

Sys 1965849



Universidade de São Paulo
Escola Politécnica
Departamento de Engenharia Mecânica

10 (dez)
km

Defletor de Feixes com Dois Graus de Liberdade

**Trabalho de formatura apresentado
à Escola Politécnica da Universidade
de São Paulo para obtenção do título
de Graduação em Engenharia.**

**elaborado por
Michel Friedhofer**

**orientador:
Oswaldo Horikawa**

**área de concentração:
Engenharia Mecânica
Automação e Sistemas**

São Paulo

- 1999 -

Friedhofer, Michel

Defletor de feixes com dois graus de liberdade, por M. Friedhofer. São Paulo, EPUSP, 1999. 131p.

Trabalho de Formatura. Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica.

1. Motores de passo. 2. Laser. 3. Estereolitografia. 4. Microstepping. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecânica. II.t.

"All these moments will be lost in time, like tears in rain"

- Rutger Hauer, como "Roy Batty" (Blade Runner)

AGRADECIMENTOS

A concepção e a implementação deste trabalho contaram com o apoio indispensável de diversas pessoas e entidades, manifestados em diferentes formas - todas igualmente apreciadas. No esforço de expressar minha mais sincera gratidão listo alguns dos nomes que me vêm à mente, e pela minha memória eventualmente falha, peço desculpas aos não mencionados.

Meus agradecimentos vão a Mário Sérgio (G.D.E. Inc. do Brasil), Kiko e Koji Kawasaki (NMB Corporation), Roberto Carlos Pieruci (National Semiconductors do Brasil), e Luiz Mendes (Analog Devices), pela ajuda prestada na obtenção de componentes essenciais; a Aparecida Tomoko e a Flavio Beni Duobles (Etelbras Eletrônica e Telecomunicações S.A.), pelo *layout* e confecção das placas de circuito impresso; a Will Roberts (Creative Technologies Inc.), pela ajuda a distância; a Jorge Szanto de Toledo (Advanced Manufacturing Systems Ltda.), pelo patrocínio injustificado; a Carlos do Prado Neto, que partilhou de sua experiência, empolgação e de seus copos de tequila de Isla Margarita; a Camilla Del Soldato Spínola, que da maneira mais doce fingiu interessar-se por todo o projeto; ao prof. Marcelo Massarani, por sua compreensão em momentos complicados, e pela valiosíssima lição de "menos iniciativa, mais acabativa"; ao prof. Marcelo Godoy Simões, que de um jeito meio atravancado contribuiu imensamente para meu domínio e interesse na Eletrônica; ao prof. Lucas Antônio Moscato, pelo apoio e compreensão, e sobretudo pelo respeito; ao prof. Oswaldo Horikawa, pela orientação, que certamente foi menor do que ele gostaria que fosse, e pela liberdade, que certamente foi maior do que ele gostaria que fosse; e finalmente, a Antonio Fernando Maiorquim, Maurício Lima e Adilson Fernandes Felix Filho, por toda a ajuda prestada - sem vocês, nada disto seria possível.

Em um âmbito menos técnico, a conclusão deste trabalho é símbolo de uma conquista alcançada depois de anos, duros anos, de batalha. Não foi fácil. Possivelmente, algumas pessoas não sabiam o que estavam fazendo quando me encorajaram e me apoiaram ao longo deste curso, do começo ao fim. Estas pessoas, certamente, também não sabem o quão imensamente grato lhes sou, agora que tudo chega ao fim. Pessoas como meu sempre amigo Tiago do Prado Barizon, que já não faz esforços para ver-se livre de mim, por saber que não vai conseguir; Pierluigi Piazzi e Renato da Silva Oliveira, talvez os grandes responsáveis pela escolha que fiz em 1994, e certamente minhas maiores fontes de inspiração (Pier e Renato: quem será o quarto?); minha amiga Ana Luiza Cesarino Junqueira, com quem fui imperdoavelmente negligente nestes anos complicados, mas que ainda assim sempre esteve pronta e disposta a me acolher e ajudar; Carlos Rogério Duarte Barreiros, meu alter-ego lírico, onírico e satírico; Karina Funcia Murgel, pelo passado, pelas passadas, pelos passados, pelos presentes e pelos ausentes, quiçá pelo futuro; os colegas da turma da Mecatrônica/95, que sabem melhor que quaisquer outras pessoas a alegria que sinto ao escrever estas palavras; meus familiares, por tudo que fizeram e representaram nestes últimos cinco - não, vinte e dois - anos; e finalmente, minha mãe, a quem não sei como agradecer - tudo isso aconteceu graças a você, todo o resto aconteceu graças a você, e acho que qualquer agradecimento, escrito ou não, jamais será suficiente.

APOIOS



National Semiconductor - (11) 3043-7378

Analog Devices - (11) 290-3433

NMB Corporation - (11) 577-5411

Atmel Corporation - (11) 5089-8090

Etelbras S.A. - (11) 274-3144

AMS Ltda. - (11) 867-0372

Índice

1. INTRODUÇÃO	1
2. ESCOPO DO TRABALHO	3
3. PARÂMETROS DE PROJETO	5
3.1. INTERFACE DE ENTRADA	5
3.2. VELOCIDADE DE VARREDURA	6
3.3. RESOLUÇÃO	7
3.4. MÉTODO DE VARREDURA	8
3.5. CONTROLE DE POTÊNCIA DO FEIXE	10
3.6. COMPENSAÇÃO DE DISTORÇÕES ÓTICAS	11
3.7. CUSTO	12
3.8. PARÂMETROS DE PROJETO: RESUMO	13
4. SOLUÇÕES PROPOSTAS	14
4.1. ALTERNATIVA A: DEFLEXÃO CARTESIANA COM ESPELHOS FIXOS A 45°	15
4.2. ALTERNATIVA B: DEFLEXÃO ESFÉRICA COM MICROSTEPPING	16
4.3. ALTERNATIVA C: DEFLEXÃO ESFÉRICA COM REDUÇÃO SEM HISTERESE	18
5. CRITÉRIOS DE SELEÇÃO	20
5.1. LINEARIDADE	20
5.2. VELOCIDADE	20
5.3. PRECISÃO/HISTERESE	21
5.4. SIMPLICIDADE	21
5.5. CUSTO	22

6. CONSIDERAÇÕES SOBRE O ASPECT RATIO	23
7. TESTES DE LABORATÓRIO: MICROSTEPPING	26
8. PRÉ-PROJETO DO SISTEMA ELETRÔNICO	36
8.1. CONTROLE DE CORRENTE DOS ENROLAMENTOS	37
8.2. CONVERSORES D/A	40
8.3. INTERFACE RS-232	41
8.4. CONTROLE DE LUMINOSIDADE DO LASER	41
8.5. MICROCONTROLADOR	42
8.6. ALIMENTAÇÃO	43
8.7. ORÇAMENTO PRELIMINAR	44
9. PRÉ-PROJETO DO SISTEMA MECÂNICO	45
9.1. SUPORTES DOS ESPELHOS	46
9.2. CONJUNTO ÓTICO	47
9.3. ESTRUTURA	48
10. PRÉ-PROJETO DE FIRMWARE	49
10.1. COMUNICAÇÃO SERIAL	50
10.2. MOVIMENTOS BÁSICOS DO MOTOR	51
10.3. ALGORITMO DE TRAÇAGEM DE LINHAS	54
10.4. CONTROLE DE INTENSIDADE DO FEIXE	60
10.5. INTERPRETAÇÃO DE COMANDOS	61
11. PROJETO MECÂNICO	62
11.1. CORREÇÃO DE FOCO	62
12. PROJETO ELETRÔNICO	66
12.1. SENSOR DE HOME	66
12.2. LAYOUT DA PCI	67

13. FIRMWARE	68
13.1. CONTROLE DE VELOCIDADE	69
13.2. PROTOCOLO DE COMUNICAÇÃO	70
13.3. ALGORITMO DE HOMING	74
13.4. ALGORITMO DE POWER DOWN	75
14. CUSTOS FINAIS	76
15. RESULTADOS	78
15.1. FOTOS	78
15.2. PROJEÇÕES	79
16. REFERÊNCIAS BIBLIOGRÁFICAS	81
17. APÊNDICES	81
APÊNDICE I: DATA SHEETS	82
APÊNDICE II: DESENHOS	88
APÊNDICE III: CABOS	101
APÊNDICE IV: LISTAGEM DO <i>FIRMWARE</i>	103
APÊNDICE V: EXEMPLOS DE PROGRAMAS	130

1. INTRODUÇÃO

A luz de um feixe *laser* tem a importante propriedade de ser colimado - i.e., o feixe é concentrado em uma única direção de propagação. Com isso, é possível concentrar uma grande quantidade de energia em um espaço relativamente pequeno. Esta característica abre portas para aplicações interessantes, tais como a prototipagem rápida por estereolitografia: um protótipo de algum sólido é construído camada por camada, através da polimerização de uma



Fig. 1.1 - Exemplo de resultado conseguido pela técnica da estereolitografia. O modelo da esquerda foi digitalizado e reconstruído por camadas, resultando no modelo da direita

resina ou pó quando localmente atingido por um feixe de comprimento de onda específico. Outro exemplo de aplicação é a gravação de mensagens e desenhos em diversos substratos. Por trás destes e de outros exemplos está um sistema que consiga defletir o feixe, de modo a projetar um desenho qualquer num plano apropriado, a partir de uma

única fonte de *laser*. É importante que este sistema possa também controlar a velocidade linear do feixe, para garantir a uniformidade da profundidade de polimerização/gravação.

Tomando as aplicações de prototipagem rápida como contexto, é preciso criar-se a habilidade de desenhar secções transversais de um sólido qualquer num plano de projeção - plano este onde se encontra a superfície livre de um tanque de resina ou pó. À medida que o feixe de *laser* percorre o desenho desta secção transversal, o material é solidificado. Repetindo-se o processo em camadas sucessivas, constrói-se um sólido arbitrariamente complexo.

Diversas são as abordagens adotadas para conseguir criar o movimento do feixe sobre o plano de projeção. Contudo, percebe-se que há a dominância de duas concepções sobre todas as outras: a que aqui chamaremos de *deflexão esférica do feixe*, onde um par de espelhos articulados em eixos reversos fornecem os dois graus de liberdade necessários para a formação do desenho, de forma que o movimento do feixe obedece uma geometria igual à de um sistema de coordenadas esféricas, e aquela que chamaremos de *deflexão cartesiana*, construtivamente similar a uma *plotter* convencional, de forma que o movimento do feixe obedece uma geometria igual à de um sistema de coordenadas cartesianas.

As aplicações da prototipagem rápida são limitadas por três fatores principais: tempo de construção dos protótipos, precisão

dimensional e custo final. O tempo de construção depende diretamente da complexidade do modelo, da velocidade do sistema defletor do feixe e do tempo de cura ou sinterização da resina ou pó utilizado. A precisão dimensional é limitada pela resolução máxima que o sistema defletor fornece, pelo diâmetro mínimo do feixe e pelas características de cura do material (incluindo sua imunidade à distorção em virtude da polimerização de regiões vizinhas). Finalmente, o custo é naturalmente influenciado pelo material empregado na construção do modelo e pelo preço do equipamento em si. É fácil perceber que o sistema de deflexão do feixe é um fator crítico no desempenho final de um sistema de prototipagem rápida, tendo influência direta em todas as suas limitações.

2. ESCOPO DO TRABALHO

A construção de um sistema completo de prototipagem rápida por estereolitografia constitui empreitada multidisciplinar e extremamente complexa, além de cara. Por outro lado, o projeto de um sistema de baixa resolução e baixo custo encontra justificativa não apenas no aspecto didático, mas também na possibilidade de aplicação real - sistemas comerciais, de altíssima precisão, são extremamente caros.

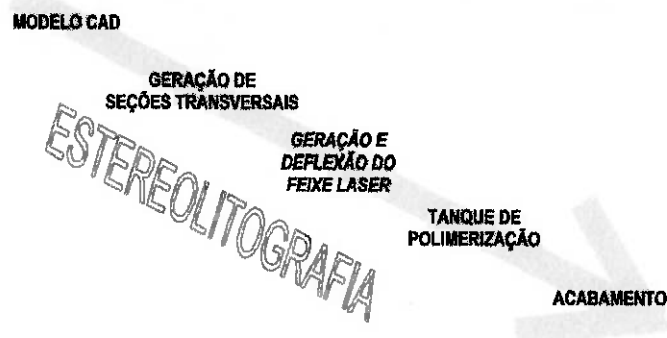


Fig. 2.1 - O escopo do trabalho se restringe ao sistema de deflexão do feixe *laser*

Entretanto, o tempo previsto para a conclusão do projeto não permite que se aborde todos os subsistemas envolvidos. Desta forma, optou-se por restringir o escopo do trabalho ao projeto e construção do sistema de deflexão do feixe, atendendo especificações de projeto correspondentes a uma configuração de baixas resolução e velocidade, porém de baixíssimo custo quando comparado a sistemas comerciais.

Um diagrama de blocos simplificado do sistema proposto é apresentado na figura 2.2. O conjunto é alimentado por uma fonte externa, conforme mostrado. As informações são trocadas por meio de uma interface padrão RS-232, de acordo com protocolo a ser definido. A unidade controla diretamente dois motores de passo, responsáveis pela movimentação dos espelhos que fornecerão os dois graus de liberdade ao feixe de *laser*. Este último é interrompido por uma interface que funcionalmente é igual a um relé. Contudo, para baixas potências, pode-se adotar um interruptor baseado em transistor, que apresenta a vantagem adicional de poder operar em altas velocidades (permitindo, por exemplo, um ajuste da potência do feixe por modulação PWM).

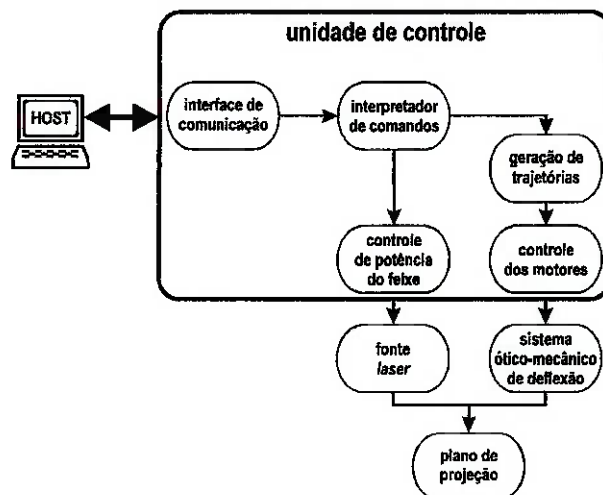


Fig. 2.2 - Diagrama de blocos simplificado

3. PARÂMETROS DE PROJETO

Conforme dito anteriormente, pretende-se projetar e construir um subsistema de deflexão de um feixe *laser*, com 2 graus de liberdade, que esteja inserido no contexto construtivo de um sistema de estereolitografia de baixos custo, resolução e velocidade. Neste sentido, e em comparação aos sistemas comerciais já estabelecidos no mercado, é conveniente caracterizarmos o desempenho do sistema proposto em alguns de seus aspectos mais relevantes.

3.1. Interface de entrada

Originalmente introduzido pela 3D Systems, Inc. em 1988, o formato de arquivos STL é hoje o padrão estabelecido entre todas as máquinas de prototipagem rápida para transferência de dados sobre os

modelos tridimensionais dos sólidos a serem construídos. O formato STL representa o sólido discretizando-o em elementos triangulares de superfície, além de conter outras informações relevantes. Contudo, novamente em virtude da necessidade de adequar-se o escopo do projeto à disponibilidade de recursos e de tempo, não será implementada uma interface STL no sistema a ser projetado.

Ao invés disso, pretende-se implementar um protocolo gráfico bidimensional, similar ao padrão HPGL criado pela *Hewlett Packard*, que represente de forma extremamente simples figuras bidimensionais vetorizadas ou *raster* (vide item 3.4). O desenvolvimento de um algoritmo de conversão de arquivos STL para o protocolo proprietário a ser implementado fica registrado como idéia para projeto de alguém que se interesse.

No nível físico, implementar-se-á uma interface serial assíncrona padrão RS-232, com parâmetros (*baud rate*, paridade, etc.) a serem estabelecidos *a posteriori*, de modo a permitir a comunicação direta com o equipamento *host*.

3.2. Velocidade de varredura

Quanto maior a velocidade máxima de varredura do feixe sobre o plano de projeção, mais rápida é a construção do modelo, caracterizando um melhor desempenho do sistema. Contudo, é preciso levar em conta a relação entre a potência do feixe e o tempo de cura do

material, com cuidado para não provocar uma cura pobre em função de uma velocidade muito alta.

fabricante	velocidade (m/s)
3D Systems	5
CMET	5
Teijin Seiki	15
EOS	2 - 2.5
DTM	2

Tab. 3.1 - Velocidades de varredura de sistemas comerciais

A título de exemplo, a Tabela 3.1 mostra algumas das velocidades de varredura atingidas por sistemas comerciais de prototipagem rápida. A velocidade de projeto para o sistema a ser desenvolvido será fixada, *a priori*, em 0.5 m/s - fazendo-se a ressalva de que este valor será adotado como mínimo aceitável, podendo ser reavaliado em função da capacidade do sistema final.

3.3. Resolução

A resolução máxima alcançada por um sistema de estereolitografia depende do diâmetro do feixe *laser* e do tamanho do passo alcançado pelo sistema de espelhos. Naturalmente, quanto maior a resolução do sistema, maior será a precisão dimensional do modelo construído. Como exemplo, a Tabela 3.2 fornece valores tanto para o diâmetro do feixe quanto para o passo de alguns sistemas comerciais.

fabricante	passo (μm)	diâmetro do feixe (μm)
3D Systems	50	80 - 250
CMET	50	130 - 1500
Teijin Seiki	12.5	150
EOS	50 - 100	200
DTM	50	400

Tab. 3.2 - Características de resolução de alguns sistemas comerciais

Adotaremos valores significativamente maiores para o passo e para o diâmetro do feixe, tendo em vista o enfoque do projeto. Será usado um feixe de *laser* vermelho (650 nm) com diâmetro de aproximadamente 3 mm, apenas a título de demonstração. Adicionalmente, adotaremos 1 mm como passo mínimo admissível. Vale lembrar que este passo deve ser medido no plano de projeção, que estipularemos como estando a uma distância aproximada de 1 m do conjunto ótico. O sistema será projetado prevendo a substituição do feixe *laser*, de modo que se possa usar outros comprimentos de onda e/ou outros diâmetros de feixe.

3.4. Método de varredura

Existem duas abordagens para a reprodução de curvas bidimensionais. A primeira é a formação de imagens *vetoriais*, onde o feixe percorre a curva de maneira suave e contínua, com movimentos conjugados dos dois espelhos. A segunda é a formação da imagem por meio de sua discretização em pontos, onde cada ponto é acessado

individualmente. Esta última abordagem é conhecida como *raster*. Em geral, imagens vetoriais fornecem resultados melhores pois minimizam a aparência descontínua de imagens rasterizadas. No entanto, o preenchimento de contornos geralmente exige um comportamento similar ao método *raster*, percorrendo-se o total da área interna de um contorno com uma sequência de segmentos de reta paralelos a algum dos eixos principais de movimentação do sistema.

A grosso modo, imagens *raster* são concluídas em um tempo menor tendo em vista a uniformidade do movimento de varredura: não é preciso computar trajetórias complexas, sendo suficiente “ligar” e “desligar” o feixe nos pontos adequados da varredura. Por outro lado, na abordagem *raster* é preciso cobrir-se toda a área projetável para cada desenho, o que em geral consome um tempo muito grande.

Uma característica importante de *todos* os sistemas de estereolitografia feitos até hoje é que mesmo que se adote uma abordagem vetorial na reprodução das secções transversais, o resultado ao longo do eixo perpendicular ao plano de projeção é discretizado (*raster*). Isto acontece porque o modelo é construído camada por camada, e cada uma destas camadas tem uma espessura mensurável.

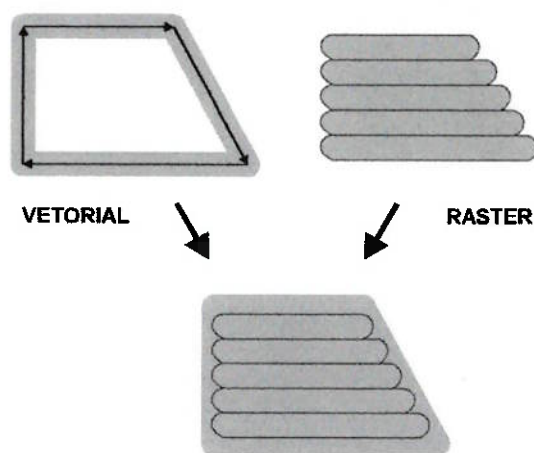


Fig. 3.1 - Métodos de varredura e sua aplicação conjunta

No sistema a ser implementado, estipula-se a capacidade de, no mínimo, traçar-se segmentos de reta em qualquer direção, vetorialmente. Desta forma, qualquer contorno pode ser aproximado por segmentos de reta e o resultado pode ser desenhado vetorialmente. Adicionalmente, prevê-se também a abordagem *raster*, tanto para preenchimentos como para desenhos em geral. Havendo a possibilidade de ampliar o protocolo de comunicação e as rotinas de geração de trajetórias ao final da implementação do sistema básico, pretende-se incluir comandos para a traçagem de outros elementos fundamentais além dos segmentos de reta, tais como arcos de circunferência.

3.5. Controle de potência do feixe

A profundidade de cura e, de maneira mais genérica, a área do material afetada pelo feixe dependem das características do próprio material, da geometria do feixe, da velocidade de varredura e da

potência nominal do feixe. Considerando-se o possível emprego do sistema a ser desenvolvido em uma aplicação com profundidade de cura variável (o que é hoje uma tendência global), é interessante criar um controle da potência do feixe.

O controle de potência também é empregado num ambiente de malha fechada em sistemas comerciais, variando em função da velocidade real do feixe. Nos trechos de aceleração e desaceleração da varredura, a potência do feixe é reduzida proporcionalmente de modo a compensar as velocidades menores inerentes a estas fases do movimento. Este controle de potência, entretanto, não será implementado no sistema a ser desenvolvido, tendo em vista que o sistema operará, *a priori*, em malha aberta.

Desta forma, o protocolo gráfico deverá prever comandos de controle da potência do feixe, de 0% a 100%, com uma resolução de 8 bits. O controle será efetivamente implementado através de um interruptor eletrônico em série com a alimentação do emissor *laser*, operando em PWM (*pulse width modulation*).

3.6. Compensação de distorções óticas

Tanto na geometria cartesiana quanto na geometria esférica (vide item 1), o comprimento total do caminho ótico percorrido pelo feixe varia sensivelmente à medida que o feixe percorre toda a área útil de projeção. Logo, para que o feixe esteja sempre perfeitamente focalizado

no ponto onde incide no plano de projeção, é preciso que a distância focal deste feixe seja controlada dinamicamente. Adicionalmente, na geometria esférica o próprio ângulo de incidência do feixe varia, fazendo com que a secção transversal resultante do feixe sobre o plano de projeção sofra distorções em posições extremas da área útil.

Logo, deve-se prever um conjunto ótico projetado para anular estas distorções, para garantir uma uniformidade do feixe *laser* sobre toda a área de projeção. Contudo, o projeto de um tal conjunto ótico é complexo e, principalmente, sua implementação é extremamente cara. Paralelamente, para um sistema em caráter demonstrativo e didático, estas distorções são plenamente admissíveis. Desta forma, *nenhuma medida* será adotada para compensar as distorções óticas do sistema.

3.7. Custo

Sistemas comerciais de estereolitografia a *laser* são extremamente caros - e como acontece em casos semelhantes, configura-se com isso um mercado de *bureaus* de serviço ao invés de vendas substanciais de equipamentos para um mercado mais significativo. Estes equipamentos são, afinal, produtos *high end* e de altíssima tecnologia, além de terem um volume de produção reduzido - fatores que contribuem para um alto preço de venda. Estes preços situam-se na casa das centenas de milhares de dólares, variando bastante em função das características de cada equipamento. É preciso

lembrar também o custo indissociável dos materiais de consumo para a operação de tais máquinas, tais como as resinas empregadas na fabricação dos modelos.

O subsistema defletor representa uma parcela significativa deste custo, tendo em vista que este é um ponto concentrador de alta tecnologia, sem mencionar o conjunto ótico associado (caro como todos os produtos óticos de alta qualidade são). Considerando todo o enfoque que está sendo adotado para o resto do projeto, estipula-se o valor máximo de US\$ 1000 (mil dólares americanos) para o equipamento a ser construído, considerando-se apenas o custo de aquisição das peças e eventuais serviços associados (confeção de placas de circuito impresso, serviços de usinagem, etc.). Prevendo-se a possibilidade de serem testadas diferentes abordagens, de modificações surgirem ao longo do desenvolvimento do projeto e de eventuais ampliações dos parâmetros ora estipulados, determina-se que este custo máximo será referido ao sistema *as built*, em sua configuração mínima.

3.8. Parâmetros de projeto: resumo

Abaixo encontram-se de forma resumida todos os parâmetros de projeto estipulados *a priori*. Conforme prega a metodologia da espiral de projeto, reserva-se a possibilidade de reavaliar estas especificações ao longo do desenvolvimento do sistema - as especificações a seguir são diretrizes, não limites.

descrição	subsistema defletor de feixes com 2 graus de liberdade, com controle de potência do feixe, para aplicação num sistema de estereolitografia a laser de baixo custo
geometria	cartesiana ou esférica
interface de comunicação	serial assíncrona padrão RS-232
protocolo de comunicação	protocolo proprietário a ser estabelecido posteriormente
fonte laser	650 nm, 5 mW, diâmetro 3 mm (<i>laser pointer</i>), substituível por outro qualquer
velocidade de varredura	50 cm/s máx. (medido no plano de projeção)
resolução	1 ponto/mm (medido no plano de projeção) total: 500 pontos em cada direção (X e Y)
método de varredura	vetorial p/ segmentos de reta e <i>raster</i>
controle de potência do feixe	interruptor eletrônico com PWM de 8 bits
compensação de distorções óticas	-
custo	US\$ 1000 máx. (<i>as built</i> , configuração mínima)

4. SOLUÇÕES PROPOSTAS

De acordo com as técnicas de metodologia do projeto, é conveniente conceber diferentes abordagens para a resolução do problema. No caso em questão, procurar-se-á aplicar este método sempre que possível e/ou necessário. Contudo, de imediato é possível identificar que a geometria dos espelhos constitui o principal aspecto do sistema - desta forma, a primeira escolha a ser feita é qual a geometria a ser adotada. Para tanto, apresentar-se-á a seguir diferentes concepções estruturais que resolvem o problema, concepções estas que serão posteriormente avaliadas de modo a escolher-se e adotar-se uma única.

4.1. Alternativa A: Deflexão cartesiana com espelhos fixos a 45°

Cada um dos dois motores de passo aciona uma correia com um carro. Nos carros estão montados os espelhos, a 45°, conforme mostra o desenho. No esquema mostrado, as duas correias são fixas, de modo que o segundo espelho precisa abranger todo o percurso do feixe defletido pelo primeiro espelho.

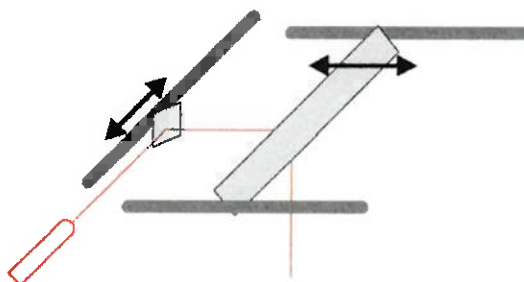


Fig. 4.1 - Alternativa A

As principais vantagens são as seguintes:

- pelo fato do movimento dos espelhos ser linear, os espelhos empregados podem ser comuns;
- é possível fazer um controle em malha fechada, já que a resolução exigida do *encoder* é relativamente baixa (apesar de não ser necessário fechar a malha nesta configuração);
- o sistema tem uma resposta linear, tendo em vista que o movimento dos espelhos é retilíneo.

As principais desvantagens são:

- a estrutura final é relativamente grande, abrangendo toda a área de projeção (i.e., o sistema ocupará um volume correspondente a toda a área de projeção, com uma certa altura;
- o segundo espelho deverá ter um comprimento mínimo de 50 cm, sendo relativamente grande;
- a montagem do segundo espelho é particularmente sensível a erros de alinhamento, gerando erros no posicionamento final do feixe;
- serão necessárias correias também relativamente grandes (certamente mais de 1 m de comprimento);
- a mecânica empregada é complicada (comparada às outras alternativas), encarecendo o projeto;
- o sistema terá uma velocidade final baixa, talvez insuficiente para atender o parâmetro de projeto.

4.2. Alternativa B: Deflexão esférica com *microstepping*

Os espelhos são montados diretamente sobre os eixos dos motores. A deflexão do feixe obedece uma geometria igual a um sistema de coordenadas esféricas. Por este motivo, é necessário empregar-se espelhos de primeira superfície, de modo a garantir que o eixo de

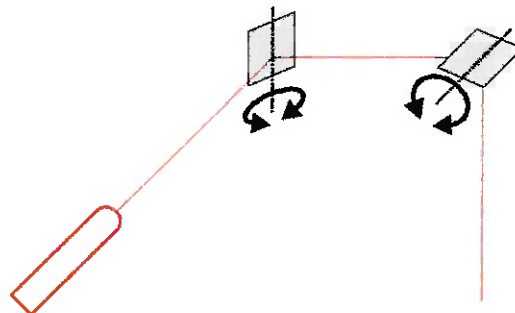


Fig. 4.2 - Alternativa B

rotação do espelho esteja contido no plano de reflexão. Os motores de passo são controlados usando *microstepping*, de modo a conseguir a resolução de posicionamento necessária (50 cm de abertura a 1 m de distância significa uma abertura do feixe de cerca de 28° , ou seja, um ângulo útil de rotação do espelho de cerca de 14° ; logo, o motor deverá ter aproximadamente 13000 passos por volta).

As principais vantagens são:

- o sistema como um todo ocupa um volume bastante reduzido;
- a velocidade final é bastante alta;
- o custo do sistema é bastante reduzido, tendo em vista que a mecânica exigida é muito simples;
- não existem perdas mecânicas ou fontes de histerese;
- é possível anular as não-linearidades inerentes à geometria esférica no próprio algoritmo de controle *microstepping*.

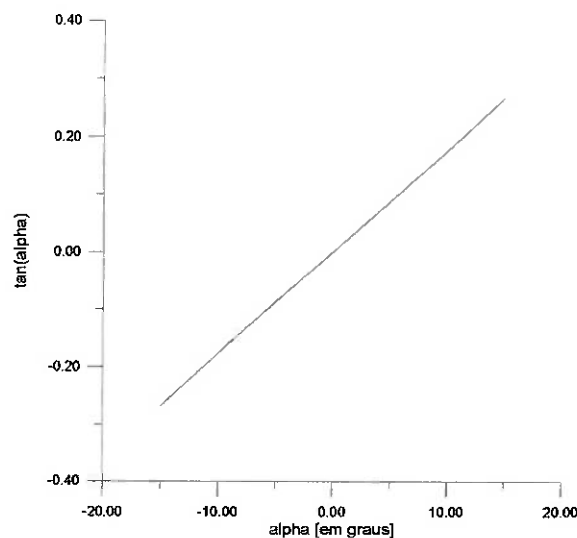


Fig. 4.3 - Não-linearidade de projeção na região de interesse (-15° a $+15^\circ$)

As principais desvantagens são:

- algoritmo para controle *microstepping* é relativamente complexo;
- há não-linearidades intrínsecas ao sistema, devidas à geometria esférica de deflexão do feixe - contudo, dentro da faixa útil de rotação dos espelhos, estas não-linearidades são pouco expressivas (vide fig. 4.3);
- os espelhos empregados deverão ser tais que permitam uma montagem que garanta que o eixo de rotação passe pelo plano de reflexão - ou seja, devem ser empregados espelhos de primeira superfície;
- a validade do controle *microstepping*, no enfoque e âmbito deste projeto, ainda precisa ser estabelecida em laboratório.

4.3. Alternativa C: Deflexão esférica com redução sem histerese

Esta alternativa é semelhante à anterior; contudo, para conseguir a resolução de projeto, ao invés de adotarmos a técnica de

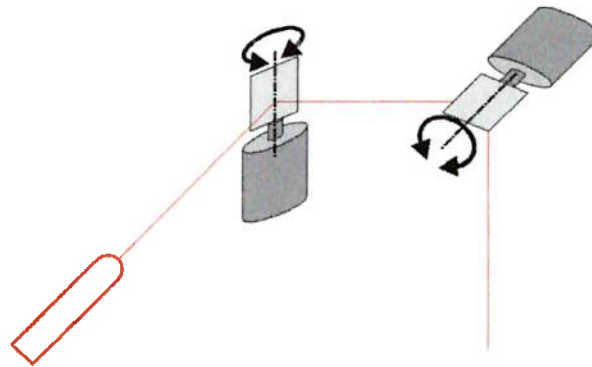


Fig. 4.4 - Alternativa C

microstepping emprega-se uma caixa de redução *zero backlash* (sem histerese mecânica). Há caixas de redução miniatura deste tipo com taxas de redução de até 1:1670, que aliadas a motores de passo comuns (200 passos por volta), forneceriam uma resolução de mais de

300000 passos por volta (cerca de 25 vezes mais do que a resolução mínima de projeto).

Conforme mostra a figura, os motores são diretamente ligados nas caixas de redução. Os espelhos são montados nos eixos de saída das caixas, sem nenhum subsistema mecânico adicional.

As principais vantagens são:

- o sistema como um todo ocupa um volume bastante reduzido;
- a resolução de projeto é facilmente atingida e ultrapassada;
- o erro de posicionamento devido a histerese é reduzido ao mínimo;
- é possível compensar as não-linearidades inerentes à geometria esférica no próprio algoritmo de controle, aproveitando a alta resolução.

As principais desvantagens são:

- o custo das caixas de redução é alto, por serem componentes de alta precisão;
- o torque de acionamento é muito maior, comparado às outras alternativas;
- as caixas de redução não podem ser acopladas a qualquer motor - é preciso adquirir os motores de passo do mesmo fabricante, aumentando o custo;
- os espelhos empregados deverão ser tais que permitam uma montagem que garanta que o eixo de rotação passe pelo plano de reflexão - ou seja, devem ser empregados espelhos de primeira superfície.

5. CRITÉRIOS DE SELEÇÃO

Abaixo são listados os critérios considerados na pré-seleção da alternativa construtiva. Naturalmente, esta primeira decisão poderá ser revista ao longo do desenvolvimento do projeto, servindo apenas de orientação decisória, não tendo caráter eliminatório.

5.1. Linearidade

É importante que o sistema seja capaz de projetar pontos equidistantes no plano de projeção. No entanto, as abordagens que adotam uma projeção esférica sofrem de uma não-linearidade intrínseca à própria geometria. Ainda assim, conforme evidencia a figura 4.3, para pequenas oscilações esta característica não-linear mal se manifesta.

Desta forma, no critério linearidade, a alternativa A se destaca, deixando as outras duas em segundo lugar.

5.2. Velocidade

Obviamente, quanto menos estágios de redução mecânica o sistema tiver, maior será a velocidade alcançável. Afinal, considerando-se o emprego do mesmo motor de passo nas três alternativas, existe uma frequência máxima com que os pulsos de acionamento podem ser enviados ao motor, e isto se traduz numa velocidade máxima do eixo do mesmo. Havendo uma redução mecânica acoplada a este, a velocidade máxima é proporcionalmente reduzida.

Logo, no critério velocidade, a alternativa B é a que mais se destaca, seguida da alternativa C e da alternativa A.

5.3. Precisão/Histerese

No tocante à histerese, novamente temos a regra empírica que diz que quanto mais simples for a mecânica, quanto menor a quantidade de partes móveis, melhor o projeto. Isto porque a histerese surge principalmente em virtude do atrito - logo, reduzindo-se as possíveis origens de atrito, reduz-se a histerese. Há ainda um certo grau de histerese inerente ao acionamento com *microstepping*, devido a não-linearidades construtivas dos motores de passo, mas a experiência mostra que esta histerese se torna significativa apenas na presença de cargas mecânicas consideráveis. Como o presente caso praticamente não apresenta carga (afinal, a massa a ser movimentada é a de um pequeno espelho), o efeito desta última histerese é negligenciável.

Quanto à precisão, as três alternativas têm um desempenho aproximadamente igual. Afinal, todas estariam sendo dimensionadas para atender ao requisito de precisão estipulado.

Com isso, neste critério classificamos as alternativas, em ordem de desempenho decrescente, da seguinte forma: B, C e A.

5.4. Simplicidade

A complexidade das alternativas A e C se concentra no aspecto mecânico, sendo que a primeira é bastante mais complexa do que a

segunda, conforme fica claro pela própria descrição das alternativas. Já a alternativa B exigirá um projeto de *software* mais complexo, além de envolver pequenas complicações na eletrônica de acionamento (inclusão de conversores D/A para gerar os *microsteps*). No entanto, *software* é muito mais “retrabalhável” do que estruturas mecânicas, que provavelmente teriam que ser reconstruídas do zero no caso de algum erro de projeto. Desta forma, como antes: a melhor alternativa é a B, seguida de C e A.

5.5. Custo

A alternativa A envolve o emprego de um espelho grande, além de fusos relativamente compridos e com alta precisão de fabricação. Além disso, há o custo envolvido na construção de toda a estrutura de suporte do sistema.

A alternativa B tem seu custo concentrado na eletrônica, tendo em vista a adição de componentes relativamente especializados. No entanto, este diferencial de custo é muito menos impactante do que o das outras alternativas.

A alternativa C é baseada no emprego de componentes mecânicos que, apesar de pré-fabricados e de prateleira, são extremamente caros. Possivelmente, a aquisição de um par de motores e um par de caixas de redução *zero backlash* por si só já ultrapassaria o custo-alvo estabelecido.

Logo, novamente temos que a melhor alternativa é a B, seguida de C e A.

6. CONSIDERAÇÕES SOBRE O *ASPECT RATIO*

Como fica claro na avaliação de cada alternativa em cada critério, a melhor opção parece ser a adoção de geometria esférica com acionamento por *microstepping*. No entanto, na caracterização desta alternativa como uma opção realista e viável, algumas considerações precisam ser feitas.

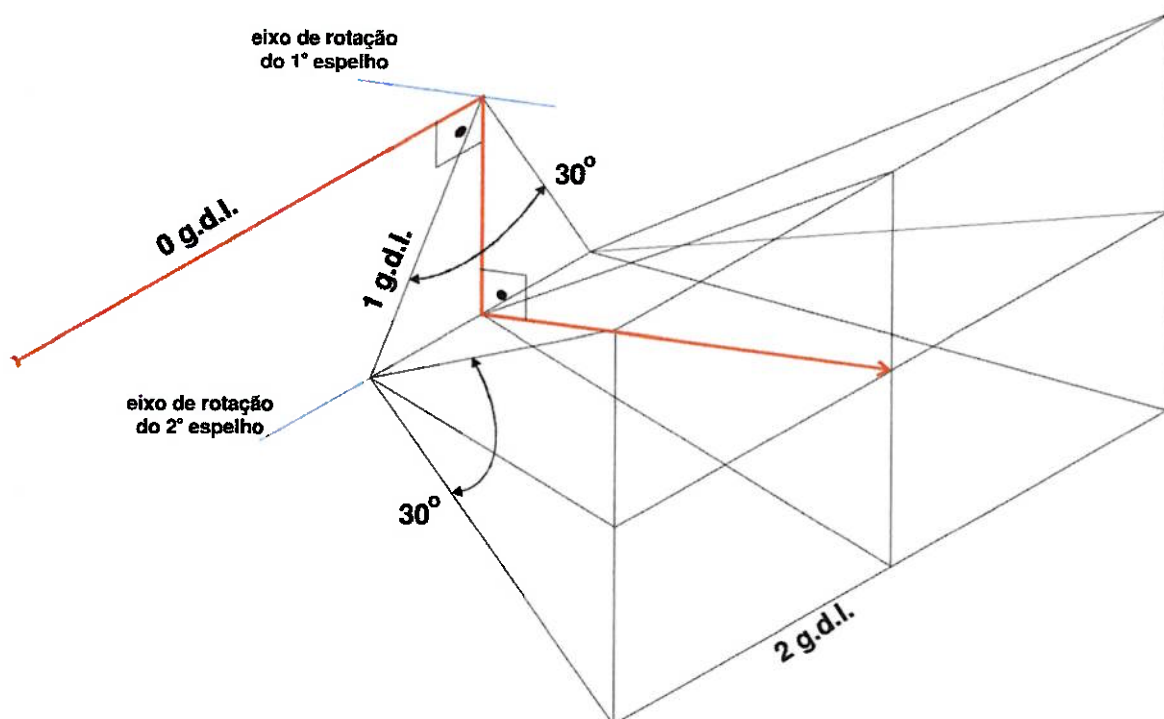


Fig. 6.1 - A origem da distorção no *aspect ratio* é a distância entre os pontos de adição dos dois graus de liberdade

A figura 6.1 mostra o caminho do feixe *laser* à medida que este percorre o interior do mecanismo. Como a figura mostra, o feixe com 0 graus de liberdade atinge o primeiro espelho, que adiciona um grau, abrindo um “leque” de 30°. Este “leque” então atinge o segundo espelho, onde o último dos dois graus é adicionado, na formação de um outro “leque” (perpendicular ao primeiro). Por uma questão de simples conveniência, chamemos o lugar geométrico da intersecção do feixe *laser* com os espelhos de “foco”. Note que o foco do segundo espelho não é, a princípio, um ponto, e sim uma reta.

Os requisitos de projeto exigem uma resolução de 1 ponto/mm, numa área de 500x500 pontos. Isto implica que o *aspect ratio* no plano de projeção é igual a 1. No entanto, com a geometria mostrada na figura 1, isto não é realizável. O motivo é que os dois focos não coincidem. Matematicamente, temos o seguinte desenvolvimento:

Seja d a distância entre os focos dos dois espelhos, e seja x a distância do plano de projeção até o segundo espelho. Conforme estabelecemos, a abertura dos dois “leques” é de 30° cada. Logo, a abertura linear do leque adicionado pelo segundo espelho, a uma distância x , é dada por:

$$V = 2x \tan(15^\circ)$$

Similarmente, a abertura linear do leque do primeiro espelho é dada por:

$$H = 2(x + d)\tan(15^\circ)$$

Logo, o *aspect ratio* é dado por:

$$\text{A.R.} = \frac{V}{H} = \frac{x}{x + d}$$

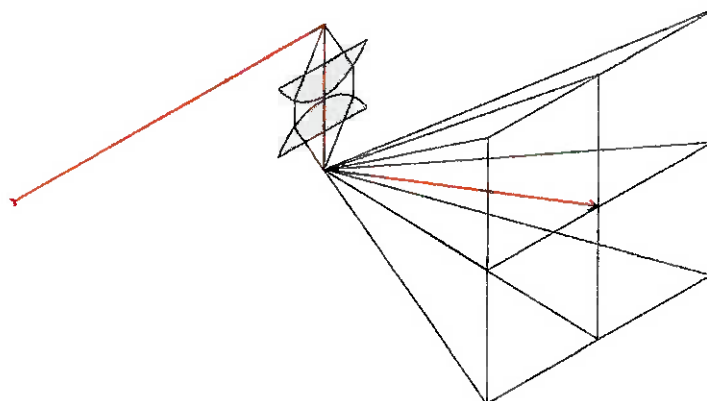


Fig. 6.2 - Correção da distorção do *aspect ratio* por meio de um conjunto ótico

Como é fácil ver, a única maneira de fazer com que o *aspect ratio* seja igual a 1 é fazer com que a distância **d** seja igual a 0. A rigor, seria também possível conseguir este resultado fazendo com que a abertura dos leques fosse diferente, mas isto implicaria num sistema de acionamento diferente para cada motor de passo - por este motivo, esta solução está relegada a um segundo plano.

Fazer com que a distância d seja igual a zero significa fazer com que os focos dos dois espelhos coincidam. Isto é facilmente realizável por meio de um par de lentes arranjadas como mostra a figura 6.2, posicionadas de modo que simplesmente adicionem um *offset* ao vértice do primeiro leque, transladando-o para que coincida com o vértice do segundo leque.

Esta correção ainda tem uma segunda vantagem, além de proporcionar um *aspect ratio* igual a 1: originalmente, o segundo espelho deveria obrigatoriamente ser maior do que o primeiro, exatamente pelo fato de seu foco não ser um ponto, e sim um segmento de reta. No entanto, com a adição do conjunto ótico, o foco do segundo espelho passa a ser um ponto, o que permite o emprego de um espelho menor - de fato, igual ao empregado na adição do primeiro grau de liberdade.

7. TESTES DE LABORATÓRIO: *MICROSTEPPING*

Tendo em vista o que foi exposto no item 5, fica claro que - *a priori* - a melhor alternativa é o emprego de *microstepping*, com os espelhos acoplados diretamente sobre o eixo do motor.

No entanto, é preciso validar esta abordagem por meio de ensaios de laboratório, para verificar se é possível alcançar a precisão e repetibilidade necessárias com um sistema de testes simplificado. Com

esse intuito, será construída uma giga de testes da seguinte maneira: um motor de passo comum (200 passos por volta) será preparado com um espelho e fixado a uma bancada. Um projetor *laser* (*laser pointer*) será também fixado, de modo que o feixe incida sobre o espelho e seja refletido sobre uma superfície distante (uma parede, por exemplo). Um circuito de acionamento do motor será construído, de acordo com o diagrama de blocos da figura 7.1.

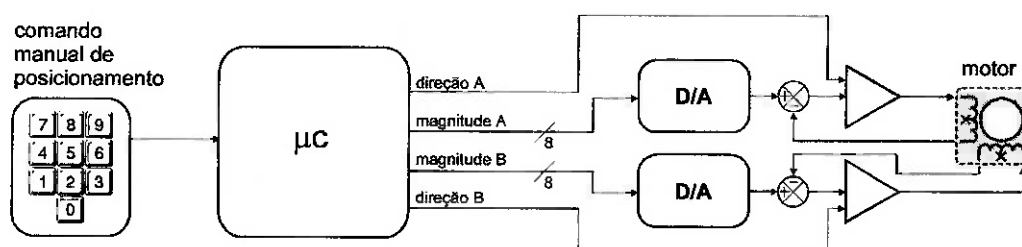


Fig. 7.1 - Diagrama de blocos do circuito proposto para validar a alternativa B

Neste circuito, temos uma entrada manual de dados (por meio de um teclado), dados estes que corresponderão de alguma forma à corrente injetada em cada um dos dois enrolamentos (será utilizado um esquema de acionamento bipolar). Desta forma, será possível controlar o campo gerado por cada um dos enrolamentos e por conseguinte, teoricamente, a posição do rotor. Estes dados são fornecidos a um microcontrolador, que gera a informação digital das correntes de cada um dos enrolamentos. A princípio, esta informação consistirá em 8 bits de magnitude e 1 bit de sinal, para cada um dos enrolamentos.

A magnitude é alimentada a conversores D/A, que geram na saída um sinal de tensão proporcional à entrada digital. Este sinal, por sua vez, servirá de referência a um circuito de ponte H, clássica implementação de controle de corrente para motores. A própria ponte H tem como entrada também o bit de sinal da corrente, alternando a polaridade de acionamento do enrolamento de acordo com esta informação.

Um esquema incompleto do circuito proposto é apresentado na figura 7.2. O esquema está incompleto por 2 motivos: o esquema mostra apenas um pedaço correspondente ao acionamento de um dos dois enrolamentos do motor. O acionamento do outro enrolamento é idêntico, e os conversores D/A compartilham o mesmo barramento de dados. Por outro lado, no esquema não aparece o microcontrolador em si (um AT90S8515, da *Atmel*), apenas um de seus barramentos de dados e os sinais de controle que este deverá prover (LATCH e DIRECTION). Tendo em vista que este circuito se presta unicamente à validação de um conceito, o controlador será usado na própria placa de demonstração que acompanha o kit de desenvolvimento, uma vez que este já se encontra à disposição. Desta forma, o programa para este teste rodará na placa de demonstração, que será conectada ao circuito da figura 7.2, assim fornecendo todos os sinais de controle. Adicionalmente, esta placa de demonstração já possui um teclado incorporado, o que facilita esta etapa do projeto.

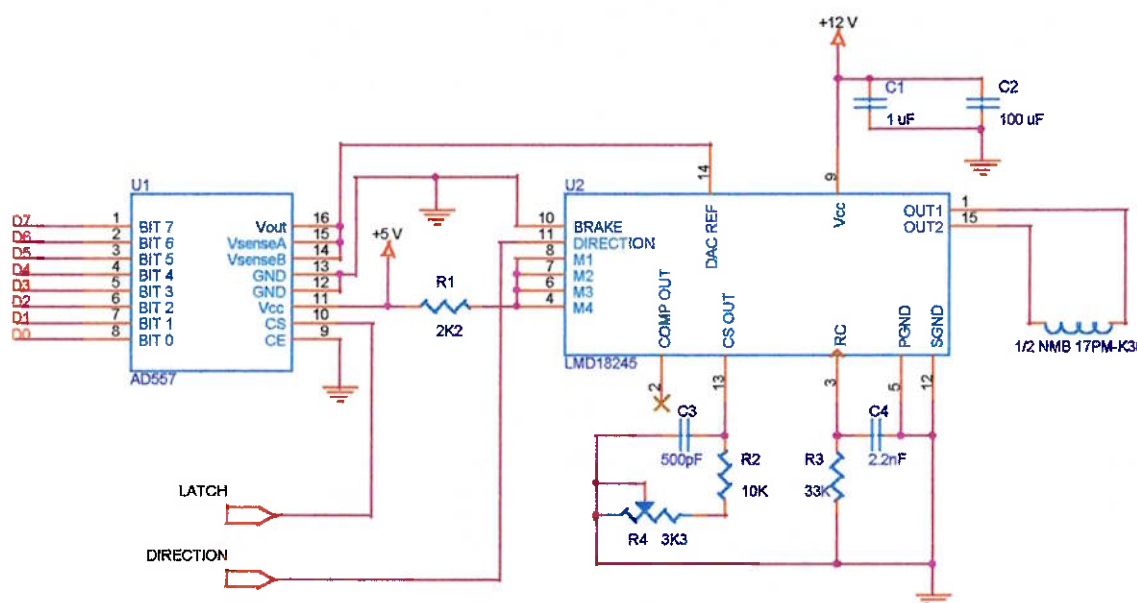


Fig. 7.2 - Esquema do circuito proposto para validar a abordagem *microstepping*

Os motores de passo adquiridos são de construção própria para o emprego de acionamento *microstepping* (17PM-K301V, da NMB Corp.), e os circuitos integrados envolvidos na construção do circuito proposto para testes (AD557 da Analog Devices e LMD18245 da National Semiconductor) foram obtidos como amostras. Esta disponibilidade foi um fator decisivo na escolha destes componentes, mas é preciso deixar claro que o mesmo resultado poderia ser obtido com diversas outras implementações.

Os conversores D/A da Analog Devices (AD557) foram escolhidos em virtude das diversas vantagens que apresentam: em primeiro lugar, a resolução é adequada (8 bits); além disso, este *chip* já tem saída em tensão, o que torna desnecessário o emprego de um estágio externo de amp-op's. Por fim, a referência de tensão de fundo de escala é definida internamente, fixa em 2.55V, o que também torna desnecessário o uso

de uma referência externa. Como mostra a figura 7.2, o pino de saída do D/A é ligado diretamente à ponte H, e os únicos componentes externos necessários para o uso deste *chip* são capacitores de desacoplamento na alimentação (não mostrados na figura 7.2).

As pontes H da National Semiconductor (LMD18245) já têm um D/A interno, de 4 bits, projetado para aplicações de *microstepping*. No entanto, devido à baixa resolução, as entradas deste D/A são todas grampeadas em *high*, efetivamente anulando a influência deste bloco funcional. Uma outra vantagem deste *chip* em particular é o modo como é implementada a malha fechada para controle de corrente: enquanto nos *chips* tradicionais o sensor de corrente é um resistor em série com os emissores da ponte, de valor baixo em função das altas correntes que circulam pelos enrolamentos, no LMD18245 existe um amplificador interno de corrente, com ganho 1/4000, de modo que o resistor *shunt* pode ter um valor bem maior (o que facilita a implementação, e proporciona melhor precisão de ajuste).

No caso do circuito apresentado, como a corrente nominal do motor é 800mA e a tensão de fundo de escala dos D/A's é 2.55V, a resistência *shunt* terá um valor de:

$$2.55 = R \cdot 0.8 \cdot \frac{1}{4000} \Rightarrow R = 12.75\text{k}\Omega$$

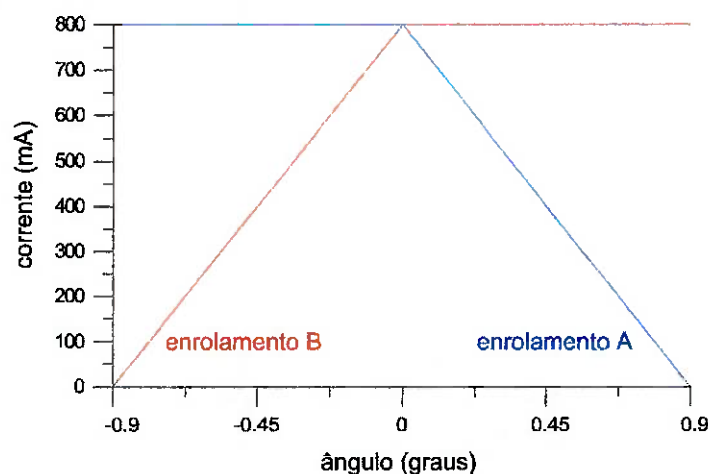


Fig. 7.3 - Perfil de acionamento dos enrolamentos para o primeiro teste

Conforme mostrado na figura 7.2, esta resistência consiste na verdade em um resistor em série com um potenciômetro, o que permite um ajuste preciso do valor.

O circuito foi implementado, constituindo então um *driver microstepping* completo para um motor de passo com as características do 17PM-K301V (vide Apêndice I), capaz de controlar digitalmente a corrente de cada enrolamento com uma resolução de 8 bits. Naturalmente, o que o sistema inteiro faz depende do *software* que roda no microcontrolador.

O primeiro *software* feito teve por objetivo verificar se de fato é possível gerar um aumento significativo da resolução do motor. Para isso, o motor foi preparado com um espelho comum em seu eixo, com o feixe *laser* diretamente apontado para este e refletido numa parede distante. O *software*, por sua vez, gera uma sequência de valores para

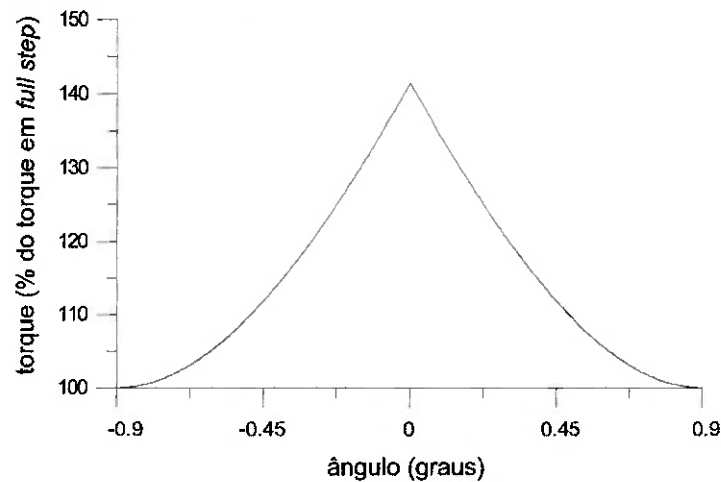


Fig. 7.4 - Variação do torque ao longo da amplitude de oscilação

as correntes do enrolamento conforme mostra a figura 7.3, englobando uma amplitude total de oscilação de um *full step* (1.8°). Na primeira metade do movimento, a corrente do enrolamento A é mantida no máximo, enquanto a corrente do enrolamento B sobe de 0mA (*full step*) a 800mA (quando os dois enrolamentos são igualmente excitados, ou seja, *half step*), sendo que esta subida tem uma resolução de 8 bits, compreendendo um total de 256 pontos. A segunda metade do movimento é simétrica à primeira, resultando no enrolamento B com 800mA e o enrolamento A com 0mA. Logo, gera-se um total de 512 combinações de valores de corrente para os enrolamentos.

O perfil de correntes mostrado na figura 7.3 gera pontos de operação totalmente não-lineares, na medida em que o torque resultante varia significativamente ao longo de toda a amplitude (figura 7.4) e também na medida em que as posições de equilíbrio não são

igualmente espaçadas (figura 7.5). Nesta última figura, a reta vermelha indica o que seria configuração correta, correspondendo a uma variação linear da posição do eixo. Estas não-linearidades ocorrem devido ao fato do perfil de correntes não ser senoidal. No entanto, o perfil usado é muito mais fácil de implementar, e cumpre a função de verificar se é possível aumentar a resolução do motor.

O resultado deste primeiro teste foi positivo. O feixe refletido caminhou suavemente ao longo dos 512 pontos gerados, embora de modo não-linear (pelos motivos expostos). Desta forma, conclui-se que a abordagem *microstepping* é válida e funcional.

Num segundo teste, o *software* já passa a ter um perfil de correntes para acionamento dos enrolamentos com característica

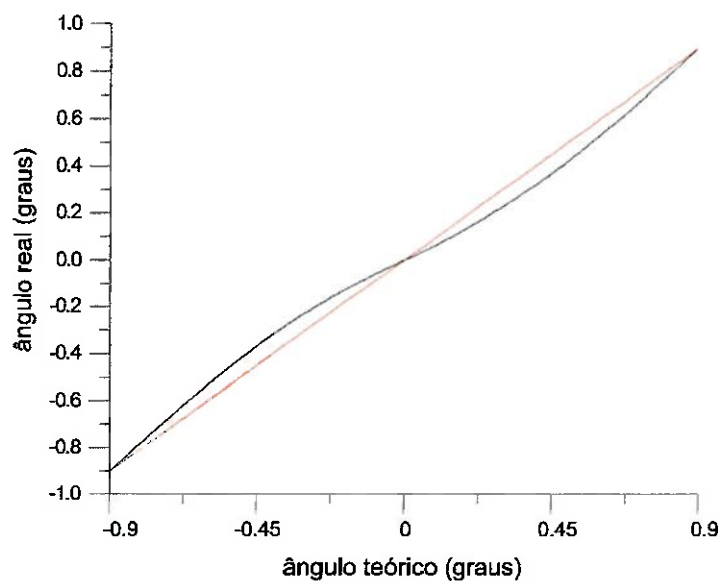


Fig. 7.5 - Comportamento não-linear da posição de equilíbrio do eixo do motor

senoidal (fig. 7.6), com um total de 1000 pontos para cada *full step* que, teoricamente, resultam em 1000 posições igualmente espaçadas numa amplitude de 1.8° de oscilação do eixo. De acordo com a resolução de projeto, precisa-se apenas de 64 pontos por *full step* - no entanto, os 1000 pontos obtidos não serão rigorosamente igualmente espaçados, devido a não-linearidades na eletrônica de acionamento e no próprio motor. Logo, o que se pretende fazer é escolher, dentre os 1000 pontos gerados, aqueles 64 que resultam em 64 pontos de incidência do feixe igualmente espaçados. Em outras palavras, o sistema será calibrado e compensado de modo a gerar 64 pontos igualmente espaçados.

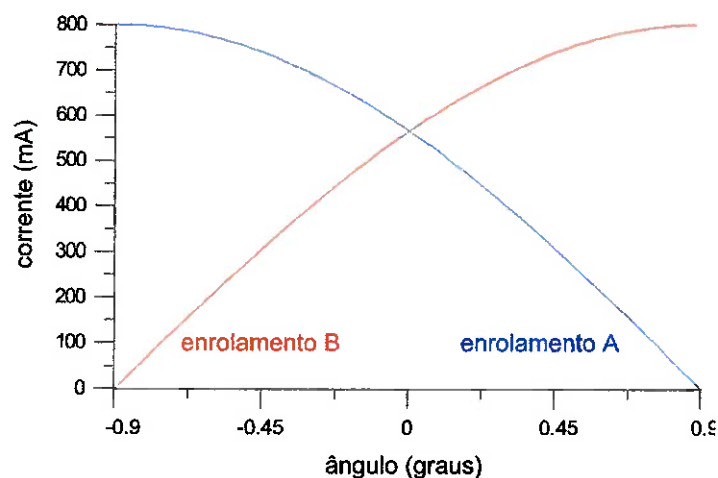


Fig. 7.6 - Perfil senoidal de correntes

Ainda nesta fase de testes preliminares, todos realizados em laboratório, dois outros aspectos funcionais do sistema foram abordados: o controle de luminosidade do feixe e a interface serial de comunicação com o *host*. O kit de desenvolvimento do microcontrolador, usado como base para o funcionamento deste

conforme explicado anteriormente, já possui um *chip* driver de RS-232. Este circuito integrado será incorporado na versão final da eletrônica do sistema, sendo sua aplicação muito fácil e comum. Optou-se, na versão final, por um DS14C232 da National Semiconductor, devido à facilidade de uso e obtenção.

Numa terceira versão do *software* de teste, foi implementado um protocolo de comunicação rudimentar, com o objetivo exclusivo de verificar o funcionamento deste *feature* do sistema. Os testes transcorreram sem maiores problemas, constatando-se a viabilidade da implementação de uma interface padrão RS-232, que no sistema final poderá atingir 38400 bps (em virtude do *clock* do microprocessador).

A última versão do *software* de teste usada em laboratório incluía um acionamento PWM da fonte *laser*, implementado por meio de um transistor bipolar NPN comum (BC337), em cuja base é injetado um sinal vindo de uma das portas de I/O do microcontrolador. Novamente, os resultados foram positivos, e constatou-se que o acionamento da fonte *laser* usando PWM é viável.

Desta forma, em virtude dos resultados obtidos com estes testes preliminares, constatou-se que a abordagem *microstepping* é capaz de fornecer a resolução desejada, além de terem sido validados outros aspectos do sistema, tais como o PWM e a interface RS-232.

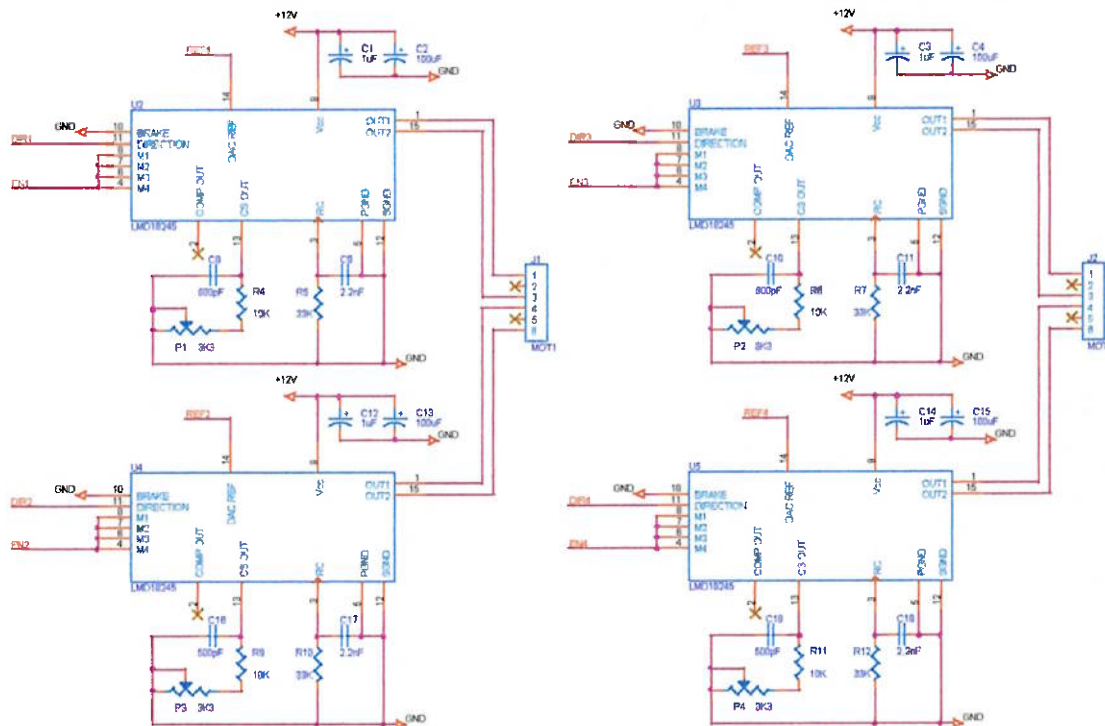


Fig. 8.1b - Circuito de controle e acionamento (parte 2 de 3)

de dimensionamento, os cálculos envolvidos serão apresentados e explicados. No entanto, a enorme maioria dos componentes foram colocados no circuito por recomendação dos fabricantes dos circuitos integrados, e desta forma, não se apresentarão justificativas. A figura 8.1 apresenta o circuito proposto nesta seção.

8.1. Controle de Corrente dos Enrolamentos

Como antes, o controle de corrente nos enrolamentos do motor será implementado com o *chip* LMD18245, da National Semiconductor. A tensão de alimentação do enrolamento será de 12V, suficientemente mais alta do que a nominal (4.8V) para que a resposta dinâmica do indutor seja otimizada.

Na versão para testes, todas as entradas digitais dos conversores D/A internos do LMD18245 eram mantidas em *high*, sendo que o *set-point* de corrente era dado exclusivamente pela referência analógica deste D/A, por sua vez gerada por um D/A externo. No entanto, na fase de testes constatou-se que, ao mudar o bit de direção de um destes *chips* (situação que só ocorre quando o valor digital da corrente controlada por este *chip* é 0) o eixo dava um salto maior do que os outros *microsteps*. Isto provavelmente ocorre porque existe uma tensão residual na entrada da referência analógica do *chip*, de forma que ao invés desta transição ser de +0 A para -0 A, ela acaba sendo de +X A para -X A, onde X é um valor próximo de 0.

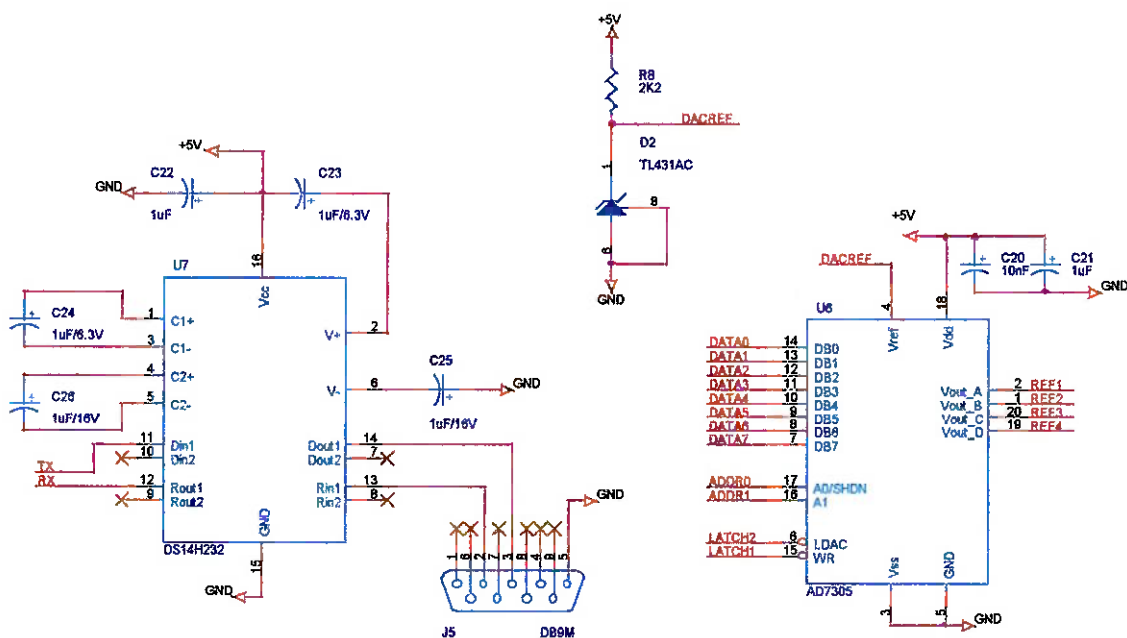


Fig. 8.1c - Circuito de controle e acionamento (parte 3 de 3)

Para eliminar o efeito desta tensão residual, as entradas do D/A interno do *chip* passam agora a ser controladas pelo microcontrolador, ficando todas em *high* quando o valor digital da corrente for maior do que 0, e indo para *low* quando o valor desejado for 0. Pretende-se, com isto, suavizar a transição do bit de direção.

A constante RC usada nos *chips* é de aproximadamente 80 μ s, dentro da faixa recomendada pelo fabricante. Esta constante de tempo corresponde a uma frequência de 12.5KHz, e de fato ouve-se um ruído agudo durante a operação do circuito. O ruído muda de frequência porque a frequência de *chopping* do motor não será de exatamente 12.5KHz o tempo todo. O valor de 80 μ s é na verdade o tempo durante o qual o enrolamento permanece desligado uma vez que a corrente ultrapasse o *set-point*. No entanto, à medida que o *set-point* muda, o tempo necessário para a corrente ultrapassar este limite muda, e com isso o período total de chaveamento do enrolamento muda. Logo, é natural que o ruído mude de frequência.

O resistor sensor de corrente, ligado ao pino CS OUT do *chip*, terá um valor ligeiramente diferente daquele usado no circuito para testes. Isto ocorre porque, enquanto no circuito de teste a tensão de fundo de escala do conversor D/A era de 2.55V (gerada no próprio D/A), no circuito final esta tensão será de 2.5V, gerada por um circuito integrado de referência de tensão (um TL431 da SGS-Thomson). Logo, de acordo com o mesmo raciocínio apresentado no cálculo deste resistor

anteriormente, obtemos um valor de R igual a $12.5\text{K}\Omega$. A mesma associação de um resistor de $10\text{K}\Omega$ com um potenciômetro de $3.3\text{K}\Omega$ será usada para obter-se este valor.

8.2. Conversores D/A

O *chip* AD557, da Analog Devices, foi usado na placa de testes por principalmente três motivos: são *chips* de pronta disponibilidade, sua saída já é em tensão (não sendo necessário assim usar um amp-op na saída para converter um sinal de corrente em um sinal de tensão) e a referência de fundo de escala é gerada internamente. Para o circuito final, no entanto, optou-se por um outro *chip*, o AD7305 (também da Analog Devices), que apesar de não ter geração interna de referência de fundo de escala, é um D/A quádruplo num único encapsulamento, com a vantagem adicional de possibilitar um *update* simultâneo de todas as quatro saídas analógicas. Desta forma, todos os enrolamentos podem ser atualizados ao mesmo tempo.

A tensão de referência de fundo de escala será gerada por um TL431 da SGS-Thomson, que produz uma referência de 2.5V . Todos os sinais de controle do conversor D/A serão gerados diretamente pelo microcontrolador. Com isso, é possível escrever os valores de cada um dos quatro canais do conversor, e em seguida promover a atualização simultânea de todas as saídas.

8.3. Interface RS-232

O microcontrolador usado já possui uma UART interna completa, capaz de atingir taxas de transmissão de até 38400 bps (usando-se um *clock* de 8 MHz). No entanto, fisicamente esta interface assíncrona trabalha com níveis de tensão TTL, ou seja, 0V e +5V. O padrão EIA-232 trabalha com outros níveis de tensão, e para fazer esta conversão de níveis de tensão de forma rápida existe uma série de circuitos integrados, dentre os quais o mais conhecido é certamente o MAX232 da Maxim. A National Semiconductor possui seu próprio *chip* de conversão TTL/EIA, o DS14C232, que requer apenas alguns capacitores externos para conseguir gerar as tensões necessárias a partir de uma alimentação padrão TTL. O lado TTL deste *chip* é conectado diretamente às portas de comunicação do microcontrolador, enquanto o lado EIA é ligado a um conector DB9 fêmea, obedecendo a pinagem padrão de microcomputadores. Com isto, basta um cabo cruzado para estabelecer a conexão.

8.4. Controle de Luminosidade do Laser

A fonte *laser* a ser usada é um *laser pointer* comum, alimentado com +5V. Seu consumo de corrente é inferior a 10mA, e seu comportamento é de uma carga resistiva (até uma determinada frequência de operação). Desta forma, para que seja possível fazer um controle PWM da luminosidade do feixe, é preciso inserir um interruptor eletrônico em série com a alimentação, interruptor este controlado

diretamente pelo microcontrolador. Este sinal de controle terá o perfil de um sinal PWM, cuja frequência deverá ser superior a cerca de 50Hz para que não se perceba o *flicker* do chaveamento.

Este circuito foi implementado por meio de um transistor bipolar NPN comum, o BC337. Trata-se de um componente extremamente barato, rápido e com uma capacidade de corrente bastante superior à necessária para esta aplicação.

8.5. Microcontrolador

Optou-se pelo uso de um AT90S8515, da Atmel Corporation, como microcontrolador a ser usado no sistema. Diversos são os motivos que levaram a esta escolha, dentre os quais podemos destacar: trata-se de um processador extremamente rápido, com muitas instruções tomando apenas um ciclo de *clock* para serem executadas; possui 4 portas de I/O totalmente configuráveis, com alta capacidade de corrente; possui UART interna; usa um conjunto de instruções extremamente fácil e versátil; tem capacidade de programação *in-circuit*, disponibilidade imediata para o projetista, incluindo ferramentas de suporte tais como placa de demonstração, compiladores e simuladores; e, finalmente, baixo custo.

Os componentes externos necessários para o emprego deste *chip* foram todos tirados de *application notes* fornecidos pelo fabricante. O *chip* trabalhará com um *clock* de 8MHz, que é a máxima velocidade

desta versão do *chip*. Todos os sinais de controle necessários para todos os outros componentes do sistema serão fornecidos diretamente pelo microcontrolador, devido à alta quantidade de portas disponíveis. Além disso, estuda-se a possibilidade de acrescentar ao sistema uma memória RAM de 32Kb (para a qual o microcontrolador já se apresenta facilmente configurável), para servir de *buffer* dos dados recebidos do computador *host*.

8.6. Alimentação

Toda a eletrônica implementada trabalha com alimentação TTL, ou seja, 0V e +5V. Os motores, no entanto, serão alimentados com +12V. Enquanto o consumo total dos motores não passará de 4A, estima-se que o consumo da linha de +5V será da ordem de 500mA.

Ainda assim, para evitar a etapa enfadonha de projetar-se uma fonte de alimentação específica para esta aplicação - atividade que demandaria tempo e dinheiro -, optou-se pelo emprego de uma fonte pronta, encontrada comercialmente. As fontes de alimentação usadas em microcomputadores fornecem tanto o +5V como o +12V, e são capazes de fornecer corrente suficiente nestes ramos para suprir as necessidades do sistema. Além disso, são produtos baratos, facilmente disponíveis e extremamente confiáveis. Em particular, será usada uma fonte de PC com proteção interna por fusíveis de disparo rápido, para proteger o sistema em caso de curto-circuito accidental.

8.7. Orçamento Preliminar

Com base no circuito proposto nesta seção, já é possível fazer um orçamento estimativo da parte eletrônica. Neste orçamento, entretanto, ainda não serão acrescentados alguns itens, tais como o *layout* e a confecção da placa de circuito impresso, cabos e outros itens secundários. Este orçamento preliminar encontra-se na Tabela 8.1, com as restrições mencionadas. Os valores indicados são em reais (R\$), correspondendo a preços médios de mercado ou preços indicados pelos distribuidores.

item	qtde.	descrição	valor	\$ unit	\$ total
01	2	Capacitor	22 pF	0,20	0,40
02	4	Capacitor	500 pF	0,20	0,80
03	4	Capacitor	2.2 nF	0,30	1,20
04	1	Capacitor	10 nF	0,30	0,30
05	8	Capacitor	1 μ F / 6.3V	0,30	2,40
06	2	Capacitor	1 μ F / 16V	0,30	0,60
07	1	Capacitor	3.3 μ F / 10V	0,10	0,10
08	4	CAPACITOR	100 μ F	0,30	1,20
09	1	Resistor	100 Ω	0,10	0,10
10	1	Resistor	330 Ω	0,10	0,10
11	2	Resistor	1 K Ω	0,10	0,20
12	2	Resistor	2.2 K Ω	0,10	0,20
13	1	Resistor	4.7 K Ω	0,10	0,10
14	4	Resistor	10 K Ω	0,10	0,40
15	4	Resistor	33 K Ω	0,10	0,40
16	4	Trim-pot	3.3 K Ω	0,70	2,80
17	1	Diodo	1N4148	0,10	0,10
18	1	LED	verde	0,20	0,20
19	1	LED	vermelho	0,20	0,20
20	1	Transistor	BC337	0,20	0,20
21	1	Cristal	8 MHz	2,00	2,00
22	1	Circuito Integrado (<i>Analog Devices</i>)	AD7305	8,00	8,00
23	1	Circuito Integrado (<i>SGS-Thomson</i>)	TL431	2,00	2,00
24	1	Circuito Integrado (<i>National Semiconductor</i>)	DS14C232	4,00	4,00
25	4	Circuito Integrado (<i>National Semiconductor</i>)	LMD18245	20,00	80,00
26	1	Microcontrolador (<i>Atmel Corporation</i>)	AT90S8515	30,00	20,00
				TOTAL	128,00

Tab. 8.1 - Orçamento preliminar da parte eletrônica

9. PRÉ-PROJETO DO SISTEMA MECÂNICO

A parte mecânica do sistema proposto deve cumprir exclusivamente uma função: servir de estrutura de suporte para todos os elementos envolvidos. Estes elementos são, em princípio, a fonte *laser*, os motores com os espelhos acoplados em seus eixos e o conjunto ótico (vide seção 6).

Apresenta-se a seguir o pré-projeto de todas as peças a serem fabricadas com este propósito, salientando-se que algumas características poderão ser alterados em virtude de revisões dos parâmetros de projeto e outros fatores similares. Por este mesmo motivo, os desenhos apresentados são apenas esquemas conceituais, sendo que a definição de medidas será feita *a posteriori*. No entanto, um comentário geral pode ser feito com relação à seleção do material de que serão fabricadas as peças: não existe solicitação mecânica significativa em nenhum dos componentes.

Desta forma, a escolha dos materiais não sofre influência de aspectos de resistência dos materiais. Por outro lado, a estrutura do sistema deve fornecer uma base sólida e firme para o conjunto, o que nos leva a preferir materiais pesados, tais como o aço. Nenhum dos desenhos desta seção apresenta escala definida.

9.1. Suportes dos Espelhos

As características mecânicas dos motores de passo adquiridos encontram-se no Apêndice I. O eixo deste motor tem 5mm de diâmetro. É preciso adaptar um espelho de 10x10x2 (medidas em milímetros) neste eixo, de modo que a superfície reflexiva contenha o eixo de rotação do espelho. Trata-se de um espelho de primeira superfície com acabamento em alumínio protegido (H45517 da Edmund Scientific Company), de modo que a superfície reflexiva é a superfície externa. Um suporte adequado apresentará as características mostradas na figura 9.1 - uma peça simples, prevendo um parafuso para fixação do suporte no eixo, e com uma superfície de apoio do espelho rebaixada de 2mm a partir do eixo da peça. A fixação do espelho no suporte será feita por um adesivo de uso geral. Quanto ao material, estas são as únicas peças que deverão ser tão leves quanto possível, para minimizar a inércia rotacional do conjunto e, com isso, otimizar a resposta dinâmica do espelho.

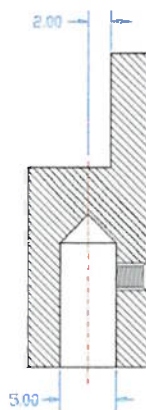


Fig. 9.1 - Suportes dos espelhos

9.2. Conjunto Ótico

Conforme discutido na seção 6, existe a necessidade de um conjunto ótico, formado por duas lentes condensadoras em oposição, para que não ocorram distorções do *aspect ratio*. Foi escolhido o modelo H43988 da Edmund Scientific Company por suas características geométricas - especificamente um anel externo que permite sua fixação de maneira bastante simplificada. De todas as lentes deste tipo, foi escolhido este modelo em particular por apresentar uma distância focal compatível com as dimensões gerais do sistema (dadas a princípio pelo próprio tamanho do motor).

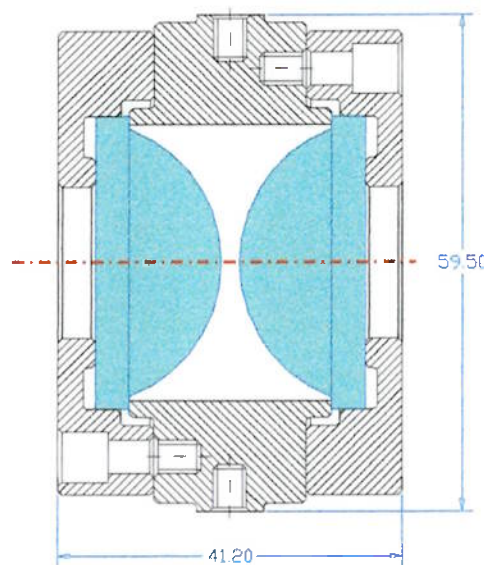


Fig. 9.2 - Conjunto ótico

A distância entre as lentes não é crítica, pois não influencia a correção ótica imposta pelo conjunto. Por outro lado, é fundamental que o centro de cada espelho esteja a uma distância de 16.4mm da face

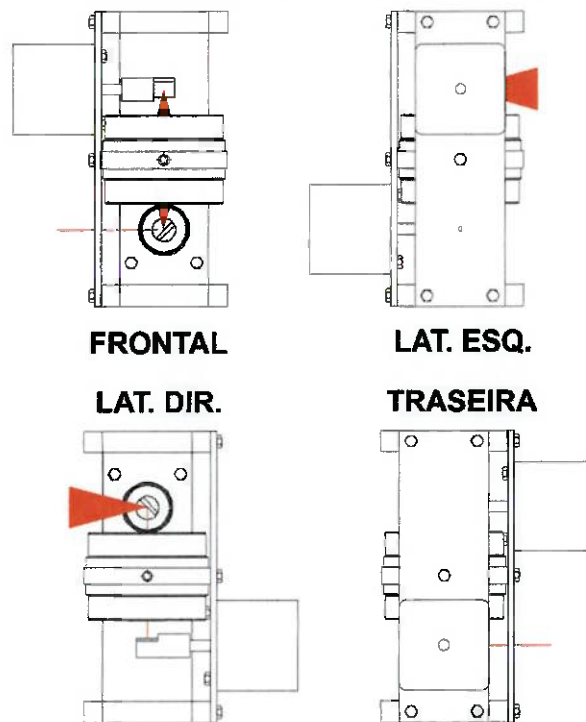


Fig. 9.3 - Vistas ortogonais do conjunto

externa da lente correspondente - ou seja, o foco da lente. As dimensões mostradas são meramente ilustrativas, não sendo valores críticos. Note que o conjunto é composto de um anel central, com furos roscados no diâmetro externo para a fixação do conjunto, e de duas tampas iguais, fixadas ao anel central por meio de três parafusos.

9.3. Estrutura

A figura 9.3 mostra uma concepção de estrutura que agrega todos os elementos do sistema, exceto a placa que abriga a parte eletrônica. Na figura não é mostrada a fixação da fonte *laser* - no entanto, na vista lateral esquerda é possível ver o orifício por onde o feixe entra no conjunto.

Como se vê, a estrutura consiste em duas bases, usinadas a partir de chapas de aço, onde são fixadas as paredes, por meio de parafusos. Estas paredes, por sua vez, servem de apoio para os motores de passo e para o tambor do conjunto ótico. As paredes têm espessura de cerca de 3mm, enquanto as bases têm altura de aproximadamente 10mm. Todos os parafusos usados são M3, exceto talvez os de fixação dos suporte dos espelhos, que podem vir a ser M2.5 por razões de espaço. O conjunto todo cabe num volume aproximado de 100x100x150 (medidas em milímetros).

10. PRÉ-PROJETO DE *FIRMWARE*

Durante a fase de testes para validação da abordagem *microstepping* (vide seção 7), diversos tópicos de programação foram abordados, muitos dos quais se aplicam à versão final do sistema. Como exemplo, recursos como o movimento dos motores e transmissão de dados via interface serial já foram implementados nesta fase de testes, embora não de forma completa e integrada, como será necessário na implementação final. Ainda assim, com base na concepção do sistema e na experiência adquirida nestas implementações preliminares, é possível delinear algumas características dos principais blocos funcionais do *software* a ser implementado.

10.1. Comunicação Serial

O microcontrolador utilizado (AT90S8515, da Atmel Corporation) possui uma USART (*Universal Serial Asynchronous Receiver and Transmitter*) *full duplex*, com diversas características que facilitam seu uso no âmbito de um projeto maior. De acordo com a documentação do produto, a maior taxa de transmissão que se pode alcançar, usando-se um cristal de 8 MHz (a maior frequência utilizável), é de 38400 bps. A taxa de transmissão *real* será ligeiramente diferente (pouco acima de 38461 bps), e esta diferença se traduz numa largura do bit real ligeiramente menor do que a ideal. Este erro é cumulativo ao longo dos bits de um determinado byte, mas não é cumulativo *entre* bytes tendo em vista que a comunicação é assíncrona - ou seja, a referência temporal é reinicializada a cada byte. Como resultado, para estes valores de taxa de transmissão, o acúmulo de erro não oferece perigo significativo.

Existe uma disponibilidade limitada e relativamente baixa de memória de trabalho no microcontrolador (512 bytes de SRAM). Isto gera a necessidade de um controle de fluxo na troca de informações entre o sistema e o *host*, controle este que pode ser feito por *hardware* (CTS/RTS) ou por *software* (Xon/Xoff). É possível ainda estabelecer um protocolo de controle de fluxo proprietário, desenvolvido especificamente para atender as necessidades do sistema. Por exemplo, uma alternativa particularmente atraente é a transmissão de conjuntos pequenos de

dados, constituindo “comandos” completos para o sistema (função a ser executada + eventuais parâmetros). Uma vez recebido cada comando, este é executado e uma resposta confirmando a liberação para o próximo comando é enviada ao *host*. Deste modo, a transmissão do *host* para o sistema fica “policiada” pelas respostas que o último envia ao primeiro, evitando perda de dados. Adicionalmente, como estas respostas para o *host* se dão a cada comando, que corresponde a uma certa quantidade de bytes, a taxa efetiva de transmissão é maior do que obteríamos, por exemplo, fazendo uma confirmação a cada byte enviado.

10.2. Movimentos Básicos do Motor

Conforme discutido anteriormente, os eixos dos motores são colocados em uma determinada posição através da alimentação de cada enrolamento com correntes de valores conhecidos (será feita uma explicação para apenas um motor, mas naturalmente o procedimento é válido para ambos os motores). Estes valores formam uma senóide, e para qualquer posição do eixo do motor, os valores das correntes estão defasados de 90°. Também foi visto que são necessários 64 *microsteps* para cada *full step*, para atingirmos a resolução de projeto. Desta forma, é possível pensar em uma implementação do tipo LUT (*look-up table*) para a senóide já discretizada em incrementos de $90^\circ/64 = 1.40625^\circ$. No entanto, simplificações adicionais na implementação ainda podem ser feitas.

Para um dado motor, o algoritmo de controle deverá estar ciente sempre dos valores da corrente de cada um dos dois enrolamentos. No entanto, como vimos, estes estão sempre defasados de 90° - logo, estes dois valores não são independentes. Por outro lado, no *hardware* projetado, o valor digital da corrente é alimentado ao estágio de potência segundo um esquema sinal/magnitude, i.e., o bit de sinal corre por uma via independente do barramento de dados dos D/A's. Como existe uma simetria entre os valores positivos e negativos de uma senóide, precisamos implementar apenas a metade positiva de um ciclo, ou seja, 180° . Fazendo assim, abrangeremos 2 *full steps*, ou 128 valores na tabela. Seria ainda possível implementar apenas 1/4 de ciclo, tendo em vista esta outra simetria existente em meio ciclo senoidal. No entanto, isto implicaria rotinas de consulta a esta tabela ligeiramente mais complexas, e por este motivo optou-se pela implementação de meio ciclo. A figura 10.1 ilustra o funcionamento desta tabela. Para um dado motor, existem dois ponteiros que indicam, respectivamente, a posição na tabela das correntes dos enrolamentos "A" e "B". Estes ponteiros

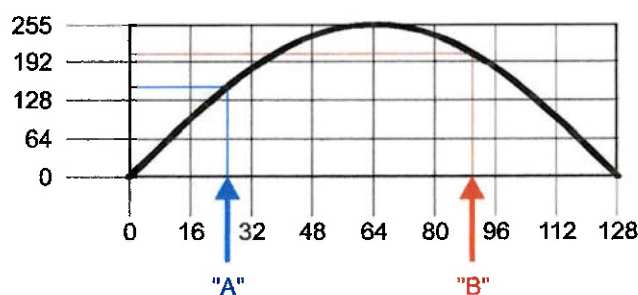


Fig. 10.1 - Implementação dos perfis de acionamento dos motores por *look-up table*

distam sempre de 90° , ou seja, 64 posições na tabela. Quando qualquer um destes ponteiros está em um dos extremos da tabela, seu próximo salto será para o extremo oposto da tabela, e o bit de sinal daquele enrolamento será invertido.

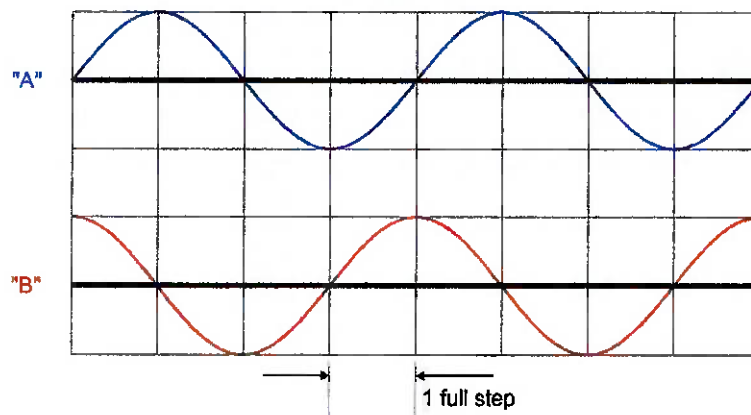


Fig. 10.2 - Resultado da implementação dos perfis por *look-up table*

Desta forma, percorrendo-se a tabela em um sentido, avançando os dois ponteiros para o mesmo lado a cada *microstep* (exceto no caso descrito acima, em que um ponteiro atinge um extremo), o eixo do motor gira num determinado sentido; percorrendo-se a tabela no outro sentido, inverte-se a rotação do eixo. Naturalmente, a cada movimento dos ponteiros o algoritmo deve atualizar os valores presentes nos conversores D/A, além dos bits de sinal. Por fim, conforme exposto na seção 8.1, sempre que o valor da corrente de um determinado enrolamento for identicamente nulo, o bit de *enable* correspondente deve ser mantido em 0. O resultado do uso desta implementação por LUT pode ser visualizado na figura 10.2 - as correntes resultantes, para cada enrolamento, obedecem as condições necessárias para o

movimento adequado do motor. Em termos de organização do *software*, é possível imaginar um conjunto de 4 rotinas responsáveis pela manipulação dos ponteiros e atualização dos valores correspondentes no *hardware*, cumprindo as funções de dar passos unitários (*microsteps*) nos sentidos horário e anti-horário, para os motores 1 e 2.

10.3. Algoritmo de Traçagem de Linhas

Uma vez implementadas funções que movimentem o feixe para cima, para baixo, para a esquerda e para direita (seção 10.2), é possível pensar em uma função no nível imediatamente superior, responsável por fazer com que o feixe percorra cada ponto do segmento de reta que une dois pontos especificados. A área de trabalho desta função será toda a área útil de projeção, de 500 x 500 pontos. Esta função será, fundamentalmente, a implementação de um algoritmo de traçagem de linhas, tendo como parâmetros de entrada as coordenadas dos pontos inicial e final do segmento. Pelo menos a princípio, faremos a suposição de que o controle da luminosidade do feixe transcorre independentemente desta função, e não trataremos ainda do problema do controle de velocidade.

Existem diversos algoritmos bastante conhecidos que tratam deste problema. Basicamente, estes algoritmos devem atender a alguns requisitos básicos, dentre os quais:

- **As retas devem “parecer” realmente retas:** as técnicas de plotagem de pontos são ótimas para traçar linhas paralelas ou a 45° dos eixos do sistema. Nos outros casos, entre os pontos inicial e final a reta passará por pontos não-endereçáveis. A qualidade da reta dependerá da escolha dos pontos intermediários, podendo parecer “quebrada” ou não;
- **As retas devem terminar corretamente:** quando se desenha uma figura geométrica é importante que cada reta termine exatamente onde começar a próxima, para evitar falhas representadas por pequenos saltos;
- **As retas devem ter densidade constante:** independentemente do comprimento ou da direção da reta, é necessário que o brilho seja uniforme para evitar a sensação de variação da espessura da linha;
- **As retas devem ser traçadas rapidamente:** como as retas são normalmente usadas em grandes quantidades em aplicações gráficas elas devem ser desenhadas rapidamente. Isto implica que se deve usar o mínimo de computação para traçá-las.

Os critérios acima foram desenvolvidos tendo em mente a aplicação de tais algoritmos na traçagem de retas em monitores de vídeo. No entanto, as idéias principais neles contidas se aplicam também ao caso em questão. Em particular, o terceiro requisito tem sua importância aumentada pelo fato de a densidade local da reta traduzir-se, no caso da estereolitografia, na profundidade de polimerização. Similarmente, como o algoritmo será implementado em

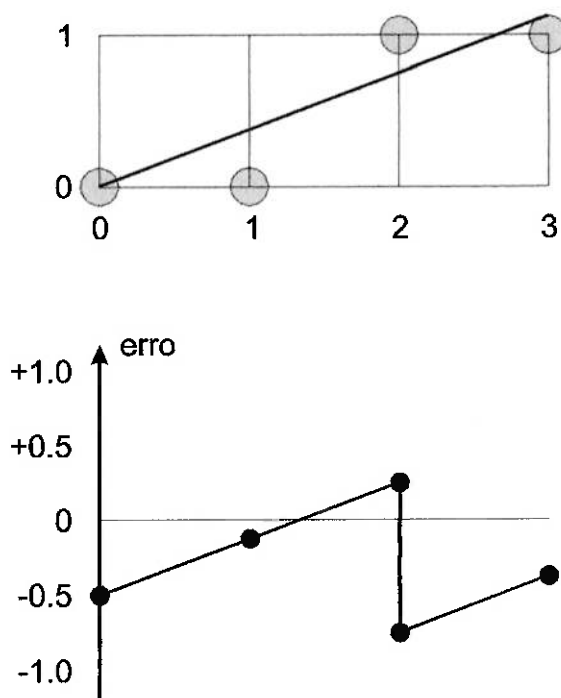


Fig. 10.3 - Exemplo de traçagem pelo algoritmo de Bresenham

microcontrolador, a necessidade de cálculos simples e rápidos expressa no último requisito também tem sua importância redobrada.

Para esta aplicação, escolheu-se o algoritmo de Bresenham com variáveis inteiras, na implementação de Rogers. O método de Bresenham para traçagem de retas consiste em sempre incrementar de uma unidade a coordenada x (no caso do primeiro octante; para os outros octantes, o raciocínio é similar), enquanto o incremento da coordenada y (0 ou 1) é determinado examinando-se a distância entre a posição da reta real e a posição do pixel mais próximo. Esta distância é denominada *erro*, e na sua definição está a eficiência deste algoritmo.

Suponha uma reta passando por $(0, 0)$. Se a inclinação for maior que 0.5, então sua intersecção com a linha $x = 1$ está mais próxima da linha $y = 1$ do que da linha $y = 0$. Assim, a melhor aproximação da reta real passa pelo ponto $(1, 1)$, e não pelo ponto $(1, 0)$. O contrário aconteceria se a inclinação da reta fosse menor que 0.5. Dessa forma, inicializando-se o erro com -0.5, ao se somar a inclinação da reta ao mesmo, é necessário apenas verificar o sinal deste parâmetro para tomar a decisão de qual é a melhor aproximação.

Note que após incrementar a coordenada y é preciso reinicializar o erro. Isto é feito subtraindo-se uma unidade de seu valor. A figura 10.3 mostra um exemplo de traçagem de uma reta com inclinação de $3/8$.

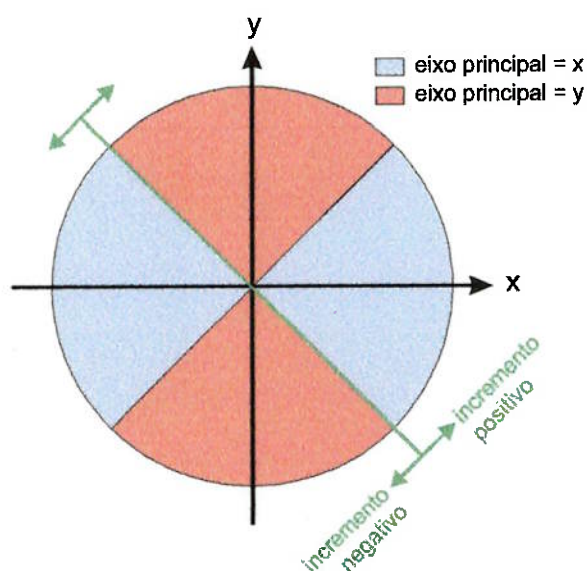


Fig. 10.4 - Mapeamento de eixos principais e incrementos no algoritmo de Bresenham

Sobre esta estrutura básica do algoritmo de Bresenham, acrescentam-se ainda duas modificações importantes. A primeira delas é uma substituição de variáveis para a fórmula do erro. Esta mudança de variáveis permite que o algoritmo seja completamente implementado fazendo-se uso apenas de variáveis inteiras.

A segunda modificação consiste em ampliar o algoritmo para que retas em todos os octantes possam ser apropriadamente tratadas. No caso do primeiro octante, como foi visto acima, a coordenada x é incrementada sempre. No caso geral, uma das coordenadas, x ou y , será incrementada ou decrementada, *sempre*. O mapeamento do sinal do incremento e de qual será a coordenada sempre incrementada pode ser melhor visto na figura 10.4. Passaremos a chamar a coordenada incrementada sempre (ou decrementada sempre) de *eixo principal*. Assim, no primeiro octante, o eixo principal é o eixo x , e seu incremento é positivo. Já para uma reta que vai de $(10, 10)$ para $(11, -200)$ o eixo principal é o eixo y , e seu incremento é negativo.

Com estas considerações, uma implementação em C do algoritmo de Bresenham, para os oito octantes e com variáveis inteiras, pode ser a seguinte:

```
Bresenham(x1, y1, x2, y2)
{
  int    x,y;
  int    delta_x, delta_y;
  int    s1, s2;
  int    e;
  int    auxiliar;
  char    troca_mutua;
  int    i;

  x = x1;
  y = y1;
  delta_x = abs(x2-x1);
  delta_y = abs(y2-y1);
  s1 = sign(x2-x1);
  s2 = sign(y2-y1);
  if(delta_y > delta_x)
  {
    auxiliar = delta_x;
    delta_x = delta_y;
    delta_y = auxiliar;
    troca_mutua = 1;
  }
  else
    troca_mutua = 0;
  e = 2 * delta_y - delta_x;
  for(i = 1; i <= delta_x; ++i)
  {
    plot(x, y)
    if(e>=0)
    {
      if(troca_mutua)
        x += s1;
      else
        y += s2;
      e -= 2 * delta_x;
    }
    if(troca_mutua)
      y += s2;
    else
      x += s1;
    e += 2 * delta_y;
  }
}
```

A função *sign(x)*, utilizada no algoritmo acima, é definida da seguinte forma:

$$\begin{aligned} \text{sign}(x) = & -1, \text{ para } x < 0 \\ & 0, \text{ para } x = 0 \\ & +1, \text{ para } x > 0 \end{aligned}$$

O algoritmo acima foi implementado, em caráter de teste, na linguagem do microcontrolador a ser usado. Em sua aplicação final, o algoritmo recebe poucas alterações. Dentre elas, destacamos que as linhas que fazem o incremento das coordenadas, seja este incremento positivo ou negativo, são substituídas por chamadas às funções que dão um *microstep* no motor apropriado. Naturalmente, o sentido de rotação deste *microstep* é determinado pelo sinal do incremento, e o motor a ser acionado é determinado pela coordenada a ser incrementada.

10.4. Controle de Intensidade do Feixe

O projeto eletrônico foi feito de modo a permitir a conexão direta de um módulo *laser* de até 100 mW @ 5 VDC. Este módulo é acionado através de um transistor bipolar simples (BC337), cuja base é excitada pelo bit 4 do port B do microcontrolador. Chaveando-se o módulo *laser* em uma frequência alta através de um perfil PWM, é possível controlar-se a intensidade média do feixe. A resolução de projeto para este controle de intensidade de feixe é de 256 níveis (8 bits). Como a velocidade máxima de varredura, de acordo com as especificações do projeto, é de 500 pontos por segundo, conclui-se que a base do transistor de acionamento do *laser* deverá ser atualizada a uma frequência mínima de 500 Hz. No entanto, a frequência de *clock* do sistema (8 MHz) permite um chaveamento muito mais rápido. A princípio, o PWM do *laser* será implementado com uma frequência de cerca de 3.9 kHz, o que permitirá varrer toda a área imprimível em

pouco mais de 2 minutos. Esta relação entre a velocidade de varredura e a frequência do PWM existe porque se o tempo de permanência do feixe em um dado ponto for menor que o período do PWM, não se garante que a luminosidade média deste ponto será igual à desejada.

10.5. Interpretação de Comandos

Será implementado um protocolo de comunicação simples e rústico, apenas suficiente para demonstrar o funcionamento do sistema. As capacidades de memória e processamento do microcontrolador utilizado permitem que se implemente um protocolo bastante mais complexo, podendo-se por exemplo adaptar a linguagem G, utilizada em equipamentos CNC, ou ainda o HPGL, utilizado em *plotters*. Devido ao curto prazo de implementação do sistema como um todo, optou-se por um protocolo proprietário simples, embora abrangente o bastante para demonstrar toda a capacidade do sistema.

A comunicação se dá basicamente no sentido do *host* para o defletor, sendo que o defletor apenas emite bytes de controle de fluxo (padrão Xon/Xoff). Este controle de fluxo é necessário devido à pequena quantidade de memória RAM do microcontrolador (512 bytes), insuficiente para armazenar grandes quantidades de dados quando o sistema está operando em modo *raster*. Adicionalmente, a comunicação se dará a 9600 bps, sendo possível aumentar esta taxa de transmissão até um teto de 38400 bps.

11. PROJETO MECÂNICO

Nenhuma alteração conceitual foi feita no pré-projeto mecânico, apresentado na seção 9. O projeto das peças foi detalhado, e alguns valores foram adaptados em virtude da disponibilidade de matéria prima e da facilidade de fabricação.

A estrutura continua sendo basicamente de chapas de aço, com o objetivo de proporcionar o mínimo de deformações. Os motores, espelhos e lentes são os mesmos anteriormente especificados, e os únicos elementos novos são a placa de circuito impresso (vide seção 12.1), o sensor de *home* (vide seção 12.2) e o módulo *laser* com ótica incorporada (vide seção 11.1).

Os desenhos de fabricação finais encontram-se no Apêndice II. A montagem do conjunto pode ser vista na seção 15.

11.1. Correção de Foco

Quando da primeira implementação do sistema, detectou-se um efeito de perda de colimação do feixe *laser* ao passar pelo conjunto ótico. Como resultado, o feixe, antes colimado (paralelo), passa a convergir para um ponto próximo à saída do conjunto ótico e, com isso, sua projeção no plano toma a forma de uma circunferência de diâmetro considerável (vide Fig. 11.1).

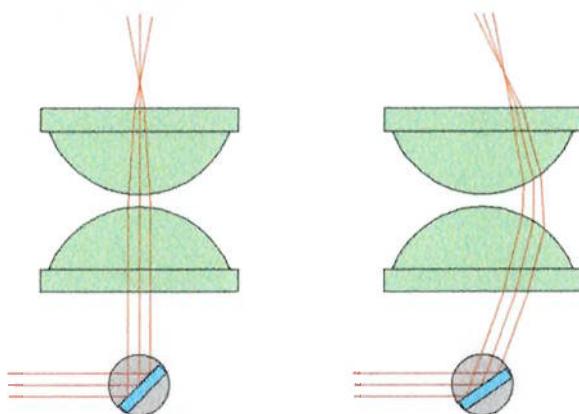


Fig. 11.1 - Perda de colimação do feixe *laser* devido à incidência fora do foco

Em primeiro lugar, foi preciso determinar o motivo desta perda de colimação. Investigações simples levaram à conclusão de que o responsável por isto é um efeito que foi absolutamente negligenciado durante a concepção do sistema. Conforme foi visto na seção 6, o conjunto ótico foi inserido no sistema para promover um *offset* no vértice de adição do primeiro grau de liberdade, de modo a coincidir com o segundo. A geometria apresentada na figura 6.2 é, de fato, válida - mas apenas para aqueles feixes que passam exatamente sobre o foco da primeira lente. O feixe real, no entanto, tem um diâmetro finito. Com isso, invariavelmente a maior parte do feixe real incidirá no primeiro espelho numa região *próxima* ao foco exato da lente, mas não exatamente *sobre* o foco. É exatamente esta parcela do feixe que não incide exatamente sobre o foco que acaba convergindo.

A primeira alternativa para solucionar o problema é reduzir o diâmetro do feixe real. Fazendo isso, a região de incidência do feixe sobre o espelho estará mais próxima do foco, e o efeito de convergência será minimizado. Isto não *corrige* o problema, apenas minimiza-o. No entanto, é possível assim minimizá-lo até que seu efeito seja arbitrariamente pequeno - desde que, obviamente, se consiga produzir tamanha redução do diâmetro do feixe.

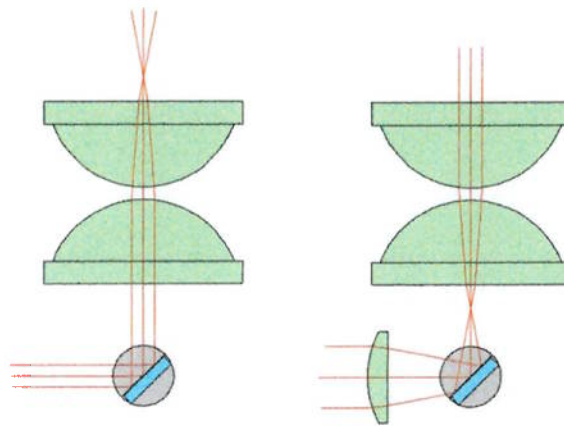


Fig. 11.2 - Correção da perda de colimação pela adição de uma lente convergente

Uma redução do diâmetro do feixe pode ser produzida usando-se um arranjo de duas lentes convergentes em oposição, preferencialmente de distâncias focais bastante diferentes (a primeira com distância focal grande, a segunda com distância focal pequena).

Uma outra alternativa para solucionar o efeito de convergência é baseado no princípio da reversão aplicável aos arranjos de lentes: basta introduzir uma convergência do feixe *antes* do conjunto ótico (vide Fig. 11.2); se esta convergência for igual àquela que seria produzida na

saída do conjunto ótico usando-se um feixe colimado, o feixe sairá do conjunto ótico colimado. Em outras palavras, para um dado arranjo de lentes, se o feixe entra colimado e sai convergente, então entrando convergente ele sai colimado. Naturalmente, tal convergência é conseguida com apenas uma lente, adequadamente selecionada.

Optou-se então por conjugar-se os *dois* métodos mencionados acima, como meio de minimizar o efeito de perda de colimação do feixe. Para isso, foi adquirido um módulo *laser* industrial (modelo ALA5-3.5-650 da Creative Technologies Inc.) com conjunto ótico incorporado, ao preço de US\$ 42,00 (quarenta e dois dólares americanos). Este módulo opera em 5 VDC, tem potência nominal de 3.5 mW e comprimento de onda 650 nm (vermelho). Incorporado ao conjunto está um arranjo de 3 lentes que permitem o ajuste independente tanto do diâmetro do feixe quanto do foco, compreendendo valores adequados à aplicação no sistema.

Uma compreensão exata da geometria ótica do sistema leva à conclusão que é vantajoso operar na maior distância focal proporcionada pelo módulo. Com isso, o diâmetro final do feixe será minimizado. Experiências, aliadas às características técnicas do módulo, mostraram que uma distância de cerca de 120 mm é ideal para a montagem. O módulo foi então acondicionado em um suporte de alumínio acoplado à estrutura mecânica do sistema (vide desenho no Apêndice II). Este suporte possui um sistema de alinhamento do feixe,

primitivo mas extremamente funcional, consistindo de parafusos que permitem ajustar a posição e direção do módulo em relação à estrutura mecânica. De fato, o emprego do módulo juntamente com o suporte se mostrou suficiente para produzir uma redução significativa do efeito de perda de colimação do feixe.

12. PROJETO ELETRÔNICO

Assim como o projeto mecânico, o projeto eletrônico praticamente não foi alterado em sua implementação final. Dentre as alterações feitas estão a substituição do conversor D/A AD557 por um AD7305, abordada na seção 8, a adição do driver RS-232 DS14C232 da National Semiconductor e a alteração do módulo *laser* (vide seção 11.1).

12.1. Sensor de *home*

A principal alteração foi a inclusão de um fototransistor ao sistema, que funciona como sensor de *home* para fins de auto-calibração. Este fototransistor (um PT-202C, da Everlight Inc.) foi ligado a uma entrada do port D do microcontrolador, tornando seu sinal acessível ao *firmware* do sistema. Fixando-se o corpo do fototransistor à estrutura do sistema e usando-se um algoritmo de busca automática deste sensor (seção 13.1), é possível criar um sistema de coordenadas absoluto, próprio do sistema.

Em sua fase de testes, o algoritmo de *homing* mostrou-se tão eficiente que sua repetibilidade atingiu níveis dificilmente mensuráveis. A eletrônica periférica necessária para a adição deste sensor ao circuito consiste, exclusivamente, de um resistor de *pull-up*.

A adição de um *segundo* sensor, idêntico ao primeiro, garantiria que o processo de auto-calibração fosse mais rápido: bastaria posicionar o primeiro sensor próximo ao primeiro espelho, e o segundo sensor próximo ao segundo espelho. Assim, primeiro faria-se a busca do feixe sobre o primeiro sensor, girando-se apenas o primeiro espelho - certamente, a referência seria encontrada dentro de, no máximo, uma volta do espelho. Uma vez encontrada esta posição de referência do primeiro espelho, este seria posicionado de forma que o feixe incidisse perpendicularmente ao conjunto ótico. Em seguida, procedimento semelhante seria feito com o segundo espelho.

12.2. *Layout da PCI*

Com o esquema do circuito eletrônico pronto e o projeto mecânico definido, partiu-se para o *layout* da placa de circuito impresso (PCI). Para tanto, foram selecionados os conectores a serem empregados, de acordo com sua disponibilidade, padronização de uso e características elétricas.

A PCI foi dimensionada de modo a ficar presa à parede 1 da estrutura mecânica. A disposição dos conectores foi planejada de modo

a racionalizar a distribuição dos cabos. Outras informações passadas ao profissional contratado para execução do *layout* diziam respeito à largura das trilhas. Naturalmente, as trilhas de corrente mais alta foram feitas proporcionalmente mais largas. Além disso, as trilhas que conduzem tensões de referência também foram alargadas, para minimizar quedas resistivas.

Uma vez pronto e aprovado o *layout*, feito para uma placa de face dupla, os documentos necessários foram encaminhados à Etelbras Eletrônica e Telecomunicações S.A., que gentilmente providenciou um desconto para a manufatura da placa. Não foi necessária nenhuma correção à placa entregue. O *layout* da PCI encontra-se no Apêndice II.

13. *FIRMWARE*

As idéias básicas apresentadas na seção 10 foram mantidas, sem quaisquer alterações. De fato, praticamente todos os problemas apresentados pela implementação foram resolvidos durante o pré-projeto. A versão final do *firmware* apresenta alguns recursos adicionais, não previstos na especificação inicial, cuja implementação se mostrou fácil e rápida.

Abaixo, alguns aspectos do *firmware* são discutidos em maior detalhe. No entanto, discussões mais aprofundadas sobre diversos

outros aspectos de implementação não serão feitas. O código é apresentado em sua totalidade no Apêndice IV. Para maiores esclarecimentos sobre a implementação, recomenda-se a consulta à literatura do fabricante do microcontrolador.

13.1. Controle de velocidade

O controle de velocidade de varredura do feixe foi implementado de forma indireta. Controla-se, na verdade, o *período* entre dois passos consecutivos do eixo principal (vide seção 10.3). Uma implementação direta da velocidade propriamente dita não foi feita pois não houve tempo hábil para implementar um algoritmo que, com base neste parâmetro, obtenha o valor do período. No entanto, o cálculo do período é facilmente feito no computador *host*.

Devido às configurações do *timer* no qual os movimentos se baseiam, a unidade de medida do período (no que tange à programação) é de 2 μ s. Conforme explicado abaixo, o parâmetro é passado em 3 dígitos hexadecimais. No caso específico do período do movimento, não se recomenda o uso de valores maiores do que 0x100 (período de 512 ms, ou cerca de 1950 passos por segundo), devido à deterioração da resposta dinâmica dos elementos mecânicos do sistema. Além disso, conforme o exposto na seção 10.4, usando-se este valor como período mínimo de varredura, garante-se que para cada ponto de incidência do feixe o *laser* será chaveado por um ou dois períodos do PWM.

Uma outra observação a ser feita quanto ao controle de velocidade é que o período dado será respeitado para o *eixo principal* de um dado segmento de reta. Desta forma, se uma dada velocidade é desejada para um segmento de reta genérico, uma correção trigonométrica deve ser feita no cálculo do período. Para os octantes em azul na figura 10.4, em que o eixo principal é o eixo X, o período a ser programado é dado por:

$$P = \frac{5 \cdot 10^5}{V|\cos\alpha|}$$

onde V é a velocidade de varredura, em pontos/segundo, e α é o ângulo de inclinação do segmento de reta. Analogamente, para os octantes em vermelho, em que o eixo principal é o eixo Y, o período é dado por:

$$P = \frac{5 \cdot 10^5}{V|\sin\alpha|}$$

13.2. Protocolo de comunicação

Os comandos do *host* para o defletor estão compreendidos em 3 grandes grupos, detalhados abaixo, sendo que alguns dos comandos exigem parâmetros numéricos (*sempre* representados por 3 dígitos hexadecimais). Para não desperdiçar tempo implementando um interpretador de comandos mais complexo, não serão feitas verificações como falta de zeros à esquerda, letras minúsculas e maiúsculas, etc.

Desta forma, nesta versão do protocolo de comunicação, a sintaxe dos comandos deve ser observada de maneira rigorosa. Obviamente, futuras versões poderão ser mais robustas, compreendendo um verificador sintático mais poderoso.

Comandos Gerais

- R** (0x52) : Reset - força uma reinicialização geral do sistema;
- H** (0x48) : Home - executa a rotina de *homing*, reiniciando o sistema de coordenadas do sistema (vide seção 13.1). A velocidade é reinicializada para o valor padrão (500 passos por segundo);
- P** (0x50) : Power Down - posiciona os espelhos em posições de *full-step* próximas à posição atual, e reduz a corrente de excitação dos motores de modo a minimizar o consumo de potência e o aquecimento dos motores, além de desligar o *laser*. Note que tanto as coordenadas da posição anterior quanto o valor da intensidade do *laser* são perdidos após a execução deste comando, mas o sistema de coordenadas é preservado;
- B** <valor> (0x42 + 3 dígitos) : Brightness - controla a intensidade do feixe *laser*. Admite-se valores entre 000 e 0FF, de modo que B000 desliga o *laser*, e B0FF liga o *laser* em sua intensidade máxima. Este comando é do tipo *one-shot*;
- S** <valor> (0x53 + 3 dígitos): Speed - controla o período de varredura. O parâmetro passado neste comando é o *período* entre dois passos da varredura, em unidades de 2 μ s. Assim, S9C4 programa uma varredura de 200 passos por segundo. Admite-se valores entre S100 (1950 pps) e SFFF (122 pps);

O (0x4F) : Origin - reposiciona a origem do sistema de coordenadas no ponto atual;

A (0x41) : Absolute - recupera o sistema de coordenadas absoluto, obtido após a execução do comando H. Desta forma, é possível redefinir a origem do sistema várias vezes (comando O), e depois voltar ao sistema absoluto.

Comandos para Imagens Vetoriais

F <coord. X> <coord. Y> (0x46 + 6 dígitos) : Fast - faz o avanço rápido dos espelhos para a posição indicada, com o *laser* desligado. O *laser* é religado na sua intensidade anterior logo após o fim do movimento, e a velocidade de avanço programada anteriormente à execução deste comando também é reestabelecida;

L <coord. X> <coord. Y> (0x4C + 6 dígitos) : Line - traça uma linha do ponto atual até o ponto indicado, com os últimos valores de intensidade de feixe e velocidade de avanço estabelecidos.

Comandos para Imagens Raster

I <valor> (0x49 + 3 dígitos) : Image - coloca o defletor no modo de imagem *raster*, preparando-o para receber sequências de dados cujo comprimento é dado por <valor>. Após este comando, o defletor deve receber as linhas da imagem *raster*, de cima para baixo, em sequência;

- + <**sequência de dados**> (0x2B + sequência) : Sequência direta - o caracter “+” no início de uma linha *raster* indica que os bytes são enviados da esquerda para a direita (sentido positivo do eixo X). A sequência de dados deve obrigatoriamente ter um tamanho igual ao comprimento indicado com o comando “I”;
- <**sequência de dados**> (0x2D + sequência) : Sequência inversa - o caracter “-” no início de uma linha *raster* indica que os bytes são enviados da direita para a esquerda (sentido negativo do eixo X). Alternando-se o envio das linhas *raster* no sentido direto e inverso, o tempo total de formação da imagem é reduzido, tendo em vista que a “impressão” é bidirecional;
- . (0x2E) : End mode - Indica o fim da imagem *raster*, recolocando o defletor no modo vetorial.

Comandos de Controle de Fluxo

- (0x11) : Xon - Ao receber este byte do defletor, o *host* está liberado para reiniciar a transmissão de dados e comandos;
- (0x13) : Xoff - Ao receber este byte do defletor, o *host* deve interromper o envio de dados ou comandos, tendo em vista que o *buffer* interno do defletor está cheio. Note que este comando poderá ser enviado pelo defletor a qualquer instante. Adicionalmente, o defletor *não* envia um Xon logo após um *reset*.

13.3. Algoritmo de *homing*

Com a adição do sensor de *home*, o sistema passa a ter um sistema de coordenadas fixo, obtido com base em uma auto-calibração. Esta auto-calibração é traduzida num algoritmo de *homing*, que pesquisa a posição do sensor e define a posição absoluta do sistema de coordenadas fixo em relação ao sensor.

O algoritmo de *homing* consiste, basicamente, em uma busca rápida e grosseira, seguida de uma busca mais restrita porém mais precisa. Para ganhar precisão, esta busca “fina” é feita iterativamente, o que equivale em termos práticos a tirar a média de várias medições.

A busca rápida é feita girando-se os espelhos em velocidade alta, de modo conjugado. Quantitativamente, trata-se de dar um passo em X para cada 100 passos em Y, o que resulta num movimento levemente “diagonal” do feixe. Este movimento é executado até que se detecte (por meio de interrupção) que o feixe bateu no sensor.

Neste instante, os eixos são freados. Devido à dinâmica do motor, cujos efeitos principais são um atraso de fase do eixo em relação ao acionamento elétrico e um *overshoot* no instante da parada, o eixo é então girado no sentido *oposto* (apenas em Y), em velocidade *baixa*, até que o feixe incida no sensor.

Garantindo-se então que os eixos estão parados e o feixe sobre o sensor, repete-se o seguinte procedimento várias vezes, para cada eixo

(de forma independente e intercalada). O eixo é girado no sentido positivo, até que o feixe saia do sensor. Em seguida, o eixo é girado no sentido negativo, até entrar novamente no sensor. Detectada a incidência, continua-se a girar o eixo no sentido negativo, até que o feixe saia do sensor, *contando-se quantos passos são dados*. Quando o feixe finalmente sai do sensor, metade dos passos contados são dados no sentido positivo do eixo.

Após a repetição do procedimento acima para cada eixo diversas vezes, o segundo motor (eixo Y) é girado 180°, de forma que o feixe esteja no centro da área projetável. As coordenadas são, então, definidas em (250,250).

Este algoritmo provou-se extremamente eficiente, demonstrando um nível de repetibilidade tal que não é possível detectar diferença de posição entre os sistemas de coordenadas obtidos em múltiplas execuções.

13.4. Algoritmo de *power down*

Durante a operação normal do sistema, os motores tendem a esquentar, até uma temperatura máxima de cerca de 80°C. Quando o sistema não está em movimento, os eixos dos motores são mantidos travados em posição, favorecendo o aquecimento e provocando um consumo de potência injustificado. Para minimizar o consumo durante

o tempo ocioso do sistema, foi implementado um algoritmo de *power down*.

O que o algoritmo faz, em linhas gerais, é posicionar os dois motores em uma posição de *full step* próxima à posição corrente (alterando a posição dos eixos, mas preservando a origem do sistema de coordenadas), e nestas posições de *full step*, excitar os enrolamentos dos motores com apenas 10% da corrente nominal - suficiente para travar a posição dos eixos, mas baixa o bastante para que o consumo seja reduzido sensivelmente. Antes deste reposicionamento dos eixos, o *laser* é desligado para que não incida em pontos não desejados.

14. CUSTOS FINAIS

Ao final do projeto, os custos da solução final, *as built*, foram contabilizados. O resultado está na Tabela 14.1. Nesta tabela, não foram incluídos os seguintes itens, por não serem considerados parte integrante do sistema ou por representarem fração irrisória do custo total: cabo serial, matéria prima estrutural, arruelas, solda, cabos diversos.

Como se vê, o custo total está próximo de R\$ 700,00 - o que, à taxa de câmbio média quando da data deste projeto, representa cerca de US\$ 350,00. O custo máximo admissível, definido nas especificações

do projeto, era de US\$ 1000,00. Obviamente, o projeto provou-se economicamente viável.

item	descrição	valor	fabricante	qtde.	\$ unit.	\$ total
1	capacitor eletrolítico	1uF/16V	-	10	R\$ 0,30	R\$ 3,00
2	capacitor eletrolítico	3,3uF/10V	-	1	R\$ 0,30	R\$ 0,30
3	capacitor eletrolítico	100uF	-	4	R\$ 0,30	R\$ 1,20
4	capacitor cerâmico	22pF	-	2	R\$ 0,20	R\$ 0,40
5	capacitor cerâmico	500pF	-	4	R\$ 0,20	R\$ 0,80
6	capacitor cerâmico	2,2nF	-	4	R\$ 0,20	R\$ 0,80
7	capacitor cerâmico	10nF	-	1	R\$ 0,20	R\$ 0,20
8	capacitor cerâmico	100nF	-	1	R\$ 0,20	R\$ 0,20
9	resistor	100R	-	1	R\$ 0,10	R\$ 0,10
10	resistor	330R	-	1	R\$ 0,10	R\$ 0,10
11	resistor	1K	-	3	R\$ 0,10	R\$ 0,30
12	resistor	2K2	-	2	R\$ 0,10	R\$ 0,20
13	resistor	4K7	-	1	R\$ 0,10	R\$ 0,10
14	resistor	10K	-	5	R\$ 0,10	R\$ 0,50
15	resistor	33K	-	4	R\$ 0,10	R\$ 0,40
16	trim-pot	3K3	-	4	R\$ 0,70	R\$ 2,80
17	transistor NPN	BC337	-	1	R\$ 0,25	R\$ 0,25
18	fototransistor	PT-202C	Everlight	1	R\$ 1,80	R\$ 1,80
19	diodo	1N4148	-	1	R\$ 0,10	R\$ 0,10
20	referência de tensão	TL431AC	Texas Instruments	1	R\$ 1,20	R\$ 1,20
21	led verde	5 mm	-	1	R\$ 0,20	R\$ 0,20
22	led vermelho	5 mm	-	1	R\$ 0,20	R\$ 0,20
23	conector PCI 2,54	22-27-2081	MOLEX	2	R\$ 0,70	R\$ 1,40
24	conector PCI 2,54	22-27-2031	MOLEX	2	R\$ 0,25	R\$ 0,50
25	conector PCI 2,54	87331-1020	MOLEX	1	R\$ 0,40	R\$ 0,40
26	conector de alimentação	9090-4R	ALEX	1	R\$ 0,80	R\$ 0,80
27	conector PCI serial	DB9 fêmea	-	1	R\$ 0,85	R\$ 0,85
28	microcontrolador	AT90S8515	ATMEL	1	R\$ 16,00	R\$ 16,00
29	ponte H	LMD18245T	National Semiconductor	4	R\$ 12,00	R\$ 48,00
30	conversor D/A	AD7305	Analog Devices	1	R\$ 4,00	R\$ 4,00
31	driver RS-232	DS14C232CN	National Semiconductor	1	R\$ 5,00	R\$ 5,00
32	cristal	8MHz	-	1	R\$ 2,00	R\$ 2,00
33	suporte DIP	16 pinos	-	1	R\$ 0,30	R\$ 0,30
34	suporte DIP	20 pinos	-	1	R\$ 0,50	R\$ 0,50
35	suporte DIP	40 pinos	-	1	R\$ 1,00	R\$ 1,00
36	conector cabo 2,54	22-01-2085	MOLEX	2	R\$ 0,60	R\$ 1,20
37	conector cabo 2,54	22-01-2035	MOLEX	2	R\$ 0,50	R\$ 1,00
38	motor de passo	17PM-K301V	NMB Corporation	2	R\$ 40,00	R\$ 80,00
39	módulo laser	ALA5-3.5-650	-	1	R\$ 84,00	R\$ 84,00
40	cooler	12V/60x60x20mm	-	2	R\$ 2,00	R\$ 4,00
41	fonte de alimentação	padrão PC	-	1	R\$ 25,00	R\$ 25,00
42	lente condensadora	H43988	Edmund Scientific	2	R\$ 51,60	R\$ 103,20
43	espelhos de primeira face	H32941	Edmund Scientific	2	R\$ 18,50	R\$ 37,00
44	placa de circuito impresso	TFMF99	Etelbras	1	R\$ 31,97	R\$ 31,97
45	parafuso	M3x10	-	8	R\$ 0,15	R\$ 1,20
46	parafuso	M3x12	-	42	R\$ 0,15	R\$ 6,30
47	parafuso	M3x5 s/ cabeça	-	2	R\$ 0,15	R\$ 0,30
48	porca	M3	-	16	R\$ 0,10	R\$ 1,60
49	horas de usinagem	-	-	40	R\$ 5,00	R\$ 200,00
					total	R\$ 672,67

Tab. 14.1 - Custos do projeto

15. RESULTADOS

15.1. Fotos

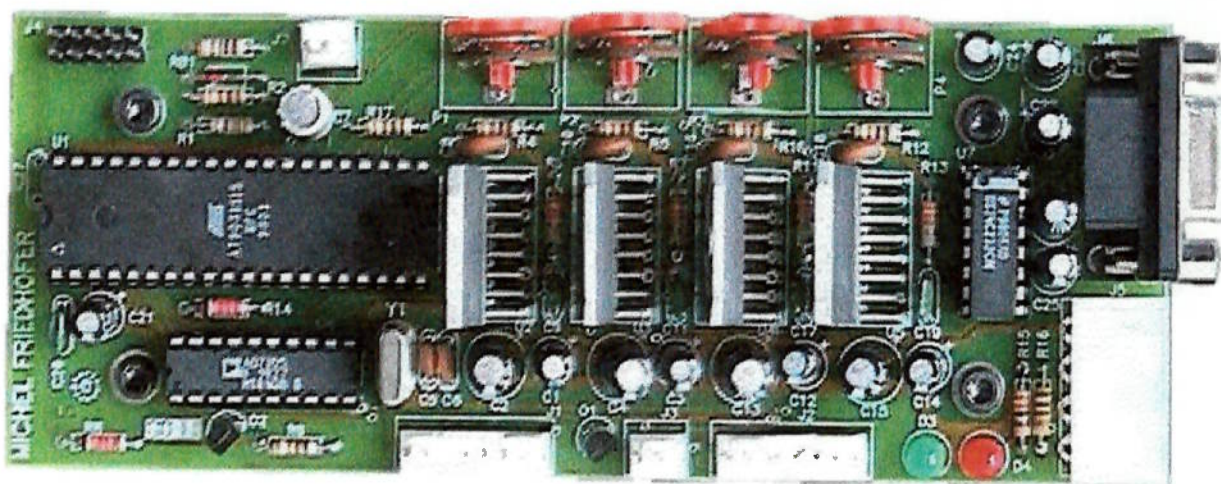


Fig. 15.1 - Placa de circuito impresso montada



Fig. 15.2 - Montagem do conjunto motor/suporte/espelho

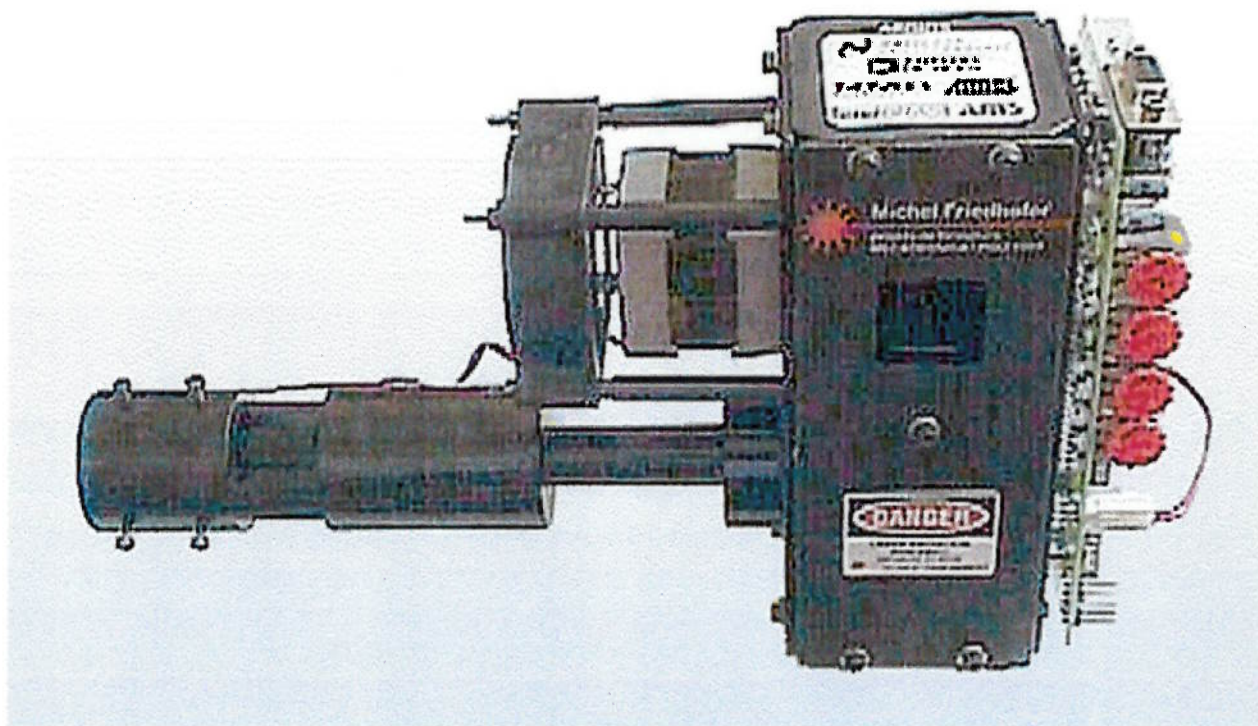


Fig. 15.3 - Conjunto

15.2. Projeções

As projeções abaixo foram fotografadas com o recurso de abertura longa do obturador (posição “Bulb” da máquina fotográfica). As distorções das figuras resultam de erros de alinhamento do *laser* e das lentes, além de falta de perpendicularismo defletor/parede e máquina fotográfica/parede.



Fig. 15.4 - Projeção em modo vetorial



Fig. 15.5 - Projeção em modo raster

16. REFERÊNCIAS BIBLIOGRÁFICAS

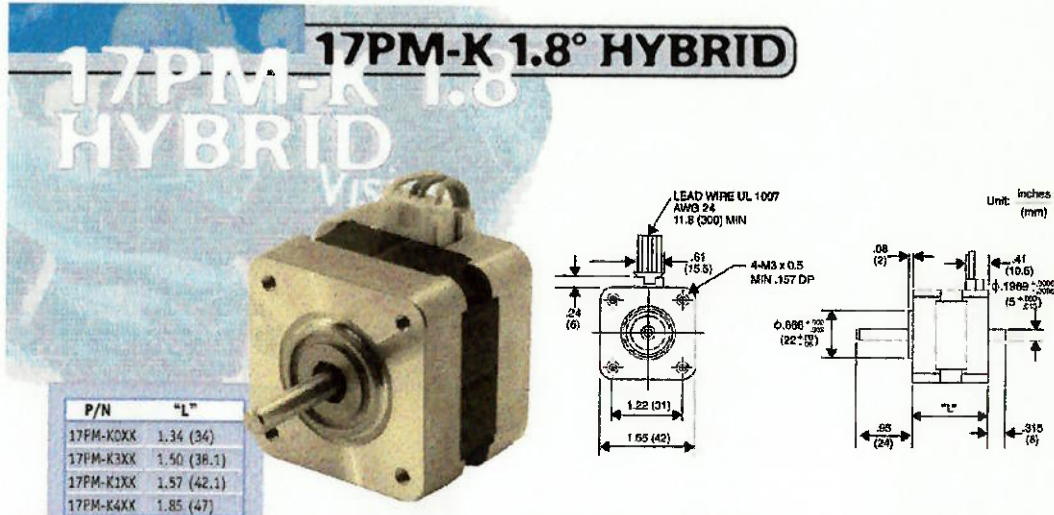
<http://www.national.com>
<http://www.analog.com>
<http://www.nmbcorp.com>
<http://www.everlight.com>
<http://www.eio.com/stepindx.htm>
<http://www.eio.com/prointro.htm>
<http://www.eio.com/lsrcint.htm>
http://www.eopc.com/eop_scan.htm
<http://www.pangolin.com/resguide01.htm>
<http://www.camtech.com/>
<http://www.cs.hut.fi/~ado/rp/rp.html>
<http://itri.loyola.edu/rp/toc.htm>

17. APÊNDICES

O Apêndice I contém as capas dos *data sheets* de alguns dos componentes do sistema. Os *data sheets* completos podem ser solicitados aos fabricantes, através de seus representantes no Brasil ou da World Wide Web (WWW). O Apêndice II contém todos os desenhos e esquemas do projeto, tanto da eletrônica quanto da mecânica. O Apêndice III mostra a pinagem dos cabos usados na operação do sistema. O Apêndice IV contém a listagem completa do *firmware* implementado no sistema. Por fim, o Apêndice V contém um exemplo de programa que demonstra um pouco da capacidade do sistema.

Apêndice I: Data Sheets

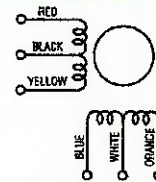
MOTOR DE PASSO



GENERAL SPECIFICATIONS

Step Angle 1.8°
 Step Angle Accuracy +/- 5%
 Temperature Rise 80° C Max
 Ambient Temperature Range -20° to +50° C
 Insulation Resistance 100MΩ Min., 500 VDC
 Dielectric Strength 500 VAC for 1 min.
 Radial Play 0.02 mm Max. (450 g-load)
 End Play 0.08 mm Max. (450 g load)
 Switching Sequence See page 31

WINDING DIAGRAM



MODEL SPECIFICATIONS

Model Number	Rated Voltage V	Rated Current/Phase A	Winding Resistance/Phase Ω	Holding Torque g-cm	Inductance mH	Rotor Inertia g-cm²	Detent Torque g-cm	Weight g
17PM-K015V	8.80	0.40	22.00	1.500	19.5	34.0	80	200
17PM-K017V	4.40	0.80	5.50	1.500	5.7	34.0	80	200
17PM-K018V	3.00	1.20	2.50	1.500	2.8	34.0	80	200
17PM-K316V	9.50	0.40	24.00	1.700	25.8	45.0	100	250
17PM-K301V	4.80	0.80	6.00	1.700	7.1	45.0	100	250
17PM-K303V	3.20	1.20	2.70	1.700	3.3	45.0	100	250
17PM-K111V	10.00	0.40	25.00	2.200	33.4	56.0	120	300
17PM-K101V	5.00	0.80	6.20	2.200	8.6	56.0	120	300
17PM-K103V	3.60	1.20	3.00	2.200	4.4	56.0	120	300
17PM-K402V	6.00	0.80	7.50	3.400	7.0	75.0	200	350

MICROCONTROLADOR

Features

- AVR - High Performance and Low Power RISC Architecture
- 118 Powerful Instructions - Most Single Clock Cycle Execution
- 8K bytes of In-System Reprogrammable Flash
 - SPI Serial Interface for Program Downloading
 - Endurance: 1,000 Write/Erase Cycles
- 512 bytes EEPROM
 - Endurance: 100,000 Write/Erase Cycles
- 512 bytes Internal SRAM
- 32 x 8 General Purpose Working Registers
- 32 Programmable I/O Lines
- Programmable Serial UART
- SPI Serial Interface
- V_{CC} : 2.7 - 6.0V
- Fully Static Operation
 - 0 - 8 MHz 4.0 - 6.0V,
 - 0 - 4 MHz 2.7 - 4.0V
- Up to 8 MIPS Throughput at 8 MHz
- One 8-Bit Timer/Counter with Separate Prescaler
- One 16-Bit Timer/Counter with Separate Prescaler and Compare and Capture Modes
- Dual PWM
- External and Internal Interrupt Sources
- Programmable Watchdog Timer with On-Chip Oscillator
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes
- Programming Lock for Software Security

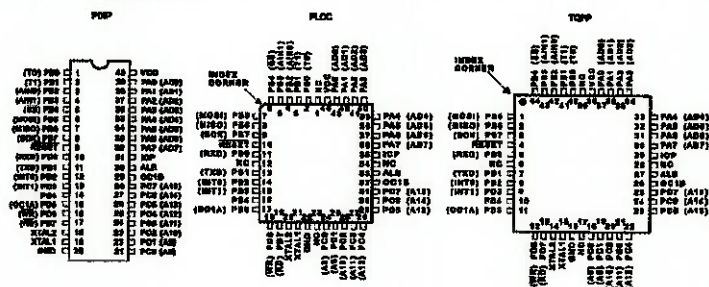
Description

The AT90S8515 is a low-power CMOS 8-bit microcontroller based on the AVR[®] enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S8515 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

(continued)

Pin Configurations



Rev. 0841DS-06/98

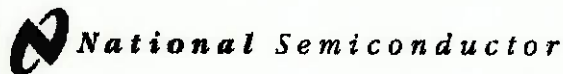


8-Bit **AVR[®]**
Microcontroller
with 8K bytes
In-System
Programmable
Flash

AT90S8515
Preliminary

Note: This is a summary document. For the complete 76 page document, please visit our Web site at www.atmel.com or e-mail literature@atmel.com and request literature number 0841D.

PONTE H



September 1996

LMD18245 3A, 55V DMOS Full-Bridge Motor Driver

General Description

The LMD18245 full-bridge power amplifier incorporates all the circuit blocks required to drive and control current in a brushed type DC motor or one phase of a bipolar stepper motor. The multi-technology process used to build the device combines bipolar and CMOS control and protection circuitry with DMOS power switches on the same monolithic structure. The LMD18245 controls the motor current via a fixed off-time chopper technique.

An all DMOS H-bridge power stage delivers continuous output currents up to 3A (8A peak) at supply voltages up to 55V. The DMOS power switches feature low $R_{DS(ON)}$ for high efficiency, and a diode intrinsic to the DMOS body structure eliminates the discrete diodes typically required to clamp bipolar power stages.

An innovative current sensing method eliminates the power loss associated with a sense resistor in series with the motor. A four-bit digital-to-analog converter (DAC) provides a digital path for controlling the motor current, and, by extension, simplifies implementation of full, half and microstep stepper motor drives. For higher resolution applications, an external DAC can be used.

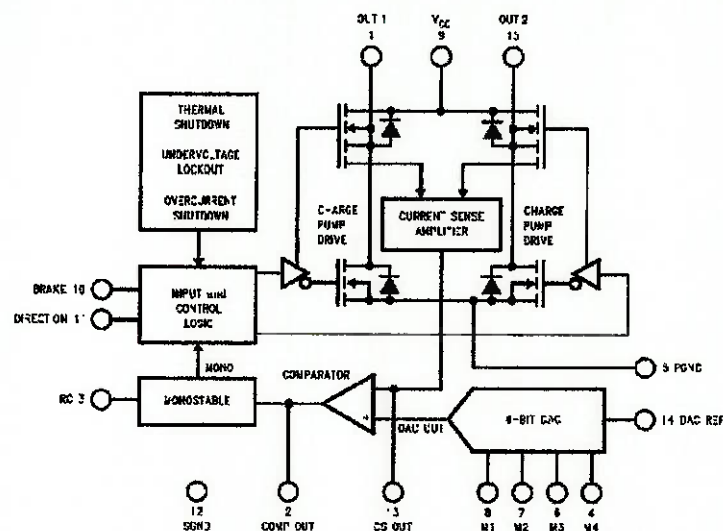
Features

- DMOS power stage rated at 55V and 3A continuous
- Low $R_{DS(ON)}$ of typically 0.3Ω per power switch
- Internal clamp diodes
- Low-loss current sensing method
- Digital or analog control of motor current
- TTL and CMOS compatible inputs
- Thermal shutdown (outputs off) at $T_J = 155^\circ\text{C}$
- Overcurrent protection
- No shoot-through currents
- 15-lead TO-220 molded power package

Applications

- Full, half and microstep stepper motor drives
- Stepper motor and brushed DC motor servo drives
- Automated factory, medical and office equipment

Functional Block and Connection Diagram (15-Lead TO-220 Molded Power Package (T))



Order Number LMD18245T
See NS Package Number TA15A

CONVERSOR D/A



**+3 V/+5 V, Rail-to-Rail
Quad, 8-Bit DAC**

AD7304/AD7305*

FEATURES

Four 8-Bit DACs in One Package
+3 V, +5 V and ± 5 V Operation
Rail-to-Rail REF-input to Voltage Output Swing
2.6 MHz Reference Multiplying Bandwidth
Compact 1.1 mm Height TSSOP 16-/20-Lead Package
Internal Power ON Reset
SPI Serial Interface Compatible—AD7304
Fast Parallel Interface—AD7305
40 μ A Power Shutdown

APPLICATIONS

Automotive Output Span Voltage
Instrumentation, Digitally Controlled Calibration
Pin-Compatible AD7226 Replacement when $V_{DD} < 5.5$ V

GENERAL DESCRIPTION

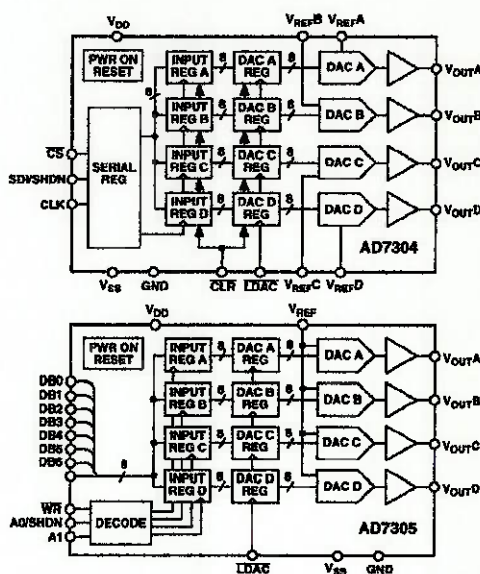
The AD7304/AD7305 are quad, 8-bit DACs that operate from a single +3 V to +5 V supply or ± 5 V supplies. The AD7304 has a serial interface, while the AD7305 has a parallel interface. Internal precision buffers swing rail-to-rail. The reference input range includes both supply rails allowing for positive or negative full-scale output voltages. Operation is guaranteed over the supply voltage range of +2.7 V to +5.5 V, consuming less than 9 mW from a +3 V supply.

The full-scale voltage output is determined by the external reference input voltage applied. The rail-to-rail V_{REF} input to DAC V_{OUT} allows for a full-scale voltage set equal the positive supply V_{DD} , the negative supply V_{SS} or any value in between.

The AD7304's doubled-buffered serial-data interface offers high speed, three-wire, SPI and microcontroller compatible inputs using data in (SDI), clock (CLK) and chip select ($\overline{CS}$) pins. Additionally, an internal power-on reset sets the output to zero scale.

The parallel input AD7305 uses a standard address decode along with the $\overline{WR}$ control line to load data into the input registers. The double buffered architecture allows all four input registers to be preloaded with new values, followed by a $\overline{LDAC}$ control strobe which copies all the new data into the DAC registers thereby updating the analog output values. When operating from less than +5.5 V, the AD7305 is pin-compatible with the popular industry standard AD7226.

FUNCTIONAL BLOCK DIAGRAMS



An internal power ON reset places both parts in the zero-scale state at turn ON. A 40 μ A power shutdown (SHDN) feature is activated on both parts by tristating the SDI/SHDN pin on the AD7304, and tristating the A0/SHDN address pin on the AD7305.

The AD7304/AD7305 are specified over the extended industrial (-40°C to $+85^{\circ}\text{C}$), and the automotive (-40°C to $+125^{\circ}\text{C}$) temperature ranges. AD7304s are available in 16-lead plastic DIP (N-16), and wide-body SOL-16 (R-16) packages. The parallel input AD7305 is available in the 20-lead plastic DIP (N-20), and the SOL-20 (R-20) surface mount package. For ultracompact applications the thin 1.1 mm TSSOP-16 (RU-16) package will be available for the AD7304, while the TSSOP-20 (RU-20) will house the AD7305.

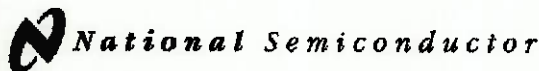
*Protected under Patent Number 5684481.

REV. A

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: <http://www.analog.com>
Fax: 781/326-8703 © Analog Devices, Inc., 1998

DRIVER RS-232



February 1996

DS14C232

Low Power +5V Powered TIA/EIA-232 Dual Driver/Receiver

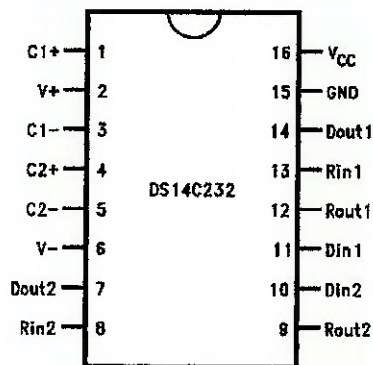
General Description

The DS14C232 is a low power dual driver/receiver featuring an onboard DC to DC converter, eliminating the need for $\pm 12V$ power supplies. The device only requires a +5V power supply. I_{CC} is specified at 3.0 mA maximum, making the device ideal for battery and power conscious applications. The drivers' slew rate is set internally and the receivers feature internal noise filtering, eliminating the need for external slew rate and filter capacitors. The device is designed to interface data terminal equipment (DTE) with data circuit-terminating equipment (DCE). The driver inputs and receiver outputs are TTL and CMOS compatible. DS14C232C driver outputs and receiver inputs meet TIA/EIA-232-E (RS-232) and CCITT V.28 standards.

Features

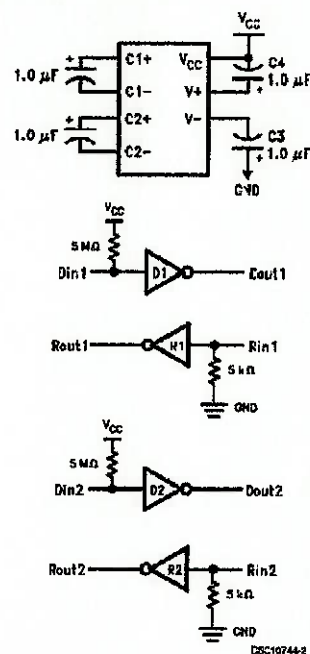
- Pin compatible with industry standard MAX232, LT1081, ICL232 and TSC232
- Single +5V power supply
- Low power — I_{CC} 3.0 mA maximum
- DS14C232C meets TIA/EIA-232-E (RS-232) and CCITT V.28 standards
- CMOS technology
- Receiver Noise Filter
- Package efficiency — 2 drivers and 2 receivers
- Available in Plastic DIP, Narrow and Wide SOIC packages
- TIA/EIA-232 compatible extended temperature range options:
DS14C232T: -40°C to $+85^{\circ}\text{C}$
DS14C232: -55°C to $+125^{\circ}\text{C}$

Connection Diagram



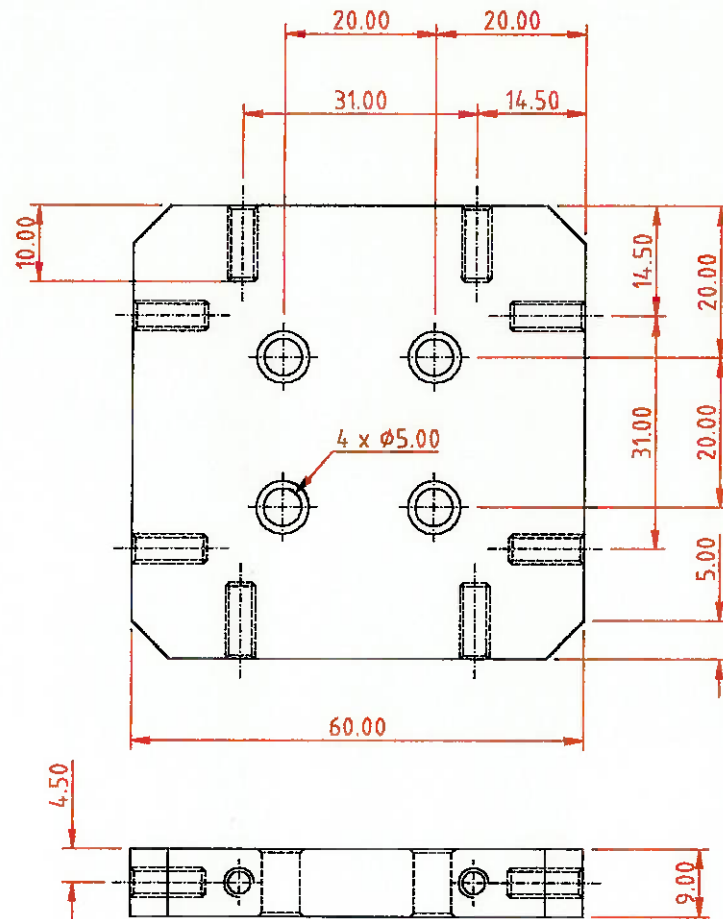
Order Number DS14C232CN, DS14C232TN,
DS14C232CM, DS14C232TM,
DS14C232CWM or DS14C232TWM
See NS Package Number N16A, M16A or M16B

Functional Diagram



Apêndice II: Desenhos

BASES

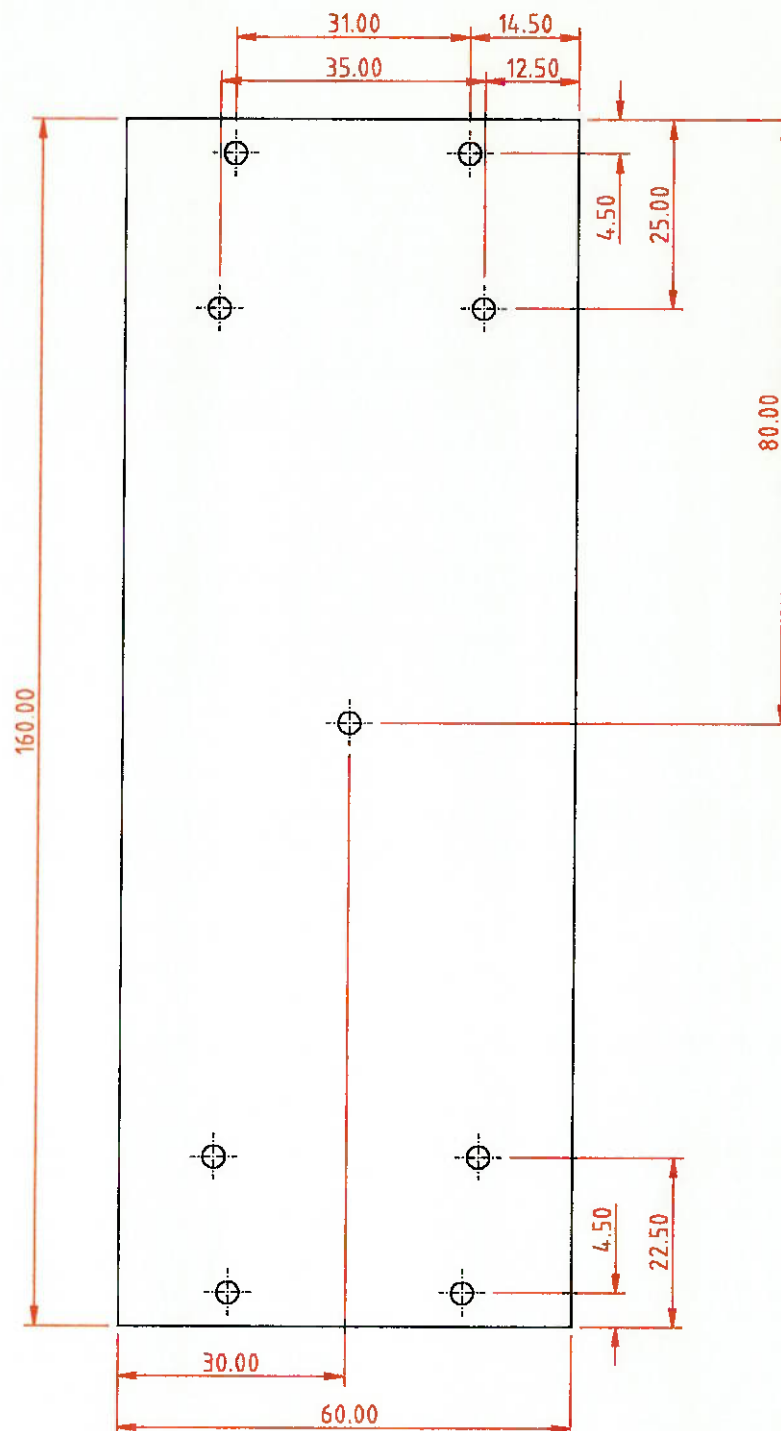


roscas: M3
 escareamentos: 0.5 x 0.5 (45 graus)
 todas as dimensões em mm

parte	quantidade	material	tolerâncias gerais	escala
BASE DA ESTRUTURA	02	AÇO	+/- 0.1 mm	1:1

PAREDE 1

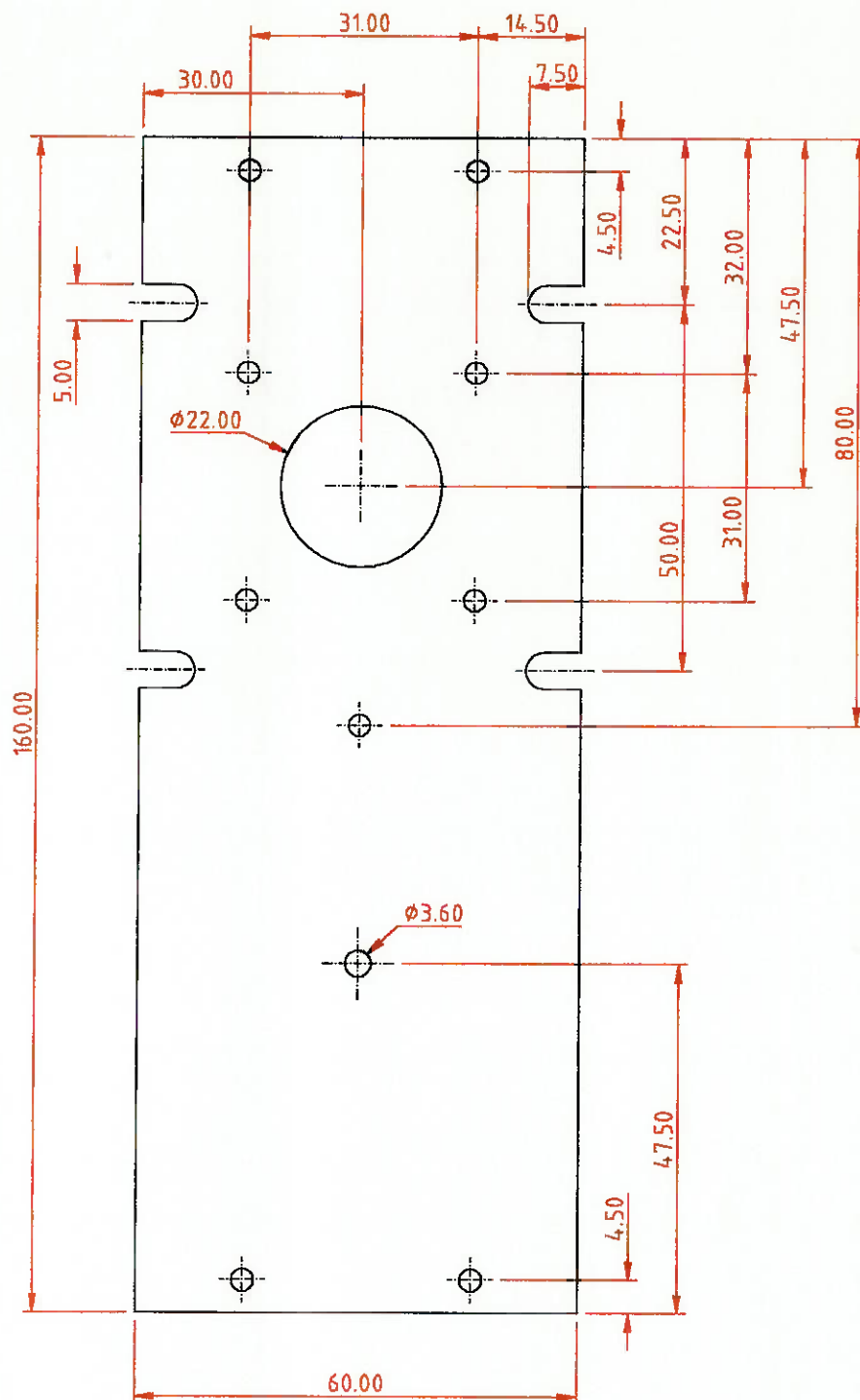
espessura da chapa: 3 mm
furos: 3 mm
escareamentos: 0.5 x 0.5 (45 graus)
todas as dimensões em mm



peça	quantidade	material	tolerâncias gerais	escala
PAREDE 1	01	AÇO	+/- 0.1 mm	1:1

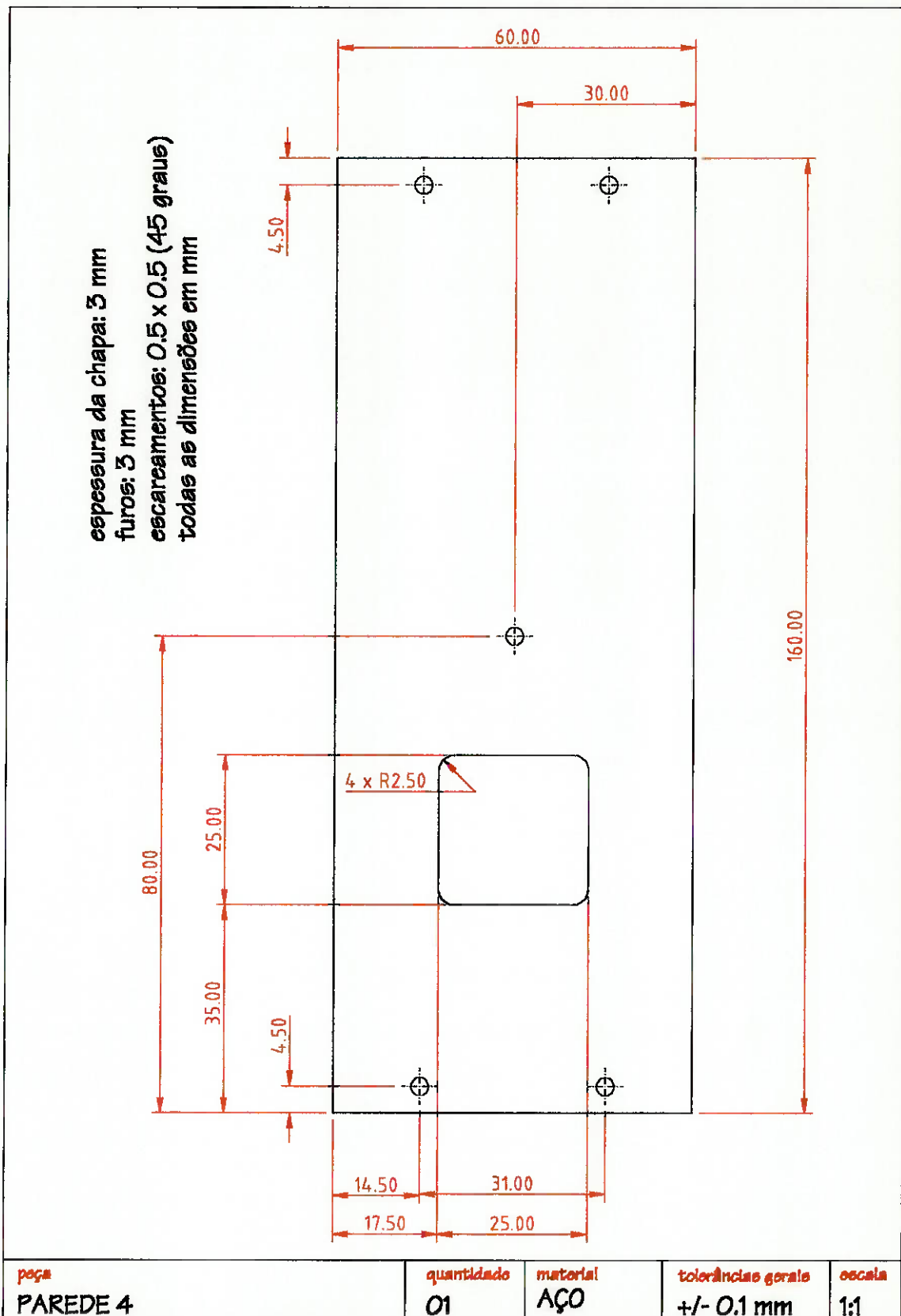
PAREDE 2

espessura da chapa: 3 mm
 furos: 3 mm
 escareamentos: 0.5 x 0.5 (45 graus)
 todas as dimensões em mm

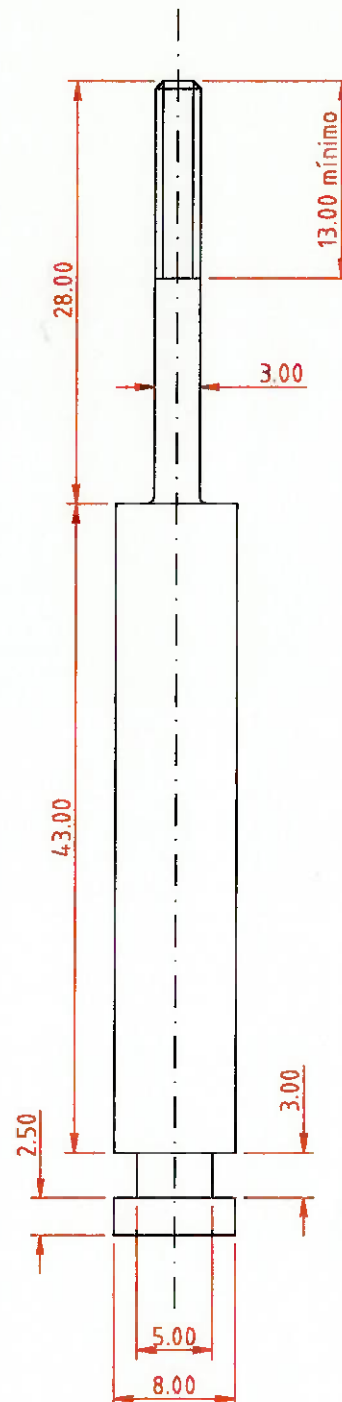


peça	quantidade	material	tolerâncias gerais	escala
PAREDE 2	01	AÇO	+/- 0.1 mm	1:1

PAREDE 4



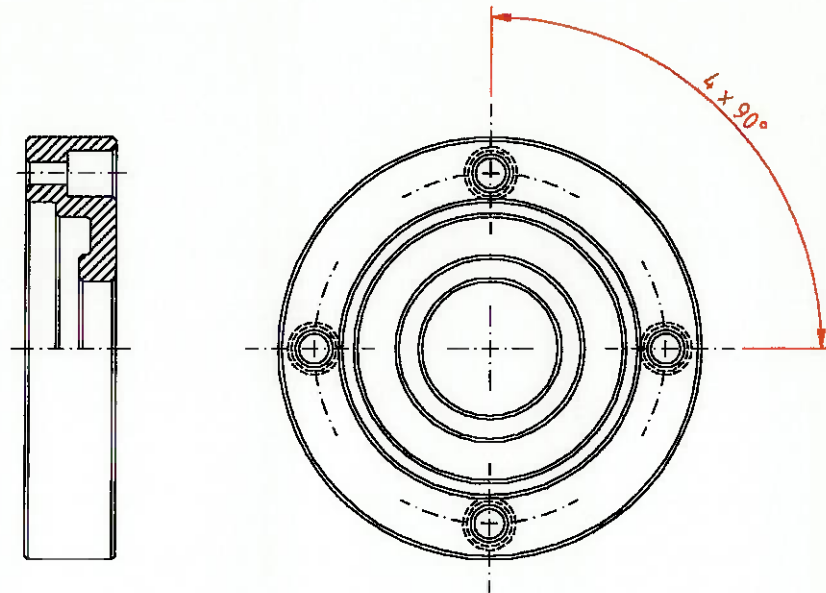
PINOS DOS COOLERS



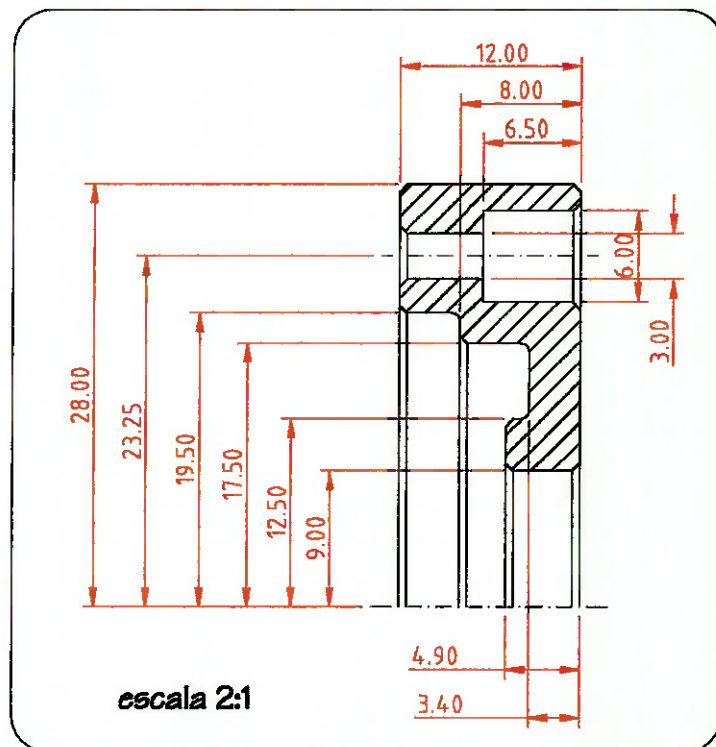
roscas: M3
 escareamentos: 0.5 x 0.5 (45 graus)
 raios de arredondamento: 0.5
 tolerâncias gerais: ± 0.1 mm
 todas as dimensões em mm

peça	quantidade	material	tolerâncias gerais	escala
PINO DO COOLER	08	ALUMÍNIO	± 0.1 mm	2:1

TAMPAS DO CONJUNTO ÓTICO

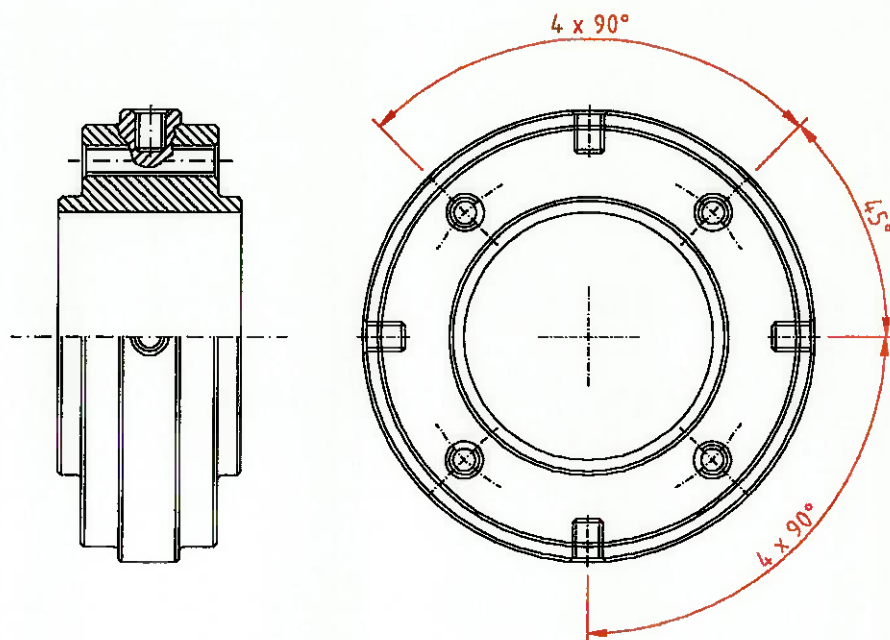


escareamentos: 0.5 x 0.5 (45 graus)
 raios de arredondamento: 0.5
 tolerâncias gerais: ± 0.1 mm
 todas as dimensões em mm

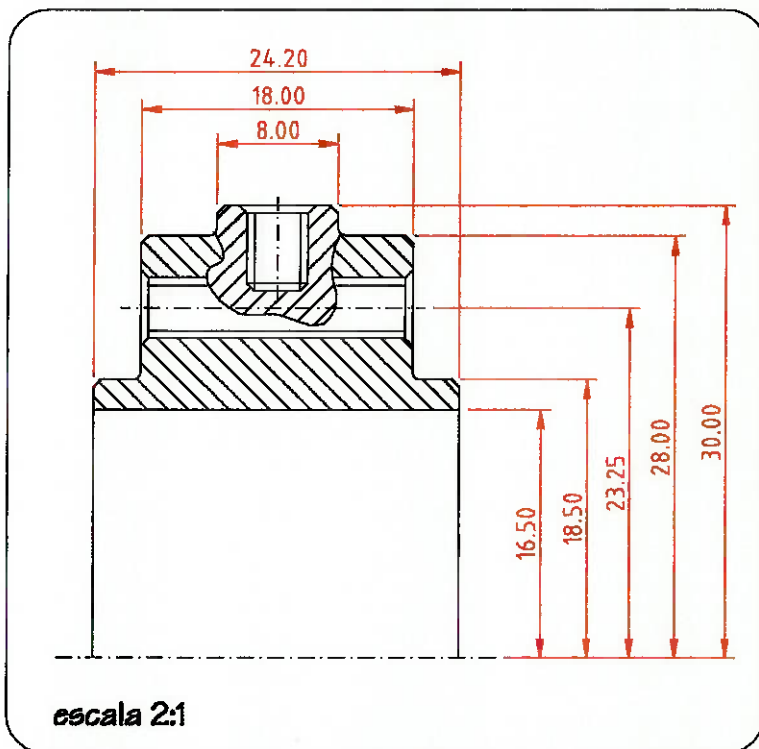


peça	quantidade	material	tolerâncias gerais	escala
TAMPA DO CONJ. ÓTICO	02	AÇO	± 0.1 mm	1:1

TUBO DO CONJUNTO ÓTICO

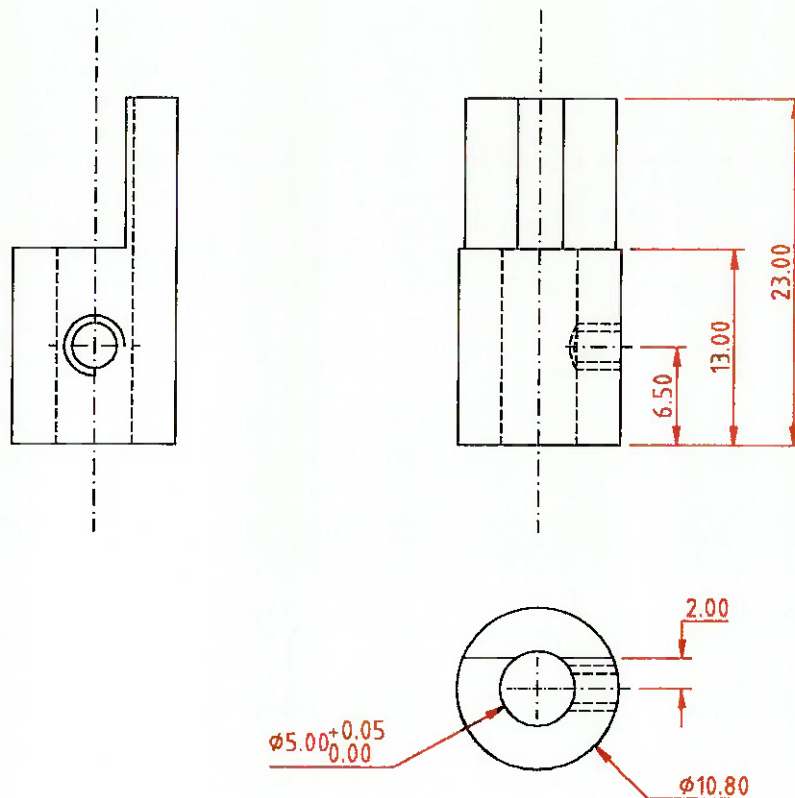


rosca: M3 / prof. mínima 10 mm
 escareamentos: 0.5 x 0.5 (45 graus)
 raios de arredondamento: 0.5
 tolerâncias gerais: +/- 0.1 mm
 todas as dimensões em mm



peça	quantidade	material	tolerâncias gerais	escala
TUBO DO CONJ. ÓTICO	01	AÇO	+/- 0.1 mm	1:1

SUPORTES DOS ESPELHOS



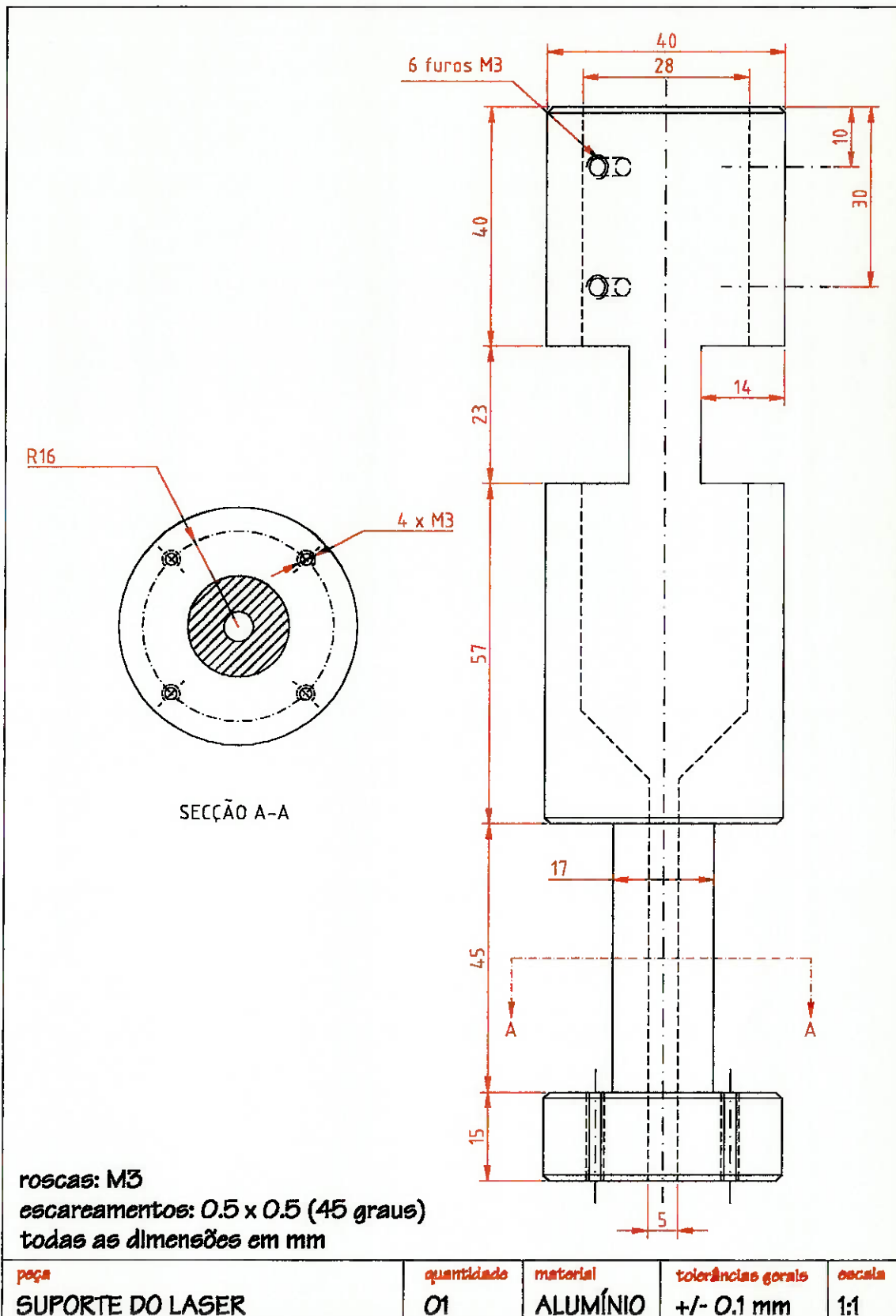
rosca: M3

escareamentos: 0.5 x 0.5 (45 graus)

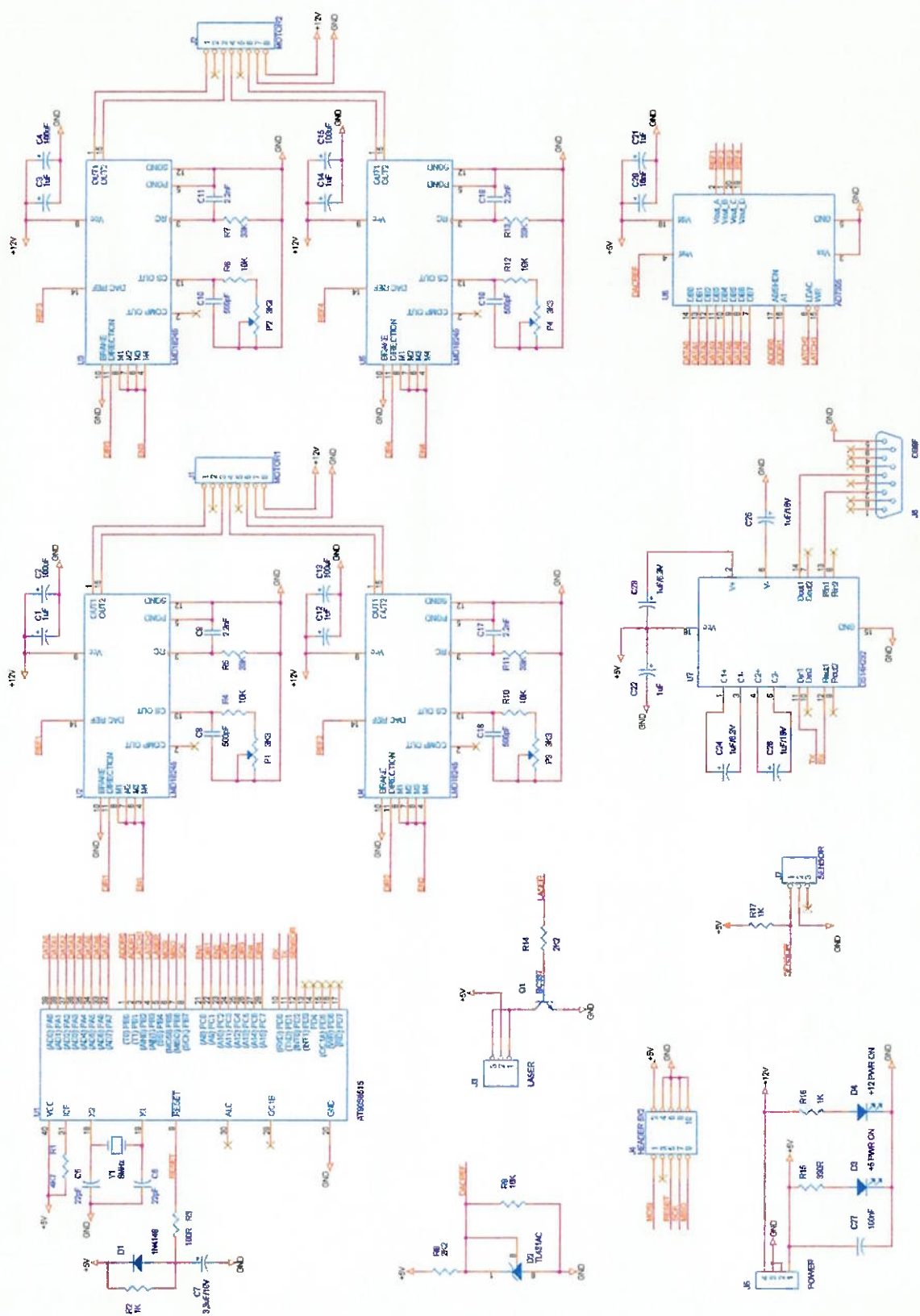
todas as dimensões em mm

parte	quantidade	material	tolerâncias gerais	escala
SUPORTE DO ESPELHO	02	ALUMÍNIO	+/- 0.1 mm	2:1

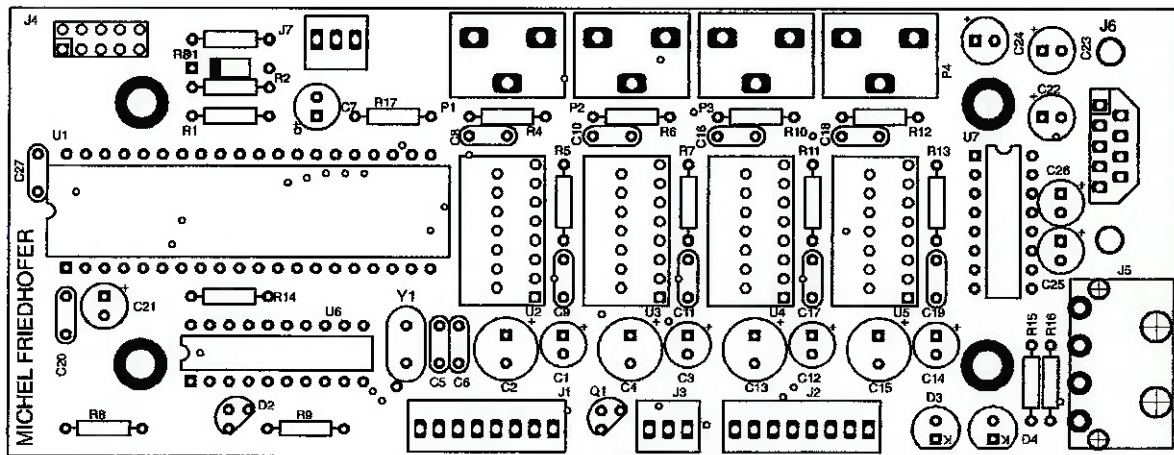
SUPORTE DO LASER



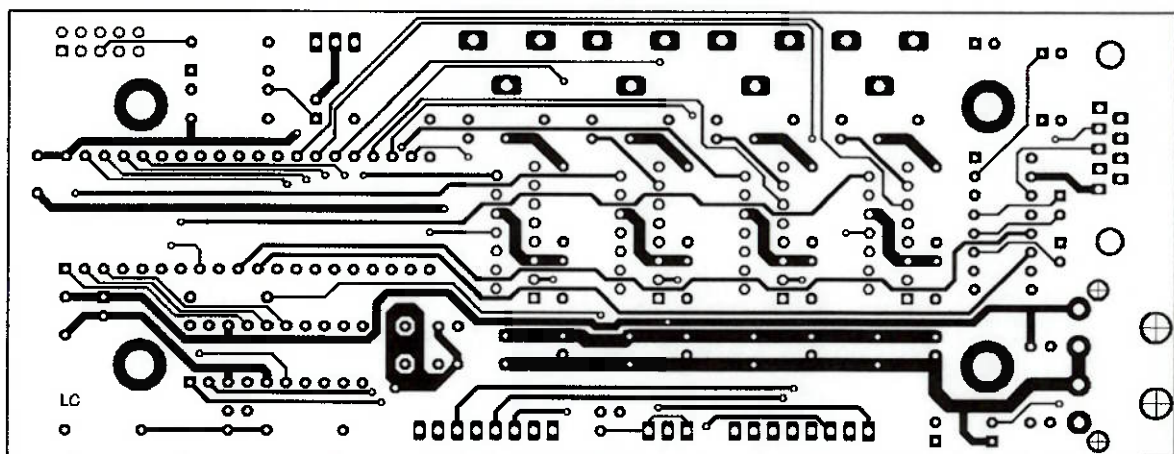
CIRCUITO ELETRÔNICO



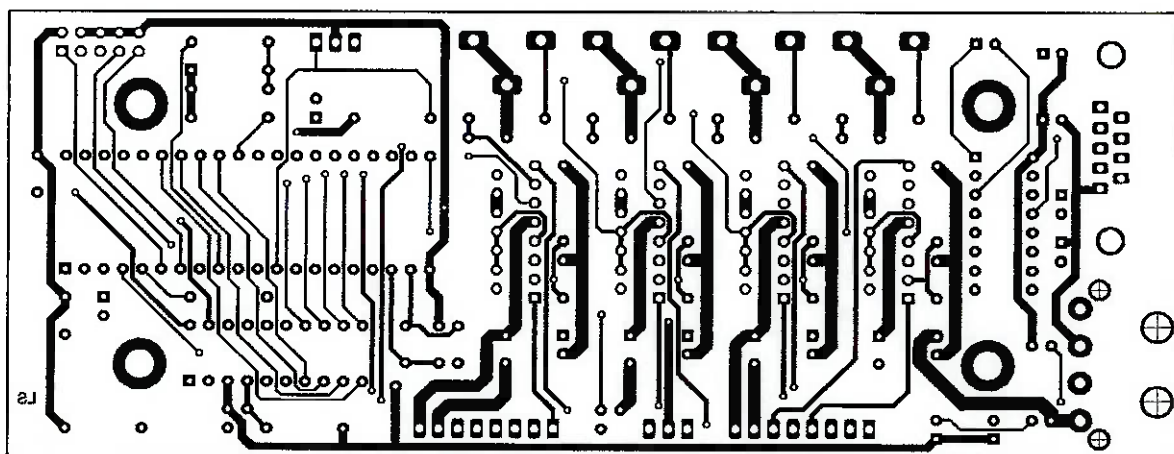
LAYOUT DA PCI



PLACEMENT



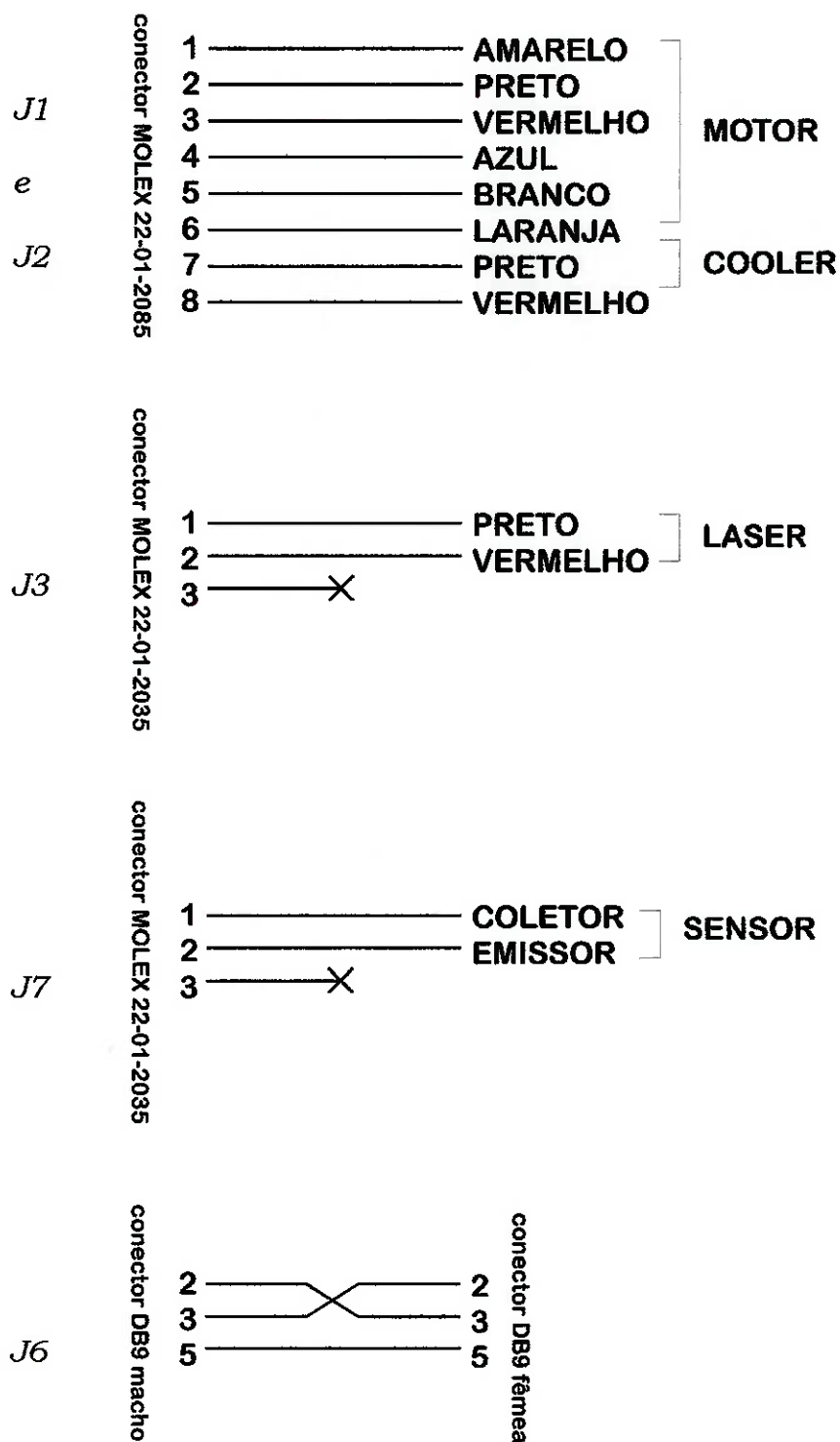
COBRE COMPONENTES



COBRE SOLDA

Apêndice III: Cabos

CABOS



Apêndice IV: Listagem do *Firmware*

```

.include "8515def.inc"

; *****
;
; PROJETO DE FORMATURA
;
;     Defletor de Feixes com
;     Dois Graus de Liberdade
;
;
;             Michel Friedhofer
;             17/11/99
; *****

; =====
; CONSTANTES
; =====

; ***** HARDWARE *****
; port A
; =====
;      label      valor      I/O  descricao
;      =====
.equ   DATA0      =0          ; 0    D/A data 0
.equ   DATA1      =1          ; 0    D/A data 1
.equ   DATA2      =2          ; 0    D/A data 2
.equ   DATA3      =3          ; 0    D/A data 3
.equ   DATA4      =4          ; 0    D/A data 4
.equ   DATA5      =5          ; 0    D/A data 5
.equ   DATA6      =6          ; 0    D/A data 6
.equ   DATA7      =7          ; 0    D/A data 7

; port B
; =====
;      label      valor      I/O  descricao
;      =====
.equ   ADDR0       =0          ; 0    D/A address 0
.equ   ADDR1       =1          ; 0    D/A address 1
.equ   LATCH0      =2          ; 0    D/A latch 0
.equ   LATCH1      =3          ; 0    D/A latch 1
.equ   LASER       =4          ; 0    laser enable

; port C
; =====
;      label      valor      I/O  descricao
;      =====
.equ   ENXA        =0          ; 0    winding XA enable
.equ   DIRXA       =1          ; 0    winding XA direction
.equ   ENXB        =2          ; 0    winding XB enable
.equ   DIRXB       =3          ; 0    winding XB direction
.equ   ENYA        =4          ; 0    winding YA enable
.equ   DIRYA       =5          ; 0    winding YA direction
.equ   ENYB        =6          ; 0    winding YB enable
.equ   DIRYB       =7          ; 0    winding YB direction

; port D
; =====
;      label      valor      I/O  descricao
;      =====
.equ   SENSOR      =2          ; I    sensor de home

; ***** CONSTANTES DOS MOTORES *****

```

	label	endereco	descricao
.equ	PD_CURR	=0x19	; corrente de power-down (10%)
.equ	FAST_PER	=1538	; periodo do movimento rapido (1E6/1538 = 650 pps)
.equ	STD_PER	=2000	; periodo padrao do movimento (1E6/2000 = 500 pps)
.equ	L2R	=0x00	; estado do ultimo movimento: left > right (raster)
.equ	R2L	=0x01	; estado do ultimo movimento: right > left (raster)

; ***** CONSTANTES DA SERIAL *****

	label	endereco	descricao
.equ	Xon	=0x11	; Xon
.equ	Xoff	=0x13	; Xoff
.equ	INIBUFL	=0x7C	; Inicio do buffer (low)
.equ	INIBUFH	=0x00	; Inicio do buffer (high)
.equ	SAFEL	=0xFA	; Inicio da area de seguranca (low)
.equ	SAFEH	=0x01	; Inicio da area de seguranca (high)
.equ	FIMBUFL	=0x0A	; Fim do buffer (low)
.equ	FIMBUFH	=0x02	; Fim do buffer (high)

; =====
; VARIAVEIS
; =====

	label	endereco	descricao
.equ	DC	=0x60	; valor do duty-cycle do laser
.equ	LAST	=0x61	; auxiliar para serial
.equ	XAL	=0x62	; ponteiro para valor do enrolamento XA (low)
.equ	XAH	=0x63	; ponteiro para valor do enrolamento XA (high)
.equ	XBL	=0x64	; ponteiro para valor do enrolamento XB (low)
.equ	XBH	=0x65	; ponteiro para valor do enrolamento XB (high)
.equ	YAL	=0x66	; ponteiro para valor do enrolamento YA (low)
.equ	YAH	=0x67	; ponteiro para valor do enrolamento YA (high)
.equ	YBL	=0x68	; ponteiro para valor do enrolamento YB (low)
.equ	YBH	=0x69	; ponteiro para valor do enrolamento YB (high)
.equ	XfinL	=0x6A	; Bresenham: coordenada X final da linha (low)
.equ	XfinH	=0x6B	; Bresenham: coordenada X final da linha (high)
.equ	YfinL	=0x6C	; Bresenham: coordenada Y final da linha (low)
.equ	YfinH	=0x6D	; Bresenham: coordenada Y final da linha (high)
.equ	deltaXL	=0x6E	; Bresenham: delta X (low)
.equ	deltaXH	=0x6F	; Bresenham: delta X (high)
.equ	deltaYL	=0x70	; Bresenham: delta Y (low)
.equ	deltaYH	=0x71	; Bresenham: delta Y (high)
.equ	RxptrL	=0x72	; Serial: ponteiro IN do buffer (low)
.equ	RxptrH	=0x73	; Serial: ponteiro IN do buffer (high)
.equ	GETptrL	=0x74	; Serial: ponteiro OUT do buffer (low)
.equ	GETptrH	=0x75	; Serial: ponteiro OUT do buffer (high)
.equ	XabsL	=0x76	; referencia para sist. de coord. abs. (X low)
.equ	XabsH	=0x77	; referencia para sist. de coord. abs. (X high)
.equ	YabsL	=0x78	; referencia para sist. de coord. abs. (Y low)
.equ	YabsH	=0x79	; referencia para sist. de coord. abs. (Y high)
.equ	WidthL	=0x7A	; largura da imagem raster (low)
.equ	WidthH	=0x7B	; largura da imagem raster (high)

; =====
; REGISTRADORES
; =====

	label	registrador	descricao
.def	valor	=r0	; valor da LUT de correntes dos enrolamentos
.def	INL	=r1	; aponta para o inicio da LUT (low)

```

.def    INH            =r2            ; aponta para o inicio da LUT (high)
.def    FIL            =r3            ; aponta para o fim da LUT (low)
.def    FIH            =r4            ; aponta para o fim da LUT (high)
.def    errL           =r5            ; Bresenham: erro (low)
.def    errH           =r6            ; Bresenham: erro (high)
.def    cntL           =r7            ; Bresenham: contador de loop (low)
.def    cntH           =r8            ; Bresenham: contador de loop (high)
.def    flag1          =r9            ; flag do temporizador de passos do motor
.def    flag2          =r10           ; flag do sensor de home
.def    temp1          =r16           ; auxiliar
.def    temp2          =r17           ; auxiliar
.def    XA             =r18           ; shadow do enrolamento XA
.def    XB             =r19           ; shadow do enrolamento XB
.def    YA             =r20           ; shadow do enrolamento YA
.def    YB             =r21           ; shadow do enrolamento YB
.def    shaB           =r22           ; shadow do port B
.def    shaC           =r23           ; shadow do port C
.def    res            =r24           ; resultado da funcao sign
.def    stat           =r25           ; Bresenham: auxiliar
.def    XL             =r26           ; coordenada X atual (low)
.def    XH             =r27           ; coordenada X atual (high)
.def    YL             =r28           ; coordenada Y atual (low)
.def    YH             =r29           ; coordenada Y atual (high)

```

```

; =====
; VETORES DE INTERRUPCAO
; =====

```

```

.org    0x0000
        rjmp    BOOT
.org    INT0addr
        rjmp    INT_SENSOR
.org    OC1Aaddr
        rjmp    INT_TMRO
.org    OVF0addr
        rjmp    INT_TMRI
.org    URXCaddr
        rjmp    INT_SERIAL

```

```

; =====
; MACROS
; =====

```

```

.macro  UPDATE_XA                                ; atualiza enrolamento XA
        cbr     shaB, (1<<ADDR0)+(1<<ADDR1)
        out     PORTB, shaB
        out     PORTA, XA
        cbr     shaB, (1<<LATCH0)
        out     PORTB, shaB
        sbr     shaB, (1<<LATCH0)
        out     PORTB, shaB
.endmacro

.macro  UPDATE_XB                                ; atualiza enrolamento XB
        cbr     shaB, (1<<ADDR1)
        sbr     shaB, (1<<ADDR0)
        out     PORTB, shaB
        out     PORTA, XB
        cbr     shaB, (1<<LATCH0)
        out     PORTB, shaB
        sbr     shaB, (1<<LATCH0)

```

```

        out        PORTB, shaB
    .endmacro

    .macro UPDATE_YA                                ; atualiza enrolamento YA
        cbr        shaB, (1<<ADDR0)
        sbr        shaB, (1<<ADDR1)
        out        PORTB, shaB
        out        PORTA, YA
        cbr        shaB, (1<<LATCH0)
        out        PORTB, shaB
        sbr        shaB, (1<<LATCH0)
        out        PORTB, shaB
    .endmacro

    .macro UPDATE_YB                                ; atualiza enrolamento YB
        sbr        shaB, (1<<ADDR0)+(1<<ADDR1)
        out        PORTB, shaB
        out        PORTA, YB
        cbr        shaB, (1<<LATCH0)
        out        PORTB, shaB
        sbr        shaB, (1<<LATCH0)
        out        PORTB, shaB
    .endmacro

    .MACRO incz                                    ; incrementa Z
        subi       ZL, 0xFF
        sbci       ZH, 0xFF
    .ENDMACRO

    .MACRO decz                                    ; decrementa Z
        subi       ZL, 0x01
        sbci       ZH, 0x00
    .ENDMACRO

; =====
; CORPO PRINCIPAL
; =====

; rotina de inicializacao
;
; inicializa parametros internos, registradores e variaveis

BOOT:
        cli                                ; DI geral
        ser        temp1
        out        DDRA, temp1            ; porta de dados
        out        DDRB, temp1
        out        DDRC, temp1            ; porta de comando
        clr        temp1
        out        DDRD, temp1            ; porta de sensores
        out        PORTA, temp1            ; PORTA acorda em 0x00
        ser        shaC                    ; PORTD acorda em 0xFF
        out        PORTC, shaC
        ldi        shaB, (1<<LATCH0)+(1<<LATCH1)
        out        PORTB, shaB
        ldi        temp1, (1<<ACD)        ; desliga analog comparator
        out        ACSR, temp1
        ldi        temp1, high(RAMEND)    ; inicializa SP
        out        SPH, temp1
        ldi        temp1, low(RAMEND)
        out        SPL, temp1
        ldi        temp1, (1<<ISC11)+(1<<ISC01) ; no SRAM/SRW/SLEEP/SM INTs = falling edge

```

```

out    MCUCR, temp1
clr    temp1
out    TCNT1H, temp1                ; timer 1 = 0
out    TCNT1L, temp1
out    TCCR1A, temp1                ; timer 1: no PWM, disconnected from OC1x
ldi    temp1, (1<<CS01)
out    TCCR0, temp1                ; timer 0: CK/8
ldi    temp1, (1<<CS11)+(1<<CTC1)   ; timer 1: CK/8, clear on match
out    TCCR1B, temp1
ldi    temp1, high(STD_PER)         ; periodo padrao de movimento (500 pps)
out    OCR1AH, temp1
ldi    temp1, low(STD_PER)
out    OCR1AL, temp1
clr    temp1                        ; INTs externas = off
out    GIMSK, temp1
ldi    temp1, (1<<OCIE1A)+(1<<TOIE0) ; matchA, INT_TMR1 = enabled
out    TIMSK, temp1
ldi    temp1, 51                    ; baud rate = 9600
out    UBRR, temp1
ldi    temp1, (1<<TXEN)+(1<<RXEN)+(1<<RXCIE)
out    UCR, temp1

clr    temp1                        ; inicializa referencia do sist. absoluto
sts    XabsL, temp1
sts    XabsH, temp1
sts    YabsL, temp1
sts    YabsH, temp1

ldi    ZH, high(faseA)              ; inicializa ponteiros da LUT
ldi    ZL, low(faseA)
lsl    ZL
rol    ZH
decz
mov    INL, ZL
mov    INH, ZH
ldi    ZH, high(fimfase)
ldi    ZL, low(fimfase)
lsl    ZL
rol    ZH
mov    FIL, ZL
mov    FIH, ZH
ldi    ZH, high(faseA)
ldi    ZL, low(faseA)
lsl    ZL
rol    ZH
sts    XAL, ZL
sts    XAH, ZH
sts    YAL, ZL
sts    YAH, ZH
lpm
mov    XA, valor
mov    YA, valor
ldi    ZH, high(faseB)
ldi    ZL, low(faseB)
lsl    ZL
rol    ZH
sts    XBL, ZL
sts    XBH, ZH
sts    YBL, ZL
sts    YBH, ZH
lpm
mov    XB, valor
mov    YB, valor

```

```

ldi    ZL, INIBUFL                ; inicializa buffer de recepcao
ldi    ZH, INIBUFH
sts    RXPtL, ZL
sts    RXPtH, ZH
sts    GETPtL, ZL
sts    GETPtH, ZH
ldi    temp1, Xon
sts    LAST, temp1

sei                                ; EI

clr    temp1                      ; desliga laser
sts    DC, temp1

rcall  PWRDWN                    ; power down
rjmp   CMD

; rotina de interpretacao de comandos da serial
;
; roda em loop infinito, interpretando
; e executando cada comando recebido

CMD:
rcall  GETBYTE                  ; le byte do buffer
cpi    temp1, 0x52              ; RESET?
bme    CMD_2
rjmp   BOOT

CMD_2:
cpi    temp1, 0x48              ; HOME?
bme    CMD_3
rcall  HOME
rjmp   CMD

CMD_3:
cpi    temp1, 0x50              ; POWER DOWN?
bme    CMD_4
rcall  PWRDWN
rjmp   CMD

CMD_4:
cpi    temp1, 0x42              ; BRIGHTNESS?
bme    CMD_5
rcall  GETPARAM
sts    DC, temp2
rjmp   CMD

CMD_5:
cpi    temp1, 0x4F              ; ORIGIN?
bme    CMD_6
lds    temp1, XabsL              ; preserva sist. absoluto
lds    temp2, XabsH
add    temp1, XL
adc    temp2, XH
sts    XabsL, temp1
sts    XabsH, temp2
lds    temp1, YabsL
lds    temp2, YabsH
add    temp1, YL
adc    temp2, YH
sts    YabsL, temp1
sts    YabsH, temp2
clr    XL
clr    XH
clr    YL
clr    YH
rjmp   CMD

```

```

CMD_6:
    cpi    temp1, 0x46                ; FAST?
    bne    CMD_7
    lds    temp1, DC                  ; guarda duty-cycle
    push   temp1
    clr    temp1                      ; desliga laser
    sts    DC, temp1
    in     temp1, OCR1AL               ; guarda periodo
    push   temp1
    in     temp1, OCR1AH
    push   temp1
    ldi    temp1, high(FAST_PER)      ; programa periodo curto
    out    OCR1AH, temp1
    ldi    temp1, low(FAST_PER)
    out    OCR1AL, temp1
    rcall  GETPARAM                   ; pega coord. X
    rcall  CONVPARAM                  ; corrige MSN para nums. negativos
    sts    XfinH, temp1
    sts    XfinL, temp2
    rcall  GETPARAM                   ; pega coord. Y
    rcall  CONVPARAM                  ; corrige MSN para nums. negativos
    sts    YfinH, temp1
    sts    YfinL, temp2
    rcall  LINHA                      ; executa movimento
    pop    temp1                      ; recupera periodo
    out    OCR1AH, temp1
    pop    temp1
    out    OCR1AL, temp1
    pop    temp1                      ; recupera DC
    sts    DC, temp1
    rjmp   CMD

CMD_7:
    cpi    temp1, 0x4C                ; LINE?
    bne    CMD_8
    rcall  GETPARAM                   ; pega coord. X
    rcall  CONVPARAM                  ; corrige MSN para nums. negativos
    sts    XfinH, temp1
    sts    XfinL, temp2
    rcall  GETPARAM                   ; pega coord. Y
    rcall  CONVPARAM                  ; corrige MSN para nums. negativos
    sts    YfinH, temp1
    sts    YfinL, temp2
    rcall  LINHA                      ; desenha segmento
    rjmp   CMD

CMD_8:
    cpi    temp1, 0x53                ; SPEED?
    bne    CMD_9
    rcall  GETPARAM                   ; pega periodo (x 2 us)
    clc                                       ; transforma em (x 1 us)
    rol    temp2
    rol    temp1
    out    OCR1AH, temp1               ; programa periodo
    out    OCR1AL, temp2
    rjmp   CMD

CMD_9:
    cpi    temp1, 0x41                ; ABSOLUTE?
    bne    CMD_10
    lds    temp1, XabsL               ; recupera sistema absoluto
    lds    temp2, XabsH
    add    XL, temp1
    adc    XH, temp2
    lds    temp1, YabsL
    lds    temp2, YabsH
    add    YL, temp1

```

```

        adc     YH, temp2
        clr     temp1
        sts     XabsL, temp1
        sts     XabsH, temp1
        sts     YabsL, temp1
        sts     YabsH, temp1
        rjmp    CMD
CMD_10:
        cpi     temp1, 0x49                ; IMAGE?
        brne    CMD_11
        rcall   GETPARAM                  ; pega tamanho da linha
        sts     WidthL, temp2
        sts     WidthH, temp1
        clr     temp1
        sts     DC, temp1
        ldi     temp2, R2L
        rcall   IMAGE
        rjmp    CMD
CMD_11:
        rjmp    CMD

; rotina de projecao de imagens raster
IMAGE:
        push    temp2
        rcall   GETBYTE
        cpi     temp1, 0x2B                ; "+"?
        breq    I_L2R
        cpi     temp1, 0x2D                ; "-"?
        breq    I_R2L
        pop     temp2
        ret
I_L2R:
        pop     temp2
        cpi     temp2, R2L                ; determina sentido da ultima linha
        breq    I_L2R_2
        lds     temp1, WidthL
        lds     temp2, WidthH
I_L2R_3:
        push    temp1
        push    temp2
        rcall   XPREV
        subi    XL, 0x01
        sbci    XH, 0x00
        rcall   UPDATE_DAC
        rcall   PERIOD
        pop     temp2
        pop     temp1
        subi    temp1, 0x01
        sbci    temp2, 0x00
        tst     temp1
        brne    I_L2R_3
        tst     temp2
        brne    I_L2R_3
        rjmp    I_L2R_4
I_L2R_2:
        rcall   XNEXT
        subi    XL, 0xFF
        sbci    XH, 0xFF
        rcall   UPDATE_DAC
I_L2R_4:
        rcall   YPREV
        subi    YL, 0x01

```

```

        sbci    YH, 0x00
        rcall   UPDATE_DAC
        rcall   PERIOD
        lds     temp1, WidthL
        lds     temp2, WidthH
I_L2R_LOOP:
        push    temp1
        push    temp2
        rcall   GETBYTE                ; pega proximo pixel
        sts     DC, temp1              ; e imprime
        rcall   XNEXT
        subi    XL, 0xff
        sbci    XH, 0xff
        rcall   UPDATE_DAC
        rcall   PERIOD
        pop     temp2
        pop     temp1
        subi    temp1, 0x01
        sbci    temp2, 0x00
        tst     temp1
        brne    I_L2R_LOOP
        tst     temp2
        brne    I_L2R_LOOP
        clr     temp1                  ; desliga laser
        sts     DC, temp1
        ldi     temp2, L2R             ; ultimo movto. = L2R
        rjmp    IMAGE

I_R2L:
        pop     temp2
        cpi     temp2, L2R             ; determina sentido da ultima linha
        breq    I_R2L_2
        lds     temp1, WidthL
        lds     temp2, WidthH
I_R2L_3:                                ; mesmo sentido -> volta para comeco
        push    temp1
        push    temp2
        rcall   XNEXT
        subi    XL, 0xFF
        sbci    XH, 0xFF
        rcall   UPDATE_DAC
        rcall   PERIOD
        pop     temp2
        pop     temp1
        subi    temp1, 0x01
        sbci    temp2, 0x00
        tst     temp1
        brne    I_R2L_3
        tst     temp2
        brne    I_R2L_3
        rjmp    I_R2L_4
I_R2L_2:                                ; sentido oposto -> reposiciona
        rcall   XPREV
        subi    XL, 0x01
        sbci    XH, 0x00
        rcall   UPDATE_DAC
I_R2L_4:                                ; retrocede 1 linha
        rcall   YPREV
        subi    YL, 0x01
        sbci    YH, 0x00
        rcall   UPDATE_DAC
        rcall   PERIOD
        lds     temp1, WidthL

```

```

        lds      temp2, WidthH
I_R2L_LOOP:
        push    temp1
        push    temp2
        rcall   GETBYTE                ; pega proximo pixel
        sts     DC, temp1              ; e imprime
        rcall   XPREV
        subi    XL, 0x01
        sbci    XH, 0x00
        rcall   UPDATE_DAC
        rcall   PERIOD
        pop     temp2
        pop     temp1
        subi    temp1, 0x01
        sbci    temp2, 0x00
        tst     temp1
        bme     I_R2L_LOOP
        tst     temp2
        bme     I_R2L_LOOP
        clr     temp1                  ; desliga laser
        sts     DC, temp1
        ldi     temp2, R2L             ; ultimo movto. = R2L
        rjmp    IMAGE

; rotina de homing automatico
;
; ao final, o sistema de coords. e' o absoluto
; e a velocidade e' a padrao

HOME:
        clr     temp1
        out     TCRC0, temp1           ; timer 0: desligado

        ldi     temp1, high(STD_PER)   ; velocidade padrao
        out     OCR1AH, temp1
        ldi     temp1, low(STD_PER)
        out     OCR1AL, temp1

        clr     temp1                  ; retrocede 500 pontos em X
        sts     XfinL, temp1
        sts     XfinH, temp1
        sts     YfinL, temp1
        sts     YfinH, temp1
        ldi     XL, low(500)
        ldi     XH, high(500)
        clr     YL
        clr     YH
        rcall   LINHA

        sbr     shaB, (1<<LASER)      ; liga laser
        out     PORTB, shaB

        ldi     temp1, (1<<ISC01)      ; habilita interrupcao do fototransistor
        out     MCUCR, temp1
        ldi     temp1, (1<<INT0)
        out     GIMSK, temp1
        nop
        clr     temp1
        mov     flag2, temp1           ; limpa flag da interrupcao

HOME0:                                     ; varredura "coarse" =

```

```

        ldi        temp1, 64                                ; 1 passo em X para cada 100 passos em Y
HOME1:   push       temp1
        rcall     XNEXT
        rcall     UPDATE_DAC
        pop       temp1
        dec       temp1
        brne      HOME1

        ldi        temp1, 100
HOME2:   push       temp1
        ldi        temp1, 64
HOME3:   push       temp1
        rcall     YNEXT
        rcall     UPDATE_DAC
        pop       temp1
        dec       temp1
        brne      HOME3

        rcall     UPDATE_DAC
        rcall     PERIOD
        mov       temp1, flag2                            ; verifica se houve interrupcao
        tst       temp1
        brne      HOME4
        pop       temp1
        dec       temp1
        brne      HOME2
        rjmp      HOME0

HOME4:   pop        temp1                                    ; balanceia stack

        clr       temp1                                    ; desabilita interrupcao do fototransistor
        out       GIMSK, temp1

        ldi       temp1, high(1000)
        out       OCR1AH, temp1
        ldi       temp1, low(1000)
        out       OCR1AL, temp1

HOME5:   in        temp1, PIND                                ; como estava em velocidade alta, pode
        sbrs      temp1, SENSOR                            ; ter ocorrido overshoot - por seguranca,
        rjmp      HOME6                                    ; retrocede ate o feixe bater no sensor
        rcall     YPREV
        rcall     UPDATE_DAC
        rcall     PERIOD
        rjmp      HOME5

HOME6:   rcall     BUSCAX                                    ; busca centro do sensor em X
        rcall     BUSCAY                                    ; busca centro do sensor em Y
        rcall     BUSCAX                                    ; busca centro do sensor em X
        rcall     BUSCAY                                    ; busca centro do sensor em Y
        rcall     BUSCAX                                    ; busca centro do sensor em X
        rcall     BUSCAY                                    ; busca centro do sensor em Y
        rcall     BUSCAX                                    ; busca centro do sensor em X

HOME7:   ldi       temp1, high(400)                        ; velocidade = 1250 pps
        out       OCR1AH, temp1
        ldi       temp1, low(400)
        out       OCR1AL, temp1

```

```

    cbr    shaB, (1<<LASER)           ; desliga laser
    out    PORTB, shaB

    clr    temp1                       ; gira o feixe 180 graus em torno de X
    sts    XfimL, temp1
    sts    XfimH, temp1
    ldi    temp1, 128
    sts    YfimL, temp1
    ldi    temp1, 12
    sts    YfimH, temp1
    clr    XL
    clr    XH
    clr    YL
    clr    YH
    rcall  LINHA

    ldi    XL, 250                     ; ao fim do processo, o feixe esta no
    clr    XH                         ; centro da area de trabalho (X = Y = 250)
    ldi    YL, 250
    clr    YH

    ldi    temp1, (1<<CS01)
    out    TCCR0, temp1               ; timer 0: CK/8

    ldi    temp1, high(STD_PER)       ; programa velocidade padrao
    out    OCRIAH, temp1
    ldi    temp1, low(STD_PER)
    out    OCRIAL, temp1

    ret

; rotina de busca centro do sensor no eixo X

BUSCAX:
BX1:
    in     temp1, PIND                ; avanca em X ate sair do sensor
    sbrc   temp1, SENSOR
    rjmp   BX2
    rcall  XNEXT
    rcall  UPDATE_DAC
    rcall  PERIOD
    rjmp   BX1

BX2:
    in     temp1, PIND                ; retrocede em X ate entrar no sensor
    sbrc   temp1, SENSOR
    rjmp   BX3
    rcall  XPREV
    rcall  UPDATE_DAC
    rcall  PERIOD
    rjmp   BX2

BX3:
    clr    XL                         ; na proxima passada, conto para depois
    clr    XH                         ; parar no meio da "largura" do sensor

BX4:
    subi   XL, 0xFF                  ; inc X
    sbci   XH, 0xFF
    rcall  XPREV                     ; avanca em X ate sair do sensor
    rcall  UPDATE_DAC
    rcall  PERIOD
    in     temp1, PIND
    sbrc   temp1, SENSOR

```

```

        rjmp    BX5
        rjmp    BX4
BX5:
        clc
        ror     XH           ; X /= 2
        ror     XL
BX6:
        rcall   XNEXT           ; retrocede ate o meio da "largura"
        rcall   UPDATE_DAC
        rcall   PERIOD
        subi    XL, 0x01
        sbci    XH, 0x00
        tst     XL
        bme     BX6
        tst     XH
        bme     BX6
        ret

; rotina de busca centro do sensor em Y

BUSCAY:
BY1:
        in      temp1, PIND           ; avanca em Y ate sair do sensor
        sbrc    temp1, SENSOR
        rjmp    BY2
        rcall   YNEXT
        rcall   UPDATE_DAC
        rcall   PERIOD
        rjmp    BY1

BY2:
        in      temp1, PIND           ; retrocede em Y ate entrar no sensor
        sbrc    temp1, SENSOR
        rjmp    BY3
        rcall   YPREV
        rcall   UPDATE_DAC
        rcall   PERIOD
        rjmp    BY2

BY3:
        clr     YL           ; na proxima passada, conto para depois
        clr     YH           ; parar no meio da "altura" do sensor

BY4:
        subi    YL, 0xFF           ; inc Y
        sbci    YH, 0xFF
        rcall   YPREV           ; avanca em Y ate sair do sensor
        rcall   UPDATE_DAC
        rcall   PERIOD
        in      temp1, PIND
        sbrc    temp1, SENSOR
        rjmp    BY5
        rjmp    BY4

BY5:
        clc
        ror     YH           ; Y /= 2
        ror     YL

BY6:
        rcall   YNEXT           ; retrocede ate o meio da "altura"
        rcall   UPDATE_DAC
        rcall   PERIOD
        subi    YL, 0x01
        sbci    YH, 0x00
        tst     YL
        bme     BY6

```

```

    tst     YH
    brne    BY6
    ret

; rotina de interrupcao do sensor
;
; apenas seta a flag que indica que houve interrupcao

INT_SENSOR:
    push    temp1
    ser     temp1
    mov     flag2, temp1
    pop     temp1
    reti

; rotina de espera de 1 periodo
;
; faz o polling da flag que e' alterada
; pela interrupcao do timer 0

PERIOD:
    push    temp1
    ser     temp1
    mov     flag1, temp1
PERIOD_1:
    mov     temp1, flag1
    tst     temp1
    brne    PERIOD_1
    pop     temp1
    ret
; ja passou 1 periodo?
; se nao -> volta
; se passou -> fim

; rotina de interrupcao do timer 0
;
; apenas seta a flag correspondente

INT_TMR0:
    push    temp1
    in      temp1, SREG
    push    temp1
    clr     temp1
    mov     flag1, temp1
    pop     temp1
    out     SREG, temp1
    pop     temp1
    reti

; rotina de interrupcao do timer 1
;
; executa o PWM do laser

INT_TMR1:
    cli
    push    temp1
    in      temp1, SREG
    push    temp1

    lds     temp1, DC
    sbrc    shaB, LASER
    rjmp    LIGA
    ; decide em que parte do PWM
    ; estou, HIGH ou LOW
DESLIGA:
    ; DI

```

```

        cpi    temp1, 0xFF
        breq   INT_TMR1_FIM
        out    TCNT0, temp1
        cbr    shaB, (1<<LASER)
        out    PORTB, shaB
        rjmp   INT_TMR1_FIM
LIGA:
        cpi    temp1, 0x00
        breq   INT_TMR1_FIM
        com    temp1
        out    TCNT0, temp1
        sbr    shaB, (1<<LASER)
        out    PORTB, shaB
INT_TMR1_FIM:
        pop    temp1
        out    SREG, temp1
        pop    temp1
        sei
        reti

; rotina de power down
;
; desliga o laser, busca um full step proximo
; e reduz a corrente dos motores

PWRDWN:
        clr    temp1
        sts    DC, temp1
; desliga laser

PWD0:
        tst    XA
        breq   PWD1
        tst    XB
        breq   PWD1
        rcall  XNEXT
        subi   XL, 0xFF
        sbci   XH, 0xFF
        rjmp   PWD0
; X em full step?
; nao -> avanca
; atualiza coordenadas para nao se perder

PWD1:
        tst    YA
        breq   PWD2
        tst    YB
        breq   PWD2
        rcall  YNEXT
        subi   YL, 0xFF
        sbci   YH, 0xFF
        rjmp   PWD1
; Y em full step?
; nao -> avanca
; atualiza coordenadas para nao se perder

PWD2:
        clr    temp1
        cpse   XA, temp1
        ldi    XA, PD_CURR
        cpse   XB, temp1
        ldi    XB, PD_CURR
        cpse   YA, temp1
        ldi    YA, PD_CURR
        cpse   YB, temp1
        ldi    YB, PD_CURR
        rcall  UPDATE_DAC
        ret
; os enrolamentos que estiverem ligados em
; 100% serao ligados em 10%, para reduzir
; consumo e aquecimento

; rotina de atualizado dos covnersores D/A
;
; atualiza todo o PORTC, alem de atualizar os

```

; valores nos buffers internos do D/A

```

UPDATE_DAC:
    cli
    push    temp1
    clr     temp1
    cbr     shaC, (1<<ENXA)+(1<<ENXB)+(1<<ENYA)+(1<<ENYB) ; desabilita todos os enrolamentos
    cpse    XA, temp1 ; e so reabilita os nao zerados
    sbr     shaC, (1<<ENXA)
    cpse    XB, temp1
    sbr     shaC, (1<<ENXB)
    cpse    YA, temp1
    sbr     shaC, (1<<ENYA)
    cpse    YB, temp1
    sbr     shaC, (1<<ENYB)
    out     PORTC, shaC
    UPDATE_XA
    UPDATE_XB
    UPDATE_YA
    UPDATE_YB
    cbr     shaB, (1<<LATCH1) ; "mostra" os novos niveis de corrente
    out     PORTB, shaB ; para as pontes H
    sbr     shaB, (1<<LATCH1)
    out     PORTB, shaB
    pop     temp1
    sei
    ret

```

; rotina de avanço no eixo X

;

; da um passo no sentido positivo de X

; (sem atualizar o DAC)

```

XNEXT:
    lds     ZL, XAL ; vai acessar a LUT para o enrolamento XA
    lds     ZH, XAH
    incz    ; incrementa ponteiro
    mov     temp1, FIL
    cpse    temp1, ZL ; chegou no fim da LUT?
    rjmp    XNEXT_2
    mov     temp2, FIH
    cpse    temp2, ZH
    rjmp    XNEXT_2
    mov     ZL, INL ; sim, chegou no fim: volta ao inicio
    mov     ZH, INH
    incz    ; incrementa ponteiro
    ldi     temp1, (1<<DIRXA) ; troca sentido de excitacao do enrolamento
    eor     shaC, temp1
XNEXT_2:
    lpm     ; pega nivel de corrente da LUT
    mov     XA, valor
    sts     XAL, ZL ; salva ponteiro
    sts     XAH, ZH

    lds     ZL, XBL ; vai acessar a LUT para o enrolamento XB
    lds     ZH, XBH
    incz    ; incrementa ponteiro
    mov     temp1, FIL
    cpse    temp1, ZL ; chegou no fim da LUT?
    rjmp    XNEXT_3
    mov     temp2, FIH
    cpse    temp2, ZH
    rjmp    XNEXT_3

```

```

        mov     ZL, INL                                ; sim, chegou no fim: volta ao inicio
        mov     ZH, INH
        incz
        ldi     temp1, (1<<DIRXB)                     ; troca sentido de excitacao do enrolamento
        eor     shaC, temp1
XNEXT_3:
        lpm
        mov     XB, valor                               ; pega nivel de corrente da LUT
        sts     XBL, ZL                                ; salva ponteiro
        sts     XBH, ZH
        ret

; rotina de avanco no eixo Y
;
; da um passo no sentido positivo de Y
; (sem atualizar o DAC)

YNEXT:
        lds     ZL, YAL                                ; vai acessar a LUT para o enrolamento YA
        lds     ZH, YAH
        incz
        mov     temp1, FIL                             ; incrementa ponteiro
        cpse    temp1, ZL                             ; chegou no fim da LUT?
        rjmp    YNEXT_2
        mov     temp2, FIH
        cpse    temp2, ZH
        rjmp    YNEXT_2
        mov     ZL, INL                                ; sim, chegou no fim: volta ao inicio
        mov     ZH, INH
        incz
        ldi     temp1, (1<<DIRYA)                     ; troca sentido de excitacao do enrolamento
        eor     shaC, temp1
YNEXT_2:
        lpm
        mov     YA, valor                               ; pega nivel de corrente da LUT
        sts     YAL, ZL                                ; salva ponteiro
        sts     YAH, ZH

        lds     ZL, YBL                                ; vai acessar a LUT para o enrolamento YB
        lds     ZH, YBH
        incz
        mov     temp1, FIL                             ; incrementa ponteiro
        cpse    temp1, ZL                             ; chegou no fim da LUT?
        rjmp    YNEXT_3
        mov     temp2, FIH
        cpse    temp2, ZH
        rjmp    YNEXT_3
        mov     ZL, INL                                ; sim, chegou no fim: volta ao inicio
        mov     ZH, INH
        incz
        ldi     temp1, (1<<DIRYB)                     ; troca sentido de excitacao do enrolamento
        eor     shaC, temp1
YNEXT_3:
        lpm
        mov     YB, valor                               ; pega nivel de corrente da LUT
        sts     YBL, ZL                                ; salva ponteiro
        sts     YBH, ZH
        ret

; rotina de retrocesso no eixo X
;
; da um passo no sentido negativo de X

```

; (sem atualizar o DAC)

```

XPREV:
    lds    ZL, XAL                ; vai acessar a LUT para o enrolamento XA
    lds    ZH, XAH
    decz                   ; decrementa ponteiro
    mov    temp1, INL             ; chegou no inicio da LUT?
    cpse   temp1, ZL
    rjmp   XPREV_2
    mov    temp2, INH
    cpse   temp2, ZH
    rjmp   XPREV_2
    mov    ZL, FIL                ; sim, chegou no inicio: volta ao fim
    mov    ZH, FIH
    decz
    ldi    temp1, (1<<DIRXA)      ; troca sentido de excitacao do enrolamento
    eor    shaC, temp1

XPREV_2:
    lpm                    ; pega nivel de corrente da LUT
    mov    XA, valor
    sts    XAL, ZL                ; salva ponteiro
    sts    XAH, ZH

    lds    ZL, XBL                ; vai acessar a LUT para o enrolamento XA
    lds    ZH, XBH
    decz                   ; decrementa ponteiro
    mov    temp1, INL             ; chegou no inicio da LUT?
    cpse   temp1, ZL
    rjmp   XPREV_3
    mov    temp2, INH
    cpse   temp2, ZH
    rjmp   XPREV_3
    mov    ZL, FIL                ; sim, chegou no inicio: volta ao fim
    mov    ZH, FIH
    decz
    ldi    temp1, (1<<DIRXB)      ; troca sentido de excitacao do enrolamento
    eor    shaC, temp1

XPREV_3:
    lpm                    ; pega nivel de corrente da LUT
    mov    XB, valor
    sts    XBL, ZL                ; salva ponteiro
    sts    XBH, ZH
    ret

```

; rotina de retrocesso no eixo Y

;

; da um passo no sentido negativo de Y

; (sem atualizar o DAC)

```

YPREV:
    lds    ZL, YAL                ; vai acessar a LUT para o enrolamento YA
    lds    ZH, YAH
    decz                   ; decrementa ponteiro
    mov    temp1, INL             ; chegou no inicio da LUT?
    cpse   temp1, ZL
    rjmp   YPREV_2
    mov    temp2, INH
    cpse   temp2, ZH
    rjmp   YPREV_2
    mov    ZL, FIL                ; sim, chegou no inicio: volta ao fim
    mov    ZH, FIH
    decz
    ldi    temp1, (1<<DIRYA)      ; troca sentido de excitacao do enrolamento

```

```

        eor        shaC, temp1
YPREV_2:
        lpm                               ; pega nivel de corrente da LUT
        mov        YA, valor
        sts        YAL, ZL                ; salva ponteiro
        sts        YAH, ZH

        lds        ZL, YBL                ; vai acessar a LUT para o enrolamento YB
        lds        ZH, YBH
        decz                               ; decrementa ponteiro
        mov        temp1, INL             ; chegou no inicio da LUT?
        cpse       temp1, ZL
        rjmp       YPREV_3
        mov        temp2, INH
        cpse       temp2, ZH
        rjmp       YPREV_3
        mov        ZL, FIL                ; sim, chegou no inicio: volta ao fim
        mov        ZH, FIH
        decz
        ldi        temp1, (1<<DIRYB)      ; troca sentido de excitacao do enrolamento
        eor        shaC, temp1
YPREV_3:
        lpm                               ; pega nivel de corrente da LUT
        mov        YB, valor
        sts        YBL, ZL                ; salva ponteiro
        sts        YBH, ZH
        ret

; rotina de calculo de valor absoluto
;
; retorna o valor absoluto de <temp2:temp1> em <temp2:temp1>

ABS:
        sbrs       temp2, 7
        ret
        com        temp1
        com        temp2
        inc        temp1
        brne       ABS_FIM
        inc        temp2
ABS_FIM:
        ret

; rotina de calculo de sinal
;
; retorna o sinal de <temp2:temp1> em res<0:1>
; (00 = 0, 10 = -, 01 = +)

SIGN:
        ldi        res, 2
        sbrc       temp2, 7
        rjmp       SIGN_FIM
        ldi        res, 1
        tst        temp2
        brne       SIGN_FIM
        tst        temp1
        brne       SIGN_FIM
        clr        res
SIGN_FIM:
        ret

```

; rotina de tracagem de segmento de reta (Bresenham)

;

; assume velocidade e PWM pre-definidos

LINHA:

```

    push    temp1
    push    temp2
    lds     temp1, XfimL                ; deltax = abs(xfim-x)
    lds     temp2, XfimH
    sub     temp1, XL
    sbc     temp2, XH
    push    temp1
    push    temp2
    rcall   ABS
    sts     deltaXL, temp1
    sts     deltaXH, temp2
    pop     temp2
    pop     temp1
    rcall   SIGN                        ; stat<0:1> = sign(xfim-x)
    mov     stat, res
    lds     temp1, YfimL                ; deltay = yfim-y
    lds     temp2, YfimH
    sub     temp1, YL
    sbc     temp2, YH
    push    temp1
    push    temp2
    rcall   ABS
    sts     deltaYL, temp1
    sts     deltaYH, temp2
    pop     temp2
    pop     temp1
    rcall   SIGN                        ; stat<2:3> = sign(yfim-y)
    lsl     res
    lsl     res
    or      stat, res
    cbr     stat, 128                  ; troca mutua?
    lds     temp1, deltaXH
    lds     temp2, deltaYH
    cp      temp1, temp2
    brlo    PREP_TROCA
    brne    PREP_ERR
    lds     temp1, deltaXL
    lds     temp2, deltaYL
    cp      temp1, temp2
    brlo    PREP_TROCA
    rjmp    PREP_ERR
PREP_TROCA:
    lds     temp1, deltaXL
    lds     temp2, deltaYL
    sts     deltaXL, temp2
    sts     deltaYL, temp1
    lds     temp1, deltaXH
    lds     temp2, deltaYH
    sts     deltaXH, temp2
    sts     deltaYH, temp1
    sbr     stat, 128
PREP_ERR:
    lds     errL, deltaYL                ; e = 2*dy-dx
    lds     errH, deltaYH
    lsl     errL
    rol     errH
    lds     temp1, deltaXL
    lds     temp2, deltaXH
    sub     errL, temp1

```

```

        sbc      errH, temp2

        lds      cntL, deltaXL          ; for cnt = dx downto 1
        lds      cntH, deltaXH

LOOP_LINHA:
        tst      cntL
        bme      L_L_1
        tst      cntH
        bme      L_L_1
        rjmp     LINHA_FIM

L_L_1:
        sbrs     errH, 7                ; if(e >= 0)
        rjmp     L_L_6
        sbrs     stat, 7                ; if(troca mutua)
        rjmp     L_L_3

        sbrs     stat, 0                ; x = x + s1
        rjmp     L_L_2
        rcall     XNEXT
        subi     XL, 0xFF
        sbci     XH, 0xFF                ; x += 1
        rjmp     L_L_5

L_L_2:
        sbrs     stat, 1
        rjmp     L_L_5
        rcall     XPREV
        subi     XL, 0x01
        sbci     XH, 0x00                ; x -= 1
        rjmp     L_L_5

L_L_3:
        sbrs     stat, 2                ; y = y + s2
        rjmp     L_L_4
        rcall     YNEXT
        subi     YL, 0xFF
        sbci     YH, 0xFF                ; y += 1
        rjmp     L_L_5

L_L_4:
        sbrs     stat, 3
        rjmp     L_L_5
        rcall     YPREV
        subi     YL, 0x01
        sbci     YH, 0x00                ; y -= 1

L_L_5:
        lds      temp1, deltaXL          ; e = e-2*dx
        lds      temp2, deltaXH
        lsl      temp1
        rol      temp2
        sub       errL, temp1
        sbc      errH, temp2

L_L_6:
        sbrs     stat, 7                ; if(troca mutua)
        rjmp     L_L_8

        sbrs     stat, 2                ; y = y + s2
        rjmp     L_L_7
        rcall     YNEXT
        subi     YL, 0xFF
        sbci     YH, 0xFF                ; y += 1
        rjmp     L_L_10

L_L_7:
        sbrs     stat, 3
        rjmp     L_L_10
        rcall     YPREV
        subi     YL, 0x01

```

```

        sbci    YH, 0x00                ; y -= 1
        rjmp   L_L_10

L_L_8:
        sbrs   stat, 0                  ; x = x + s1
        rjmp   L_L_9
        rcall  XNEXT
        subi   XL, 0xFF
        sbci   XH, 0xFF                ; x += 1
        rjmp   L_L_10

L_L_9:
        sbrs   stat, 1
        rjmp   L_L_10
        rcall  XPREV
        subi   XL, 0x01
        sbci   XH, 0x00                ; x -= 1

L_L_10:
        lds    temp1, deltaYL           ; e = e+2*dy
        lds    temp2, deltaYH
        lsl    temp1
        rol    temp2
        add    errL, temp1
        adc    errH, temp2

        push   temp1
        push   temp2
        mov    temp1, cntL
        mov    temp2, cntH
        subi   temp1, 0x01
        sbci   temp2, 0x00
        mov    cntL, temp1
        mov    cntH, temp2
        pop    temp2
        pop    temp1

        rcall  UPDATE_DAC
        rcall  PERIOD
        rjmp   LOOP_LINHA

LINHA_FIM:
        pop    temp2
        pop    temp1
        ret

; rotina de interrupcao de recepcao de byte na serial
;
; armazena o que for recebido no buffer, e
; se necessario, emite Xoff

INT_SERIAL:
        cli                                ; DI
        push   temp1
        in     temp1, SREG
        push   temp1
        push   temp2
        push   ZL
        push   ZH

        in     temp1, UDR                  ; pega byte recebido
        lds    ZL, RXptrL                 ; recupera ponteiro do buffer de recepcao
        lds    ZH, RXptrH
        cpi    ZL, FIMBUFL                ; fim do buffer?
        brne   RX_2
        cpi    ZH, FIMBUFH

```

```

    breq    RX_FIM                                ; fim do buffer: ignora byte recebido
RX_2:
    cpi     ZL, SAFEL                             ; fim da area segura?
    brne    RX_3
    cpi     ZH, SAFEH
    brne    RX_3                                ; fim da area segura: envia Xoff
    ldi     temp2, Xoff
    out     UDR, temp2
    sts     LAST, temp2
RX_3:
    st      Z+, temp1                             ; guarda byte no buffer
    sts     RXptrL, ZL
    sts     RXptrH, ZH
RX_FIM:
    pop     ZH
    pop     ZL
    pop     temp2
    pop     temp1
    out     SREG, temp1
    pop     temp1
    sei
    reti

; rotina de renocao de um byte do buffer
;
; se buffer vazio, espera ate que haja algo

GETBYTE:
    push    ZL
    push    ZH
    push    YL
    push    YH
GET_2:
    sei
    nop
    nop
    nop
    cli
    lds     ZL, RXptrL                            ; se RXptr != GETptr, tem byte no buffer
    lds     ZH, RXptrH
    lds     YL, GETptrL
    lds     YH, GETptrH
    cp      ZL, YL
    brne    GET_3
    cp      ZH, YH
    breq    GET_2                                ; como RXptr = GETptr, espera byte
GET_3:
    ld      temp1, Y+                             ; pega byte do buffer e incrementa ponteiro
    cp      ZL, YL                                ; fim dos bytes do buffer?
    brne    GET_4
    cp      ZH, YH
    brne    GET_4
    ldi     ZL, INIBUFL
    ldi     YL, INIBUFL
    ldi     ZH, INIBUFH
    ldi     YH, INIBUFH
    lds     temp2, LAST
    cpi     temp2, Xoff
    brne    GET_4                                ; sim -> reseta ponts. para inicio do buffer
    ldi     temp2, Xon
    sts     LAST, temp2
    out     UDR, temp2

```

```

GET_4:
    sts    RXPtRl, ZL                ; guarda ponteiros
    sts    RXPtRH, ZH
    sts    GETpTrL, YL
    sts    GETpTrH, YH
    pop    YH
    pop    YL
    pop    ZH
    pop    ZL
    sei                    ; EI
    ret

; rotina de leitura de parametro de 3 digitos
;
; ignorando espacos e tabs, pega os proximos 3 bytes e
; converte para hexadecimal, em <temp1:temp2>

GETPARAM:
    rcall  GETBYTE                    ; pega MSN
    cpi    temp1, 0x20                ; se espaco em branco, ignora
    breq   GETPARAM
    cpi    temp1, 0x09                ; se tabulacao, ignora
    breq   GETPARAM
    rcall  ASC2HEX
    push   temp1
    rcall  GETBYTE                    ; pega nibble do meio
    rcall  ASC2HEX
    swap   temp1                      ; coloca o nibble do meio no lugar certo
    push   temp1
    rcall  GETBYTE                    ; pega LSN
    rcall  ASC2HEX
    pop    temp2
    or     temp2, temp1                ; junta nibble do meio com LSN
    pop    temp1
    ret                                ; <temp1:temp2> = parametro

; rotina de correcao de numeros negativos
;
; converte o numero de 12 bits
; em numero de 16 bits

CONVPARAM:
    sbrs   temp1, 3
    ret
    ori    temp1, 0xF0
    ret

; rotina de conversao ASCII > HEX
;
; caracteres ilegais sao transformados em 0

ASC2HEX:
    cpi    temp1, 0x30                ; < '0' ?
    brlo   A2H_ERR
    cpi    temp1, 0x3A                ; >= ('9'+1) ?
    brsh   A2H_ALFA
    subi   temp1, 0x30                ; corrige
    ret

A2H_ALFA:
    cbr    temp1, (1<<5)              ; tudo maiusculas
    cpi    temp1, 'A'                 ; < 'A' ?

```

```

        brlo    A2H_ERR
        cpi     temp1, 'G'                ; >= ('F'+1) ?
        brsh    A2H_ERR
        subi    temp1, 0x37                ; corrige
        ret
A2H_ERR:
        clr     temp1                    ; erro: nibble = 0
        ret

```

```

; perfil de acionamento microstepping
; implementado como look-up table

```

```
faseA:
```

```

.DB    0, 6
.DB    12, 18
.DB    24, 31
.DB    37, 43
.DB    49, 55
.DB    61, 68
.DB    74, 79
.DB    85, 91
.DB    97, 103
.DB    109, 114
.DB    120, 125
.DB    131, 136
.DB    141, 146
.DB    151, 156
.DB    161, 166
.DB    171, 175
.DB    180, 184
.DB    188, 193
.DB    197, 201
.DB    204, 208
.DB    212, 215
.DB    218, 221
.DB    224, 227
.DB    230, 233
.DB    235, 237
.DB    240, 242
.DB    244, 245
.DB    247, 248
.DB    250, 251
.DB    252, 253
.DB    253, 254
.DB    254, 254

```

```
faseB:
```

```

.DB    255, 254
.DB    254, 254
.DB    253, 253
.DB    252, 251
.DB    250, 248
.DB    247, 245
.DB    244, 242
.DB    240, 237
.DB    235, 233
.DB    230, 227
.DB    224, 221
.DB    218, 215
.DB    212, 208
.DB    204, 201
.DB    197, 193
.DB    188, 184
.DB    180, 175

```

.DB 171, 166
.DB 161, 156
.DB 151, 146
.DB 141, 136
.DB 131, 125
.DB 120, 114
.DB 109, 103
.DB 97, 91
.DB 85, 79
.DB 74, 68
.DB 61, 55
.DB 49, 43
.DB 37, 31
.DB 24, 18
.DB 12, 6
fimfase:
.DB 0,0

Apêndice V: Exemplos de Programas

A	L04807A
S3E8	L04802C
F0000BD	L03102C
O	F08B000
B0FF	O
L00002C	L00002C
L00B02C	L00B02C
L00B07A	L00B07A
L00007A	L00007A
L0000A5	L0000A5
L0640A5	L04A0A5
L080088	L04A07A
L080050	L03C07A
L062034	L03C02C
L03C034	L04E02C
L03C02C	L04E045
L04702C	L07B045
L047000	L07B000
L000000	L000000
F03C062	F08C000
L03C07A	O
L05007A	L00002C
L050062	L00B02C
L03C062	L00B07A
F091000	L00007A
O	L0000A5
F00001D	L0470A5
L000088	L04707A
L01D0A5	L03C07A
L05C0A5	L03C02C
L079088	L04702C
L07901D	L047000
L05C000	L000000
L01D000	B000
L00001D	A
F03102C	P
L03107A	



Fig. 17.1 - Resultado do programa-exemplo acima