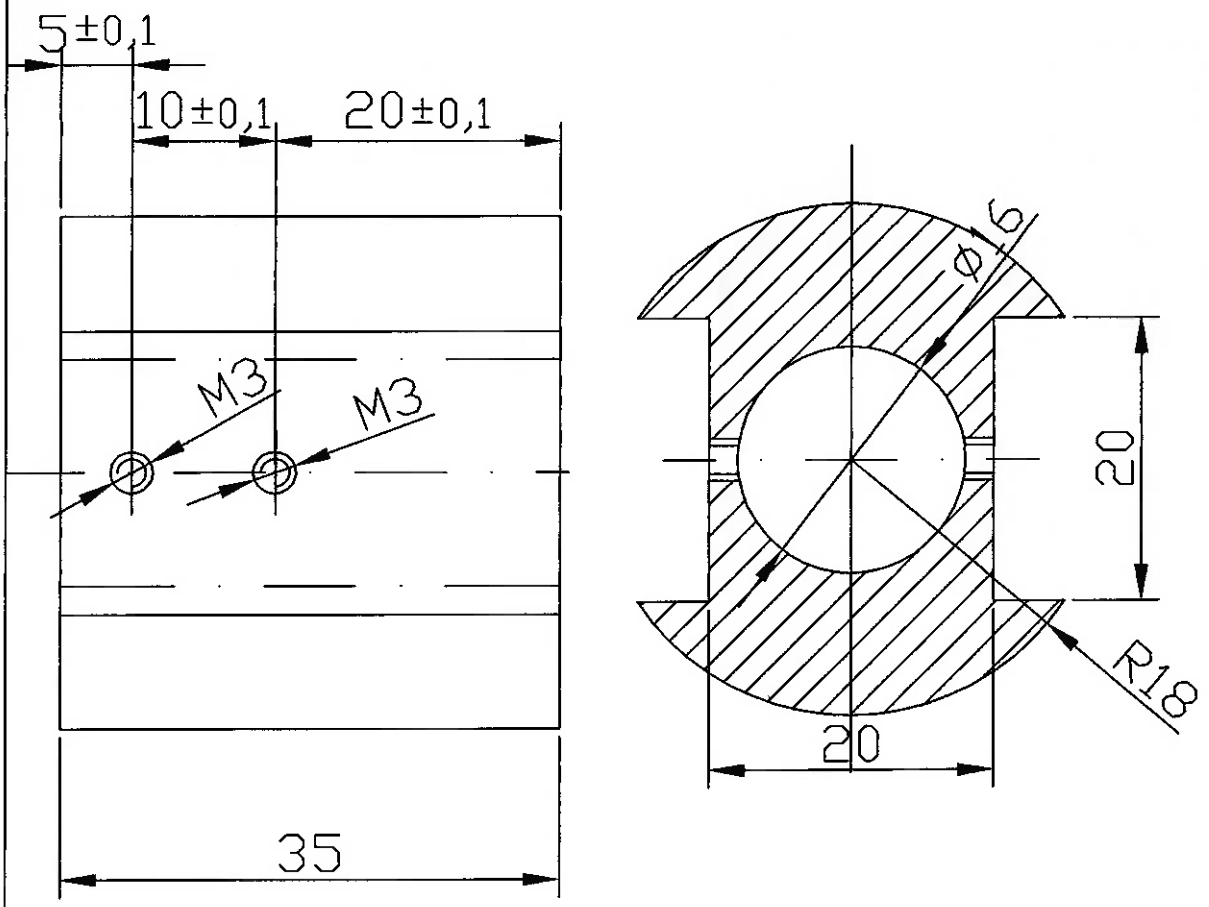
Título: Fixador do eixo à base secundária



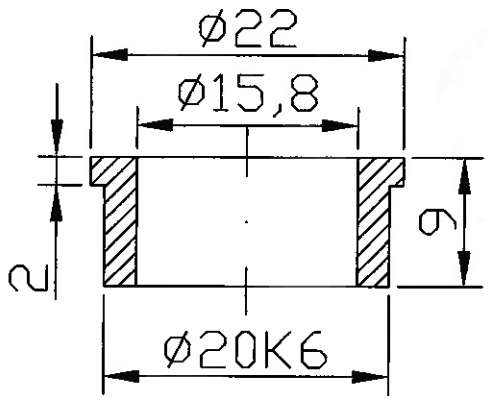
Item: 06	Quantidade: 02	Material: PP	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 5:1

Título

Mancais do rolamento da polia terminal



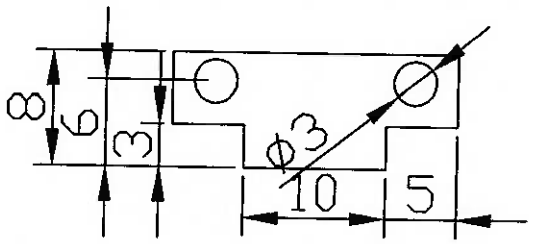
Item: 07	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1
Título Conector entre trilhos e o tubo			

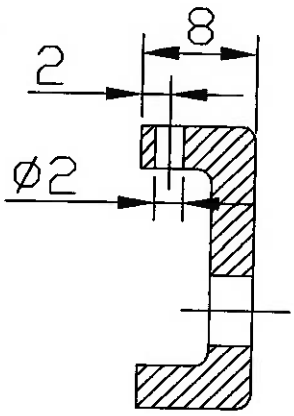


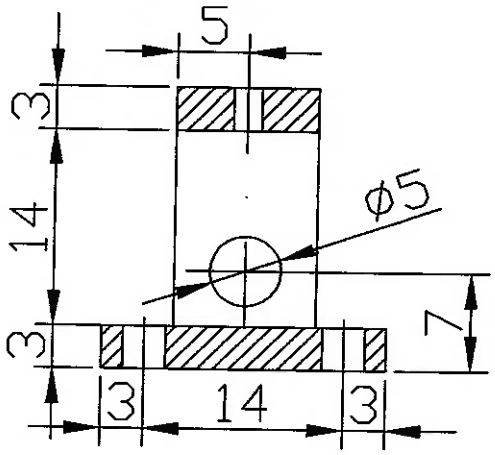
Item: 08	Quantidade: 2	Material: PP	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1

Título

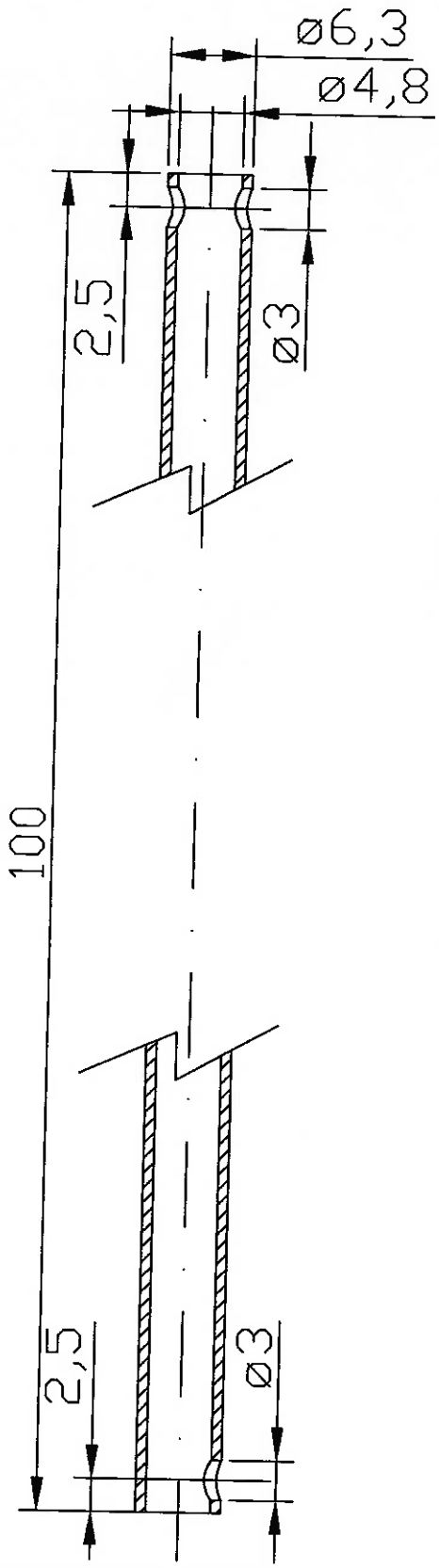
Bucha dos mancais da base principal



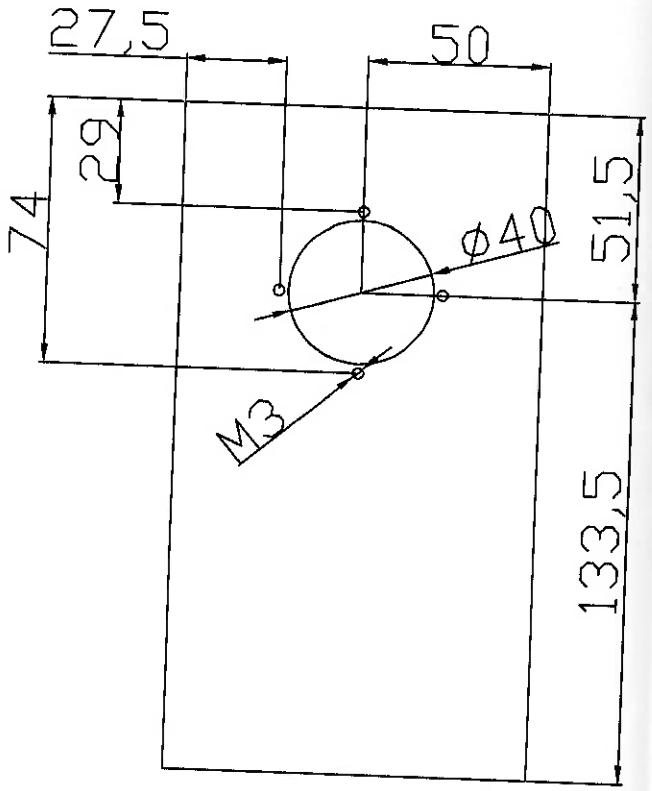


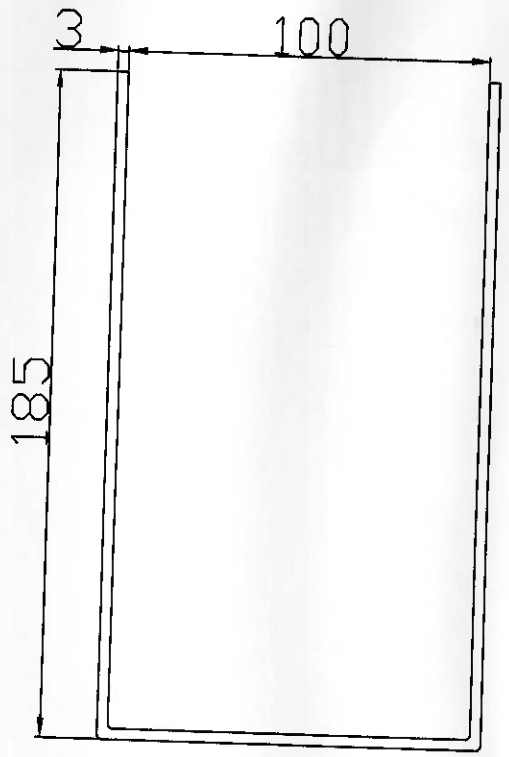


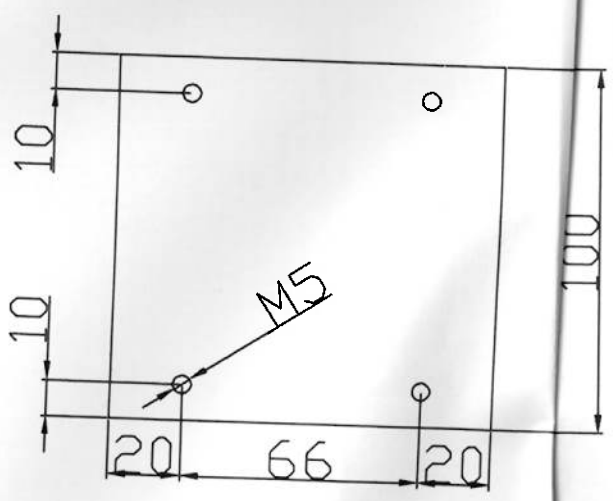
Item: 09	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1
Título Terminador-guia do cabo-mov. de zoom			

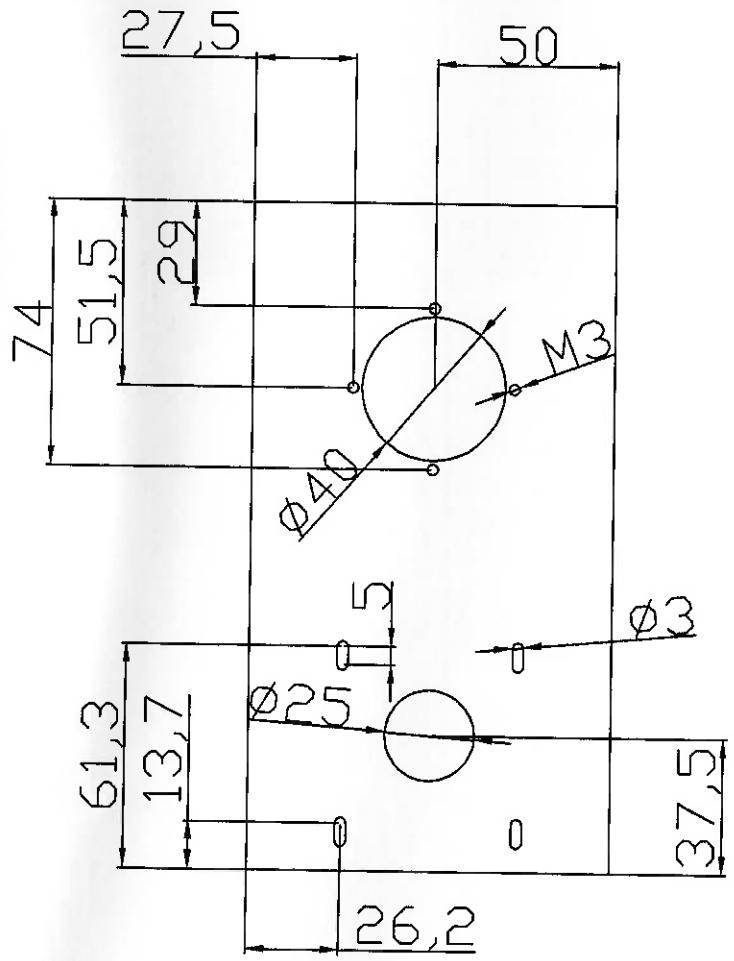


Item: 10	Quantidade: 02	Material: Cobre	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1
Título Guias lineares do mov. z			

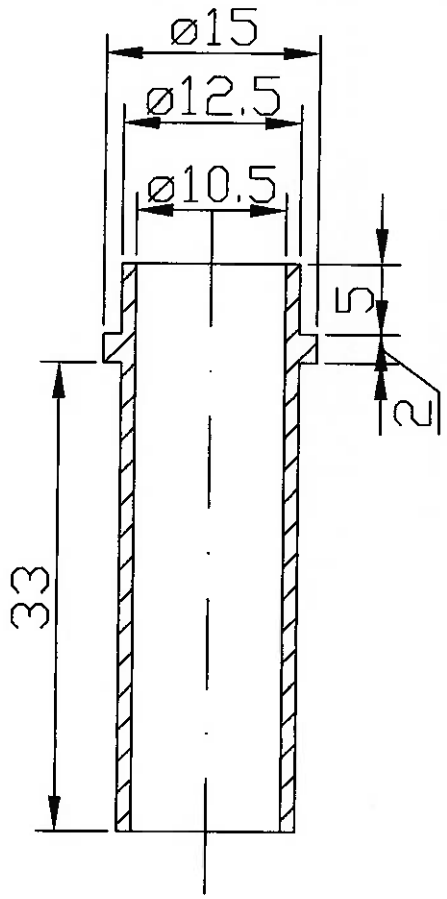








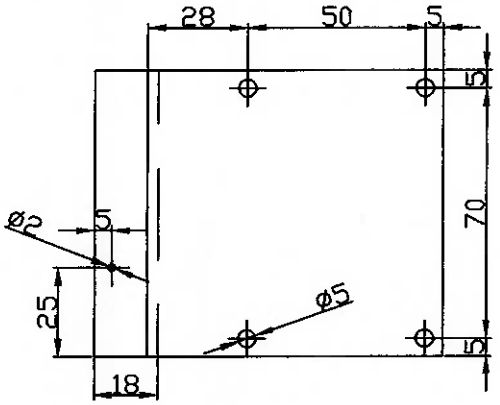
Item: 11	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:2
Título Base do conjunto principal			

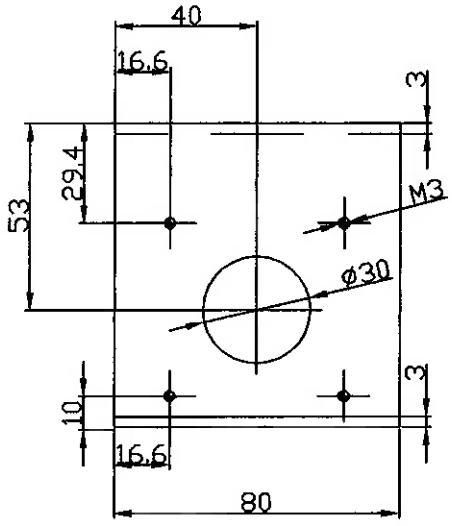


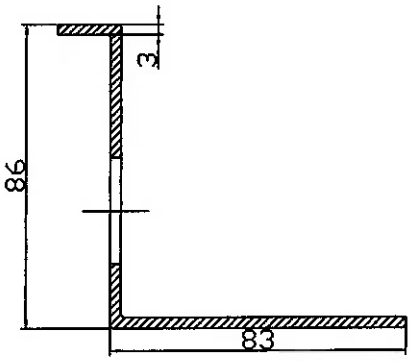
Item: 12	Quantidade: 01	Material: PP	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1

Título

Bucha de deslizamento do endoscópio



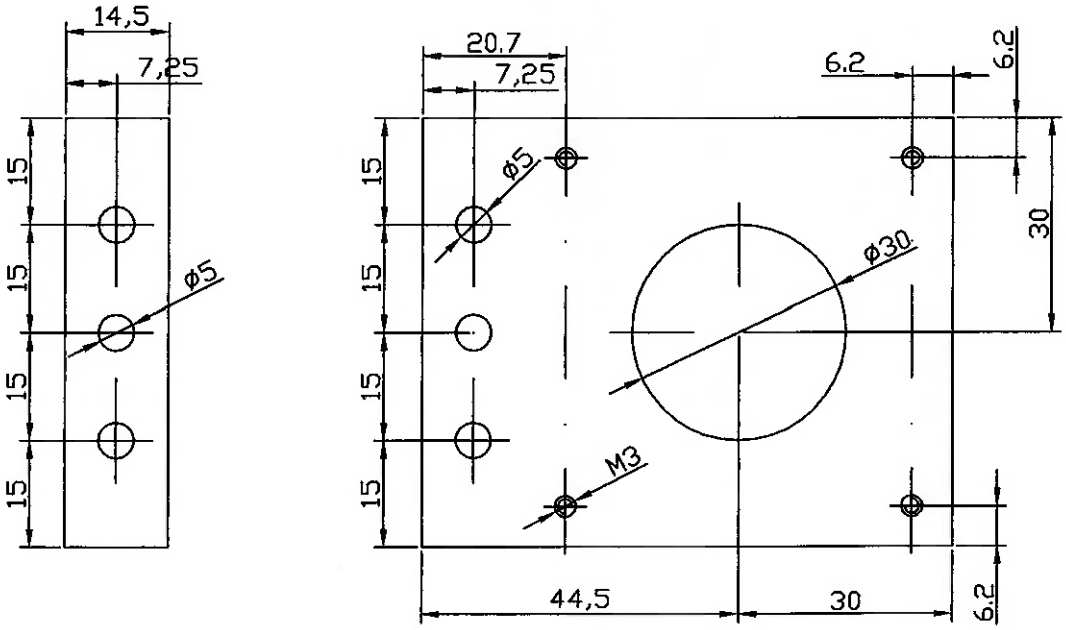




Item: 13	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:2

Título

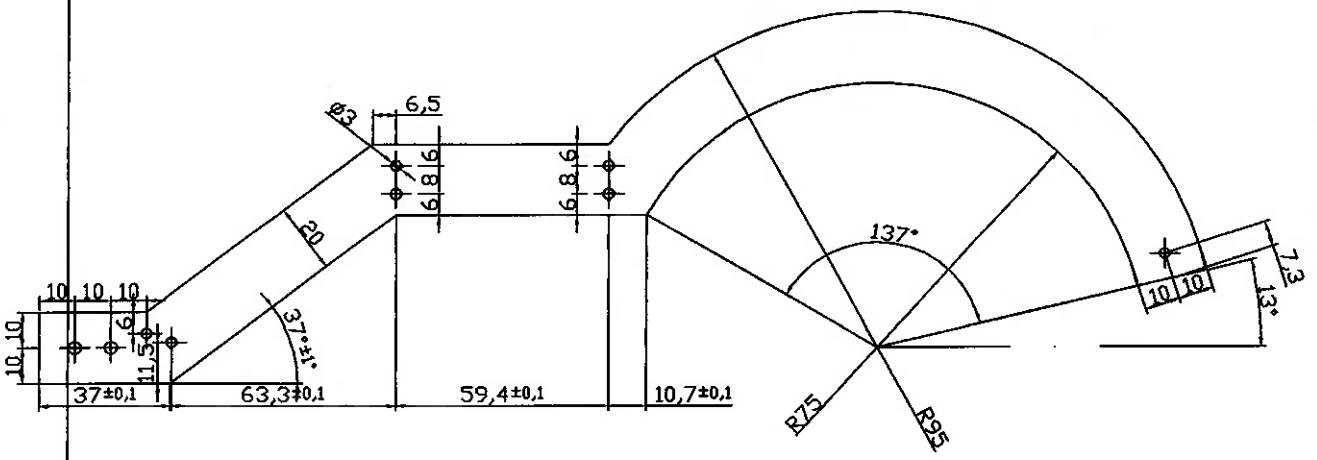
Base de motores do conjunto 3



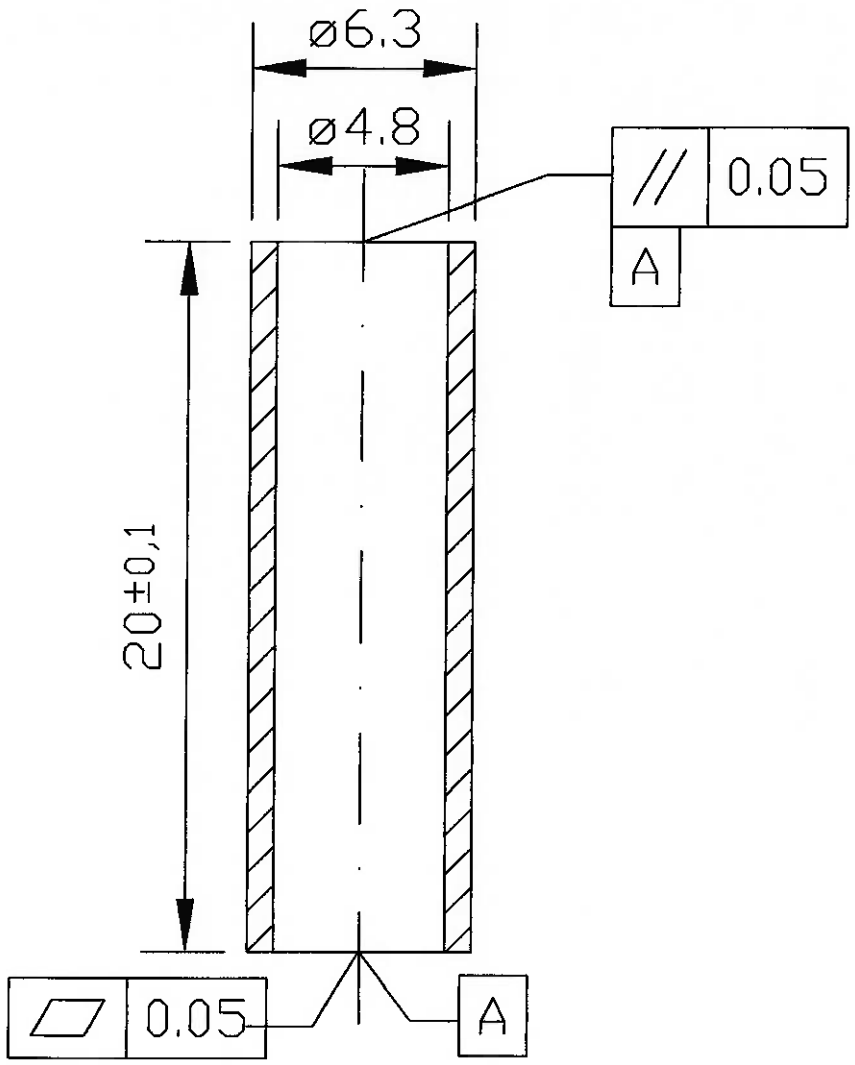
Item: 14	Quantidade: 01	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:1

Título

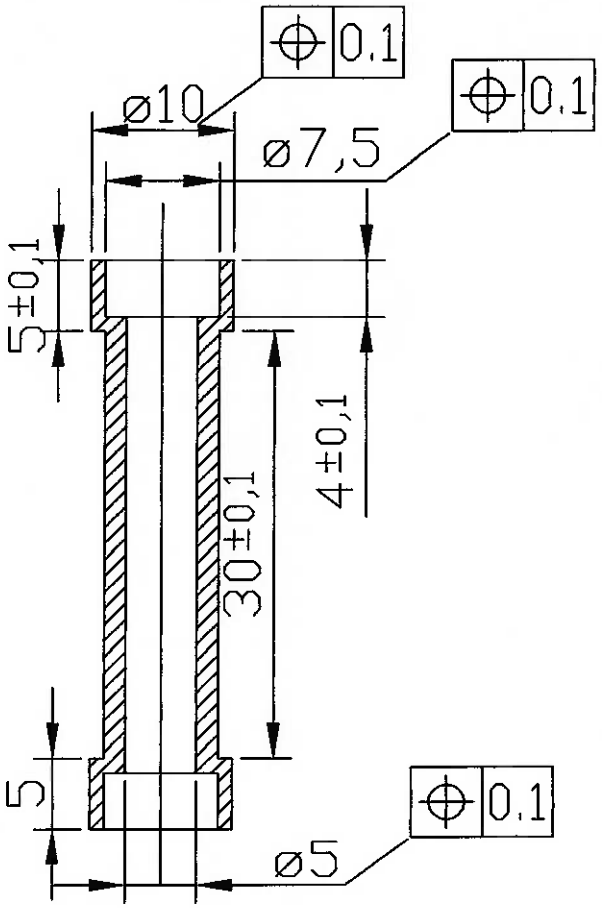
Base de motores secundária



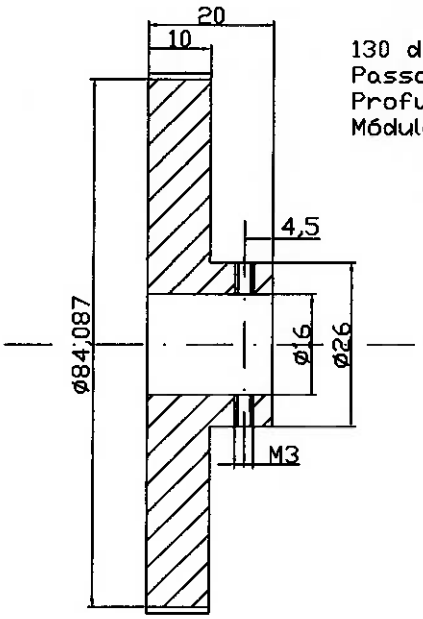
Item: 15	Quantidade: 2	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:2
Título Trilhos			



Item: 16	Quantidade: 14	Material: Cobre	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:1
Título Separadores			

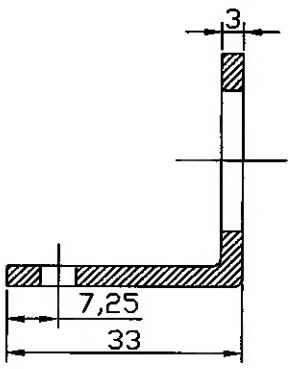


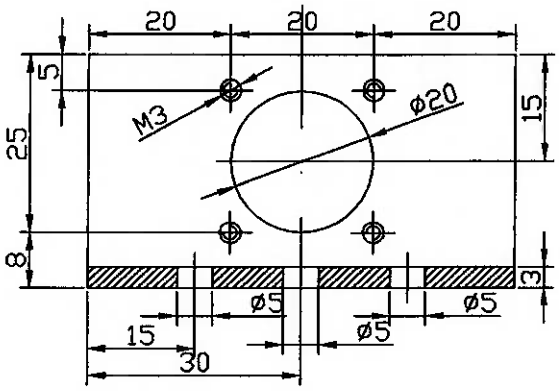
Item: 17	Quantidade: 4	Material: PP	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:1
Título Roletes para mov. do carro			



130 dentes
Passo = 0,8" (2,032mm)
Profundidade = 1,0mm
Módulo = 0,6468mm

Item: 18	Quantidade: 1	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:1
Título Polia de acoplamento			

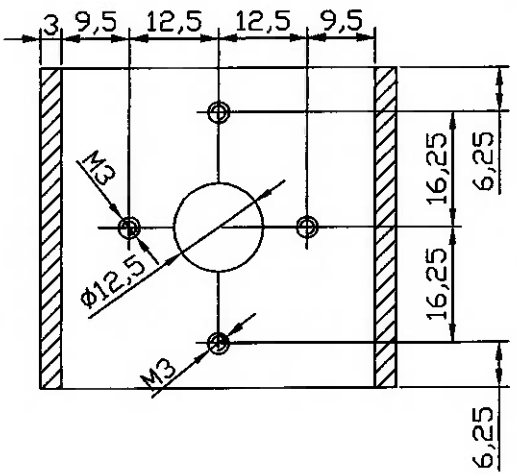


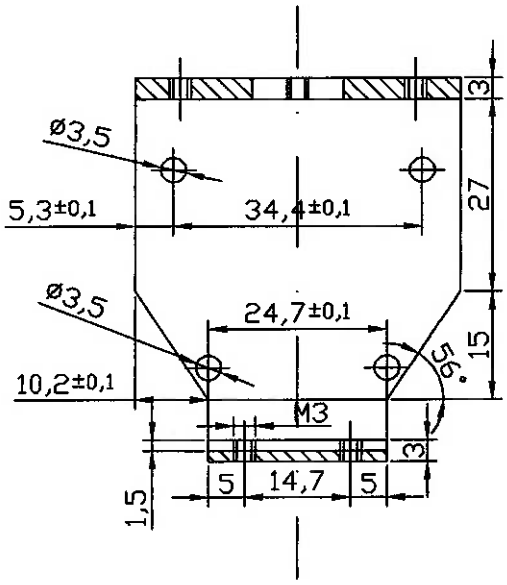


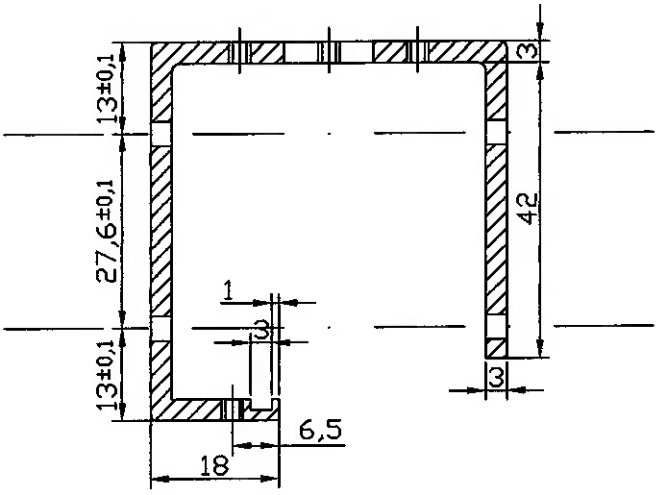
Item: 19	Quantidade: 1	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:1

Título

Base secundária - Peça 2



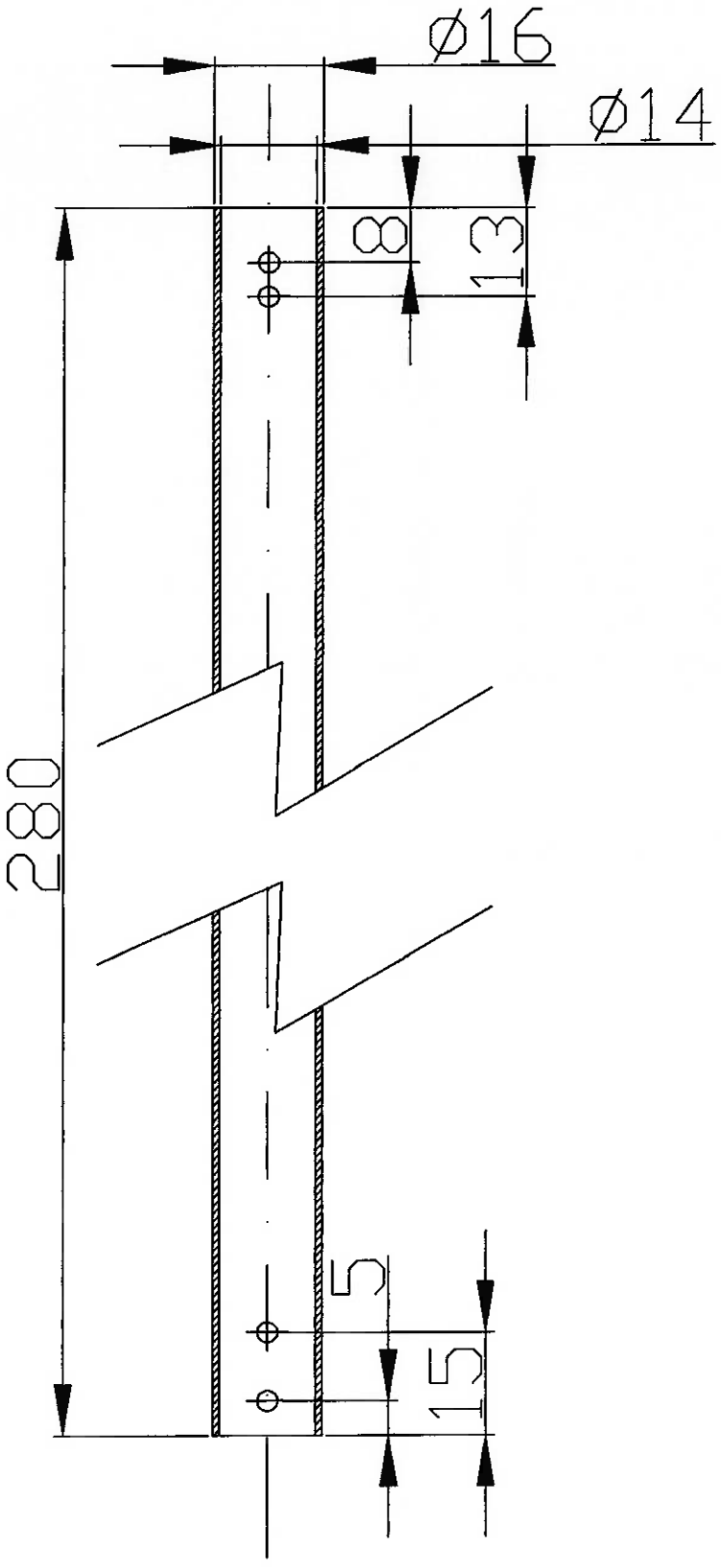




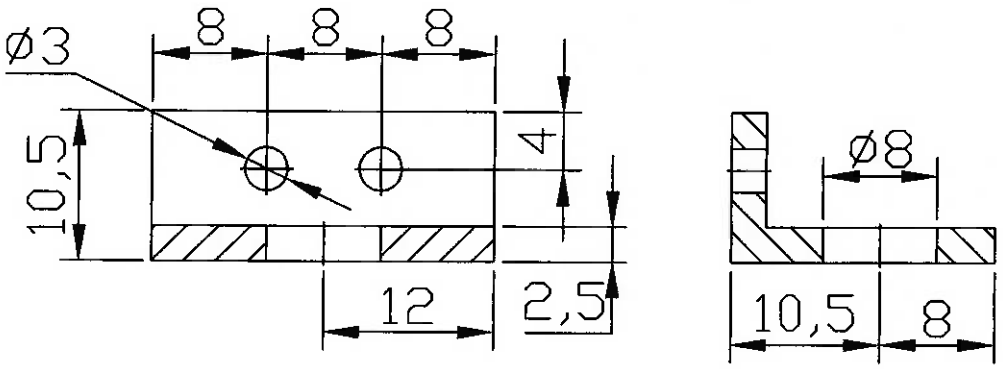
Item: 20	Quantidade: 1	Material: Alumínio

Responsável	Data	Escala
Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452	23/11/01	1:1

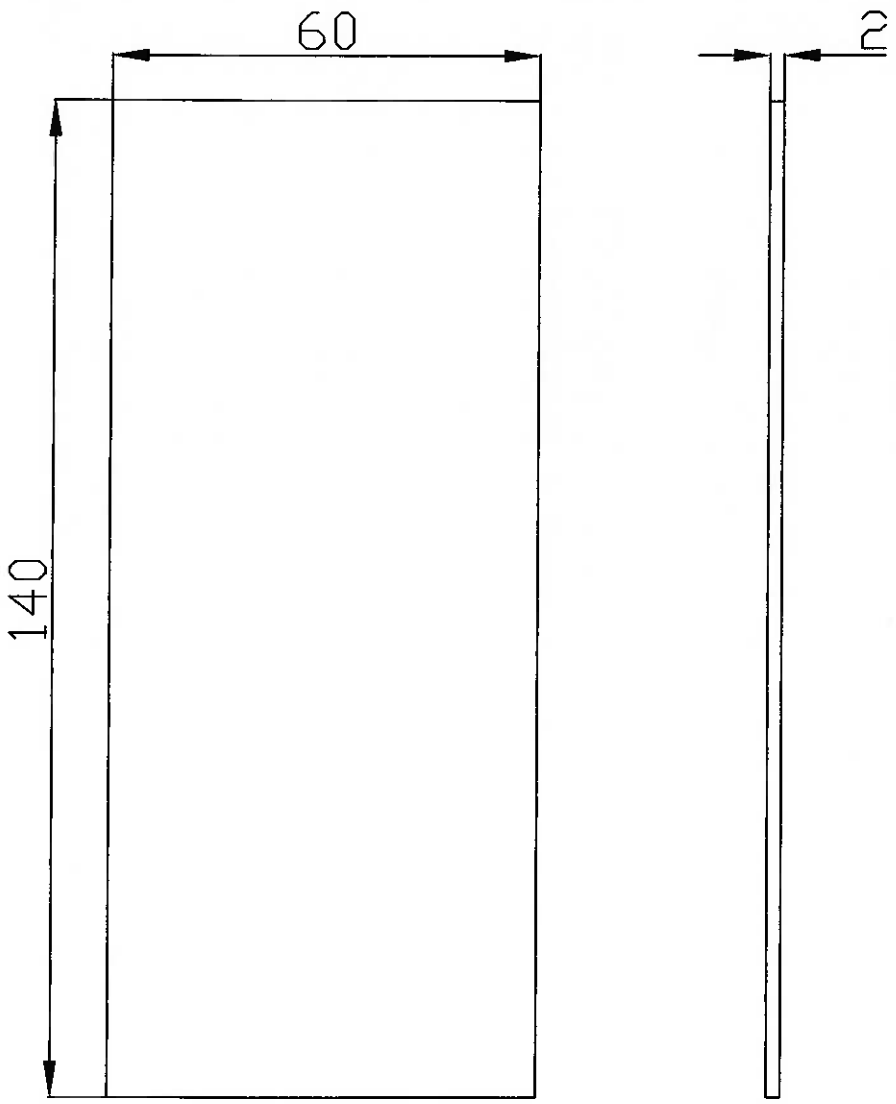
Título: Carro do endoscópio



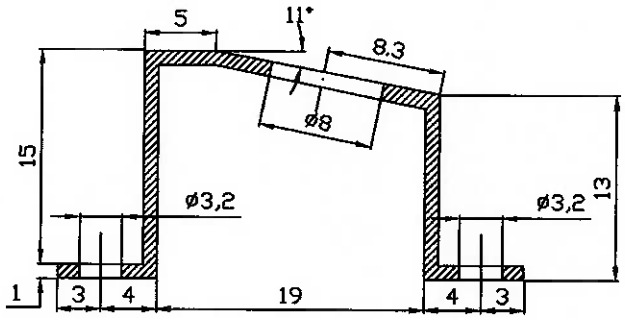
Item: 21	Quantidade: 1	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:1
Título Tubo de movimentação			

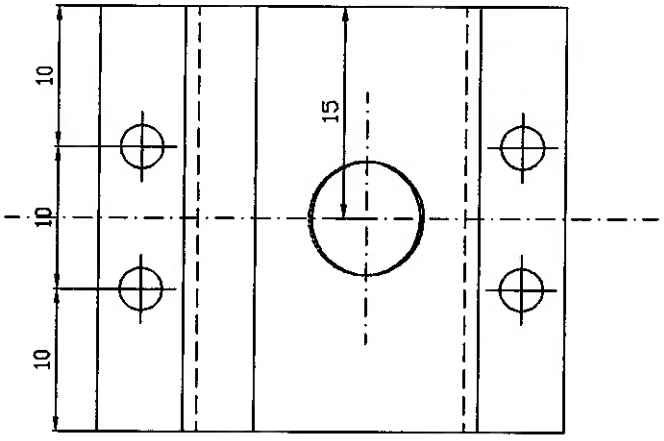


Item: 22	Quantidade: 1	Material: ABNT-1020	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1
Título Suporte do botão 3			

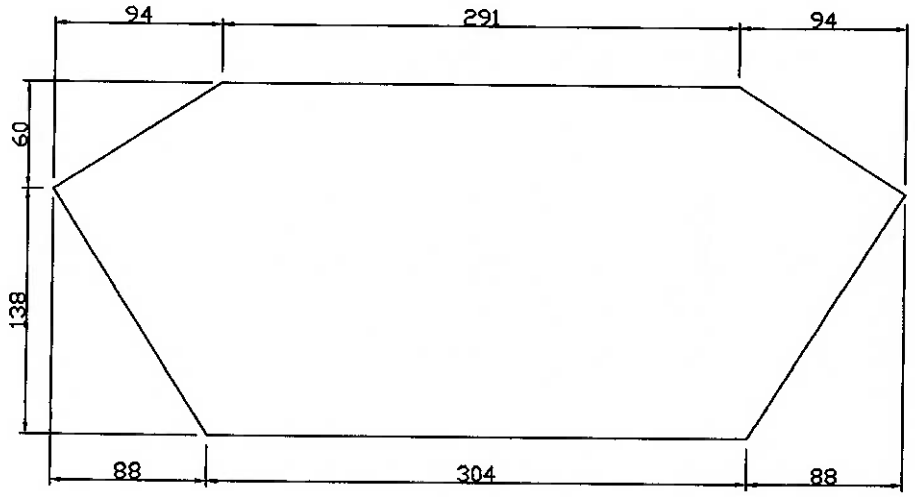


Item: 23	Quantidade: 2	Material: ABNT-1020	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/2001	Escala 1:1
Título Pedaleira			



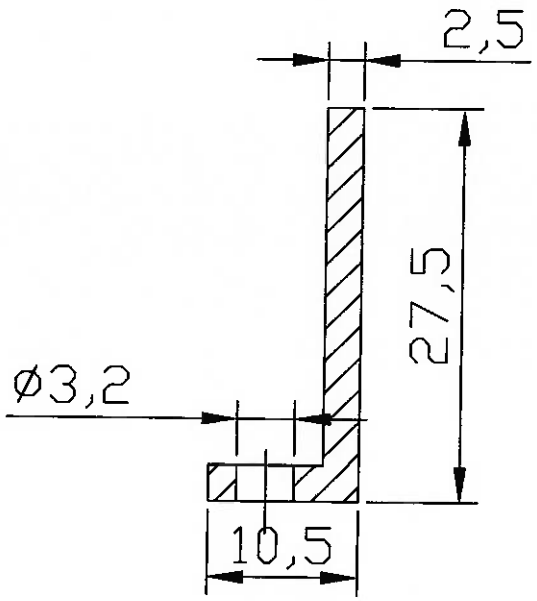


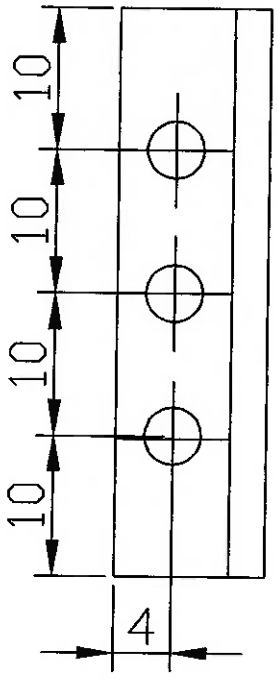
Item: 24	Quantidade: 2	Material: ABNT-1020	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1
Título Suporte para contato			



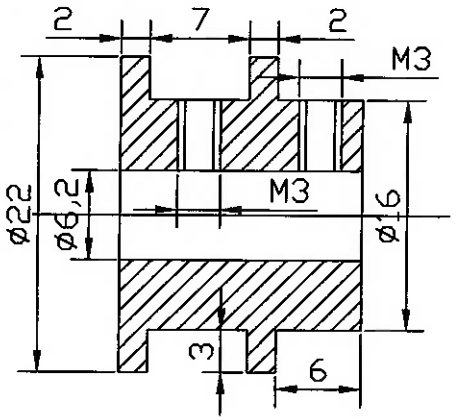
Espessura: 2mm

Item: 25	Quantidade: 1	Material: ABNT-1020	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 1:4
Título Base dos pedais			

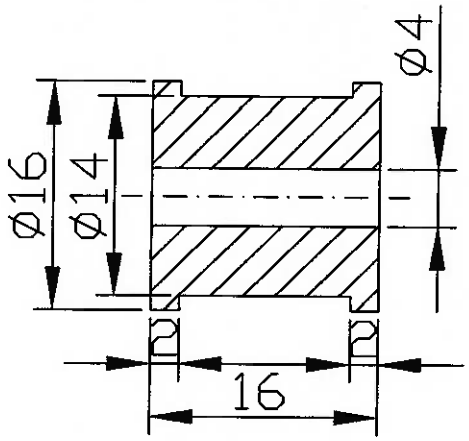




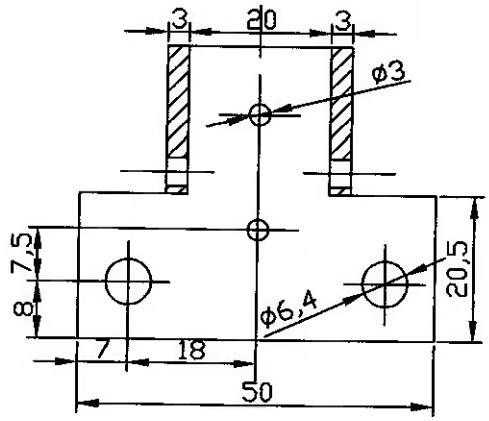
Item: 26	Quantidade: 2	Material: ABNT-1020	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 23/11/01	Escala 2:1
Título Apoio da pedaleira			

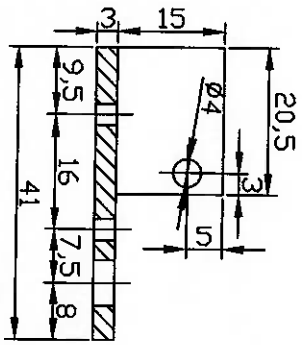


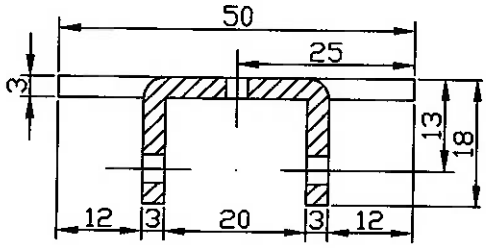
Item: 27	Quantidade: 2	Material: Latão	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 02/12/2001	Escala 2:1
Título Polias para o enrolamento dos cabos			



Item: 28	Quantidade: 28	Material: Latão	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 02/12/2001	Escala 2:1
Título Polia de acionamento do zoom			







Item: 28	Quantidade: 1	Material: Alumínio	
Responsável Marcos Alexandre Scholtz - 2367330 Hugo Leonardo Paiva Rodrigues - 2367452		Data 02/12/2001	Escala 1:1
Título Movimentador do endo. - Peça 3			

ANEXO III - LISTAGEM DOS ARQUIVOS FONTE

```
/*****************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO   ***
 ***                                                                     ***
 ***  Arquivo   :  VIDEO.H                                               ***
 ***  Conteudo  :  Procedimentos para geracao de saidas no video         ***
 ***  Descricao :  Contem todos os procedimentos utilizados para gerar   ***
 ***               saidas no video ou monitor, incluindo inicializacao   ***
 ***               e recuperacao do video e criacao de formas graficas   ***
 ***               utilizadas no software.                               ***
 ***  Autoria   :  Marcos Alexandre Scholtz       -  2367330            ***
 ***               Hugo Leonardo Paiva Rodrigues  -  2367452            ***
 ***  Data      :  13/09/2001                                            ***
 *****************************************************************************/

//Inicializa o video, em modo de exibicao 640x480, 16 cores
//Argumentos : nenhum
//Retorno    : nenhum
void inicializaVideo(void) {
   _setvideomode(_VRES16COLOR);
}

//Retorna o video para o modo padrao de exibicao
//Argumentos : nenhum
//Retorno    : nenhum
void recuperaVideo(void) {
   _setvideomode(_DEFAULTMODE);
}

//Limpa a tela completamente
//Argumentos : nenhum
//Retorno    : nenhum
void limpaTela(void) {
   _clearscreen(_GCLEARSCREEN);
}

//Altera a cor de um pixel da tela
//Argumentos : posX - Posicao horizontal do pixel na tela
//             posY - Posicao vertical do pixel na tela
//             cor  - Cor a atribuir ao pixel
//Retorno    : nenhum
void alteraPixel(int posX, int posY, int cor) {
   _setcolor(cor);             //Define a cor
   _setpixel(posX, posY);      //Desenha o pixel na cor
}

//Procedimento que recupera a cor de um pixel
//Argumentos : posX - Posicao horizontal do pixel na tela
//             posY - Posicao vertical do pixel na tela
//Retorno    : O codigo da cor do pixel na posicao passada
int recuperaPixel(int posX, int posY) {
   return _getpixel(posX, posY);   //Retorna a cor do pixel
}

//Desenha uma linha nas coordenadas passadas, na cor desejada
//Argumentos : Xini - Posicao horizontal do ponto inicial
//             Yini - Posicao vertical do ponto inicial
//             Xfim - Posicao horizontal do ponto final
//             Yfim - Posicao vertical do ponto final
```

```
//              cor  - Cor a desenhar a linha
//Retorno     : nenhum
void linha(int Xini, int Yini, int Xfim, int Yfim, int cor) {
   _setcolor(cor);        //Define a cor
   _moveto(Xini, Yini);   //Posiciona o inicio da linha
   _lineto(Xfim, Yfim);   //Desenha a linha
}

//Desenha um retangulo, com coordenadas, cor e preenchimento (1) ou nao (0)
//Argumentos : Xini - Posicao horizontal do ponto inicial
//             Yini - Posicao vertical do ponto inicial
//             Xfim - Posicao horizontal do ponto final
//             Yfim - Posicao vertical do ponto final
//             cor  - Cor a desenhar o retangulo
//             fill - Indica se usa preenchimento (1) ou nao (0)
//Retorno     : nenhum
void retangulo(int Xini, int Yini, int Xfim, int Yfim, int cor, int fill) {
   short control;

   if(fill == 0)                  //Determina a constante de acordo com fill
      control = _GBORDER;
   if(fill == 1)
      control = _GFILLINTERIOR;
   _setcolor(cor);                //Define a cor
   _rectangle(control, Xini, Yini, Xfim, Yfim);  //Desenha o retangulo
}

//Desenha uma elipse, com coordenadas, cor e preenchimento (1) ou nao (0)
//Argumentos : Xini - Posicao horizontal do ponto inicial
//             Yini - Posicao vertical do ponto inicial
//             Xfim - Posicao horizontal do ponto final
//             Yfim - Posicao vertical do ponto final
//             cor  - Cor a desenhar a elipse
//             fill - Indica se usa preenchimento (1) ou nao (0)
//Retorno     : nenhum
void elipse(int Xini, int Yini, int Xfim, int Yfim, int cor, int fill) {
   short control;

   if(fill == 0)                  //Determina a constante de acordo com fill
      control = _GBORDER;
   if(fill == 1)
      control = _GFILLINTERIOR;
   _setcolor(cor);                //Define a cor
   _ellipse(control, Xini, Yini, Xfim, Yfim);  //Desenha a elipse
}

//Procedimento que desenha uma seta para a direita, na posicao e cor dadas
//Argumentos : posX - Posicao horizontal do ponto inicial (Superior Esquerdo)
//             posY - Posicao vertical do ponto inicial (Superior Esquerdo)
//             cor  - Cor a desenhar a forma grafica
//Retorno     : nenhum
void setaDireita(int posX, int posY, int cor) {
   linha(posX,posY+7,posX,posY+19, cor);
   linha(posX,posY+7,posX+28,posY+7,cor);
   linha(posX,posY+19,posX+28,posY+19,cor);
   linha(posX+28,posY,posX+28,posY+7,cor);
   linha(posX+28,posY+19,posX+28,posY+26,cor);
   linha(posX+28,posY,posX+41,posY+13,cor);
```

```
    linha(posX+28,posY+26,posX+41,posY+13,cor);
}

//Procedimento que desenha uma seta para a esquerda, na posicao e cor dadas
//Argumentos : posX - Posicao horizontal do ponto inicial (Superior Esquerdo)
//             posY - Posicao vertical do ponto inicial (Superior Esquerdo)
//             cor  - Cor a desenhar a forma grafica
//Retorno    : nenhum
void setaEsquerda(int posX, int posY, int cor) {
   linha(posX+41,posY+7,posX+41,posY+19, cor);
   linha(posX+13,posY+7,posX+41,posY+7,cor);
   linha(posX+13,posY+19,posX+41,posY+19,cor);
   linha(posX+13,posY,posX+13,posY+7,cor);
   linha(posX+13,posY+19,posX+13,posY+26,cor);
   linha(posX+13,posY,posX,posY+13,cor);
   linha(posX+13,posY+26,posX,posY+13,cor);
}

//Procedimento que desenha uma seta para baixo, na posicao e cor dadas
//Argumentos : posX - Posicao horizontal do ponto inicial (Superior Esquerdo)
//             posY - Posicao vertical do ponto inicial (Superior Esquerdo)
//             cor  - Cor a desenhar a forma grafica
//Retorno    : nenhum
void setaBaixo(int posX, int posY, int cor) {
   linha(posX+7,posY,posX+19,posY, cor);
   linha(posX+7,posY,posX+7,posY+28,cor);
   linha(posX+19,posY,posX+19,posY+28,cor);
   linha(posX,posY+28,posX+7,posY+28,cor);
   linha(posX+19,posY+28,posX+26,posY+28,cor);
   linha(posX,posY+28,posX+13,posY+41,cor);
   linha(posX+26,posY+28,posX+13,posY+41,cor);
}

//Procedimento que desenha uma seta para cima, na posicao e cor dadas
//Argumentos : posX - Posicao horizontal do ponto inicial (Superior Esquerdo)
//             posY - Posicao vertical do ponto inicial (Superior Esquerdo)
//             cor  - Cor a desenhar a forma grafica
//Retorno    : nenhum
void setaCima(int posX, int posY, int cor) {
   linha(posX+7,posY+41,posX+19,posY+41, cor);
   linha(posX+7,posY+41,posX+7,posY+13,cor);
   linha(posX+19,posY+41,posX+19,posY+13,cor);
   linha(posX,posY+13,posX+7,posY+13,cor);
   linha(posX+19,posY+13,posX+26,posY+13,cor);
   linha(posX,posY+13,posX+13,posY,cor);
   linha(posX+26,posY+13,posX+13,posY,cor);
}

//Procedimento que desenha uma lupa com um "+", indicando zoom in, na
//posicao e cor passadas no argumento.
//Argumentos : posX - Posicao horizontal do ponto inicial (Superior Esquerdo)
//             posY - Posicao vertical do ponto inicial (Superior Esquerdo)
//             cor  - Cor a desenhar a forma grafica
//Retorno    : nenhum
void zoomIn(int posX, int posY, int cor) {
   elipse(posX,posY,posX+20,posY+20,cor,0);
   elipse(posX+3,posY+3,posX+17,posY+17,cor,0);
   linha(posX+16,posY+19,posX+24,posY+27,cor);
```

```
    linha(posX+19,posY+16,posX+27,posY+24,cor);
    linha(posX+25,posY+27,posX+27,posY+25,cor);
    linha(posX+6,posY+10,posX+14,posY+10,cor);
    linha(posX+6,posY+11,posX+14,posY+11,cor);
    linha(posX+10,posY+6,posX+10,posY+14,cor);
    linha(posX+11,posY+6,posX+11,posY+14,cor);
}

//Procedimento que desenha uma lupa com um "-", indicando zoom out, na
//posicao e cor passadas no argumento.
//Argumentos : posX - Posicao horizontal do ponto inicial (Superior Esquerdo)
//             posY - Posicao vertical do ponto inicial (Superior Esquerdo)
//             cor  - Cor a desenhar a forma grafica
//Retorno    : nenhum
void zoomOut(int posX, int posY, int cor) {
   elipse(posX,posY,posX+20,posY+20,cor,0);
   elipse(posX+3,posY+3,posX+18,posY+18,cor,0);
   linha(posX+16,posY+19,posX+24,posY+27,cor);
   linha(posX+19,posY+16,posX+27,posY+24,cor);
   linha(posX+25,posY+27,posX+27,posY+25,cor);
   linha(posX+6,posY+10,posX+14,posY+10,cor);
   linha(posX+6,posY+11,posX+14,posY+11,cor);
}
```

```
/*****************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO   ***
 ***                                                                     ***
 ***  Arquivo    :  TECLADO.H                                           ***
 ***  Conteudo   :  Procedimentos de leitura do teclado                 ***
 ***  Descricao  :  Contem alguns procedimentos utilizados no software  ***
 ***                para facilitar a leitura de dados do teclado. De    ***
 ***                maneira geral, verifica-se se ha' dados no buffer do***
 ***                teclado, e le-se estes dados de formas diferentes.  ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330           ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452           ***
 ***  Data       :  13/09/2001                                          ***
 *****************************************************************************/

//Procedimento que limpa o buffer do teclado
//Argumentos : nenhum
//Retorno    : nenhum
void limpaTeclado(void) {
   char tecla;

   while(kbhit())        //Executa enquanto o buffer nao estiver vazio
      tecla = getch();    //Remove a tecla do buffer
}

//Funcao que verifica se ha teclas a serem pegas do teclado.
//Pega apenas a PROXIMA tecla da lista. Retorna o codigo ASC.
//Argumentos : nenhum
//Retorno    : Codigo ASC da tecla pressionada, ou 0 caso nenhuma
int verificaTecla(void) {
   char tecla;
   int cod;

   cod = 0;            //Caso nao haja tecla, retorna 0
   if(kbhit()) {       //Executa caso haja teclas aguardando no buffer
      tecla = getch();   //Caso haja, recupera a tecla do buffer
      cod = (int)tecla;  //recupera o codigo da tecla
   }
   return cod;     //Retorna o codigo ASC, ou 0
}

//Funcao que verifica se ha teclas a serem pegas do teclado.
//Pega apenas a ULTIMA tecla da lista. Retorna o codigo ASC.
//Argumentos : nenhum
//Retorno    : Codigo ASC da tecla pressionada, ou 0 caso nenhuma
int verificaUltimaTecla(void) {
   char tecla;
   int cod;

   cod = 0;            //Caso nao haja tecla, retorna 0
   while(kbhit()) {       //Utiliza um while, retornara apenas a ultima tecla
      tecla = getch();       //Caso haja, recupera a tecla do buffer
      cod = (int)tecla;      //recupera o codigo da tecla
   }
   return cod;    //Retorna o codigo ASC, ou 0
}

//Funcao que verifica se ha teclas a serem pegas do teclado.
//Pega apenas a PRIMEIRA tecla da lista. Retorna o codigo ASC.
```

```
//Argumentos : nenhum
//Retorno    : Codigo ASC da tecla pressionada, ou 0 caso nenhuma
int verificaPrimeiraTecla(void) {
   char tecla;
   int cod;

   cod = 0;            //Caso nao haja tecla, retorna 0
   if(kbhit()) {          //Executa caso haja teclas aguardando no buffer
      tecla = getch();    //Caso haja, recupera a tecla do buffer
      cod = (int)tecla;  //recupera o codigo da tecla
   }
   while(kbhit())        //Utiliza um while para limpar o buffer do teclado
      tecla = getch();
   return cod;         //Retorna codigo ASC ou 0
}

//Funcao que retorna a ultima tecla digitada, OU entao aguarda a digitacao
//de uma tecla para retorno. Retorna o codigo ASC.
//Argumentos : nenhum
//Retorno    : Codigo ASC da tecla pressionada, ou 0 caso nenhuma
int recuperaUltimaTecla(void) {
   char tecla;
   int cod;

   if(kbhit()) {                    //Caso haja teclas digitadas aguardando
      cod = verificaUltimaTecla();  //retorna apenas a ultima delas.
   } else {
      tecla = getch();              //Caso contrario, aguarda digitar
      cod = (int)tecla;             //uma tecla para retorna-la.
   }
   return cod;    //Retorna o codigo ASC, ou 0
}
```

```
/*****************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO     ***
 ***                                                                       ***
 ***  Arquivo    :  MOUSE.H                                               ***
 ***  Conteudo   :  Procedimentos de leitura e configuracao do mouse      ***
 ***  Descricao  :  Contem os procedimentos que efetuam a interacao com   ***
 ***                um mouse convencional. Utiliza-se de interrupcoes de  ***
 ***                software e das funcoes do driver para mouse criado    ***
 ***                pela Microsoft (versao 7.0 ou superior). Possui       ***
 ***                procedimentos de configuracao (como sensibilidade),   ***
 ***                posicionamento e leitura dos dados do mouse. Inclui   ***
 ***                a definicao e propriedades do dispositivo de entrada  ***
 ***                de dados (mouse adaptado).                            ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330             ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452             ***
 ***  Data       :  13/09/2001                                            ***
 *****************************************************************************/

//Constantes que identificam como o sistema deve transformar os pulsos dos
//encoders do mouse em posicionamento na tela. As constantes sÆo
//multiplicadas pelos pulsos para compor os sinais.
#define MULT_POSX_ENCODERX 0
#define MULT_POSX_ENCODERY +1
#define MULT_POSY_ENCODERX +1
#define MULT_POSY_ENCODERY 0

//Define a estrutura que contem os dados do dispositivo de entrada (mouse)
struct DISPOSITIVO {
   int visivel;         //Indica se o cursor esta visivel
   int botE;            //Indica o status do botao esquerdo, a nivel
   int botD;            //Indica o status do botao direito, a nivel
   int botM;            //Indica o status do botao central, a nivel
   int botEborda;       //Status do botao esquerdo, sensivel a borda descida
   int botDborda;       //Status do botao direito, sensivel a borda descida
   int botMborda;       //Status do botao central, sensivel a borda descida
   int micX;            //Numero de 'Mickeys' ABSOLUTOS no eixo X
   int micY;            //Numero de 'Micheys' ABSOLUTOS no eixo Y
   int sensX;           //Sensibilidade horizontal (mickeys / 10 pixels)
   int sensY;           //Sensibilidade vertical (mickeys / 10 pixels)
   int posX;            //Posicao horizontal atual
   int posY;            //Posicao vertical atual
   int posXant;         //Posicao horizontal antiga
   int posYant;         //Posicao vertical antiga
} dispositivo;

int corPixel[10];          //Contem as cores dos pixels sob o cursor

//Procedimento que atualiza o cursor, apos mover o mouse
//Argumentos : nenhum
//Retorno    : nenhum
void atualizaCursor(void) {
   //So' executa codigo caso o cursor esteja visivel
   if(dispositivo.visivel == 1) {
      //Apaga o cursor antigo (redesenha o fundo)
      alteraPixel(dispositivo.posXant - 2, dispositivo.posYant, corPixel[0]);
      alteraPixel(dispositivo.posXant - 1, dispositivo.posYant, corPixel[1]);
      alteraPixel(dispositivo.posXant, dispositivo.posYant, corPixel[2]);
      alteraPixel(dispositivo.posXant + 1, dispositivo.posYant, corPixel[3]);
```

```
      alteraPixel(dispositivo.posXant + 2, dispositivo.posYant, corPixel[4]);
      alteraPixel(dispositivo.posXant, dispositivo.posYant - 2, corPixel[5]);
      alteraPixel(dispositivo.posXant, dispositivo.posYant - 1, corPixel[6]);
      alteraPixel(dispositivo.posXant, dispositivo.posYant + 1, corPixel[7]);
      alteraPixel(dispositivo.posXant, dispositivo.posYant + 2, corPixel[8]);
      //Recupera o fundo da posicao atual onde o cursor sera desenhado
      corPixel[0] = recuperaPixel(dispositivo.posX - 2, dispositivo.posY);
      corPixel[1] = recuperaPixel(dispositivo.posX - 1, dispositivo.posY);
      corPixel[2] = recuperaPixel(dispositivo.posX, dispositivo.posY);
      corPixel[3] = recuperaPixel(dispositivo.posX + 1, dispositivo.posY);
      corPixel[4] = recuperaPixel(dispositivo.posX + 2, dispositivo.posY);
      corPixel[5] = recuperaPixel(dispositivo.posX, dispositivo.posY - 2);
      corPixel[6] = recuperaPixel(dispositivo.posX, dispositivo.posY - 1);
      corPixel[7] = recuperaPixel(dispositivo.posX, dispositivo.posY + 1);
      corPixel[8] = recuperaPixel(dispositivo.posX, dispositivo.posY + 2);
      //Desenha o cursor na posicao atual (cruz 5 x 5 pixels)
      linha(dispositivo.posX - 2, dispositivo.posY, dispositivo.posX + 2,
dispositivo.posY, 12);
      linha(dispositivo.posX, dispositivo.posY - 2, dispositivo.posX,
dispositivo.posY + 2, 12);
   }
}

//Procedimento que inicia o driver do mouse (reset)
//Argumentos : nenhum
//Retorno    : Status do driver do mouse (0 caso nao instalado)
int iniciaMouse(void) {
   int retorno;
   //Funcao : 00h
   //ax   : registrador de funcao do driver
   //33h : interrupcao de software do driver do mouse
   //Retorno : ax - status do driver de mouse
   _asm {
      mov ax,0
      int 33h
      mov retorno,ax
   }
   dispositivo.visivel = 0;   //Inicializa com o cursor nao visivel
   dispositivo.micX = 0;      //Inicializa com o cursor centralizado
   dispositivo.micY = 0;
   dispositivo.posX = 319;
   dispositivo.posY = 239;
   dispositivo.botE = 0;      //Inicializa sem botoes pressionados
   dispositivo.botD = 0;
   dispositivo.botEborda = 0;
   dispositivo.botDborda = 0;
   dispositivo.sensX = 10;    //Inicializa a sensibilidade
   dispositivo.sensY = 10;
   return retorno;         //Retorna o status do driver de mouse
}

//Procedimento que posiciona o cursor do mouse
//Argumentos : posX - Posicao horizontal a colocar o cursor
//             posY - Posicao vertical a colocar o cursor
//Retorno    : nenhum
void posicionaCursor(int posX, int posY) {
   //Atualiza a contagem de pulsos dos encoders
   dispositivo.micX = (int)((posX - 319) * dispositivo.sensX / 10);
```

```
   dispositivo.micY = (int)((posY - 239) * dispositivo.sensY / 10);
   //Atualiza a posicao de acordo com a contagem
   dispositivo.posX = 319 + (int)((dispositivo.micX * 10) / dispositivo.sensX);
   dispositivo.posY = 239 + (int)((dispositivo.micY * 10) / dispositivo.sensY);
   atualizaCursor();   //Move o cursor para a nova posicao
}

//Procedimento que atualiza todos os parametros do dispositivo.
//Argumentos : nenhum
//Retorno    : nenhum
void atualizaMouse(void) {
   int botoes, novoMicX, novoMicY, novoPosX, novoPosY, novoBotao;

   //Funcao : 03h
   //ax  : registrador de funcao do driver
   //33h : interrupcao de software do driver do mouse
   //Retorno : bx - status dos botoes
   //          cx - coordenada horizontal do cursor
   //          dx - coordenada vertical do cursor
   //Funcao : 1bh
   //ax  : registrador de funcao do driver
   //33h : interrupcao de software do driver do mouse
   //Retorno : cx - movimento incremental (em mickeys), eixo X
   //          dx - movimento incremental (em mickeys), eixo Y
   _asm {
      mov ax,3
      int 33h
      mov botoes,bx
      mov ax,0bh
      int 33h
      mov novoMicX,cx
      mov novoMicY,dx
   }
   if(botoes & 0x01 == 1) {    //Retorna o bit 0 (botao esquerdo)
      novoBotao = 0;   //E' preciso inverter o bit
   } else {
      novoBotao = 1;
   }
   if(novoBotao != dispositivo.botE) {
      dispositivo.botEborda = novoBotao;  //Ha borda, o status retorna o
   } else {                               //valor do botao.
      dispositivo.botEborda = 0;     //Nao ha borda
   }
   dispositivo.botE = novoBotao;
   if((botoes & 0x02) / 2 == 1) {    //Retorna o bit 1 (botao direito)
      novoBotao = 0;          //E' preciso inverter o bit
   } else {
      novoBotao = 1;
   }
   if(novoBotao != dispositivo.botD) {
      dispositivo.botDborda = novoBotao;  //Ha borda, o status retorna o
   } else {                               //valor do botao.
      dispositivo.botDborda = 0;     //Nao ha borda
   }
   dispositivo.botD = novoBotao;
   if((botoes & 0x04) / 4 == 1) {    //Retorna o bit 2 (botao central)
      novoBotao = 0;          //E' preciso inverter o bit
   } else {
```

```
      novoBotao = 1;
   }
   if(novoBotao != dispositivo.botM) {
      dispositivo.botMborda = novoBotao;  //Ha borda, o status retorna o
   } else {                               //valor do botao.
      dispositivo.botMborda = 0;      //Nao ha borda
   }
   dispositivo.botM = novoBotao;
   //Soma o movimento, transformando o valor incremental em valor absoluto
   dispositivo.micX = dispositivo.micX + (MULT_POSX_ENCODERX * novoMicX) +
(MULT_POSX_ENCODERY * novoMicY);
   dispositivo.micY = dispositivo.micY + (MULT_POSY_ENCODERX * novoMicX) +
(MULT_POSY_ENCODERY * novoMicY);
   //Calcula as novas posicoes de acordo com a contagem dos encoders
   novoPosX = 319 + (int)((dispositivo.micX * 10) / dispositivo.sensX);
   novoPosY = 239 + (int)((dispositivo.micY * 10) / dispositivo.sensY);
   //Verifica se houve movimentacao
   if(novoPosX != dispositivo.posX || novoPosY != dispositivo.posY) {
      dispositivo.posXant = dispositivo.posX;   //Grava posicao anterior
      dispositivo.posYant = dispositivo.posY;
      dispositivo.posX = novoPosX;         //Seta a posicao atual
      dispositivo.posY = novoPosY;
      atualizaCursor();        //Atualiza o cursor na tela
   }
}

//Procedimento que ativa ou desativa o cursor
//Argumentos : ativado - Indica se quer ativar (1) ou desativar (!=1) cursor
//Retorno    : nenhum
void ativaCursor(int ativado) {
   if(ativado == 1) {    //Caso se esteja ativando o cursor
      //Recupera o fundo da posicao atual do cursor
      corPixel[0] = recuperaPixel(dispositivo.posX - 2, dispositivo.posY);
      corPixel[1] = recuperaPixel(dispositivo.posX - 1, dispositivo.posY);
      corPixel[2] = recuperaPixel(dispositivo.posX, dispositivo.posY);
      corPixel[3] = recuperaPixel(dispositivo.posX + 1, dispositivo.posY);
      corPixel[4] = recuperaPixel(dispositivo.posX + 2, dispositivo.posY);
      corPixel[5] = recuperaPixel(dispositivo.posX, dispositivo.posY - 2);
      corPixel[6] = recuperaPixel(dispositivo.posX, dispositivo.posY - 1);
      corPixel[7] = recuperaPixel(dispositivo.posX, dispositivo.posY + 1);
      corPixel[8] = recuperaPixel(dispositivo.posX, dispositivo.posY + 2);
      //Desenha o cursor na posicao atual (cruz 5 x 5)
      linha(dispositivo.posX - 2, dispositivo.posY, dispositivo.posX + 2,
dispositivo.posY, 12);
      linha(dispositivo.posX, dispositivo.posY - 2, dispositivo.posX,
dispositivo.posY + 2, 12);
      dispositivo.visivel = 1;   //Indica que o cursor esta visivel
   } else {   //Caso se esteja desativando o cursor
      //Apaga o cursor (redesenha o fundo)
      alteraPixel(dispositivo.posX - 2, dispositivo.posY, corPixel[0]);
      alteraPixel(dispositivo.posX - 1, dispositivo.posY, corPixel[1]);
      alteraPixel(dispositivo.posX, dispositivo.posY, corPixel[2]);
      alteraPixel(dispositivo.posX + 1, dispositivo.posY, corPixel[3]);
      alteraPixel(dispositivo.posX + 2, dispositivo.posY, corPixel[4]);
      alteraPixel(dispositivo.posX, dispositivo.posY - 2, corPixel[5]);
      alteraPixel(dispositivo.posX, dispositivo.posY - 1, corPixel[6]);
      alteraPixel(dispositivo.posX, dispositivo.posY + 1, corPixel[7]);
      alteraPixel(dispositivo.posX, dispositivo.posY + 2, corPixel[8]);
```

```
        dispositivo.visivel = 0;    //Indica que o cursor nao esta' visivel
    }
}
```

```
/*****************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO    ***
 ***                                                                      ***
 ***  Arquivo    :  TIMER.H                                               ***
 ***  Conteudo   :  Procedimentos de configuracao e utilizacao do timer   ***
 ***  Descricao  :  Contem toda a programacao do timer, que reconfigura   ***
 ***                a interrupcao de clock do PC (IRQ 0) de forma a se    ***
 ***                atingir precisoes de microsegundos. Inclui procedim.  ***
 ***                para a inicializacao do timer, recuperacao ao seu     ***
 ***                estado original e leitura do mesmo, alem de outros    ***
 ***                mais altos para a espera de periodos determinados,    ***
 ***                considerando ou nao o tempo gasto na execucao de      ***
 ***                outras funcoes ou procedimentos.                      ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330             ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452             ***
 ***  Data       :  13/09/2001                                            ***
 *****************************************************************************/

//Resolucao do Timer (em ciclos por milisegundo)
#define Resolucao 1193.181667

//Variaveis de controle global de tempo decorrido
long Tempo_Inicial;
long Tempo_Final;

//Funcao que trata de numeros negativos do contador
//Argumentos : l  - Numero longo que se deseja tratar
//Retorno    : O numero apos ser tratado o caso de numero negativo
double cardinal(long l) {
   //Caso menor que zero, retorna o valor somado da constante
   return((l<0)?4294967296.0 + (long)l : (long)l);
}

//Funcao que calcula o tempo decorrido entre dois tempos (em ms)
//Argumentos : inicio - O tempo de inicio (em contagens de CLOCK)
//             fim    - O tempo de fim (em contagens de CLOCK)
//Retorno    : Diferenca de tempo entre inicio e fim, em ms
double tempoDecorrido(long inicio, long fim) {
   //Retorna a diferenca de clocks / resolucao do timer (clocks/ms)
   return(cardinal(fim - inicio) / Resolucao);
}

//Procedimento que inicializa o timer (altera a frequencia do mesmo)
//Argumentos : nenhum
//Retorno    : nenhum
void inicializaTimer(void) {
   outp(0x043,0x034);
   _asm {
      jmp short NullJump1
      NullJump1:;
   }
   outp(0x040,0x000);
   _asm {
      jmp short NullJump2
      NullJump2:;
   }
   outp(0x040,0x000);
}
```

```
//Procedimento que recupera o timer (volta a frequencia original)
//Argumentos : nenhum
//Retorno    : nenhum
void recuperaTimer(void) {
   outp(0x043,0x036);
   _asm {
      jmp short NullJump1
     NullJump1:;
   }
   outp(0x040,0x000);
   _asm {
      jmp short NullJump2
     NullJump2:;
   }
   outp(0x040,0x000);
}

//Funcao que le o valor do timer
//Argumentos : nenhum
//Retorno    : Um valor longo indicando a quantia de clocks do contador
long leTimer(void) {
   //Executa todo o codigo em assembler
   _asm {
      cli               // Desabilita interrupcoes
      mov dx,020h       // endereco PIC ocw3
      mov al,00Ah       // Pede para ler irr
      out dx,al
      mov al,00h        // Latch do timer 0
      out 043h,al
      in al,dx          // Le irr
      mov di,ax         // Grava o valor em DI
      in al,040h        // Contador --> bx
      mov bl,al         // LSB em BL
      in al,040h
      mov bh,al         // MSB em BH
      not bx            // E' necessario contagem crescente
      in al,021h        // Le PIC imr
      mov si,ax         // Grava o valor em SI
      mov al,00FFh      // Mascara as interrupcoes
      out 021h,al
      mov ax,040h       // Le o tempo
      mov es,ax         // da area de dados da BIOS
      mov dx,es:[06Ch]
      mov ax,si         // Recupera imr de SI
      out 021h,al
      sti               // Habilita interrupcoes
      mov ax,di         // Recupera irr antigo
      test al,001h      // Contador atingiu 0 ?
      jz done           // Pula caso nao
      cmp bx,0FFh       // Contador > 0x0FF ?
      ja done           // Terminado, caso verdadeiro
      inc dx            // Caso contrario conta interrupcao
     done:;
      mov ax,bx         // Atribui o resultado da funcao
   }
}
```

```
//Funcao que espera uma determinada quantia de tempo, em MICROSEGUNDOS
//Argumentos : tempo - o tempo total a aguardar, em microsegundos
//Retorno    : nenhum
void espera(long tempo) {
   long inicio, fim;

   inicio = leTimer();      //Le o valor no inicio
   fim = leTimer();         //Inicializa o fim
   while(tempoDecorrido(inicio, fim)*1000 < tempo)
      fim = leTimer();      //Le fim ate que o tempo desejado seja atingido
}

//Procedimento que inicia uma contagem de tempo que considera tempo decorrido
//em outros procedimentos ou funcoes
//Argumentos : nenhum
//Retorno    : nenhum
void iniciaContagem(void) {
   Tempo_Inicial = leTimer();   //Le o tempo atual e grava na variavel global
}

//Procedimento que termina uma contagem de tempo, considerando o tempo
//decorrido em outros procedimentos ou funcoes, com periodo passado.
//O periodo e' dado em MICROSEGUNDOS.
//Argumentos : tempo - Tempo que se deseja aguardar, em microsegundos
//Retorno    : nenhum
void finalizaContagem(long tempo) {
   Tempo_Final = leTimer();       //Inicializa o tempo de fim
   //Aguarda ate que o tempo decorrido tenha passado
   while(tempoDecorrido(Tempo_Inicial, Tempo_Final)*1000 < tempo)
      Tempo_Final = leTimer();    //Atualiza a variavel global de tempo fim
}
```

```
/*************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO   ***
 ***                                                                     ***
 ***  Arquivo    :  MOTORES.H                                           ***
 ***  Conteudo   :  Procedimentos de operacao/controle motores de passo ***
 ***  Descricao  :  Este arquivo contem as definicoes de cada motor e   ***
 ***                os procedimentos para o controle dos mesmos. Inclui ***
 ***                procedimentos para comunicacao com a porta paralela ***
 ***                e envio de sinal aos motores, assim como para a     ***
 ***                execucao de ciclos de operacao (para a geracao do   ***
 ***                trem de pulsos).                                    ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330           ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452           ***
 ***  Data       :  13/09/2001                                          ***
 *************************************************************************/

//Constante com o periodo principal (em microsegundos)
#define PERIODO 500     //Vel. maxima : 5 Hz no rotor do motor

//Constante com o endereco da porta paralela do micro (8 bits de saida)
#define PORTA_PARALELA 0x0378

//Constantes que indicam os bits que correspondem a cada saida
#define BITDIRECAO1 5
#define BITDIRECAO2 3
#define BITDIRECAO3 1
#define BITPULSOS1 4
#define BITPULSOS2 2
#define BITPULSOS3 0

//Estrutura de dados com os parametros e variaveis de um motor
struct MOTOR {
   long Limite_Inicial;  //Limites de operacao do motor (em PASSOS)
   long Limite_Final;
   long Posicao;         //Posicao do motor (em PASSOS, a partir do 0)
   long Velocidade;      //Velocidade do motor (em no. de periodos por inversao)
   int Direcao;          //Direcao do motor (bit, 0 ou 1)
   int Movendo;          //Indica se esta se movendo (0 - falso, 1 - verdadeiro)
   int Contagem;         //Contem a contagem de periodos ja passados
   int Sinal;            //Contem o sinal atual sendo enviado (1 ou 0) no trem
};

//Declara globalmente os tres motores, com valores padrao
//Motor 1 : Movimento principal (rotacao da estrutura)
//Motor 2 : Movimento do carro
//Motor 3 : Movimento de zoom
struct MOTOR motor1 = {-9000000,9000000,0,25,0,0,0,0};  //Vel : 0.2 Hz
struct MOTOR motor2 = {-9000000,9000000,0,12,0,0,0,0};  //Vel : 0.4 Hz
struct MOTOR motor3 = {-9000000,9000000,0,10,0,0,0,0};  //Vel : 0.5 Hz

//Procedimento que liga (ativa a movimentacao) de um motor
//Argumentos : *mot   - Aponta para uma estrutura MOTOR
//Retorno    : nenhum
void ligaMotor(struct MOTOR *mot) {
   if(mot->Direcao == 1) {      //Verifica a direcao do motor
      if(mot->Posicao < mot->Limite_Final) {
         mot->Movendo = 1;     //Aqui, o movimento e' permitido
      } else {
```

```
            mot->Movendo = 0;      //Aqui, o motor ja' esta' no limite
         }
      } else {
         if(mot->Posicao > mot->Limite_Inicial) {
            mot->Movendo = 1;      //Aqui, o movimento e' permitido
         } else {
            mot->Movendo = 0;      //Aqui, o motor ja' esta' no limite
         }
      }
}

//Procedimento que executa uma inversao de sinal em um motor
//Argumentos : *mot    - Aponta para uma estrutura MOTOR
//Retorno    : nenhum
void executaMotor(struct MOTOR *mot) {
   mot->Sinal = mot->Sinal + 1;    //Incrementa o sinal
   if(mot->Sinal == 2) {            //Caso seja uma borda de descida
      mot->Sinal = 0;             //O sinal e' 0 ou 1 apenas
      if(mot->Direcao == 1) {    //Verifica a direcao
         mot->Posicao = mot->Posicao + 1;     //Atualiza a posicao
         if(mot->Posicao >= mot->Limite_Final) {  //Se chegou ao limite, deve
            mot->Movendo = 0;                     //parar o motor.
         }
      } else {    //Aqui o motor esta indo para tras
         mot->Posicao = mot->Posicao - 1;     //Atualiza a posicao
         if(mot->Posicao <= mot->Limite_Inicial) { //Se chegou ao limite, deve
            mot->Movendo = 0;                      //parar o motor.
         }
      }
   }
}

//Procedimento que cria e envia o sinal a porta paralela
//Argumentos : nenhum
//Retorno    : nenhum
void enviaSinal(void) {
   int temporario, sinalFinal;

   sinalFinal = 0;                //Inicializa todos os bits com 0
   temporario = motor1.Direcao;             //Recupera a direcao do mot. 1
   temporario = temporario << BITDIRECAO1;  //Coloca no bit designado
   sinalFinal = sinalFinal | temporario;    //Associa ao sinal final
   //Repete o mesmo procedimento para todos os parametros
   temporario = motor2.Direcao;
   temporario = temporario << BITDIRECAO2;    //Direcao mot. 2
   sinalFinal = sinalFinal | temporario;
   temporario = motor3.Direcao;
   temporario = temporario << BITDIRECAO3;    //Direcao mot. 3
   sinalFinal = sinalFinal | temporario;
   temporario = motor1.Sinal;
   temporario = temporario << BITPULSOS1;      //Sinal de pulso mot. 1
   sinalFinal = sinalFinal | temporario;
   temporario = motor2.Sinal;
   temporario = temporario << BITPULSOS2;      //Sinal de pulso mot. 2
   sinalFinal = sinalFinal | temporario;
   temporario = motor3.Sinal;
   temporario = temporario << BITPULSOS3;      //Sinal de pulso mot. 3
   sinalFinal = sinalFinal | temporario;
```

```
   //Envia o sinal final para a porta paralela
   outp(PORTA_PARALELA, sinalFinal);
}

//Procedimento que executa um ciclo nos motores
//Argumentos : nenhum
//Retorno    : nenhum
void executaMotores(void) {
   int executado = 0;    //Indica se algum motor executou inversao de sinal

   if(motor1.Movendo == 1) {        //Verifica se o motor esta se movendo
      motor1.Contagem = motor1.Contagem + 1;   //Atualiza contagem de periodos
      if(motor1.Contagem >= motor1.Velocidade) {
         motor1.Contagem = 0;      //Terminou a contagem, inicia outra
         executaMotor(&motor1);    //Executa a inversao do sinal (pulso)
         executado = 1;            //Informa a execucao de inversao
      }
   }
   //Faz o mesmo para os outros dois motores
   if(motor2.Movendo == 1) {
      motor2.Contagem = motor2.Contagem + 1;
      if(motor2.Contagem >= motor2.Velocidade) {
         motor2.Contagem = 0;
         executaMotor(&motor2);
         executado = 1;
      }
   }
   if(motor3.Movendo == 1) {
      motor3.Contagem = motor3.Contagem + 1;
      if(motor3.Contagem >= motor3.Velocidade) {
         motor3.Contagem = 0;
         executaMotor(&motor3);
         executado = 1;
      }
   }
   if(executado == 1)        //Caso tenha executado um ciclo, constroi e
      enviaSinal();          //envia o sinal para a porta paralela.
}
```

```
/*****************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO   ***
 ***                                                                     ***
 ***  Arquivo    :  CLBRMOUS.C                                          ***
 ***  Conteudo   :  Procedimentos de calibracao do dispositivo de entrada ***
 ***  Descricao  :  Contem o procedimento que efetua a calibracao do     ***
 ***                mouse, definindo os parametros de sensibilidade,     ***
 ***                alem da definicao do retangulo limite para comandos  ***
 ***                durante a operacao normal do robo.                   ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330           ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452           ***
 ***  Data       :  13/09/2001                                          ***
 *****************************************************************************/

//Variaveis com as posicoes limite de comando (retangulo). Define padrao.
int limEsq = 318;
int limDir = 320;
int limSup = 238;
int limInf = 240;

//Procedimento que centraliza o mouse na tela (na posicao desejada)
//Argumentos : nenhum
//Retorno    : nenhum
void centralizaCursor(void) {
   int codigo;

   limpaTela();           //Limpa a tela
   _settextposition(2,28);
   _settextcolor(14);
   _outtext("DETERMINA€AO DO CENTRO");
   _settextcolor(7);
   _settextposition(4,13);
   _outtext("Posicione a cabe‡a na posi‡ao desejada para o centro.");
   _settextcolor(12);
   _settextposition(5,19);
   _outtext("Pressione qualquer pedal para prosseguir.");
   _settextcolor(7);
   codigo = 0;
   //Aguarda que algum botao seja pressionado
   while(dispositivo.botEborda != 1 && dispositivo.botDborda != 1 && codigo != 27)
{
      atualizaMouse();           //Atualiza os dados do dispositivo
      codigo = verificaTecla();  //Checa o teclado (ESC)
   }
   posicionaCursor(319,239);        //Coloca a seta no centro
}

//Procedimento que efetua toda a calibracao do dispositivo de entrada (mouse)
//Argumentos : nenhum
//Retorno    : nenhum
void calibraDispositivo(void) {
   int codigo;
   int antigoLim;
   int antigoSens;

   dispositivo.botEborda = 0;   //Forca o usuario a pressionar um botao
   dispositivo.botDborda = 0;
   ativaCursor(0);             //Garante que o cursor nao esta visivel
```

```
//Centraliza o cursor do dispositivo para iniciar calibracao
centralizaCursor();

//DETERMINACAO DA SENSIBILIDADE HORIZONTAL
//Desenha na tela as instrucoes
limpaTela();
_settextposition(2,19);
_settextcolor(14);
_outtext("DETERMINAÇAO DA SENSIBILIDADE HORIZONTAL");
_settextcolor(7);
_settextposition(4,6);
_outtext("Mova a cabeŧa no sentido horizontal (inclinaŧao lateral da cabeŧa).");
_settextposition(5,10);
_outtext("Idealmente, o cursor deve atingir as extremidades da tela.");
_settextcolor(9);
_settextposition(7,10);
_outtext("PEDAL ESQUERDO - Reduz a Sensibilidade");
_settextposition(8,10);
_outtext("PEDAL DIREITO  - Aumenta a Sensibilidade");
_settextposition(10,19);
_settextcolor(12);
_outtext("Pressione qualquer tecla para continuar.");
_settextcolor(7);
antigoSens = dispositivo.sensY;  //Guarda a sensibilidade em Y, e coloca
dispositivo.sensY = 5000;        //um valor proximo de infinito para Y.
_settextposition(25,45);
printf("Sensibilidade Horizontal : %d   ", dispositivo.sensX);
ativaCursor(1);   //Aciona o cursor do mouse para visualizacao
//Aguarda ate' que alguma tecla seja pressionada
while(verificaTecla() < 1) {
   atualizaMouse();       //Atualiza os dados do dispositivo
   if(dispositivo.botEborda == 1) {
      //Caso tenha pressionado o botao esquerdo, reduz sensibilidade
      dispositivo.sensX--;
      if(dispositivo.sensX < 1)    //O limite minimo e' 1
         dispositivo.sensX = 1;
   }
   if(dispositivo.botDborda == 1) {
      //Caso tenha pressionado o botao direito, aumenta sensibilidade
      dispositivo.sensX++;
      if(dispositivo.sensX > 500)   //O limite maximo e' 500
         dispositivo.sensX = 500;
   }
   if(dispositivo.botMborda == 1) {
      //Caso tenha pressionado o botao central
      ativaCursor(0);
      posicionaCursor(319,239);        //Coloca a seta no centro
      ativaCursor(1);
   }
   //Caso tenha alterado a sensibilidade de alguma forma
   if(dispositivo.botEborda || dispositivo.botDborda) {
      _settextposition(25,45);   //Informa a nova sensibilidade ao usuario
      printf("Sensibilidade Horizontal : %d   ", dispositivo.sensX);
   }
}
dispositivo.sensY = antigoSens;   //Recupera o antigo set para sensib. Y

//DETERMINACAO DA SENSIBILIDADE VERTICAL
```

```
ativaCursor(0);    //E' preciso esconder o cursor antes de alterar a tela
//Desenha na tela as instrucoes
limpaTela();
_settextcolor(14);
_settextposition(2,20);
_outtext("DETERMINA€AO DA SENSIBILIDADE VERTICAL");
_settextcolor(7);
_settextposition(4,7);
_outtext("Mova a cabe‡a no sentido vertical (inclina‡ao frontal da cabe‡a).");
_settextposition(5,10);
_outtext("Idealmente, o cursor deve atingir as extremidades da tela.");
_settextposition(7,10);
_settextcolor(9);
_outtext("PEDAL ESQUERDO - Reduz a Sensibilidade");
_settextposition(8,10);
_outtext("PEDAL DIREITO  - Aumenta a Sensibilidade");
_settextposition(10,19);
_settextcolor(12);
_outtext("Pressione qualquer tecla para continuar.");
_settextcolor(7);
antigoSens = dispositivo.sensX;  //Guarda a sensibilidade em X, e coloca
dispositivo.sensX = 5000;        //um valor proximo de infinito para X.
_settextposition(25,45);     //Informa o valor ao usuario
printf("Sensibilidade Vertical : %d   ", dispositivo.sensY);
ativaCursor(1);       //Ativa o cursor, permitindo a visualizacao
//Aguarda ate' que alguma tecla seja pressionada
while(verificaTecla() < 1) {
   atualizaMouse();        //Atualiza os dados do dispositivo
   if(dispositivo.botEborda == 1) {
      //Caso tenha pressionado o botao esquerdo, reduz a sensibilidade
      dispositivo.sensY--;
      if(dispositivo.sensY < 1)    //O limite minimo e' 1
         dispositivo.sensY = 1;
   }
   if(dispositivo.botDborda == 1) {
      //Caso tenha pressionado o botao direiro, aumenta a sensibilidade
      dispositivo.sensY++;
      if(dispositivo.sensY > 500)   //O limite maximo e' 500
         dispositivo.sensY = 500;
   }
   if(dispositivo.botMborda == 1) {
      //Caso tenha pressionado o botao central
      ativaCursor(0);
      posicionaCursor(319,239);        //Coloca a seta no centro
      ativaCursor(1);
   }
   //Verifica se a sensibilidade foi alterada de alguma forma
   if(dispositivo.botEborda || dispositivo.botDborda) {
      _settextposition(25,45);     //Informa o novo valor ao usuario
      printf("Sensibilidade Vertical : %d   ", dispositivo.sensY);
   }
}
dispositivo.sensX = antigoSens;   //Recupera o antigo set para sensib. X

ativaCursor(0);       //Esconde o cursor antes de alterar a tela
centralizaCursor();   //Centraliza o cursor para continuar a operacao

//DETERMINACAO LIMITE ESQUERDO - RETANGULO LIMITE DE POSICAO PARA COMANDOS
```

```
    //Desenha na tela as instrucoes
    limpaTela();
    _settextcolor(14);
    _settextposition(2,17);
    _outtext("DETERMINA€AO DO RETANGULO LIMITE PARA COMANDOS");
    _settextposition(3,33);
    _settextcolor(9);
    _outtext("LIMITE ESQUERDO");
    _settextposition(5,17);
    _settextcolor(7);
    _outtext("Incline lateralmente a cabe‡a para a esquerda.");
    _settextposition(6,12);
    _settextcolor(12);
    _outtext("Pressione qualquer contato ao atingir a posi‡ao desejada.");
    _settextcolor(7);
    _settextposition(25,45);
    printf("Limite Esquerdo do retangulo : %d    ",limEsq);
    //Desenha o retangulo inicial
    retangulo(limEsq, limSup, limDir, limInf, 10, 0);
    ativaCursor(1);   //Ativa o cursor para permitir visualizacao
    codigo = 0;
    dispositivo.botEborda = 0;      //Forca a zerar o status dos botoes (ou
    dispositivo.botDborda = 0;      //seja, o usuario e' forcado a apertar)
    //Aguarda ate' que o usuario pressione um botao
    while(dispositivo.botEborda != 1 && dispositivo.botDborda != 1 && codigo != 27)
{
       if(dispositivo.botMborda == 1) {
          //Caso tenha pressionado o botao central
          ativaCursor(0);
          posicionaCursor(319,239);         //Coloca a seta no centro
          ativaCursor(1);
       }
       atualizaMouse();         //Atualiza os dados do dispositivo
       codigo = verificaTecla();   //Verificacao de tecla (ESC)
       antigoLim = limEsq;         //Mantem em variavel o limite antigo
       if(dispositivo.posX < 318) {   //O limite esquerdo minimo e' 318.
          limEsq = dispositivo.posX;  //Atualiza o limite
       } else {
          limEsq = 318;         //Aqui, mantem o minimo
       }
       if(antigoLim != limEsq) {      //Caso tenha alterado o limite
          ativaCursor(0);       //Esconde o cursor antes de alterar a tela
          //Apaga o retangulo antigo
          retangulo(antigoLim, limSup, limDir, limInf, 0, 0);
          _settextposition(25,45);
          //Informa o limite ao usuario
          printf("Limite Esquerdo do retangulo : %d    ",limEsq);
          //Desenha o novo retangulo
          retangulo(limEsq, limSup, limDir, limInf, 10, 0);
          ativaCursor(1);   //Reativa o cursor para visualizacao
       }
    }

    //DETERMINACAO LIMITE DIREITO - RETANGULO LIMITE DE POSICAO PARA COMANDOS
    ativaCursor(0);    //Esconde cursor antes de alterar a tela
    //Desenha na tela as instrucoes
    limpaTela();
    _settextposition(2,17);
```

```
    _settextcolor(14);
    _outtext("DETERMINA€AO DO RETANGULO LIMITE PARA COMANDOS");
    _settextposition(3,33);
    _settextcolor(9);
    _outtext("LIMITE DIREITO");
    _settextposition(5,17);
    _settextcolor(7);
    _outtext("Incline lateralmente a cabe†a para a direita.");
    _settextposition(6,12);
    _settextcolor(12);
    _outtext("Pressione qualquer contato ao atingir a posi†ao desejada.");
    _settextposition(25,45);
    _settextcolor(7);
    printf("Limite Direito do retangulo : %d    ",limDir);
    retangulo(limEsq, limSup, limDir, limInf, 10, 0);  //Desenha retangulo
    ativaCursor(1);   //Ativa o cursor para visualizacao
    codigo = 0;
    dispositivo.botEborda = 0;      //Forca a zerar o status dos botoes (ou
    dispositivo.botDborda = 0;      //seja, o usuario e forcado a apertar)
    //Aguarda ate' que algum botao seja pressionado
    while(dispositivo.botEborda != 1 && dispositivo.botDborda != 1 && codigo != 27)
{
       if(dispositivo.botMborda == 1) {
          //Caso tenha pressionado o botao central
          ativaCursor(0);
          posicionaCursor(319,239);         //Coloca a seta no centro
          ativaCursor(1);
       }
       atualizaMouse();              //Atualiza os dados do dispositivo
       codigo = verificaTecla();     //Checa teclado (ESC)
       antigoLim = limDir;           //Guarda em variavel o limite antigo
       if(dispositivo.posX > 320) {     //O limite direito minimo e' 320
          limDir = dispositivo.posX;    //Atualiza o limite direito
       } else {
          limDir = 320;      //Neste caso, mantem o minimo
       }
       if(antigoLim != limDir) {  //Caso tenha alterado o limite
          ativaCursor(0);     //Esconde o cursor antes de alterar a tela
          //Apaga o retangulo anterior
          retangulo(limEsq, limSup, antigoLim, limInf, 0, 0);
          _settextposition(25,45);
          //Informa ao usuario o novo limite
          printf("Limite Direito do retangulo : %d    ",limDir);
          //Desenha o novo retangulo
          retangulo(limEsq, limSup, limDir, limInf, 10, 0);
          ativaCursor(1);   //Reativa o cursor para permitir visualizacao
       }
    }

    //DETERMINACAO LIMITE SUPERIOR - RETANGULO LIMITE DE POSICAO PARA COMANDOS
    ativaCursor(0);   //Esconde cursor antes de alterar a tela
    //Desenha na tela as instrucoes
    limpaTela();
    _settextposition(2,17);
    _settextcolor(14);
    _outtext("DETERMINA€AO DO RETANGULO LIMITE PARA COMANDOS");
    _settextposition(3,33);
    _settextcolor(9);
```

```
    _outtext("LIMITE SUPERIOR");
    _settextposition(5,19);
    _settextcolor(7);
    _outtext("Incline frontalmente a cabe†a para cima.");
    _settextcolor(12);
    _settextposition(6,12);
    _outtext("Pressione qualquer contato ao atingir a posi†ao desejada.");
    _settextcolor(7);
    _settextposition(25,45);
    printf("Limite Superior do retangulo : %d    ",limSup);
    retangulo(limEsq, limSup, limDir, limInf, 10, 0);  //Desenha o retangulo
    ativaCursor(1);   //Ativa o cursor para permitir a visualizacao
    codigo = 0;
    dispositivo.botEborda = 0;      //Forca a zerar o status dos botoes (ou
    dispositivo.botDborda = 0;      //seja, o usuario e forcado a apertar)
    //Aguarda ate' que o usuario pressione algum botao
    while(dispositivo.botEborda != 1 && dispositivo.botDborda != 1 && codigo != 27)
{
       if(dispositivo.botMborda == 1) {
          //Caso tenha pressionado o botao central
          ativaCursor(0);
          posicionaCursor(319,239);         //Coloca a seta no centro
          ativaCursor(1);
       }
       atualizaMouse();            //Atualiza os dados do dispositivo
       codigo = verificaTecla();   //Checa o teclado (ESC)
       antigoLim = limSup;         //Guarda o antigo limite em variavel
       if(dispositivo.posY < 238) {     //O limite maximo superior e' 238
          limSup = dispositivo.posY;    //Atualiza o limite
       } else {
          limSup = 238;     //Neste caso, mantem o maximo
       }
       if(antigoLim != limSup) {    //Caso tenha alterado o limite
          ativaCursor(0);    //Esconde o cursor antes de alterar a tela
          //Apaga o retangulo antigo
          retangulo(limEsq, antigoLim, limDir, limInf, 0, 0);
          //Aqui, e' possivel que se tenha o retangulo sobre as instrucoes,
          //portanto e' preciso redesenha-las.
          _settextposition(2,17);
          _settextcolor(14);
          _outtext("DETERMINA€AO DO RETANGULO LIMITE PARA COMANDOS");
          _settextposition(3,33);
          _settextcolor(9);
          _outtext("LIMITE SUPERIOR");
          _settextposition(5,19);
          _settextcolor(7);
          _outtext("Incline frontalmente a cabe†a para cima.");
          _settextcolor(12);
          _settextposition(6,12);
          _outtext("Pressione qualquer contato ao atingir a posi†ao desejada.");
          _settextcolor(7);
          _settextposition(25,45);
          //Informa o novo limite ao usuario
          printf("Limite Superior do retangulo : %d    ",limSup);
          //Desenha o novo retangulo
          retangulo(limEsq, limSup, limDir, limInf, 10, 0);
          ativaCursor(1);  //Reativa o cursor, permitindo sua visualizacao
       }
```

```
    }

    //DETERMINACAO LIMITE INFERIOR - RETANGULO LIMITE DE POSICAO PARA COMANDOS
    ativaCursor(0);   //Esconde o cursor antes de alterar a tela
    //Desenha na tela as instrucoes
    limpaTela();
    _settextposition(2,17);
    _settextcolor(14);
    _outtext("DETERMINA€AO DO RETANGULO LIMITE PARA COMANDOS");
    _settextposition(3,33);
    _settextcolor(9);
    _outtext("LIMITE INFERIOR");
    _settextposition(5,19);
    _settextcolor(7);
    _outtext("Incline frontalmente a cabe‡a para baixo.");
    _settextposition(6,12);
    _settextcolor(12);
    _outtext("Pressione qualquer contato ao atingir a posi‡ao desejada.");
    _settextcolor(7);
    _settextposition(25,45);
    printf("Limite Inferior do retangulo : %d    ",limInf);
    retangulo(limEsq, limSup, limDir, limInf, 10, 0);   //Desenha o retangulo
    ativaCursor(1);   //Ativa o cursor, permitindo a visualizacao
    codigo = 0;
    dispositivo.botEborda = 0;     //Forca a zerar o status dos botoes (ou
    dispositivo.botDborda = 0;     //seja, o usuario e forcado a apertar)
    //Executa ate' que o usuario pressione algum botao
    while(dispositivo.botEborda != 1 && dispositivo.botDborda != 1 && codigo != 27)
{
      if(dispositivo.botMborda == 1) {
         //Caso tenha pressionado o botao central
         ativaCursor(0);
         posicionaCursor(319,239);        //Coloca a seta no centro
         ativaCursor(1);
      }
      atualizaMouse();             //Atualiza os dados do dispositivo
      codigo = verificaTecla();    //Checa o teclado (ESC)
      antigoLim = limInf;          //Guarda o antigo limite em variavel
      if(dispositivo.posY > 240) {   //O limite minimo inferior e' 240
         limInf = dispositivo.posY;  //Atualiza o limite inferior
      } else {
         limInf = 240;     //Neste caso, mantem o minimo
      }
      if(antigoLim != limInf) {     //Caso tenha alterado o limite inferior
         ativaCursor(0);     //Esconde o cursor antes de alterar a tela
         //Apaga o retangulo antigo
         retangulo(limEsq, limSup, limDir, antigoLim, 0, 0);
         _settextposition(25,45);
         //Informa ao usuario o novo limite
         printf("Limite Inferior do retangulo : %d    ",limInf);
         //Desenha o novo retangulo
         retangulo(limEsq, limSup, limDir, limInf, 10, 0);
         ativaCursor(1);  //Reativa o cursor para permitir sua visualizacao
      }
   }
   ativaCursor(0);   //Ao final da calibracao, desativa o cursor
}
```

```
/*************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO   ***
 ***                                                                     ***
 ***  Arquivo    :  CNFGMOT.C                                           ***
 ***  Conteudo   :  Procedimentos de configuracao dos parametros do robo ***
 ***  Descricao  :  Contem os procedimentos para a configuracao dos      ***
 ***                parametros do robo, que sao posicoes limite durante  ***
 ***                a operacao (por contagem de passos), velocidades de  ***
 ***                movimentacao e posicionamento do robo para o inicio  ***
 ***                da operacao normal (posicao 0).                      ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330            ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452            ***
 ***  Data       :  13/09/2001                                          ***
 *************************************************************************/

//Variavel que contem o numero de passos por volta dos motores utilizados
#define PASSOS_POR_VOLTA 200

//Procedimento que define o limite superior ou inferior de operacao do motor
//Argumentos : *mot     - aponta para uma estrutura MOTOR
//             superior - indica se esta' definindo limite sup. ou inferior
//             numero   - uma variavel que diz o numero do motor
//Retorno    : nenhum
void defineLimite(struct MOTOR *mot, int superior, int numero) {
   int codigo;
   char tecla;

   limpaTela();                        //Limpa a tela
   retangulo(109,79,549,419,8,1);      //Desenha a tela
   retangulo(99,69,539,409,3,1);
   retangulo(99,69,539,409,15,0);
   retangulo(155,99,479,176,0,1);
   retangulo(155,99,479,176,15,0);
   _settextcolor(12);
   _settextposition(8,27);
   _outtext("POSICOES LIMITE DOS MOTORES");
   _settextcolor(15);
   _settextposition(10,28);
   if(superior == 1) {                    //Verifica de qual limite se trata
      _outtext("LIMITE SUPERIOR , MOTOR ");
   } else {
      _outtext("LIMITE INFERIOR , MOTOR ");
   }
   _settextposition(10,52);
   if(numero == 1)             //Verifica e poe na tela o numero passado,
      _outtext("1  ");         //indicando qual motor se esta' tratando
   if(numero == 2)
      _outtext("2  ");
   if(numero == 3)
      _outtext("3  ");
   while(codigo != 27 && codigo != 13) {  //Executa ate pressionar ENTER
      //Desenha a `janela` com os comandos
      retangulo(125,210,515,380,0,1);
      retangulo(125,210,515,380,15,0);
      _settextcolor(9);
      _settextposition(15,18);
      _outtext("Selecione a Opera‡ao :");          //Indica as operacoes
      if(mot->Posicao >= mot->Limite_Final) {
```

```
         _settextcolor(7);    //No caso de estar no limite, muda cor para
      } else {                //indicar a nao possibilidade do comando.
         _settextcolor(14);
      }
      _settextposition(18,18);
      _outtext("       [ + ]      Move para a frente");
      if(mot->Posicao <= mot->Limite_Inicial) {
         _settextcolor(7);       //Novamente, caso no limite, muda a cor
      } else {                   //indicando a proibicao do comando.
         _settextcolor(14);
      }
      _settextposition(20,18);
      _outtext("       [ - ]      Move para tras");
      if((mot->Posicao >= mot->Limite_Final && superior == 0) || (mot->Posicao <=
mot->Limite_Inicial && superior == 1)) {
         _settextcolor(7);     //Caso esteja definindo limite superior, e
      } else {                 //o motor esteja no limite inferior, ou o
         _settextcolor(14);    //contrario, nao se pode aceitar este limite.
      }
      _settextposition(22,18);
      _outtext("      [ENTER]     Aceita a posi‡ao atual");
      codigo = 0;
      //Aguarda teclar uma operacao valida
      while(codigo != 43 && codigo != 45 && codigo != 13 && codigo != 27) {
         tecla = getch();
         codigo = (int)tecla;
      }
      if(codigo == 43 || codigo == 45) {  //Caso tenha selecionado para mover
         if(codigo == 43) {      //Altera a direcao do motor de acordo
            mot->Direcao = 1;
         } else {
            mot->Direcao = 0;
         }
         //Verifica se e' valida a movimentacao selecionada
         if(! (mot->Direcao == 0 && mot->Posicao <= mot->Limite_Inicial)
               && ! (mot->Direcao == 1 && mot->Posicao >= mot->Limite_Final)) {
            //Redesenha a `janela` de comandos, indicando o movimento
            retangulo(125,210,515,380,0,1);
            retangulo(125,210,515,380,15,0);
            _settextcolor(9);
            _settextposition(15,18);
            _outtext("Selecione a Opera‡ao :");
            _settextcolor(14);
            _settextposition(20,18);       //Indica o comando de parada
            _outtext(" [QUALQUER TECLA]  PARA TERMINAR MOVIMENTA€AO");
            mot->Movendo = 1;          //Coloca o motor para se movimentar
            //Executa ate' que o usuario tecle uma tecla, ou o motor pare
            //naturalmente por ter atingido seu limite de operacao.
            while(! kbhit() && mot->Movendo == 1) {
               iniciaContagem();           //Inicia uma contagem de tempo
               executaMotores();           //Executa movimentacao
               finalizaContagem(PERIODO);  //Finaliza contagem de um periodo
            }
            mot->Movendo = 0;      //Indica ao motor a parada
            while(kbhit())         //Limpa o buffer do teclado
               tecla = getch();
         }
      }
```

```
        //Verifica se o limite selecionado e' valido
        if(codigo == 13 && ((mot->Posicao >= mot->Limite_Final && superior == 0) ||
   (mot->Posicao <= mot->Limite_Inicial && superior == 1))) {
           //Caso nao seja valido, cancela o comando e avisa o usuario
           codigo = 0;
           retangulo(115,188,535,235,8,1);
           retangulo(110,183,530,230,0,1);
           retangulo(110,183,530,230,15,0);
           _settextposition(13,16);
           _settextcolor(15);
           _outtext("O LIMITE PASSADO E'INVALIDO,MOTOR SEM MOVIMENTACAO");
           _settextposition(14,28);
           _settextcolor(7);
           _outtext("<Pressione qualquer tecla>");
           tecla = getch();   //Suspende o programa ate' pressionar tecla
           //E' preciso redesenhar a parte fixa da tela
           limpaTela();                        //Limpa a tela
           retangulo(109,79,549,419,8,1);    //Desenha a tela
           retangulo(99,69,539,409,3,1);
           retangulo(99,69,539,409,15,0);
           retangulo(155,99,479,176,0,1);
           retangulo(155,99,479,176,15,0);
           _settextcolor(12);
           _settextposition(8,27);
           _outtext("POSICOES LIMITE DOS MOTORES");
           _settextcolor(15);
           _settextposition(10,28);
           if(superior == 1) {            //Verifica de qual limite se trata
              _outtext("LIMITE SUPERIOR , MOTOR ");
           } else {
              _outtext("LIMITE INFERIOR , MOTOR ");
           }
           _settextposition(10,52);
           if(numero == 1)             //Verifica e poe na tela o numero passado,
              _outtext("1  ");         //indicando qual motor se esta' tratando
           if(numero == 2)
              _outtext("2  ");
           if(numero == 3)
              _outtext("3  ");
        }
     }
     //Verifica se trata-se do limite superior ou inferior
     if(superior == 1) {
        mot->Limite_Final = mot->Posicao;     //Atualiza a variavel de limite
     } else {
        mot->Limite_Inicial = mot->Posicao;  //Atualiza a variavel de limite
     }
}

//Procedimento que define a velocidade de um motor
//Argumentos : *mot      - aponta para uma estrutura MOTOR
//             numero    - uma variavel que diz o numero do motor
//Retorno    : nenhum
void defineVelocidade(struct MOTOR *mot, int numero) {
   int codigo;
   char tecla;
   double velocidade;
```

```
 limpaTela();                        //Limpa a tela
 retangulo(109,79,549,419,8,1);      //Desenha a tela
 retangulo(99,69,539,409,3,1);
 retangulo(99,69,539,409,15,0);
 retangulo(155,99,479,176,0,1);
 retangulo(155,99,479,176,15,0);
 _settextcolor(12);
 _settextposition(8,27);
 _outtext("VELOCIDADES DOS MOTORES");
 _settextcolor(15);
 _settextposition(10,30);
 _outtext("VELOCIDADE DO MOTOR ");
 _settextposition(10,50);
 if(numero == 1)              //Verifica e poe na tela o numero passado,
    _outtext("1  ");          //indicando o motor de que se trata.
 if(numero == 2)
    _outtext("2  ");
 if(numero == 3)
    _outtext("3  ");
 codigo = 0;
 retangulo(125,210,515,380,0,1);
 retangulo(125,210,515,380,15,0);
 _settextcolor(9);
 _settextposition(15,18);
 _outtext("Selecione a Opera‡ao :");          //Indica as operacoes
 _settextcolor(14);
 _settextposition(18,18);
 _outtext("        [ + ]       Aumenta a Velocidade");
 _settextposition(20,18);
 _outtext("        [ - ]       Reduz a Velocidade");
 _settextposition(22,18);
 _outtext("      [ENTER]       Aceita a velocidade atual");
 codigo = 0;
 //Inicia movendo para tras, pois a determinacao do limite superior ocorreu
 //por ultimo, sendo esta a posicao atual do motor
 mot->Direcao = 0;
 mot->Movendo = 1;       //Coloca o motor para movimentar-se
 while(codigo != 27 && codigo != 13) {  //Executa ate pressionar ENTER
    iniciaContagem();                  //Inicia a contagem de um periodo
    codigo = verificaTecla();          //Verifica se teclou algo
    if(codigo == 45)                   //Checa se houve mudanca de velocidade
       mot->Velocidade = mot->Velocidade + 1;   //Atualiza a velocidade
    if(codigo == 43 && mot->Velocidade > 1)    //A velocidade maxima e' 1
       mot->Velocidade = mot->Velocidade - 1;   //Atualiza a velocidade
    //Verifica se chegou `a posicao limite
    if(mot->Direcao == 0 && mot->Posicao <= mot->Limite_Inicial) {
       mot->Direcao = 1;     //Inverte a dire‡Æo
       mot->Movendo = 1;     //O algoritmo para o motor automaticamente,
    }                        //e' preciso reativa-lo.
    if(mot->Direcao == 1 && mot->Posicao >= mot->Limite_Final) {
       mot->Direcao = 0;     //Da mesma forma, inverte a direcao e
       mot->Movendo = 1;     //reativa a movimentacao do motor.
    }
    executaMotores();            //Executa um ciclo nos motores
    finalizaContagem(PERIODO);   //Finaliza a contagem de um periodo
 }
 mot->Movendo = 0;            //Para o motor
 //Mostra a velocidade selecionada ao usuario
```

```
   velocidade = (1/(2*(double)PASSOS_POR_VOLTA))/((double)mot-
>Velocidade*((double)PERIODO/1000000));
   retangulo(115,188,535,235,8,1);
   retangulo(110,183,530,230,0,1);
   retangulo(110,183,530,230,15,0);
   _settextposition(13,19);
   _settextcolor(15);
   _outtext("A VELOCIDADE SELECIONADA FOI DE ");
   printf("%f Hz",velocidade);
   _settextposition(14,28);
   _settextcolor(7);
   _outtext("<Pressione qualquer tecla>");
   tecla = getch();   //Interrompe a execucao ate' ser pressionada uma tecla
}

//Procedimento de colocar o robo na posicao de insercao
//Argumentos : posZero   - Indica se deve atribuir a posicao 0 (1) ou nao (0)
//Retorno    : nenhum
void posicionaRobo(int posZero) {
   int motorAtual;
   int codigo;
   char tecla;

   motorAtual = 1;      //Inicia com motor 1 selecionado
   limpaTela();                        //Limpa a tela
   retangulo(109,79,549,419,8,1);     //Desenha a tela
   retangulo(99,69,539,409,3,1);
   retangulo(99,69,539,409,15,0);
   retangulo(155,99,479,176,0,1);
   retangulo(155,99,479,176,15,0);
   _settextcolor(12);
   _settextposition(8,29);
   _outtext("POSICIONAMENTO DO ROBO");
   _settextcolor(15);
   _settextposition(10,33);
   _outtext("MOTOR ATUAL : ");
   codigo = 0;
   while(codigo != 27 && codigo != 13) {    //Aguarda ate' pressionar ENTER
      //Desenha a parte variavel da tela (janela de comandos e numero motor)
      retangulo(125,210,515,380,0,1);
      retangulo(125,210,515,380,15,0);
      _settextposition(10,47);
      printf("%d  ",motorAtual);   //Informa de qual motor se trata
      _settextcolor(9);
      _settextposition(15,18);
      _outtext("Selecione a Opera‡ao :");
      _settextcolor(14);
      _settextposition(17,18);
      _outtext(" [ 1 ] [ 2 ] [ 3 ]   Seleciona motor");
      //A cor depende se esta na posicao limite
      if(motorAtual == 1) {
         if(motor1.Posicao >= motor1.Limite_Final) {
            _settextcolor(7);     //Motor na posicao limite, nao permite
         } else {                 //este comando.
            _settextcolor(14);
         }
      }
      if(motorAtual == 2) {
```

```
      if(motor2.Posicao >= motor2.Limite_Final) {
         _settextcolor(7);      //Motor na posicao limite, nao permite
      } else {                  //este comando.
         _settextcolor(14);
      }
   }
   if(motorAtual == 3) {
      if(motor3.Posicao >= motor3.Limite_Final) {
         _settextcolor(7);      //Motor na posicao limite, nao permite
      } else {                  //este comando.
         _settextcolor(14);
      }
   }
   _settextposition(19,18);
   _outtext("       [ + ]          Move para a frente");
   //Verifica se o motor se encontra na sua posicao limite
   if(motorAtual == 1) {
      if(motor1.Posicao <= motor1.Limite_Inicial) {
         _settextcolor(7);      //Motor na posicao limite, nao permite
      } else {                  //este comando.
         _settextcolor(14);
      }
   }
   if(motorAtual == 2) {
      if(motor2.Posicao <= motor2.Limite_Inicial) {
         _settextcolor(7);
      } else {
         _settextcolor(14);
      }
   }
   if(motorAtual == 3) {
      if(motor3.Posicao <= motor3.Limite_Inicial) {
         _settextcolor(7);
      } else {
         _settextcolor(14);
      }
   }
   _settextposition(21,18);
   _outtext("       [ - ]          Move para tras");
   _settextcolor(14);
   _settextposition(23,18);
   _outtext("      [ENTER]         Aceita posi‡oes atuais");
   //Aguarda ate' que uma tecla valida seja digitada
   codigo = 0;
   while(codigo != 43 && codigo != 45 && codigo != 13 && codigo != 27 && codigo
!= 49 && codigo != 50 && codigo != 51) {
      tecla = getch();        //Recupera uma tecla do teclado
      codigo = (int)tecla;
   }
   //Verifica se o motor atual esta na posicao limite, caso sim, cancela
   //a execucao do comando de movimentacao.
   if(codigo == 43 && motorAtual == 1 && motor1.Posicao >= motor1.Limite_Final)
      codigo = 0;
   if(codigo == 43 && motorAtual == 2 && motor2.Posicao >= motor2.Limite_Final)
      codigo = 0;
   if(codigo == 43 && motorAtual == 3 && motor3.Posicao >= motor3.Limite_Final)
      codigo = 0;
```

```
      if(codigo == 45 && motorAtual == 1 && motor1.Posicao <=
motor1.Limite_Inicial)
         codigo = 0;
      if(codigo == 45 && motorAtual == 2 && motor2.Posicao <=
motor2.Limite_Inicial)
         codigo = 0;
      if(codigo == 45 && motorAtual == 3 && motor3.Posicao <=
motor3.Limite_Inicial)
         codigo = 0;
      //Caso o comando seja de movimentacao
      if(codigo == 43 || codigo == 45) {
         //Redesenha a parte variavel da tela
         retangulo(125,210,515,380,0,1);
         retangulo(125,210,515,380,15,0);
         _settextposition(10,47);
         printf("%d  ",motorAtual);       //Indica o motor se movendo
         _settextcolor(9);
         _settextposition(15,18);
         _outtext("Selecione a Opera‡ao :");
         _settextcolor(14);
         _settextposition(20,18);
         //Indica qual o comando de parada do motor
         _outtext(" [QUALQUER TECLA]  PARA TERMINAR MOVIMENTA€AO");
         if(motorAtual == 1) {        //Verifica qual o motor
            if(codigo == 43) {          //Atribui a direcao
               motor1.Direcao = 1;
            } else {
               motor1.Direcao = 0;
            }
            motor1.Movendo = 1;      //Manda o motor se mover
            //Aguarda ate que alguma tecla seja pressionada, ou atinja limite
            while(! kbhit() && motor1.Movendo == 1) {
               iniciaContagem();           //Inicia uma contagem de tempo
               executaMotores();           //Executa um ciclo nos motores
               finalizaContagem(PERIODO);  //Termina a contagem de um periodo
            }
            motor1.Movendo = 0;      //Manda o motor parar
            while(kbhit())           //Limpa o buffer do teclado
               getch();
         }
         if(motorAtual == 2) {        //Verifica qual o motor
            if(codigo == 43) {          //Atribui a direcao
               motor2.Direcao = 1;
            } else {
               motor2.Direcao = 0;
            }
            motor2.Movendo = 1;      //Manda o motor se mover
            //Aguarda ate que alguma tecla seja pressionada, ou atinja limite
            while(! kbhit() && motor2.Movendo == 1) {
               iniciaContagem();           //Inicia uma contagem de tempo
               executaMotores();           //Executa um ciclo nos motores
               finalizaContagem(PERIODO);  //Termina a contagem de um periodo
            }
            motor2.Movendo = 0;      //Manda o motor parar
            while(kbhit())           //Limpa o buffer do teclado
               getch();
         }
         if(motorAtual == 3) {        //Verifica qual o motor
```

```
            if(codigo == 43) {          //Atribui a direcao
               motor3.Direcao = 1;
            } else {
               motor3.Direcao = 0;
            }
            motor3.Movendo = 1;      //Manda o motor se mover
            //Aguarda ate que alguma tecla seja pressionada, ou atinja limite
            while(! kbhit() && motor3.Movendo == 1) {
               iniciaContagem();           //Inicia uma contagem de tempo
               executaMotores();           //Executa um ciclo nos motores
               finalizaContagem(PERIODO);  //Termina a contagem de um periodo
            }
            motor3.Movendo = 0;      //Manda o motor parar
            while(kbhit())           //Limpa o buffer do teclado
               getch();
         }
      }
      //Caso tenha selecionado outro motor, atribui nova variavel
      if(codigo == 49 || codigo == 50 || codigo == 51)
         motorAtual = codigo - 48;
   }
   if(posZero == 1) {
      //Atualiza os parametros de posicao dos motores ao final do posicionamento
      //para fazer com que a posicao final seja a posicao 0.
      motor1.Limite_Inicial = motor1.Limite_Inicial - motor1.Posicao;
      motor1.Limite_Final = motor1.Limite_Final - motor1.Posicao;
      motor1.Posicao = 0;
      motor2.Limite_Inicial = motor2.Limite_Inicial - motor2.Posicao;
      motor2.Limite_Final = motor2.Limite_Final - motor2.Posicao;
      motor2.Posicao = 0;
      motor3.Limite_Inicial = motor3.Limite_Inicial - motor3.Posicao;
      motor3.Limite_Final = motor3.Limite_Final - motor3.Posicao;
      motor3.Posicao = 0;
   }
}

//Procedimento que efetua a configuracao completa dos motores
//Argumentos : nenhum
//Retorno    : nenhum
void configuraMotores(void) {
   //Coloca os limites inicialmente em valores proximos do infinito
   motor1.Limite_Inicial = -9000000;
   motor1.Limite_Final = 9000000;
   motor2.Limite_Inicial = -9000000;
   motor2.Limite_Final = 9000000;
   motor3.Limite_Inicial = -9000000;
   motor3.Limite_Final = 9000000;
   //Define os limites dos tres motores
   defineLimite(&motor1, 0, 1);
   defineLimite(&motor1, 1, 1);
   defineLimite(&motor2, 0, 2);
   defineLimite(&motor2, 1, 2);
   defineLimite(&motor3, 0, 3);
   defineLimite(&motor3, 1, 3);
   //Define as velocidades dos tres motores
   defineVelocidade(&motor1, 1);
   defineVelocidade(&motor2, 2);
   defineVelocidade(&motor3, 3);
```

```
    posicionaRobo(1);     //Posiciona o robo na posicao 0 desejada
}
```

```
/*****************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO    ***
 ***                                                                      ***
 ***  Arquivo    :  OPERACAO.C                                           ***
 ***  Conteudo   :  Procedimentos da operacao normal do robo             ***
 ***  Descricao  :  Contem o procedimento que efetua a operacao do robo  ***
 ***                durante a cirurgia. Inicia-se centralizando o cursor  ***
 ***                e procede-se `a cirurgia, com input pelo mouse e      ***
 ***                output atraves de setas e figuras na tela.            ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330            ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452            ***
 ***  Data       :  13/09/2001                                           ***
 *****************************************************************************/

//Definicoes dos motores e direcoes correspondentes a cada movimento
#define MOTED 1    //Motor que efetua movimento esquerda/direita
#define MOTSI 2    //Motor que efetua movimento superior/inferior
#define MOTZM 3    //Motor que efetua movimento zoom in / zoom out
#define DIRESQ 0   //Direcao para mover a esquerda
#define DIRDIR 1   //Direcao para mover a direita
#define DIRSUP 1   //Direcao para mover para cima
#define DIRINF 0   //Direcao para mover para baixo
#define DIRZIN 0   //Direcao para dar zoom in
#define DIRZOU 1   //Direcao para dar zoom out

int modo;   //Variavel global que contem o modo de exibicao

//Procedimento que altera o modo de exibicao
//Argumentos : nenhum
//Retorno    : nenhum
void alteraModo(void) {
   modo++;            //Atualiza o modo, que alterna entre 0 e 3
   if(modo > 3)
      modo = 0;
   if(modo == 0) {       //Modo 0, cursor E retangulo ativos
      ativaCursor(0);
      retangulo(limEsq, limSup, limDir, limInf, 10, 0);
      ativaCursor(1);
   }
   if(modo == 1)         //Modo 1, cursor inativo, retangulo ativo
      ativaCursor(0);
   if(modo == 2)         //Modo 2, cursor E retangulo inativos
      retangulo(limEsq, limSup, limDir, limInf, 0, 0);
   if(modo == 3)         //Modo 3, cursor ativo, retangulo inativo
      ativaCursor(1);
}

//Procedimento de execucao da operacao normal do robo
//Argumentos : nenhum
//Retorno    : nenhum
void efetuaOperacao(void) {
   int codigo;
   int antigoX;   //Variaveis para as antigas posicoes do mouse. Nao se pode
   int antigoY;   //utilizar as proprias do dispositivo pois so' sao
                  //atualizadas para alterar o cursor, quando altera posicao.
   dispositivo.botEborda = 0;  //Forca o usuario a pressionar um botao
   dispositivo.botDborda = 0;
   modo = 0;                   //Inicia com modo 0
```

```
    ativaCursor(0);              //Garante que o cursor nao esta visivel
    //Mostra uma tela de instrucoes antes de iniciar a operacao
    limpaTela();
    retangulo(109,49,549,449,8,1);
    retangulo(99,39,539,439,3,1);
    retangulo(99,39,539,439,15,0);
    retangulo(125,69,515,410,0,1);
    retangulo(125,69,515,410,15,0);
    _settextcolor(12);
    _settextposition(6,29);
    _outtext("INSTRU€OES DE OPERA€AO");
    _settextcolor(9);
    _settextposition(8,18);
    _outtext("1) Centraliza†ao do Cursor");
    _settextcolor(7);
    _settextposition(9,21);
    _outtext("Posicione-se da forma mais conveniente para");
    _settextposition(10,18);
    _outtext("a cirurgia e pressione qualquer pedal.");
    _settextcolor(9);
    _settextposition(12,18);
    _outtext("2) Opera†ao");
    _settextcolor(7);
    _settextposition(13,21);
    _outtext("Durante a opera†ao, incline a cabe†a para");
    _settextposition(14,18);
    _outtext("mover o cursor.");
    _settextposition(15,21);
    _outtext("Ao sair do retangulo limite de comandos,");
    _settextposition(16,18);
    _outtext("graficos indicarao a opera†ao desejada");
    _settextposition(17,18);
    _outtext("(amarelo, pedal esquerdo, branco, pedal");
    _settextposition(18,18);
    _outtext("direito). Pressione o pedal para iniciar");
    _settextposition(19,18);
    _outtext("a opera†ao. Solte o pedal para termina-la.");
    _settextposition(20,21);
    _outtext("Dentro do retangulo, o pedal esquerdo");
    _settextposition(21,18);
    _outtext("for†a a centraliza†ao do cursor, e o direito");
    _settextposition(22,18);
    _outtext("alterna o modo de visualiza†ao.");
    _settextposition(23,21);
    _outtext("Qualquer tecla encerra a opera†ao.");
    _settextcolor(15);
    _settextposition(25,20);
    _outtext("<Pressione qualquer contato para iniciar>");
    codigo = 0;
    //Aguarda que algum botao seja pressionado
    while(dispositivo.botEborda != 1 && dispositivo.botDborda != 1 && codigo != 27)
{
       atualizaMouse();          //Atualiza os dados do dispositivo
       codigo = verificaTecla(); //Checa o teclado (ESC)
    }
    dispositivo.botEborda = 0;  //Forca o usuario a pressionar um botao
    dispositivo.botDborda = 0;
    centralizaCursor();   //Procedimento para usuario centralizar o cursor
```

```
antigoX = 319;   //Inicializa as variaveis das posicoes antigas
antigoY = 239;
limpaTela();
//Desenha o retangulo limite e ativa o cursor
retangulo(limEsq, limSup, limDir, limInf, 10, 0);
ativaCursor(1);
//Executa ate que alguma tecla seja pressionada
while(verificaTecla() < 1) {
   atualizaMouse();    //Atualiza os dados do dispositivo  (mouse)
   //Verifica se as posicoes relativas do cursor mudaram, e caso sim,
   //altera a exibicao ou nao dos elementos indicativos.
   if(dispositivo.posX < limEsq && antigoX >= limEsq) {
      //Liga a seta para a esquerda
      if(modo == 0 || modo == 3)
         ativaCursor(0);
      setaEsquerda(5,230,14);
      if(modo == 0 || modo == 3)
         ativaCursor(1);
   }
   if(dispositivo.posX >= limEsq && antigoX < limEsq) {
      //Desliga a seta para a esquerda
      if(modo == 0 || modo == 3)
         ativaCursor(0);
      setaEsquerda(5,230,0);
      if(modo == 0 || modo == 3)
         ativaCursor(1);
   }
   if(dispositivo.posX > limDir && antigoX <= limDir) {
      //Liga a seta para a direita
      if(modo == 0 || modo == 3)
         ativaCursor(0);
      setaDireita(595,230,14);
      if(modo == 0 || modo == 3)
         ativaCursor(1);
   }
   if(dispositivo.posX <= limDir && antigoX > limDir) {
      //Desliga a seta para a direita
      if(modo == 0 || modo == 3)
         ativaCursor(0);
      setaDireita(595,230,0);
      if(modo == 0 || modo == 3)
         ativaCursor(1);
   }
   if(dispositivo.posY < limSup && antigoY >= limSup) {
      //Liga a seta para cima e o Zoom Out
      if(modo == 0 || modo == 3)
         ativaCursor(0);
      setaCima(280,5,14);
      zoomOut(330,10,15);
      if(modo == 0 || modo == 3)
         ativaCursor(1);
   }
   if(dispositivo.posY >= limSup && antigoY < limSup) {
      //Desliga a seta para cima e o Zoom Out
      if(modo == 0 || modo == 3)
         ativaCursor(0);
      setaCima(280,5,0);
      zoomOut(330,10,0);
```

```
            if(modo == 0 || modo == 3)
               ativaCursor(1);
         }
         if(dispositivo.posY > limInf && antigoY <= limInf) {
            //Liga a seta para baixo e o Zoom In
            if(modo == 0 || modo == 3)
               ativaCursor(0);
            setaBaixo(280,435,14);
            zoomIn(330,440,15);
            if(modo == 0 || modo == 3)
               ativaCursor(1);
         }
         if(dispositivo.posY <= limInf && antigoY > limInf) {
            //Desliga a seta para baixo e o Zoom In
            if(modo == 0 || modo == 3)
               ativaCursor(0);
            setaBaixo(280,435,0);
            zoomIn(330,440,0);
            if(modo == 0 || modo == 3)
               ativaCursor(1);
         }
         antigoX = dispositivo.posX;   //Atualiza as variaveis com antigas pos.
         antigoY = dispositivo.posY;
         //Verifica se o botao esquerdo foi pressionado
         if(dispositivo.botEborda == 1) {
            //Verifica se existe a necessidade de se executar comandos
            if(dispositivo.posX < limEsq || dispositivo.posX > limDir ||
dispositivo.posY < limSup || dispositivo.posY > limInf) {
               if(modo == 0 || modo == 3)
                  ativaCursor(0);
               //Verifica quais as setas a serem ressaltadas, assim como
               //os motores e direcoes para efetuar o movimento.
               if(dispositivo.posX < limEsq) {
                  setaEsquerda(5,230,12);
                  if(MOTED == 1) {
                     motor1.Direcao = DIRESQ;    //Move para a esquerda
                     ligaMotor(&motor1);         //Ativa o motor, se permitido
                  }
                  if(MOTED == 2) {
                     motor2.Direcao = DIRESQ;
                     ligaMotor(&motor2);
                  }
                  if(MOTED == 3) {
                     motor3.Direcao = DIRESQ;
                     ligaMotor(&motor3);
                  }
               }
               if(dispositivo.posX > limDir) {    //Move para a direita
                  setaDireita(595,230,12);
                  if(MOTED == 1) {
                     motor1.Direcao = DIRDIR;
                     ligaMotor(&motor1);
                  }
                  if(MOTED == 2) {
                     motor2.Direcao = DIRDIR;
                     ligaMotor(&motor2);
                  }
                  if(MOTED == 3) {
```

```
            motor3.Direcao = DIRDIR;
            ligaMotor(&motor3);
        }
    }
    if(dispositivo.posY < limSup) {  //Move para cima
        setaCima(280,5,0);
        zoomOut(330,10,0);
        setaCima(310,5,12);
        if(MOTSI == 1) {
            motor1.Direcao = DIRSUP;
            ligaMotor(&motor1);
        }
        if(MOTSI == 2) {
            motor2.Direcao = DIRSUP;
            ligaMotor(&motor2);
        }
        if(MOTSI == 3) {
            motor3.Direcao = DIRSUP;
            ligaMotor(&motor3);
        }
    }
    if(dispositivo.posY > limInf) {     //Move para baixo
        setaBaixo(280,435,0);
        zoomIn(330,440,0);
        setaBaixo(310,435,12);
        if(MOTSI == 1) {
            motor1.Direcao = DIRINF;
            ligaMotor(&motor1);
        }
        if(MOTSI == 2) {
            motor2.Direcao = DIRINF;
            ligaMotor(&motor2);
        }
        if(MOTSI == 3) {
            motor3.Direcao = DIRINF;
            ligaMotor(&motor3);
        }
    }
    //Nao termina a execucao ate que o usuario solte o botao
    while(dispositivo.botE == 1) {
        iniciaContagem();       //Inicia a contagem de tempo
        atualizaMouse();        //Atualiza os dados do dispositivo
        executaMotores();       //Executa um ciclo nos motores
        finalizaContagem(PERIODO);  //Termina a contagem do periodo
    }
    motor1.Movendo = 0;    //Forca a parada de todos os motores
    motor2.Movendo = 0;
    motor3.Movendo = 0;
    limpaTeclado();       //Limpa o buffer do teclado
    atualizaMouse();            //Atualiza com os dados mais recentes
    //Recupera a tela a sua forma atual
    setaEsquerda(5,230,0);
    setaDireita(595,230,0);
    setaCima(310,5,0);
    setaBaixo(310,435,0);
    if(dispositivo.posX < limEsq)
        setaEsquerda(5,230,14);
    if(dispositivo.posX > limDir)
```

```
            setaDireita(595,230,14);
         if(dispositivo.posY < limSup) {
            setaCima(280,5,14);
            zoomOut(330,10,15);
         }
         if(dispositivo.posY > limInf) {
            setaBaixo(280,435,14);
            zoomIn(330,440,15);
         }
         if(modo == 0 || modo == 3)
            ativaCursor(1);
      } else {
         //Deve centralizar o cursor do dispositivo
         ativaCursor(0);
         posicionaCursor(319,239);          //Coloca a seta no centro
         if(modo == 0 || modo == 3)
            ativaCursor(1);
      }
   }
   //Verifica se o usuario pressionou o botao direito do mouse
   if(dispositivo.botDborda == 1) {
      //Verifica se ha comando a ser executado pelos motores
      if(dispositivo.posY < limSup || dispositivo.posY > limInf) {
         if(modo == 0 || modo == 3)
            ativaCursor(0);
         //Atualiza a tela para ressaltar o comando passado
         if(dispositivo.posX < limEsq) {
            setaEsquerda(5,230,0);
         }
         if(dispositivo.posX > limDir) {
            setaDireita(595,230,0);
         }
         if(dispositivo.posY < limSup) {     //Executa o zoom out
            setaCima(280,5,0);
            zoomOut(330,10,0);
            zoomOut(310,10,12);
            if(MOTZM == 1) {
               motor1.Direcao = DIRZOU;    //Define a direcao
               ligaMotor(&motor1);         //Ativa o motor, se permitido
            }
            if(MOTZM == 2) {
               motor2.Direcao = DIRZOU;
               ligaMotor(&motor2);
            }
            if(MOTZM == 3) {
               motor3.Direcao = DIRZOU;
               ligaMotor(&motor3);
            }
         }
         if(dispositivo.posY > limInf) {    //Efetua o Zoom In
            setaBaixo(280,435,0);
            zoomIn(330,440,0);
            zoomIn(310,440,12);
            if(MOTZM == 1) {
               motor1.Direcao = DIRZIN;
               ligaMotor(&motor1);
            }
            if(MOTZM == 2) {
```

```
                motor2.Direcao = DIRZIN;
                ligaMotor(&motor2);
            }
            if(MOTZM == 3) {
                motor3.Direcao = DIRZIN;
                ligaMotor(&motor3);
            }
        }
        //Executa ate que o usuario solte o botao do mouse
        while(dispositivo.botD == 1) {
            iniciaContagem();       //Inicia a contagem de tempo
            atualizaMouse();        //Atualiza os dados do dispositivo
            executaMotores();       //Executa um ciclo nos motores
            finalizaContagem(PERIODO);  //Termina a contagem do periodo
        }
        motor1.Movendo = 0;    //Forca a parada de todos os motores
        motor2.Movendo = 0;
        motor3.Movendo = 0;
        limpaTeclado();        //Limpa o buffer do teclado
        atualizaMouse();            //Atualiza com os dados mais recentes
        //Recupera a tela a sua forma atual
        zoomOut(310,10,0);
        zoomIn(310,440,0);
        if(dispositivo.posX < limEsq)
            setaEsquerda(5,230,14);
        if(dispositivo.posX > limDir)
            setaDireita(595,230,14);
        if(dispositivo.posY < limSup) {
            setaCima(280,5,14);
            zoomOut(330,10,15);
        }
        if(dispositivo.posY > limInf) {
            setaBaixo(280,435,14);
            zoomIn(330,440,15);
        }
        if(modo == 0 || modo == 3)
            ativaCursor(1);
    } else {
        //Verifica se o cursor esta no meio do retangulo. Caso sim,
        //deve alterar o modo de exibicao.
        if(dispositivo.posX >= limEsq && dispositivo.posX <= limDir)
            alteraModo();
    }
}
//Verifica se o usuario pressionou o botao central do mouse
if(dispositivo.botMborda == 1) {
    //Deve centralizar o cursor do dispositivo
    ativaCursor(0);
    posicionaCursor(319,239);        //Coloca a seta no centro
    if(modo == 0 || modo == 3)
        ativaCursor(1);
}
}
}
```

```
/*****************************************************************************
 ***   SOFTWARE DE CONTROLE - ROBO MANIPULADOR DE ENDOSCOPIO CIRURGICO   ***
 ***                                                                     ***
 ***  Arquivo    :  ENDO.C                                               ***
 ***  Conteudo   :  Programa Principal                                   ***
 ***  Descricao  :  Contem o programa principal do software, um menu ao  ***
 ***                usuario que permite ao mesmo selecionar entre a      ***
 ***                calibracao do dispositivo de entrada, a configuracao ***
 ***                dos parametros do robo e a operacao do mesmo em si.  ***
 ***                Tambem efetua a inicializacao do mouse, video e timer***
 ***                assim como sua recuperacao ao status normal.         ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330           ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452           ***
 ***  Data       :  13/09/2001                                          ***
 *****************************************************************************/

#include<stdio.h>       //Inclui as bibliotecas padrao do C
#include<stdlib.h>
#include<dos.h>
#include<graph.h>
#include"video.h"       //Inclui os arquivos com as funcoes e procedimentos
#include"mouse.h"       //utilizados no controle do robo.
#include"teclado.h"
#include"motores.h"
#include"timer.h"
#include"clbrmous.c"
#include"cnfgmot.c"
#include"operacao.c"

//Programa Principal
//Argumentos : nenhum
//Retorno    : nenhum
void main(void) {
   char tecla;
   int codigo;
   int realizouCalibracao = 0;   //Indica se a calibracao do mouse foi feita
   int realizouConfiguracao = 0; //Indica configuracao dos motores foi feita

   //Inicializacao de video e mouse
   inicializaVideo();
   if(iniciaMouse() == 0) {        //Caso o driver nao esteja instalado
      _settextposition(14,22);     //avisa o usuario e sai do programa
      printf("O DRIVER DO MOUSE NAO FOI ENCONTRADO.");
      _settextposition(15,9);
      printf("INSTALE O DRIVER DE MOUSE CTMOUSE (OPCAO /3 PARA 3 BOTOES)");
      getch();
      recuperaVideo();    //Retorna o video a sua configuracao padrao
      return;     //Finaliza a execucao
   }
   //Inicializa o timer de precisao de microsegundos
   inicializaTimer();
   //Permanece executando ate usuario escolher sair (codigo de tecla ESC)
   //<ESC> e' uma tecla de emergencia que abandona praticamente qualquer
   //procedimento em todo o software.
   while(codigo != 27) {
      //Desenha o menu principal
      limpaTela();
      retangulo(109,79,549,434,8,1);
```

```
retangulo(99,69,539,424,3,1);
retangulo(99,69,539,424,15,0);
retangulo(155,99,479,176,0,1);
retangulo(155,99,479,176,15,0);
retangulo(125,200,515,395,0,1);
retangulo(125,200,515,395,15,0);
_settextcolor(12);
_settextposition(8,27);
_outtext("R O B O   C I R U R G I C O");
_settextcolor(9);
_settextposition(10,24);
_outtext("(MODULO MANIPULADOR DE ENDOSCOPIO)");
_settextcolor(9);
_settextposition(14,18);
_outtext("Selecione a Opera‡ao :");
if(realizouCalibracao != 1) {        //Cor das opcoes depende do
   _settextcolor(14);                //que ja foi realizado.
} else {
   _settextcolor(7);
}
_settextposition(16,18);
_outtext("1 - Calibrar Dispositivo de Entrada");
if(realizouConfiguracao != 1) {      //Cor das opcoes depende do
   _settextcolor(14);                //que ja foi realizado.
} else {
   _settextcolor(7);
}
_settextposition(18,18);
_outtext("2 - Definir Parametros de Configuracao do Robo");
if(realizouConfiguracao == 1) {
   _settextcolor(14);
} else {
   _settextcolor(7);
}
_settextposition(20,18);
_outtext("3 - Posicionar Robo (via teclado)");
if(realizouCalibracao == 1 && realizouConfiguracao == 1) {
   _settextcolor(14);                //Cor das opcoes depende do
} else {                             //que ja foi realizado.
   _settextcolor(7);
}
_settextposition(22,18);
_outtext("4 - Iniciar Opera‡ao (efetuar cirurgia)");
_settextcolor(7);
_settextposition(24,18);
_outtext("5 - Abandonar o Aplicativo");
codigo = 0;
//Recupera um comando valido do teclado
while((codigo < 49 || codigo > 53) && codigo != 27) {
   tecla = getch();       //Recupera uma tecla pressionada
   codigo = (int)tecla;
}
if(codigo == 49) {
   //Calibra o dispositivo de entrada (mouse)
   calibraDispositivo();
   realizouCalibracao = 1;    //Informa a realizacao da calibraca
}
if(codigo == 50) {
```

```
         //Configura os parametros dos motores
         configuraMotores();
         realizouConfiguracao = 1;  //Informa a realizacao da configuracao
      }
      if(codigo == 51) {   //Posiciona manualmente o robo (teclado)
         //So' permite o posicionamento caso tenha configurado motores
         if(realizouConfiguracao == 1) {
            posicionaRobo(0);      //Procedimento de posicionamento
         } else {   //Caso contrario mostra mensagem de erro
            retangulo(115,188,535,235,8,1);
            retangulo(110,183,530,230,0,1);
            retangulo(110,183,530,230,15,0);
            _settextposition(13,16);
            _settextcolor(15);
            _outtext("REALIZE A CONFIGURA€AO DO ROBO (INCLUI POSICIONAM.)");
            _settextposition(14,28);
            _settextcolor(7);
            _outtext("<Pressione qualquer tecla>");
            tecla = getch();   //Suspende o programa ate' pressionar tecla
         }
      }
      if(codigo == 52) {    //Comando de operacao normal do robo
         //So' permite a operacao caso tenha calibrado o mouse e configurado
         //os motores.
         if(realizouCalibracao == 1 && realizouConfiguracao == 1) {
            efetuaOperacao();       //Inicia a operacao normal do robo
         } else {
            //Mostra mensagem de erro ao usuario
            retangulo(115,188,535,235,8,1);
            retangulo(110,183,530,230,0,1);
            retangulo(110,183,530,230,15,0);
            _settextposition(13,16);
            _settextcolor(15);
            _outtext("E' OBRIGATORIO EFETUAR A CALIBRA€AO E CONFIGURA€AO");
            _settextposition(14,28);
            _settextcolor(7);
            _outtext("<Pressione qualquer tecla>");
            tecla = getch();   //Suspende o programa ate' pressionar tecla
         }
      }
      if(codigo == 53) {    //Codigo de saida do programa
         //Mostra mensagem perguntando se o usuario deseja mesmo sair
         retangulo(140,188,510,250,8,1);
         retangulo(135,184,505,246,0,1);
         retangulo(135,184,505,246,15,0);
         _settextposition(13,20);
         _settextcolor(15);
         _outtext("DESEJA REALMENTE ABANDONAR O APLICATIVO ?");
         _settextposition(15,20);
         _settextcolor(12);
         _outtext("    <S> - Sim              <N> - Nao    ");
         //Aguarda que o usuario tecle `S` ou `N`
         while(tecla != 'S' && tecla != 's' && tecla != 'N' && tecla != 'n' &&
codigo != 27) {
            tecla = getch();     //Recupera uma tecla pressionada
            codigo = (int)tecla;
         }
         if(tecla == 'N' || tecla == 'n') {
```

```
            codigo = 0;     //Caso nao queira sair, cancela o comando
         } else {
            codigo = 27;    //Caso queira sair, simula ter pressionado ESC
         }
      }
   }
   //Reseta o driver do mouse e volta a configuracao padrao de video
   iniciaMouse();
   recuperaVideo();
   //Retorna o timer (IRQ 0) para sua configuracao padrao
   recuperaTimer();
}
```

```
/*************************************************************************
 ***                  TESTE COMPLETO DE PRECISAO DE TIMER                ***
 ***                                                                     ***
 ***  Arquivo    :  TESTIMER.C                                           ***
 ***  Conteudo   :  Teste de precisao do timer utilizado                 ***
 ***  Descricao  :  Contem inumeros testes de precisao do timer utilizado ***
 ***                no software de controle do robo manipulador de       ***
 ***                endoscopio cirurgico, timer este utilizado para gerar ***
 ***                os ciclos que originam os trens de pulsos que movem  ***
 ***                os motores de passo. Inclui testes dos procedimentos ***
 ***                de leitura do timer, alem de procedimentos de espera ***
 ***                e contagem de tempo, teste de repetibilidade dos     ***
 ***                periodos e teste de tempo necessario para a execucao ***
 ***                de um ciclo do software real utilizado.              ***
 ***  Autoria    :  Marcos Alexandre Scholtz       -  2367330           ***
 ***                Hugo Leonardo Paiva Rodrigues  -  2367452           ***
 ***  Data       :  11/09/2001                                           ***
 *************************************************************************/

#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<graph.h>
#include<dos.h>
#include"video.h"
#include"timer.h"
#include"mouse.h"
#include"motores.h"

//Programa principal, realiza os testes com o timer de timer.h
void main(void) {
   long inicio;    //Variaveis tipo longo para conter contagens de clock
   long fim;
   double tempo;  //Variavel tipo duplo para conter tempo em ms ou microseg.
   double total;   //Variavel para calculo da media dos valores
   double maximo;  //Variavel para conter o maximo valor
   double minimo;  //Variavel para conter o minimo valor
   int I;           //Variaveis tipo integer para teste com calculos
   int numero;

   //Inicializa o timer com precisao de microsegundos
   inicializaTimer();

   //Verifica quanto tempo o procedimento de leitura demora para contar o
   //tempo decorrido no contador.
   inicio = leTimer();   //Le o timer, colocando o clock na variavel inicio
   fim = leTimer();      //Imediatamente, le novamente, colocando no fim
   tempo = tempoDecorrido(inicio, fim);   //Calcula a diferenca entre os dois
   //Informa a diferenca detectada na chamada
   printf("\nTempo entre chamadas sucessivas : %f ms (%f micros)\n",tempo,
tempo*1000.0);

   //Verifica a precisao de um procedimento de espera, aguardando 5 ms, 1 ms,
   //0.5 ms e 0.1 ms, sucessivamente.
   inicio = leTimer();   //Faz a leitura de inicio
   espera(5000);         //Manda aguardar 5 ms (5000 microsegundos)
   fim = leTimer();      //Faz a leitura de final
   tempo = tempoDecorrido(inicio, fim);    //Calcula o tempo decorrido
```

```
   printf("\nTempo para a chamada de 'espera(5000)' : %f ms (%f micros)\n",tempo,
tempo*1000.0);
   inicio = leTimer();   //Faz a leitura de inicio
   espera(1000);         //Manda aguardar 1 ms (1000 microsegundos)
   fim = leTimer();      //Faz a leitura de final
   tempo = tempoDecorrido(inicio, fim);    //Calcula o tempo decorrido
   printf("Tempo para a chamada de 'espera(1000)' : %f ms (%f micros)\n",tempo,
tempo*1000.0);
   inicio = leTimer();   //Faz a leitura de inicio
   espera(500);         //Manda aguardar 0.5 ms (500 microsegundos)
   fim = leTimer();      //Faz a leitura de final
   tempo = tempoDecorrido(inicio, fim);    //Calcula o tempo decorrido
   printf("Tempo para a chamada de 'espera(500)'  : %f ms (%f micros)\n",tempo,
tempo*1000.0);
   inicio = leTimer();   //Faz a leitura de inicio
   espera(100);         //Manda aguardar 0.1 ms (100 microsegundos)
   fim = leTimer();      //Faz a leitura de final
   tempo = tempoDecorrido(inicio, fim);    //Calcula o tempo decorrido
   printf("Tempo para a chamada de 'espera(100)'  : %f ms (%f micros)\n",tempo,
tempo*1000.0);

   //Verifica a precisao do procedimento de espera que considera codigo
   //executado, sem prejudicar o teste pelo tempo entre chamadas sucessivas.
   //Executa o teste para os valores de 5 ms, 1 ms, 0.5 ms e 0.1 ms.
   iniciaContagem();         //Inicia a contagem de tempo
   finalizaContagem(5000);   //Finaliza a contagem ao atingir 5 ms
   //Calcula o tempo decorrido com as variaveis globais envolvidas no processo
   tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);
   printf("\nTempo para contagem em vazio, 5000 microseg. : %f ms (%f
micros)\n",tempo,tempo*1000.0);
   iniciaContagem();         //Inicia a contagem de tempo
   finalizaContagem(1000);   //Finaliza a contagem ao atingir 1 ms
   //Calcula o tempo decorrido com as variaveis globais envolvidas no processo
   tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);
   printf("Tempo para contagem em vazio, 1000 microseg. : %f ms (%f
micros)\n",tempo,tempo*1000.0);
   iniciaContagem();         //Inicia a contagem de tempo
   finalizaContagem(500);   //Finaliza a contagem ao atingir 0.5 ms
   //Calcula o tempo decorrido com as variaveis globais envolvidas no processo
   tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);
   printf("Tempo para contagem em vazio, 500 microseg.  : %f ms (%f
micros)\n",tempo,tempo*1000.0);
   iniciaContagem();         //Inicia a contagem de tempo
   finalizaContagem(100);   //Finaliza a contagem ao atingir 0.1 ms
   //Calcula o tempo decorrido com as variaveis globais envolvidas no processo
   tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);
   printf("Tempo para contagem em vazio, 100 microseg.  : %f ms (%f
micros)\n",tempo,tempo*1000.0);

   //Verifica a repetibilidade do tempo decorrido neste mesmo procedimento,
   //buscando um total de 1000 valores e calculando sua media, assim como
   //os valores maximo e minimo obtidos. O teste sera feito com um periodo
   //de contagem de 500 microsegundos.
   maximo = 0;
   minimo = 100000;
   total = 0;
   printf("\nCalculando repetibilidade (1000 amostras, 500 microsegundos)...\n");
   for(I=1;I<=1000;I++) {
```

```
        iniciaContagem();
        finalizaContagem(500);
        tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);
        if(tempo > maximo)
           maximo = tempo;
        if(tempo < minimo)
           minimo = tempo;
        total = total + tempo;
     }
     tempo = total / 1000.0;
     printf("Tempo medio para 500 microseg.      : %f ms (%f
 micros)\n",tempo,tempo*1000.0);
     printf("Tempo maximo encontrado na amostra : %f ms (%f
 micros)\n",maximo,maximo*1000.0);
     printf("Tempo minimo encontrado na amostra : %f ms (%f
 micros)\n",minimo,minimo*1000.0);

     //Verifica a manutencao da precisao deste procedimento, com determinado
     //codigo sendo executado durante a contagem de tempo. No caso, serao
     //gerados numeros pseudo-aleatorios (uma operacao trigonometrica complexa)
     //com o intuito de gastar determinado tempo. Testes novamente para 5 ms,
     //1 ms, 0.5 ms e 0.1 ms .
     srand(1);         //Inicializa o randomizador do C
     inicio = leTimer();    //Guarda o tempo de inicio
     for(I=1;I<800;I++)    //Executa varias vezes o calculo de num. aleatorio
        numero = rand();
     fim = leTimer();      //Le o timer para verificar o tempo decorrido
     tempo = tempoDecorrido(inicio, fim);     //Calcula o tempo decorrido
     printf("\nTempo estimado p/ calculo dos num. aleatorios : %f ms (%f
 micros)\n",tempo, tempo*1000.0);
     iniciaContagem();      //Inicia uma contagem de tempo
     for(I=1;I<800;I++)    //Executa o mesmo codigo
        numero = rand();
     finalizaContagem(5000);    //Finaliza a contagem com 5 ms
     tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);  //Calcula o tempo
     printf("Tempo contado com calculos, 5000 microseg. : %f ms (%f
 micros)\n",tempo,tempo*1000.0);
     iniciaContagem();      //Inicia uma contagem de tempo
     for(I=1;I<800;I++)    //Executa o mesmo codigo
        numero = rand();
     finalizaContagem(1000);    //Finaliza a contagem com 1 ms
     tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);  //Calcula o tempo
     printf("Tempo contado com calculos, 1000 microseg. : %f ms (%f
 micros)\n",tempo,tempo*1000.0);
     iniciaContagem();      //Inicia uma contagem de tempo
     for(I=1;I<800;I++)    //Executa o mesmo codigo
        numero = rand();
     finalizaContagem(500);    //Finaliza a contagem com 0.5 ms
     tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);  //Calcula o tempo
     printf("Tempo contado com calculos, 500 microseg.  : %f ms (%f
 micros)\n",tempo,tempo*1000.0);
     iniciaContagem();      //Inicia uma contagem de tempo
     for(I=1;I<800;I++)    //Executa o mesmo codigo
        numero = rand();
     finalizaContagem(100);    //Finaliza a contagem com 0.1 ms
     tempo = tempoDecorrido(Tempo_Inicial, Tempo_Final);  //Calcula o tempo
     printf("Tempo contado com calculos, 100 microseg.  : %f ms (%f
 micros)\n",tempo,tempo*1000.0);
```

```
   //Executa um teste simples do tempo necessario para gerar uma saida
   //simples para o video do computador, uma operacao reconhecidamente
   //lenta em um microcomputador.
   inicio = leTimer();   //Faz a leitura de inicio
   printf("\nTestando tempo necessario para saida no video...\n");
   fim = leTimer();      //Faz a leitura de final
   tempo = tempoDecorrido(inicio, fim);    //Calcula o tempo decorrido
   printf("Tempo estimado p/ saida de 1 linha no video : %f ms (%f
micros)\n",tempo, tempo*1000.0);

   //Executa um teste do tempo necessario para a execucao do codigo real
   //do software de controle do robo manipulador de endoscopio cirurgico.
   //Sabe-se que a cada ciclo e' executada uma comparacao (saida do ciclo,
   //verificacao de botao do mouse), uma atualizacao dos parametros do
   //dispositivo (mouse) e a execucao de um ciclo nos motores. Para tanto,
   //considera-se o pior caso, em que se faz necessaria a inversao de algum
   //sinal do motor.
   printf("\nCalculando tempo maximo para execucao de ciclo real no
software...\n");
   iniciaMouse();          //Inicia o driver do mouse  (reset)
   dispositivo.posX = 0;   //Zera as posicoes para forcar uma atualizacao
   dispositivo.posY = 0;   //das variaveis de posicao do dispositivo
   motor1.Velocidade = 5;  //Seta a velocidade do motor e a contagem atual
   motor1.Contagem = 5;    //de execucoes de ciclos para forcar o motor a
   motor1.Movendo = 1;     //executar uma inversao de sinal
   motor2.Velocidade = 5;
   motor2.Contagem = 5;
   motor2.Movendo = 1;
   motor3.Velocidade = 5;
   motor3.Contagem = 5;
   motor3.Movendo = 1;
   inicio = leTimer();   //Le o tempo de inicio
   atualizaMouse();            //Codigo de atualizacao do mouse
   executaMotores();           //Codigo de execucao de ciclo dos motores
   if(dispositivo.botE == 1)   //Simula a comparacao de saida do ciclo
      I = dispositivo.botE;
   fim = leTimer();     //Le o tempo de fim
   tempo = tempoDecorrido(inicio, fim);    //Calcula o tempo decorrido
   printf("Tempo estimado p/ executar 1 ciclo real : %f ms (%f micros)\n",tempo,
tempo*1000.0);

   recuperaTimer();   //Recupera o status normal do IRQ 0 antes de sair
}
```

APÊNDICE I - FOTOS DOS COMPONENTES E SISTEMAS DO ROBÔ

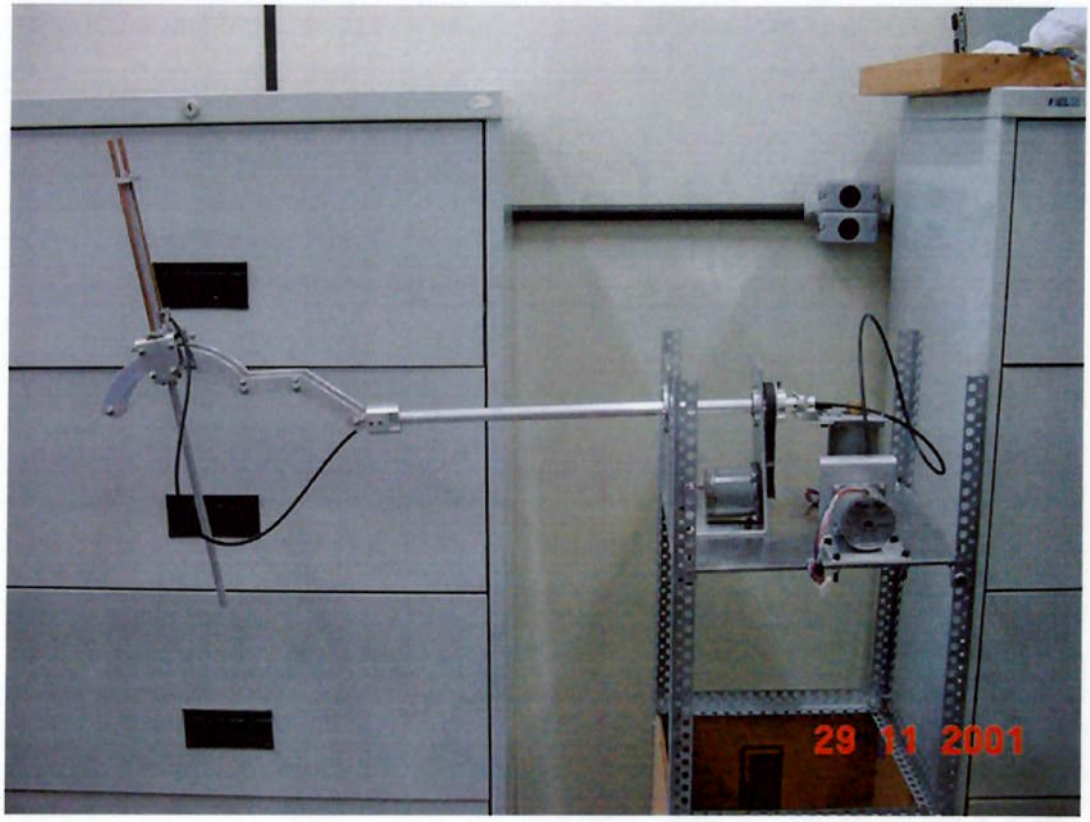


Foto 1 – Visão geral do mecanismo com a base

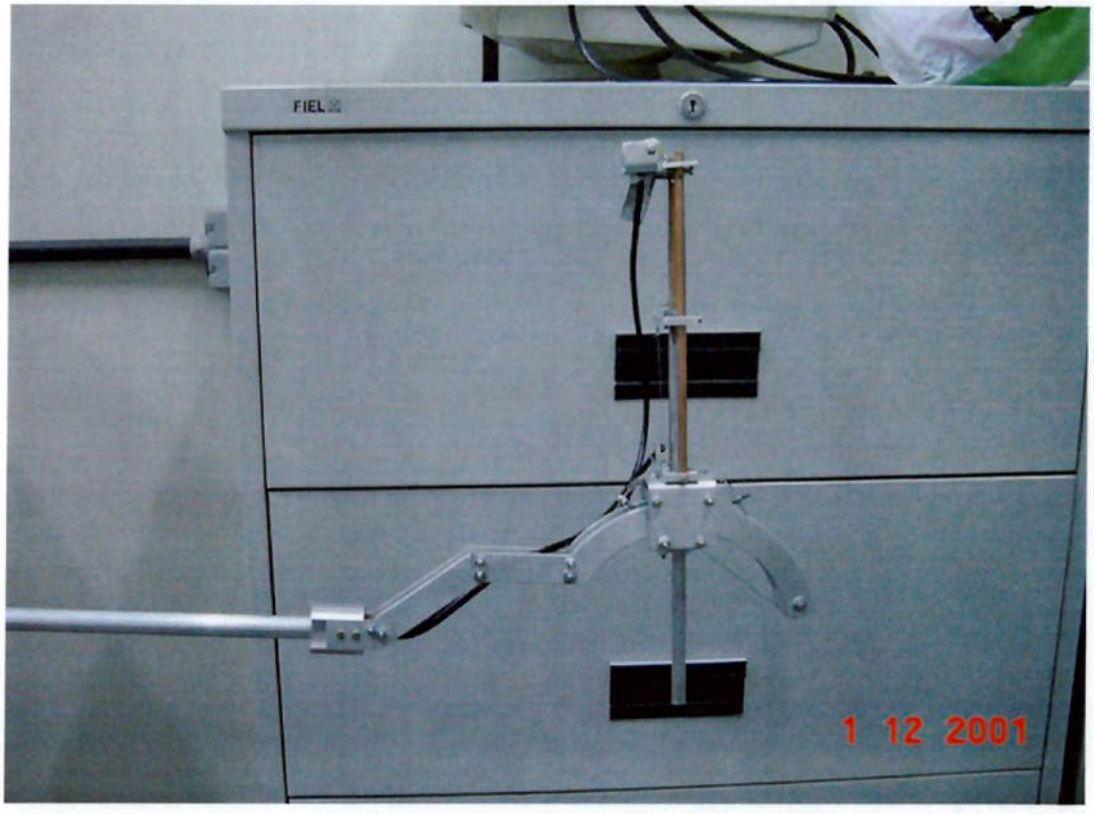


Foto 2 – Mecanismo de manipulação do endoscópio

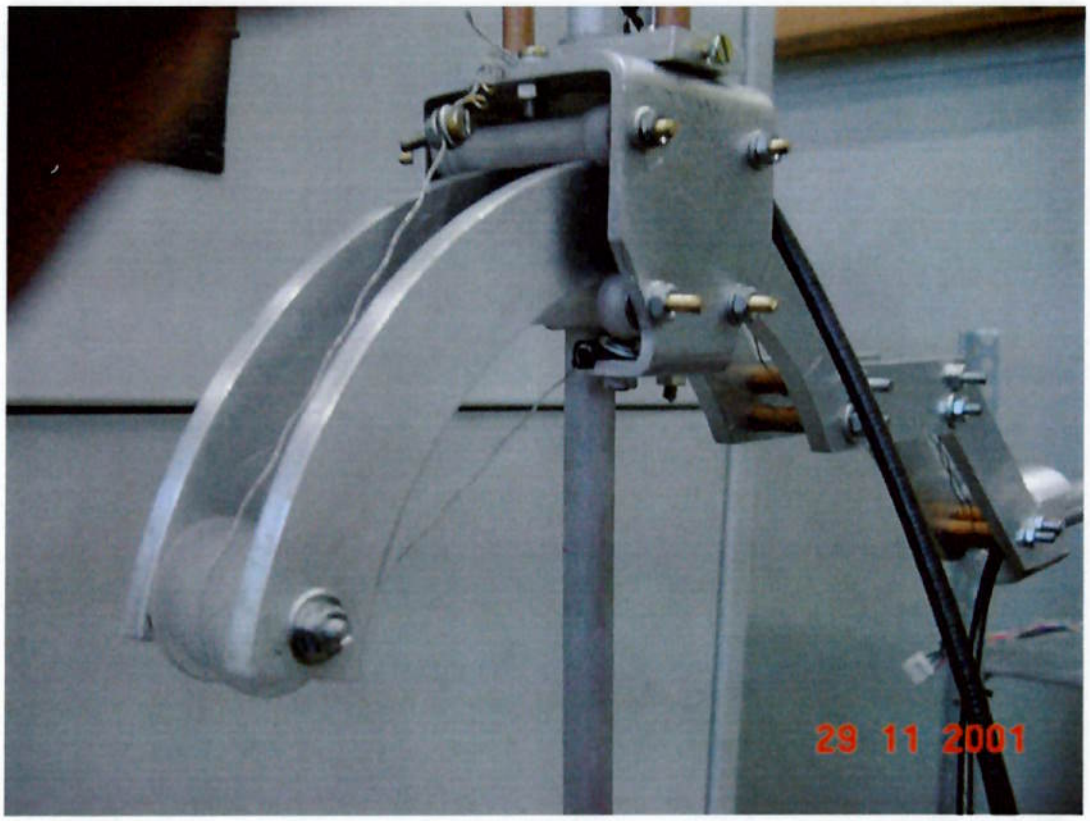


Foto 3 – Vista frontal dos trilhos, mostrando cabo de acionamento do carro

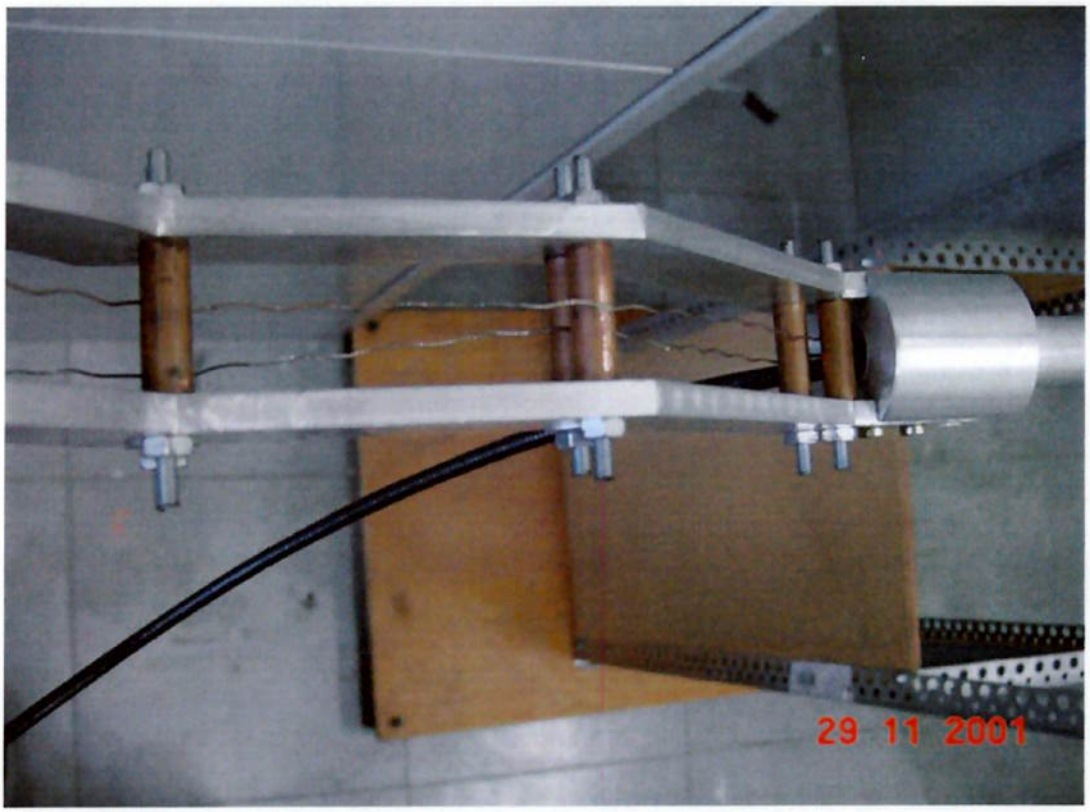


Foto 4 – Vista superior dos trilhos, mostrando caminho percorrido pelos cabos

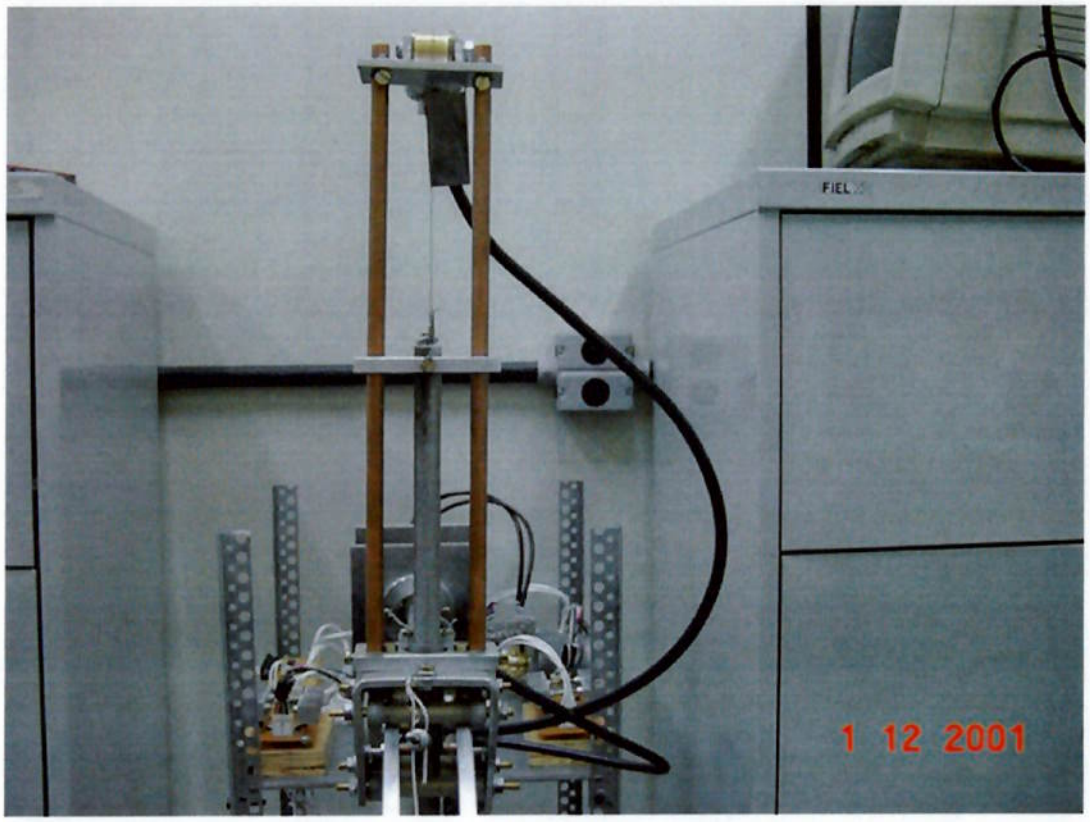


Foto 5 – Vista frontal do acionamento do zoom

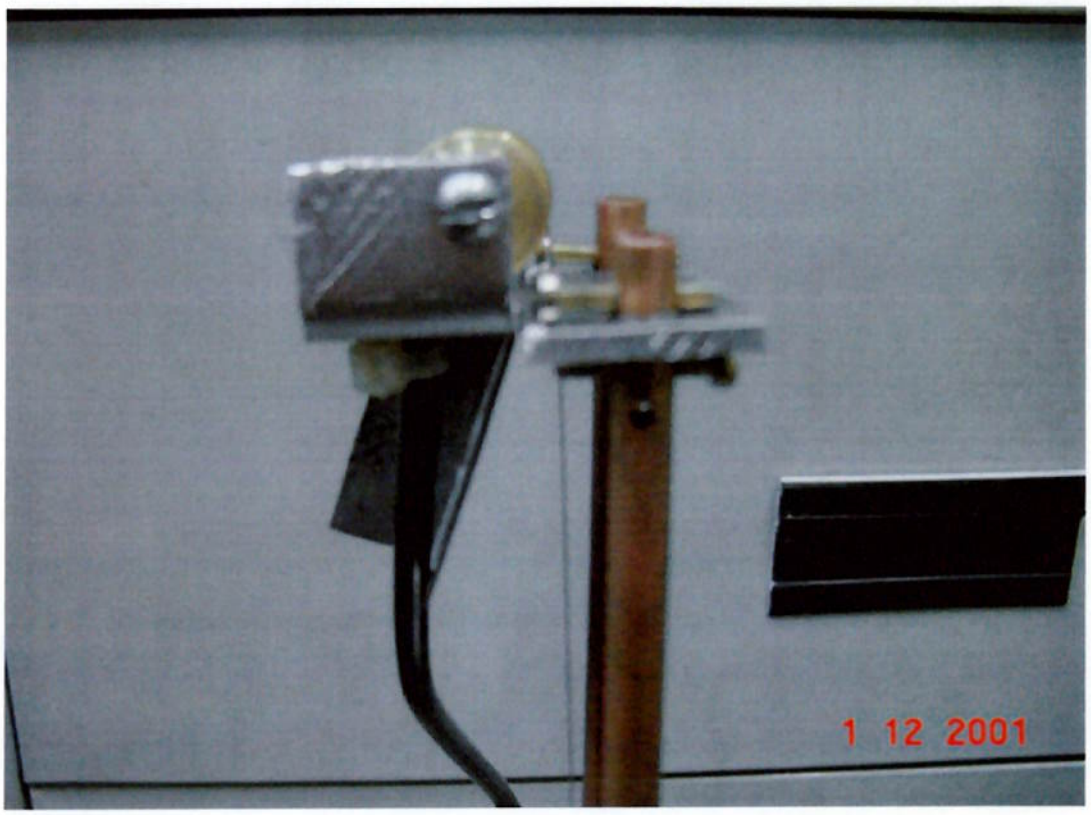


Foto 6 – Parte superior do acionamento do zoom

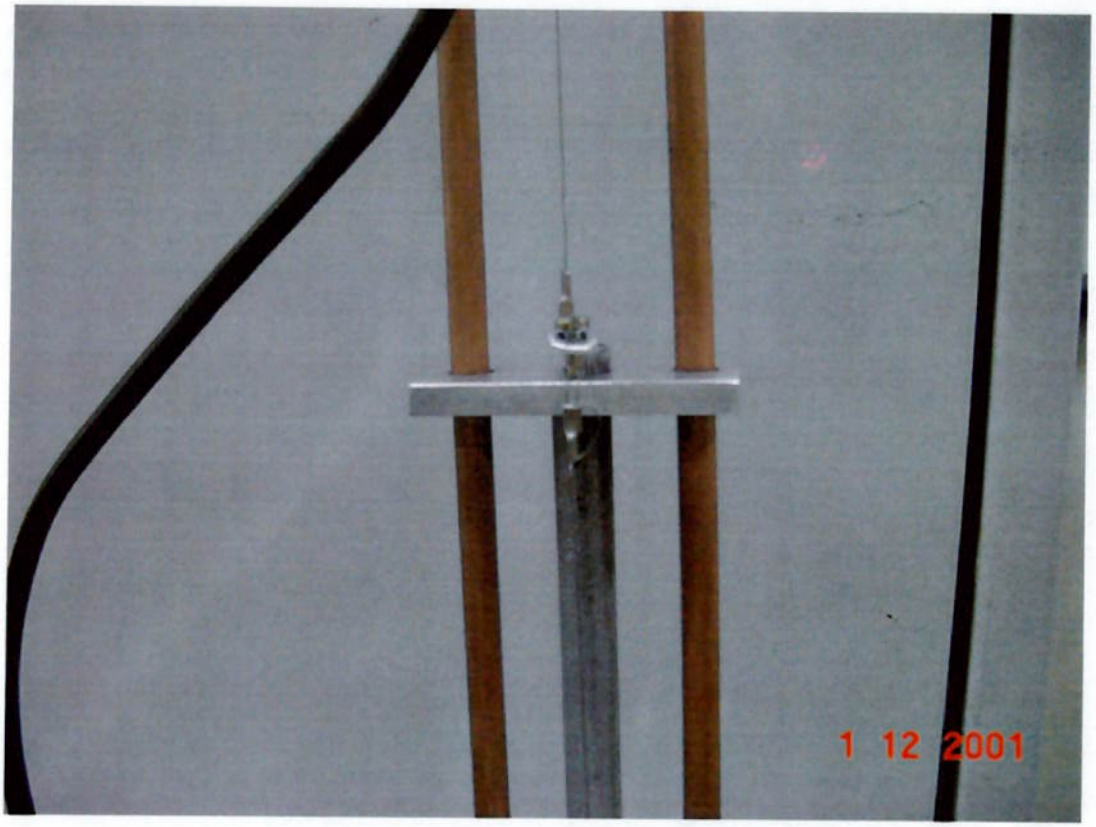


Foto 7 – Detalhe da fixação do suporte do endoscópio

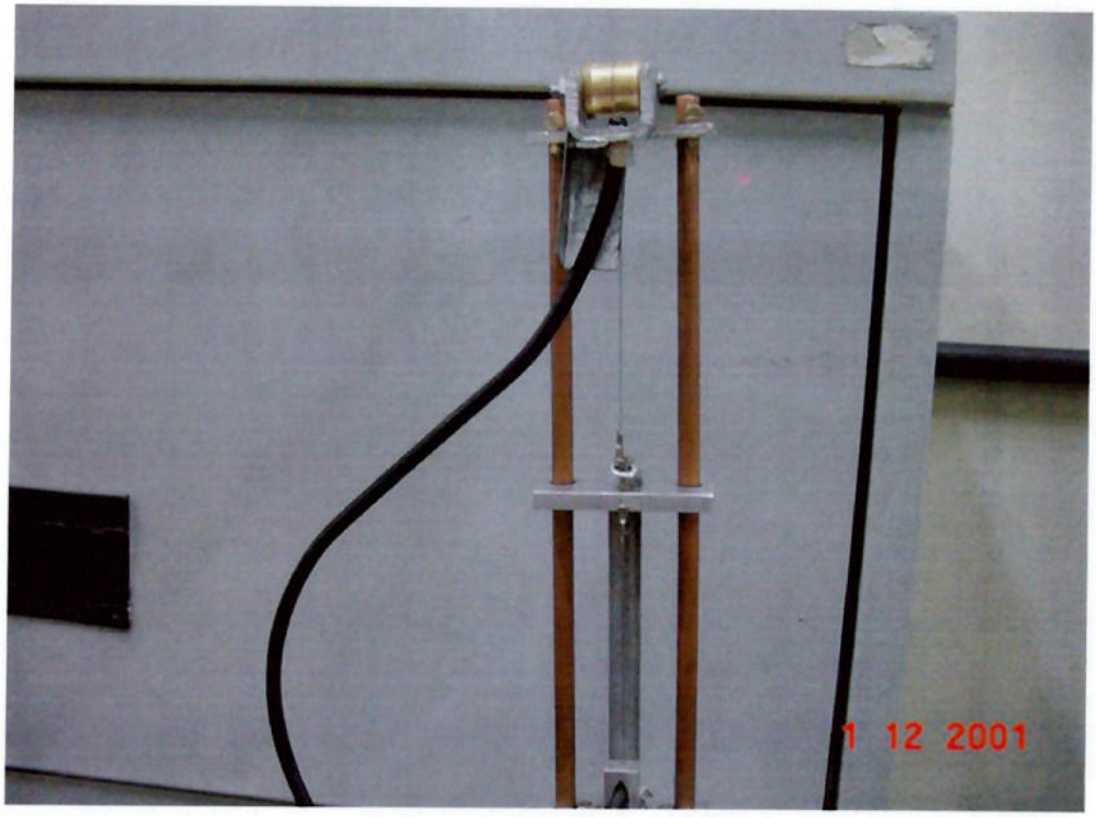


Foto 8 – Acionamento do zoom

Foto 9 – Vista lateral da base com os motores

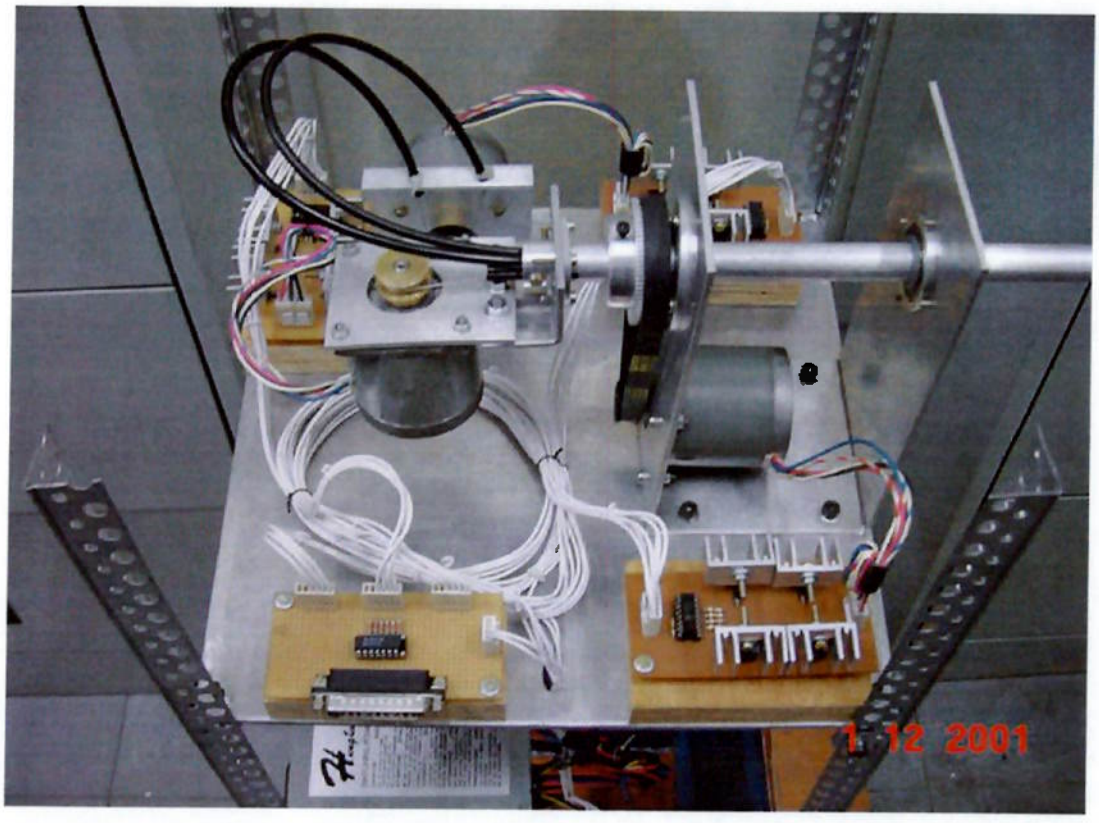


Foto 10 – Base com os motores e sistemas eletrônicos

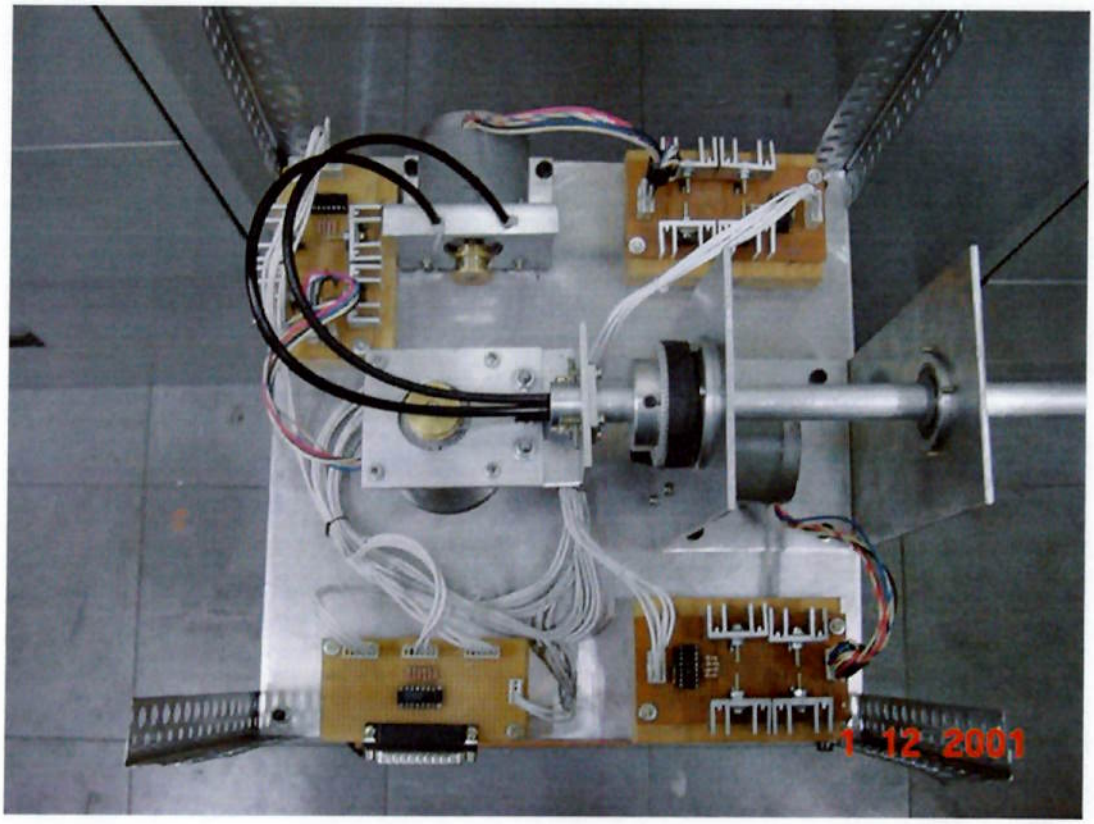


Foto 11 – Vista superior da base

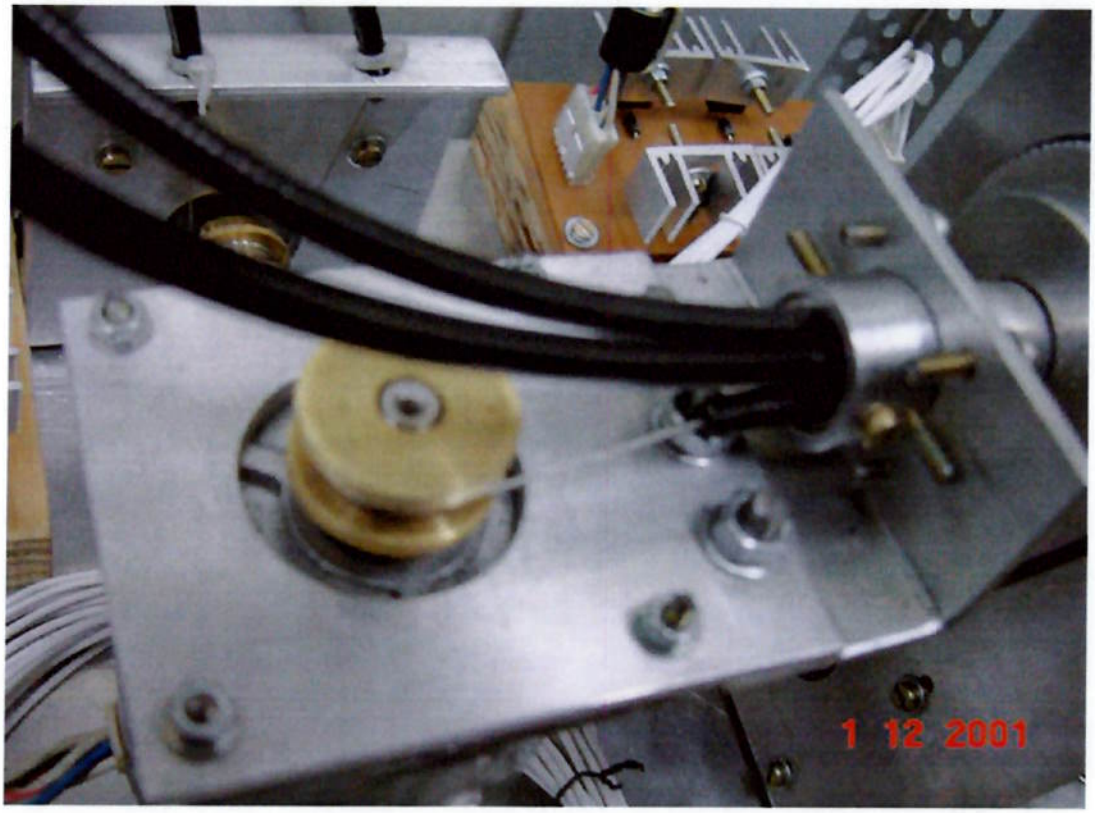


Foto 12 – Detalhe do acionamento dos cabos que movem o carro

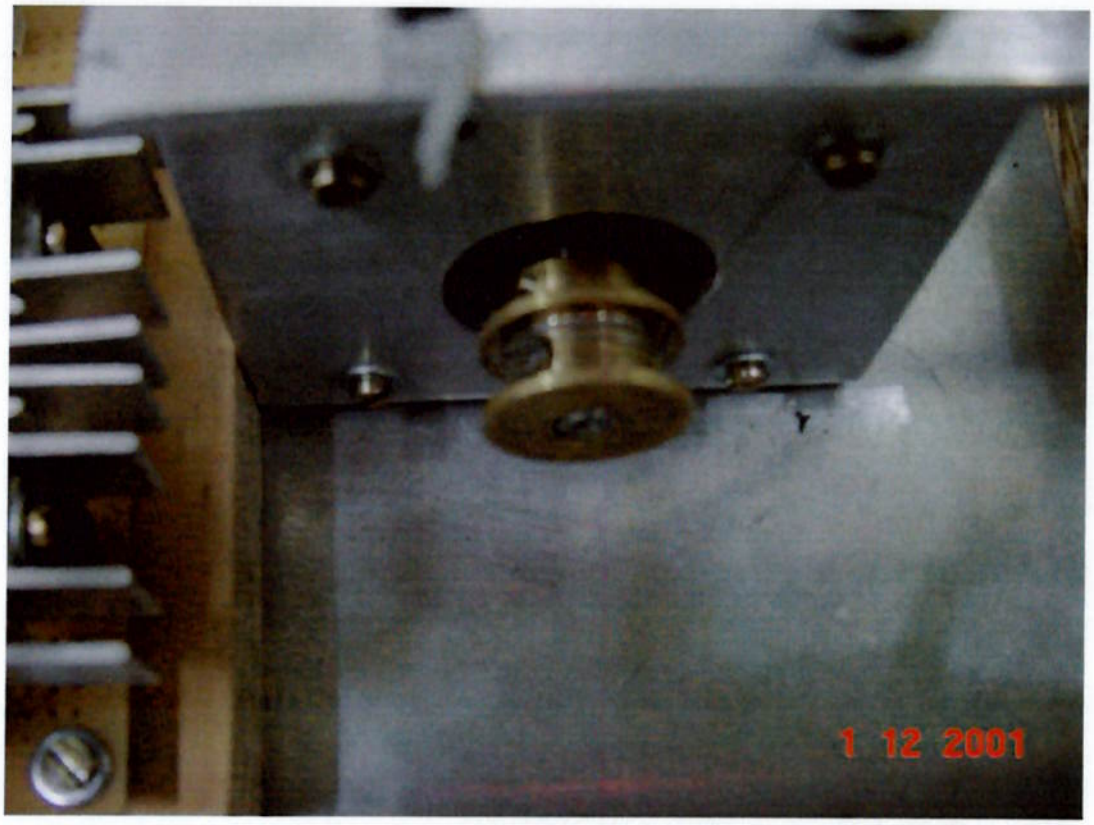


Foto 13 – Detalhe do acionamento dos cabos que movem o zoom

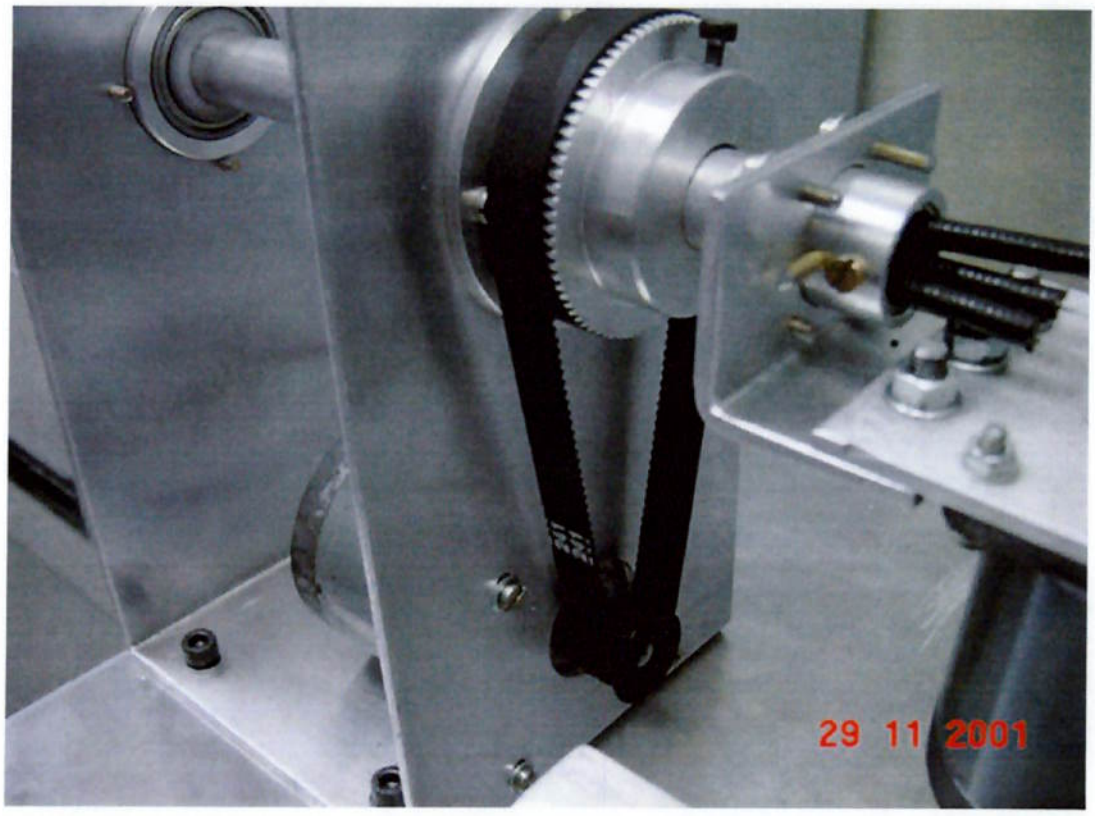


Foto 14 – Detalhe do acionamento principal (rotação da estrutura)

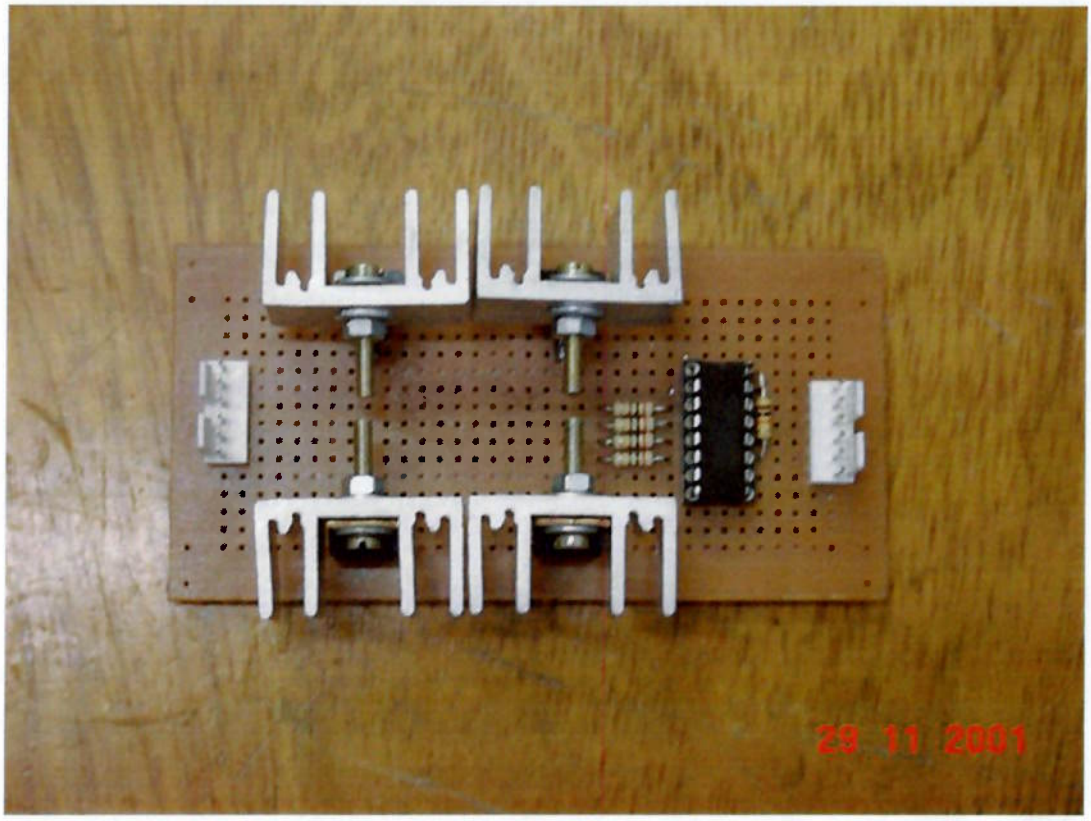


Foto 15 – Vista superior de um driver de motor de passo

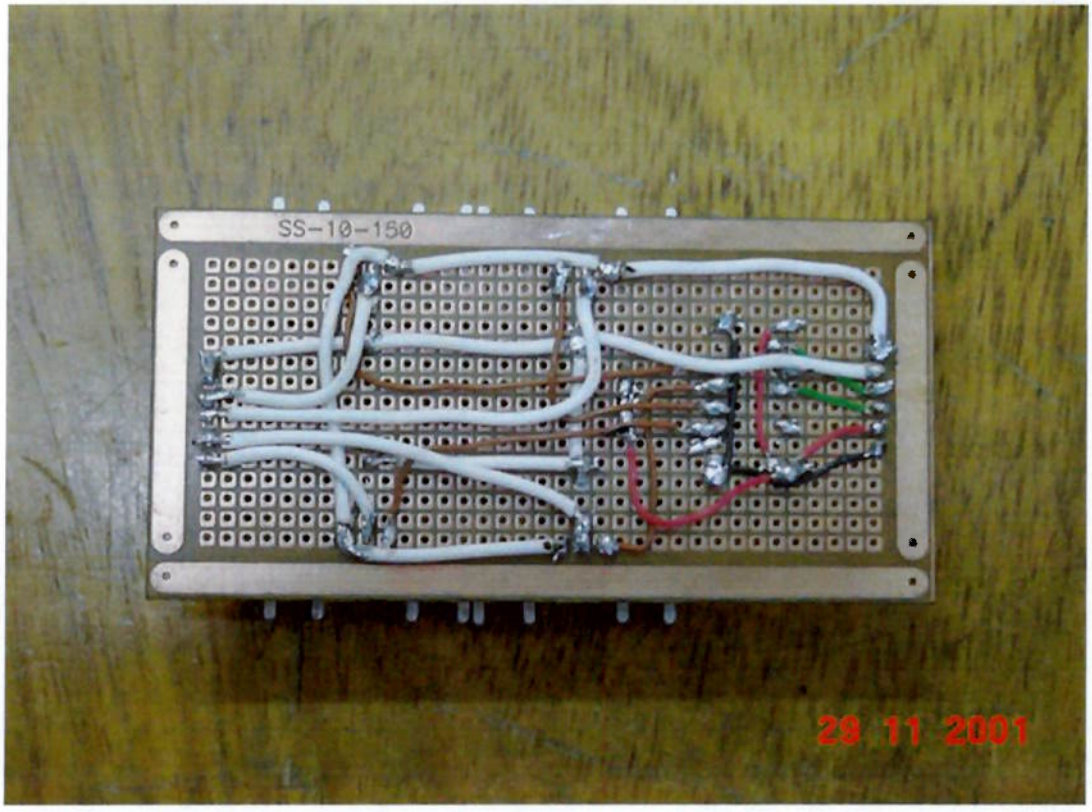


Foto 16 – Vista inferior de um driver de motor de passo

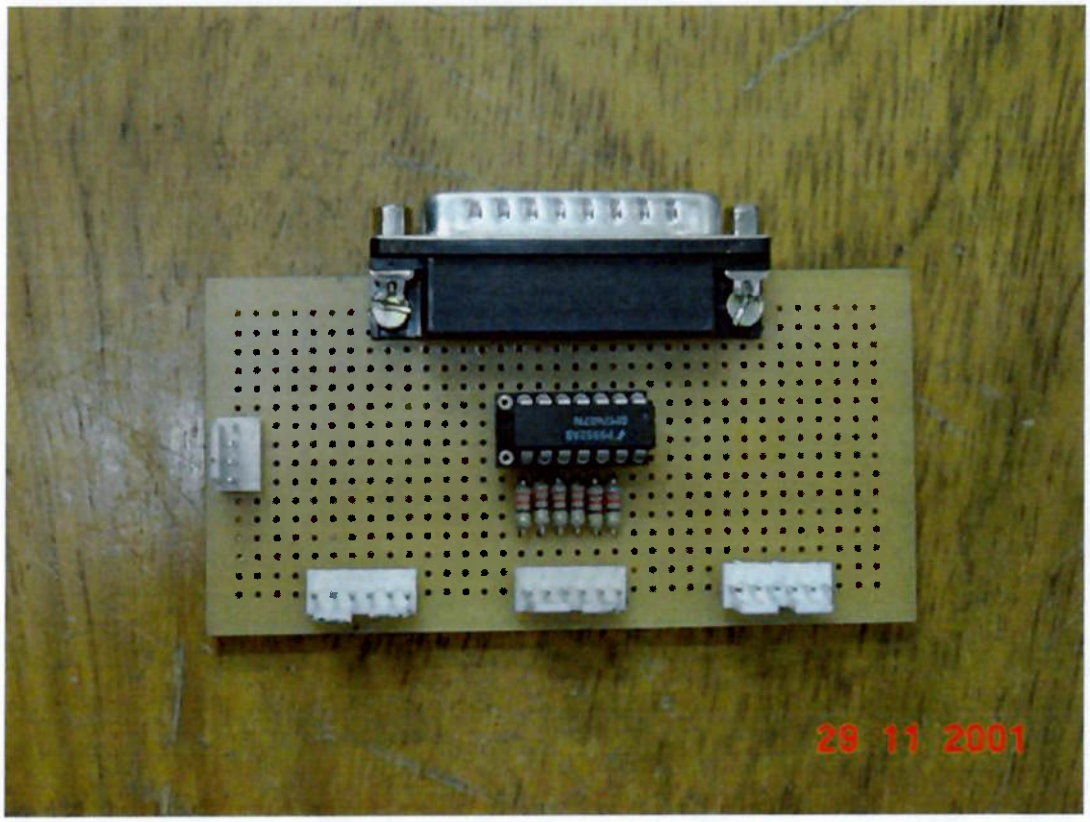


Foto 17 – Vista superior da placa de comunicação com o PC

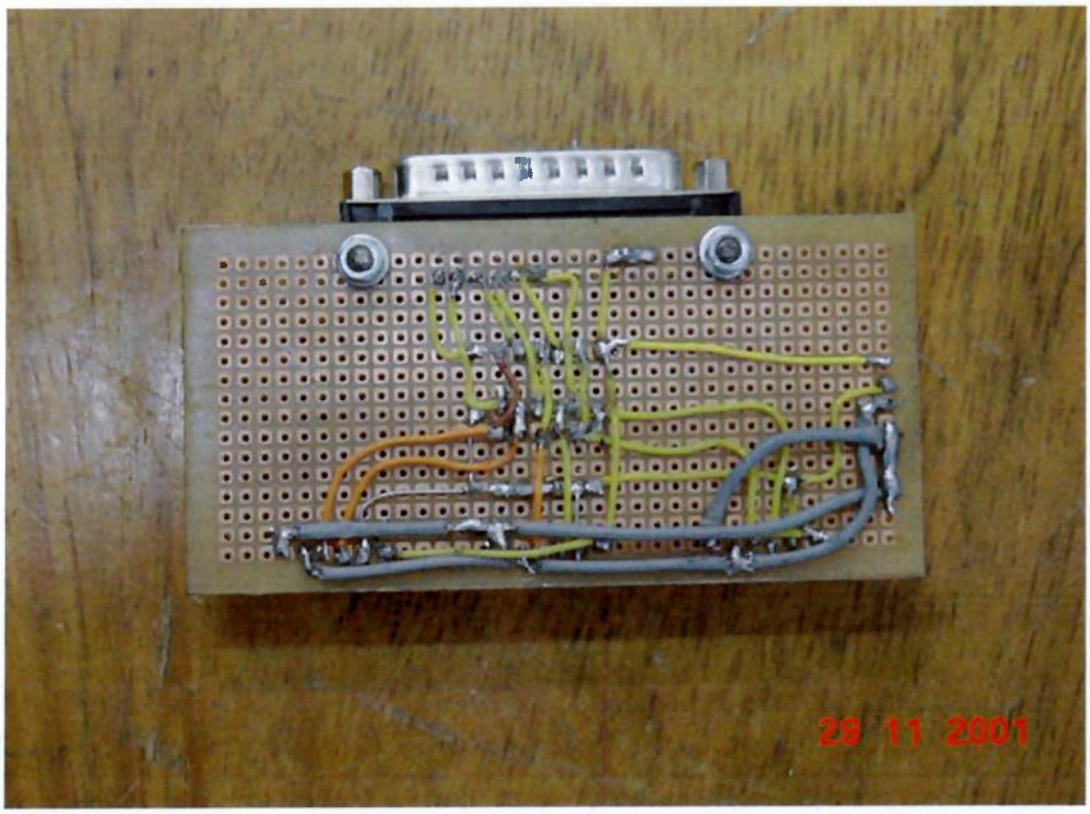


Foto 18 – Vista inferior da placa de comunicação com o PC

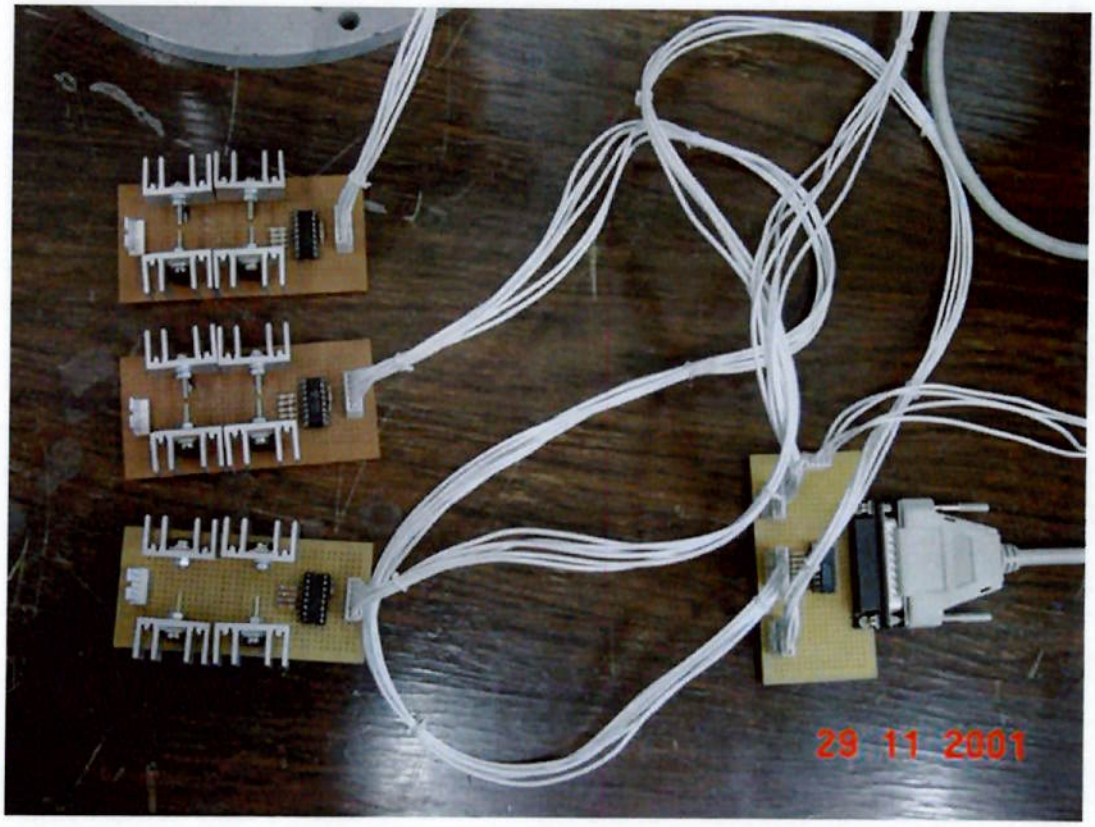


Foto 19 – Esquema elétrico mostrando a ligação dos elementos eletrônicos

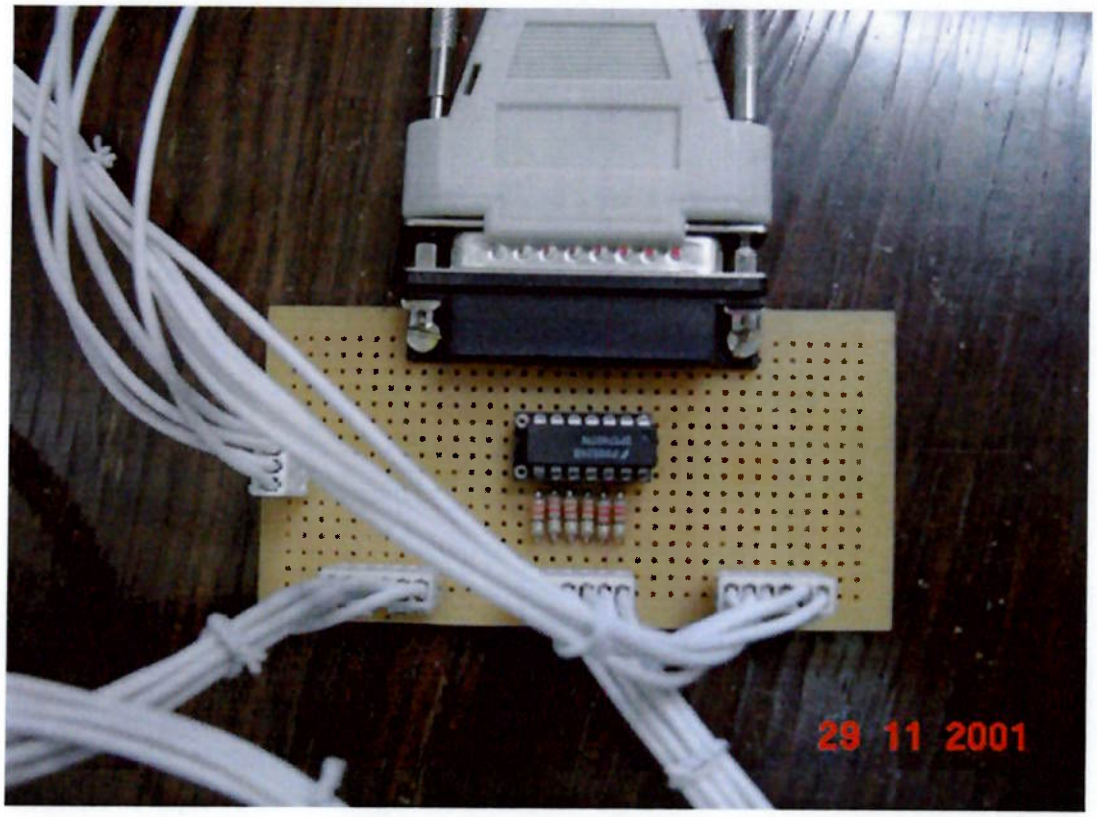


Foto 20 – Placa de comunicação com o PC com as ligações

Foto 21 – Drivers de motor de passo com as ligações

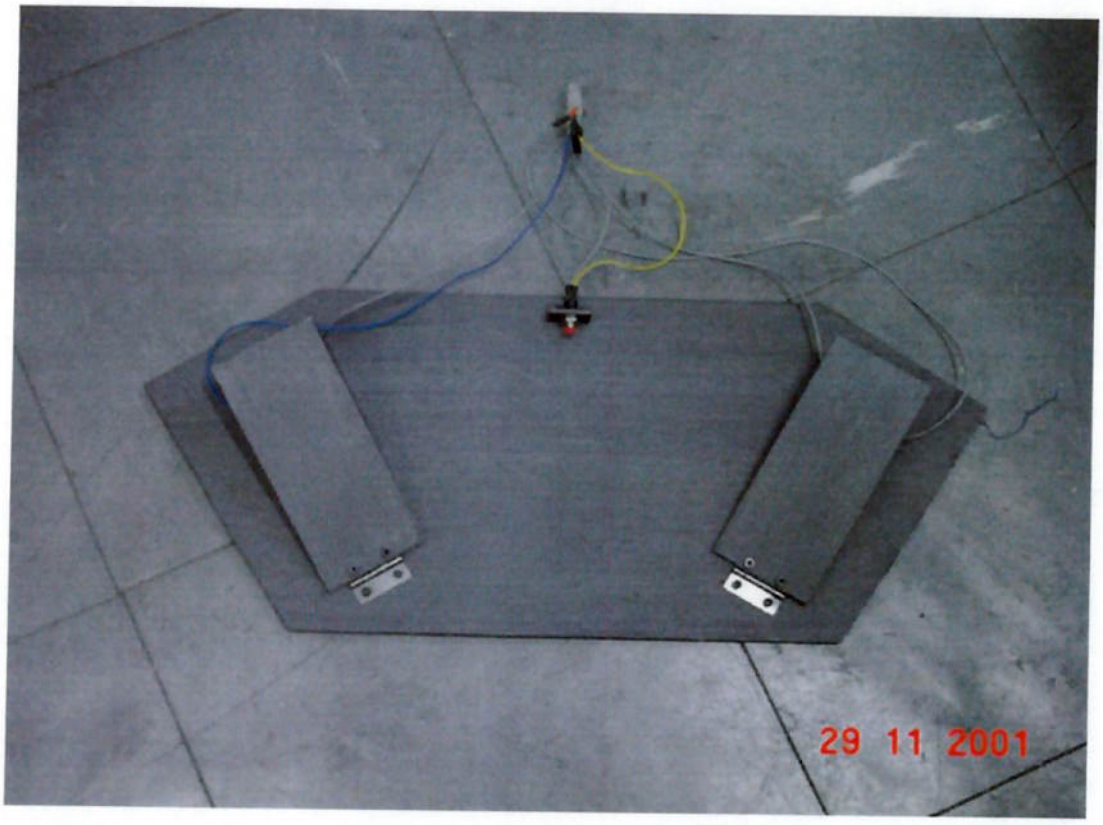


Foto 22 – Sistema de pedais

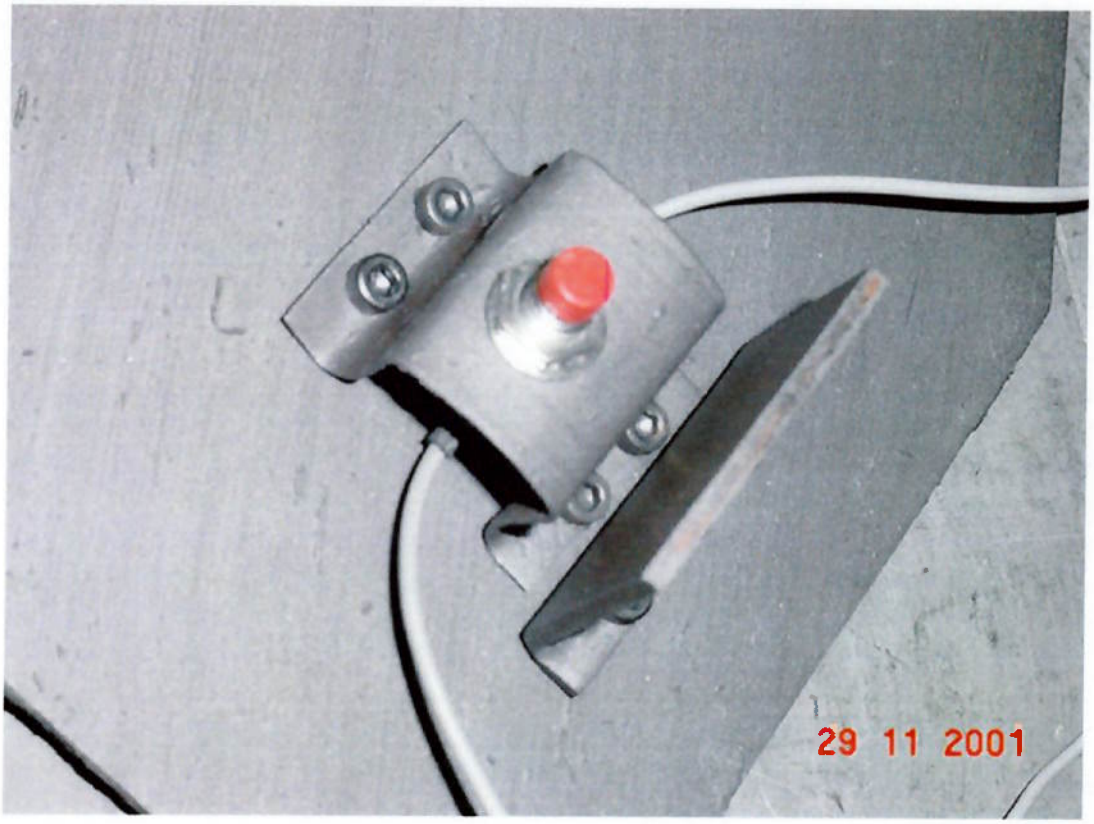


Foto 23 – Detalhe de um botão do sistema de pedais

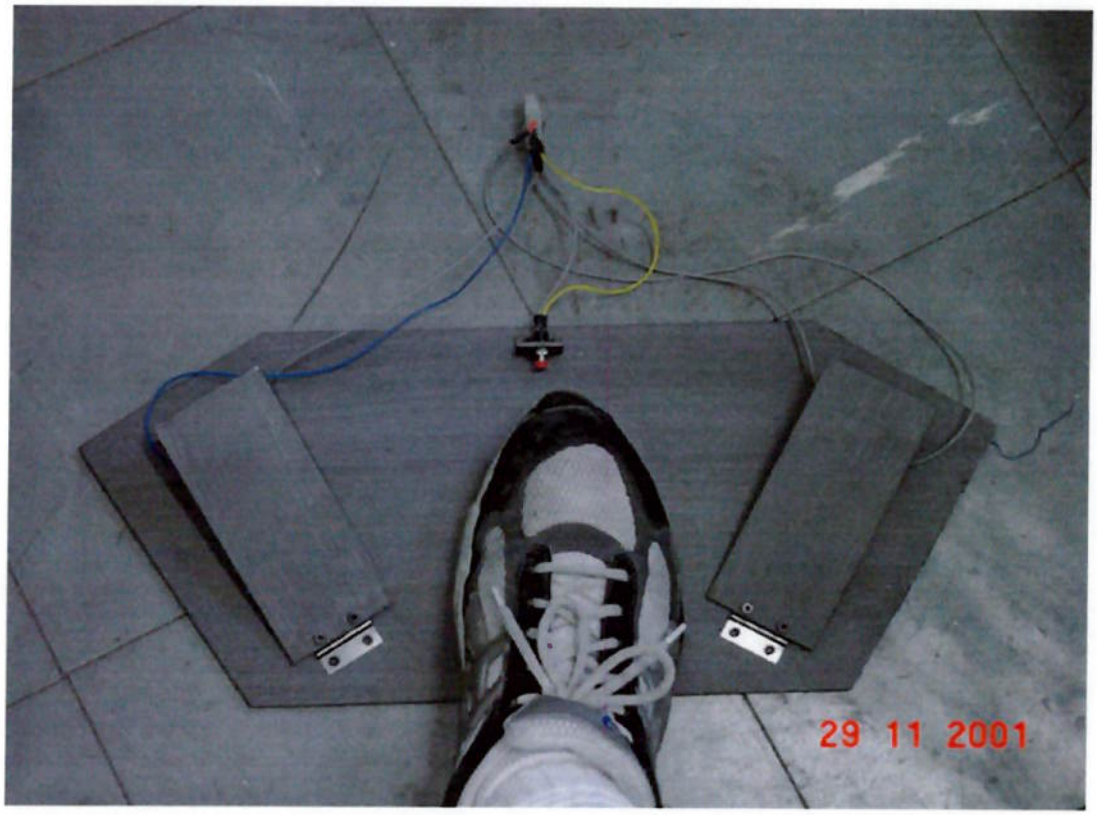


Foto 24 – Sistema de pedais, mostrando tamanho relativo a um pé

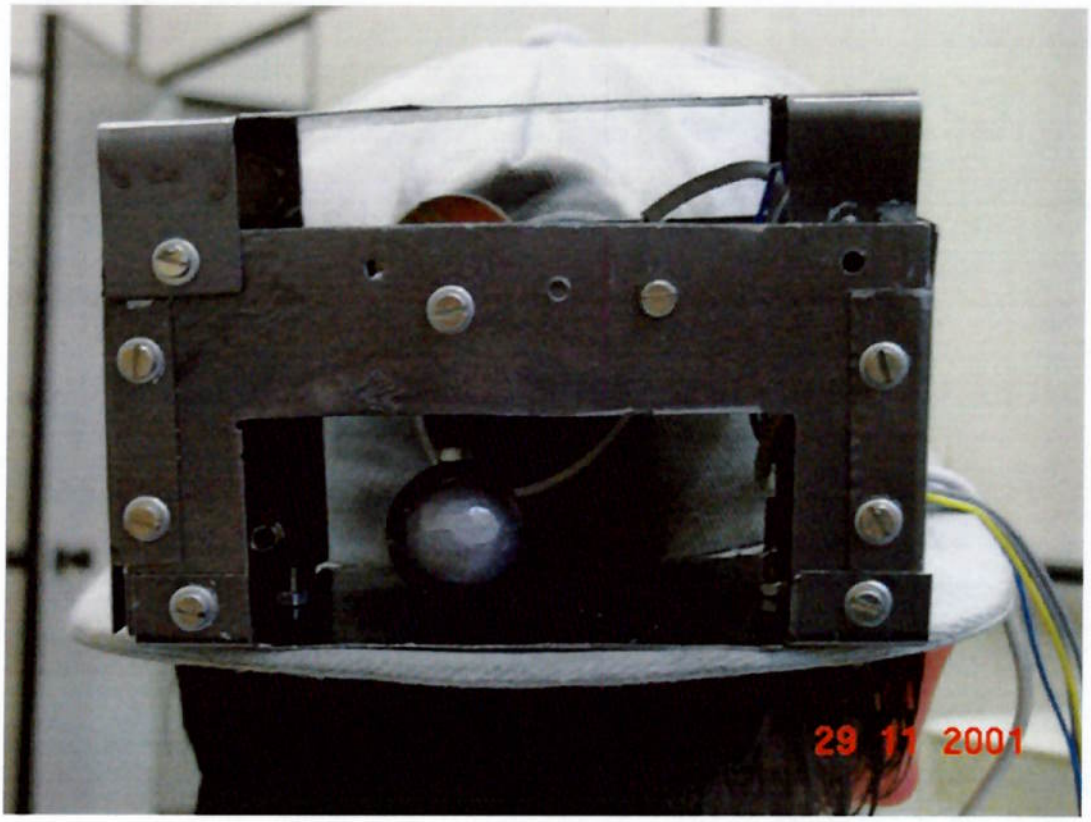


Figura 25 – Vista frontal do mouse de cabeça

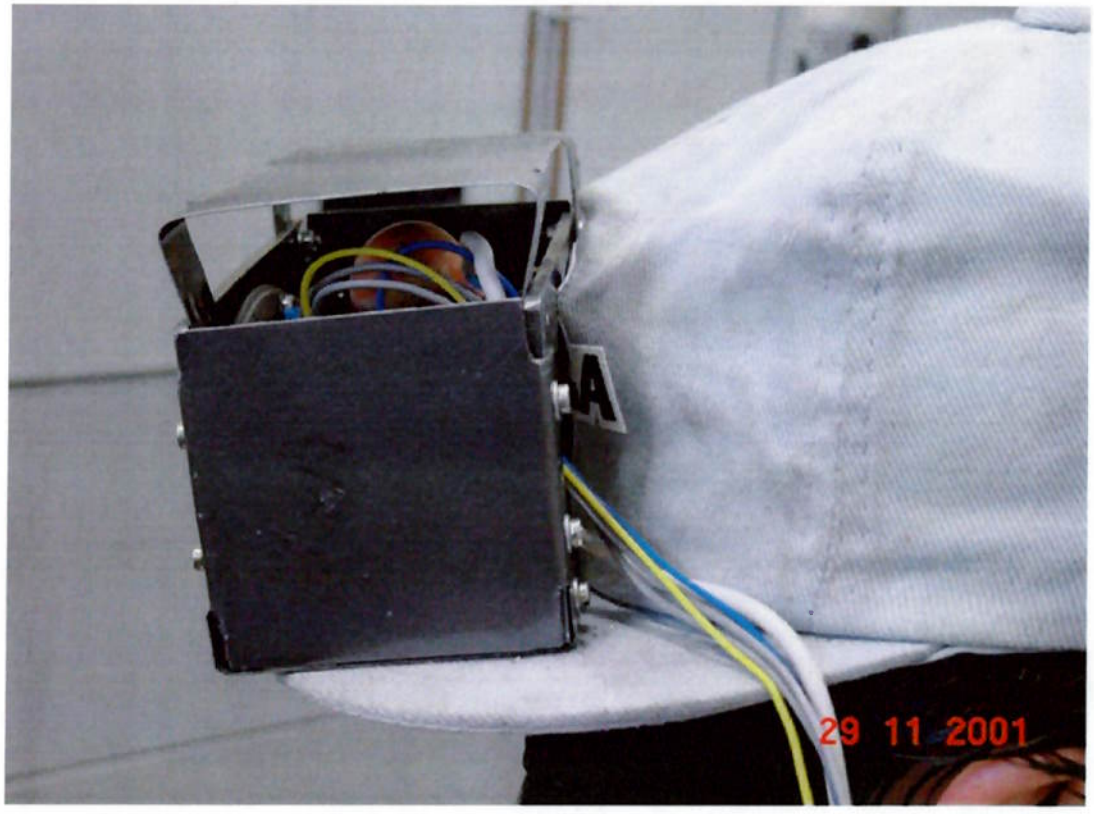


Foto 26 – Vista lateral do mouse de cabeça

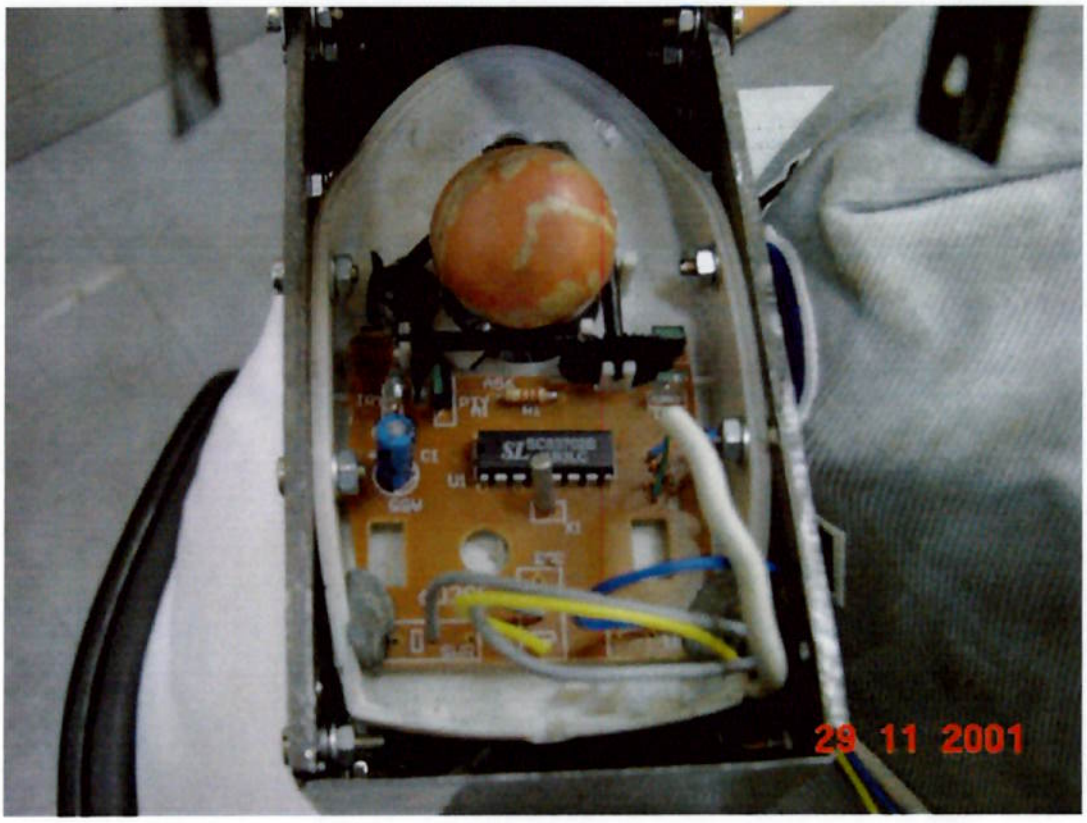


Foto 27 – Detalhe do funcionamento interno do mouse de cabeça

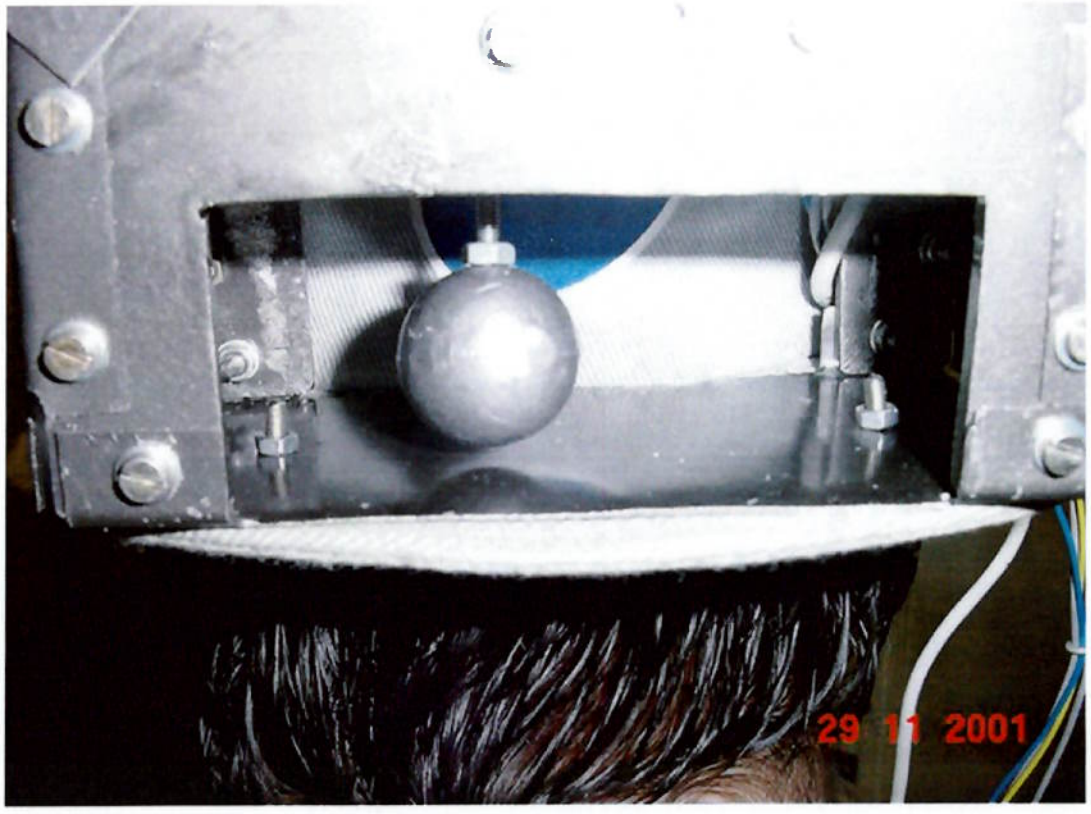


Foto 28 – Detalhe do pêndulo do mouse de cabeça

Foto 29 – Fonte utilizada para suprir os sistemas elétricos (fixa na base)