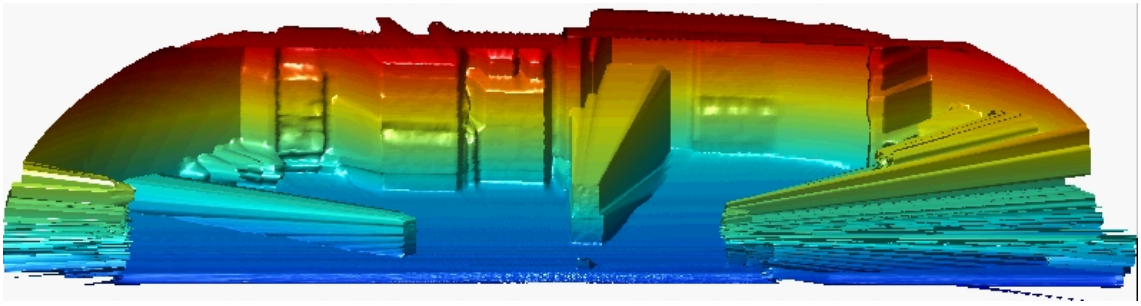
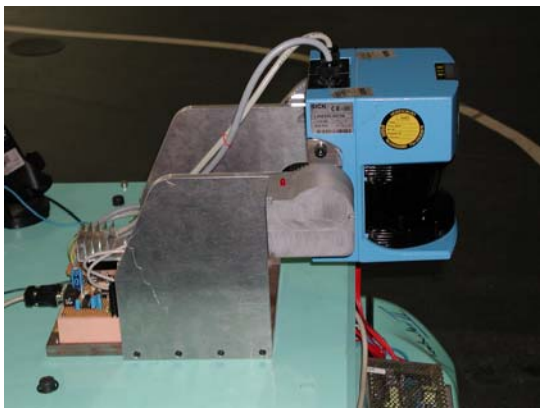
Universidade de Aveiro
Departamento de Engenharia Mecânica

Projecto de Automação
(Relatório Final)

Scanner 3D para Aplicações em Modelação e Navegação





Miguel Matos Dias
N.º 13821
a13821@alunos.mec.ua.pt

Agradecimentos

Prof. Dr. Vítor Santos
Eng. António Festas
A todos os colegas de curso

Índice

1. Introdução

Com o crescente desenvolvimento tecnológico, os sistemas laser mais económicos geram perfis planares de distâncias – 2D. Com um grau de liberdade adicional, podem obter-se "imagens" do espaço a 3 dimensões a custos reduzidos (**Fig. 1**). As aplicações de tais capacidades em sistemas artificiais são de tal forma vastas que é interessante e estimulante contribuir para o desenvolvimento deste tipo de sistemas.

As aplicações, após os devidos desenvolvimentos, poderão ir desde a reconstrução a 3 dimensões de ambientes estáticos (salas, edifícios, objectos, etc.) até á sua utilização em sistemas de navegação autónoma (desvio de obstáculos, planeamento, etc.).

Fig. 1 – Scan de uma escola na Alemanha [12].

2. Objectivos

- Concepção e implementação de um sistema de percepção 3D com base num sensor 2D;
- Desenvolvimento da estrutura mecânica adequada;
- Selecção e implementação de uma interface standard de comando (RS232, USB, ou outra);
- Concepção da unidade de controlo (hardware/software) do sistema;
- Estudo da influência da dinâmica do movimento sobre desempenho do sensor original;
- Desenvolvimento de software base para aquisição 3D.

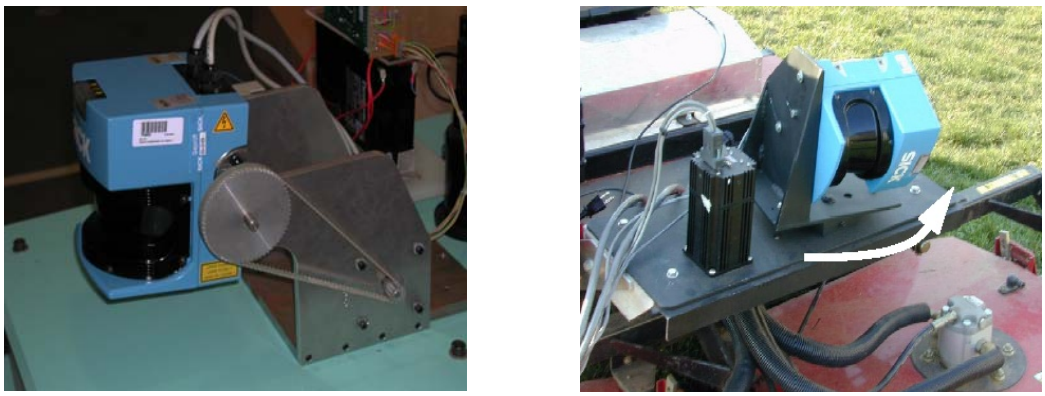
3. Estrutura Mecânica

No desenvolvimento da estrutura mecânica tomou-se em conta que esta tinha que:

- Ter resistência mecânica para suportar o laser bem como o seu movimento;
- Permitir que o laser efectuasse scan's de 270º, sem que nenhuma parte mecânica interferisse na leitura e obtenção dos dados;
- Servir de suporte do motor bem como da placa de controlo e todo o hardware associado a esta.

3.1 Colocação do Sensor

Após a definição dos requisitos da estrutura, levantou-se outra questão. Qual a forma mais vantajosa de colocação do laser para fazer o "varrimento": Na horizontal (**Fig. 2 - a**), ou na vertical (**Fig. 2 - b**)?



a) b)

Fig. 2 – Forma de "varrimento" do laser, a) na horizontal, b) na vertical [7].

Após o estudo das duas soluções, optou-se por colocar o laser na posição horizontal, uma vez que permitia um aproveitamento total do campo de visão do laser e uma melhor e mais rápida detecção de obstáculos verticais (pilares, portas, etc.).

3.2 Dimensionamento da Estrutura

Para a modelação da estrutura foi utilizado um programa (software) de CAD, *SolidWorks 2000*®, tendo em vista uma maior compreensão das peças a modelar e sua facilidade de manipulação para posterior "assembly" montagem.

Um dos materiais escolhidos para a concepção da estrutura foi uma liga de Alumínio (liga 1060), permitindo assim que a estrutura não se tornasse "muito" pesada. Esta liga foi apenas utilizada na concepção dos suportes laterais da estrutura.

As principais características mecânicas deste tipo de liga são:

Densidade = 2700 kg/m^3
σ_r = 68,94*106 N/m^2
σ_{Ced} = 27,57*106 N/m^2
E = 6,9*1010 N/m^2

Todos os outros componentes (veios, base, suportes do sensor e esticador do motor) foram concebidos em aço CK45 que tem como principais características mecânicas seguinte:

Densidade = 7700 kg/m^3
σ_r = 7,24*108 N/m^2
σ_{Ced} = 6,20*108 N/m^2
E = 2,1*1011 N/m^2

Note-se que ainda os batentes do sensor e a caixa do potenciómetro foram concebidos num polímero altamente resistente (Uriol).

3.3 Possibilidades para a Estrutura

Apresentam-se na **Fig. 3** algumas hipóteses estruturais estudadas, mas abandonadas por não cumprirem alguns dos requisitos abordados no ponto 3, nomeadamente no aproveitamento total do campo de visão do laser sem nenhuma interferência mecânica.

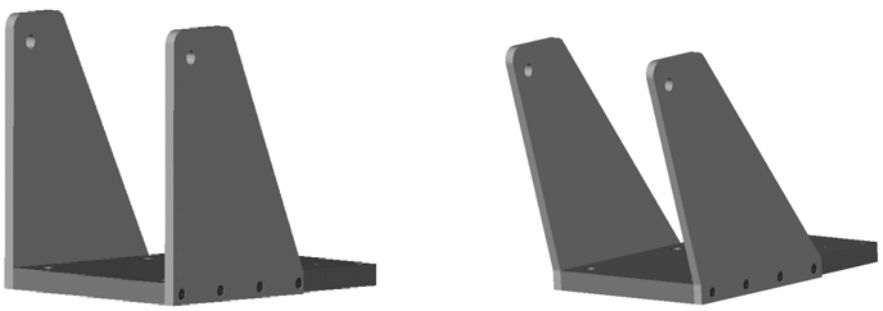


Fig. 3 – Algumas das hipóteses estruturais estudadas.

Após uma melhor análise do sensor chegou-se à seguinte estrutura mecânica final que cumpre todos os requisitos anteriormente referidos (**Fig. 4**).

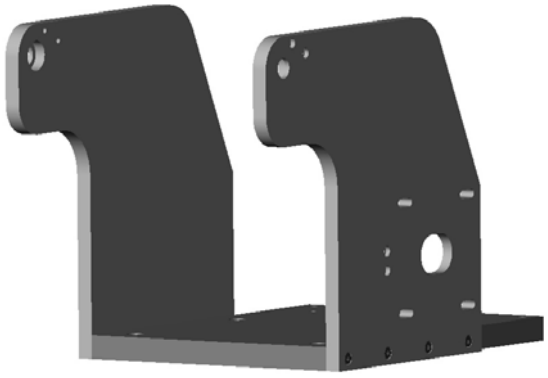
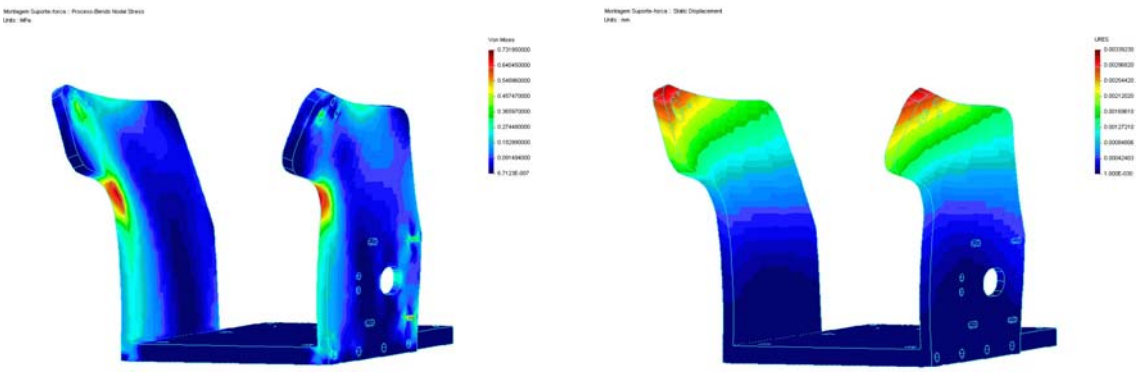


Fig. 4 – Estrutura mecânica final de suporte do laser.

3.4 Estudo Estático da Estrutura

Afim de verificar as capacidades mecânicas das peças modeladas foram realizados ensaios de elementos finitos recorrendo ao software *COSMOSWorks 4.0*®. Estes ensaios possibilitaram a verificação dos esforços a que toda a estrutura estava sujeita, permitindo visualizar os seus pontos fracos, as tensões máximas e os deslocamentos (**Fig. 5**) e caso fosse necessário proceder às devidas correcções.

Os suportes laterais foram sujeitos a esforços estáticos verticais no valor de 60N, aplicados na zona onde iriam ser montados os rolamentos de suporte do sensor laser. O critério usado para a análise foi o da Tensão Máxima de Von Misses.



a) b)

Fig. 5 – Análise estática da estrutura , a) tensões máximas , b) deslocamentos máximos.

Através da análise dos resultados obtidos, em que a tensão máxima na estrutura é de 0,73MPa e o deslocamento máximo de 0,0034mm, concluiu-se que a mesma estava suficientemente bem dimensionada.

3.5 Dimensionamento e Escolha de Alguns Componentes

Dada a disponibilidade imediata de alguns dos componentes no laboratório, nomeadamente de um motor passo a passo, procedeu-se ao dimensionamento das rodas dentadas e da correia plana dentada a utilizar no projecto.

Rodas dentadas:

Segundo dados dos fabricantes:

Binário máximo do motor = 1,2Nm
Massa do sensor = 4,5Kg

Momento na polia (**Fig. 6)**:

R = 79,5mm
$M = 79{,}5x10^{-3}x4{,}5x9{,}8 = 3{,}51Nm$

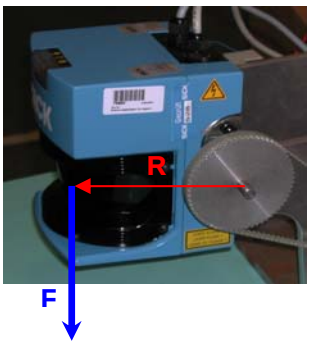


Fig. 6 – Diagrama de Forças.

Relação de transmissão:

$nx1{,}2 \geq 3{,}51 \Leftrightarrow n \geq 2{,}93$

Aplicando um coeficiente de segurança no cálculo do momento da polia de ≈ 2, escolheu-se uma relação de transmissão n = 5.

Através da consulta do url www.amidata.es, escolheram-se as seguintes polias disponíveis comercialmente:

Motor ⇒ Z1 = 12 dentes, passo 5mm, largura 10mm, referência 744-952;
Sensor ⇒ Z2 = 60 dentes, passo 5mm, largura 10mm, referência 286-5720.

Correia plana dentada:

$a_1 = 210{,}09\text{mm}$
$a_2 = 218{,}04\text{mm}$
$d_1 = 18{,}25\text{mm}$
$d_2 = 94{,}65\text{mm}$

Como o comprimento da correia é:

$$L = 2a + \frac{\pi}{2}(d_2 + d_1) + \frac{1}{4a}(d_2 - d_1)^2$$

então:

$L_{min} = 604{,}47\text{mm}$
$L_{max} = 620{,}37\text{mm}$

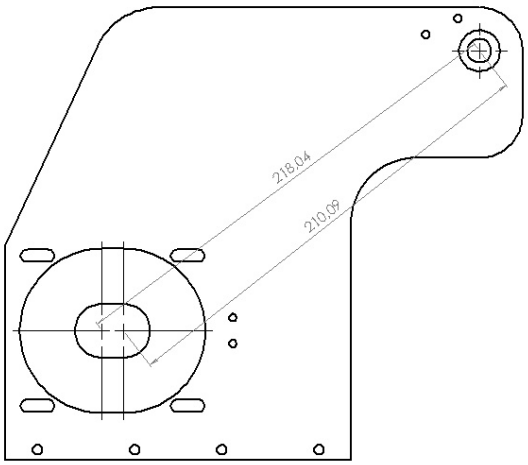


Fig. 6 – Distâncias dos entre-eixos.

Assim, de acordo com os valores obtidos anteriormente para L, escolheu-se:

Correia plana dentada, passo 5mm, 10mm largura e 610mm de perímetro, referência 359-8869.

4. Hardware

4.1 Introdução

Um dos objectivos deste projecto, era a concepção e desenvolvimento de uma unidade de controlo, que permitisse comunicar com o PC através de uma interface standard de comando (RS232, USB, ou outra), mas também o controlo do motor passo a passo, o posicionamento absoluto do laser que assim permite fazer o seu controlo e inicialização mas que também permitisse mais tarde eventuais evoluções.

4.2 Solução Inicial

A solução inicial (testada com sucesso), para o controlo do motor passo a passo e posteriormente da posição do laser, baseava-se num PLC *Mitsubishi FX$_{2N}$ -16MR-DS* [8], numa carta *FX$_{2N}$ -1PG (Pulse Generator Unit)* e numa carta de potência já existentes no laboratório do departamento. O PLC controlava a carta *FX$_{2N}$ -1PG* que por sua vez gerava os pulsos para o controlo do motor passo a passo através da carta de potência.

Esta solução foi um primeiro teste, para validar a abordagem mas que foi abandonada visto que todo o conjunto (PLC + FX$_{2N}$ -1PG + carta de potência) se revelava não muito compacta, bastante volumosa e cara (**Fig. 7**). Visto que todo o processo tinha tido sucesso decidiu-se encontrar uma solução mais atraente, menos volumosa e que também permitisse comunicar com o PC através de um interface standard de comando, sem a necessidade da instalação de software adicional (*MelDDE*

como acontecia para se aceder ás variáveis do PLC através de um programa em *Visual Basic*) além do próprio programa de controlo.

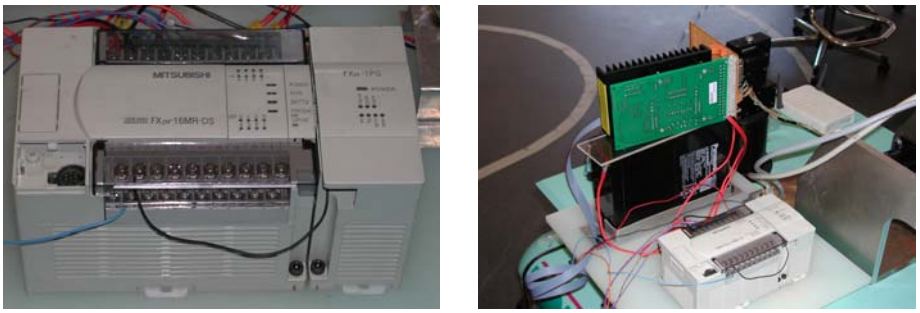


Fig. 7 – Solução inicial – PLC.

4.3 Solução Final

Tomando em conta que a escolha do microcontrolador que iria substituir o PLC teria que satisfazer os seguintes requisitos:

- Possuir entradas analógicas e I/O digitais;
- Porta de comunicação série;
- Gerador de PWM para controlo do motor passo a passo;
- Memória interna para guardar o programa de controlo;
- Temporizadores;
- "Interrupts";
- Fácil programação;
- Software de desenvolvimento económico;
- Informação disponível (manuais, etc.);
- Custo (ser um microcontrolador económico);

após intensa pesquisa, quer em livros quer na Internet, sobre circuitos para controlo de motores passo a passo e de microcontroladores, optou-se pelos microcontroladores PIC da Microchip® [14], visto que existia muita documentação sobre tais integrados (manuais, exemplos de aplicação, etc.) e ao mesmo tempo também preenchiam todos os requisitos acima enunciados. De salientar ainda a enorme gama de microcontroladores que a Microchip® [14] possui, permitindo assim uma fácil escolha do microcontrolador a usar.

4.3.1 O que é um PIC?

Um PIC é um CPU RISC de alto desempenho, concebido em torno da arquitectura Harvard. A utilização deste tipo de arquitectura na construção dos microcontroladores, permite que estes sejam rápidos e baratos.

O PIC utilizado para substituir o PLC foi o modelo 16F876, que se descreve e ilustra na **Fig. 8.**

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F876
Operating Frequency	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory	256
Interrupts	13
I/O Ports	Ports A,B,C
Timers	3
Capture/Compare/PWM Modules	2
Serial Communications	MSSP, USART
Parallel Communications	—
10-bit Analog-to-Digital Module	5 input channels
Instruction Set	35 instructions

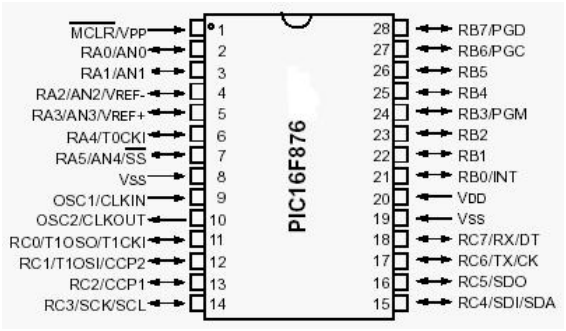
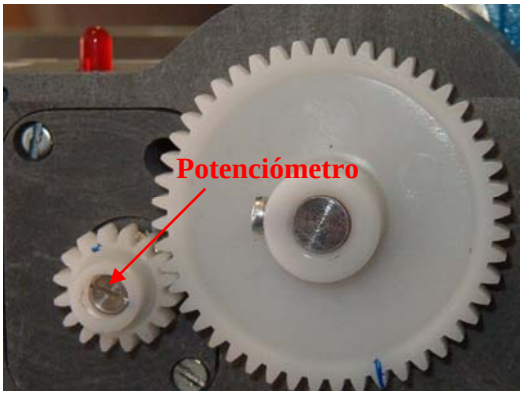


Fig. 8 – Principais características do PIC16F876.

4.3.2 Medição da Posição Absoluta do Sensor

Uma das formas encontradas para fazer a medição da posição absoluta do sensor, foi a da utilização de um potenciómetro de 3 voltas ligado á entrada da ADC do PIC (pino RA0), que permite converter os valores de tensão em números entre 0 e 1024 (resolução de 10bit's). O potenciómetro encontra-se montado numa caixa (**Fig. 9 – a**) acoplada á estrutura e está ligado ao eixo de movimento do sensor através de engrenagens (**Fig. 9 – b**) plásticas, com uma relação de transmissão igual a 3.

a)



b)

Fig. 9 – a) Caixa de montagem do potenciómetro, b) engrenagens.

4.3.3 Placa Electrónica de Controlo

Na construção da placa de controlo do sistema teve-se em atenção que a mesma tinha que ter uma porta de comunicação RS232, uma entrada analógica que estaria ligada directamente a uma das ADC's do PIC, um saída PWM para gerar pulsos para o controlador do motor passo a passo e uma entrada digital para saber quando o laser estava travado ou não. De acordo com estes factos e com base na "application note" AN822 da Microchip® [14] e nos manuais dos circuitos integrados L298 e L297 [19], o circuito electrónico da placa é o apresentado na **Fig. 10** e **11**.

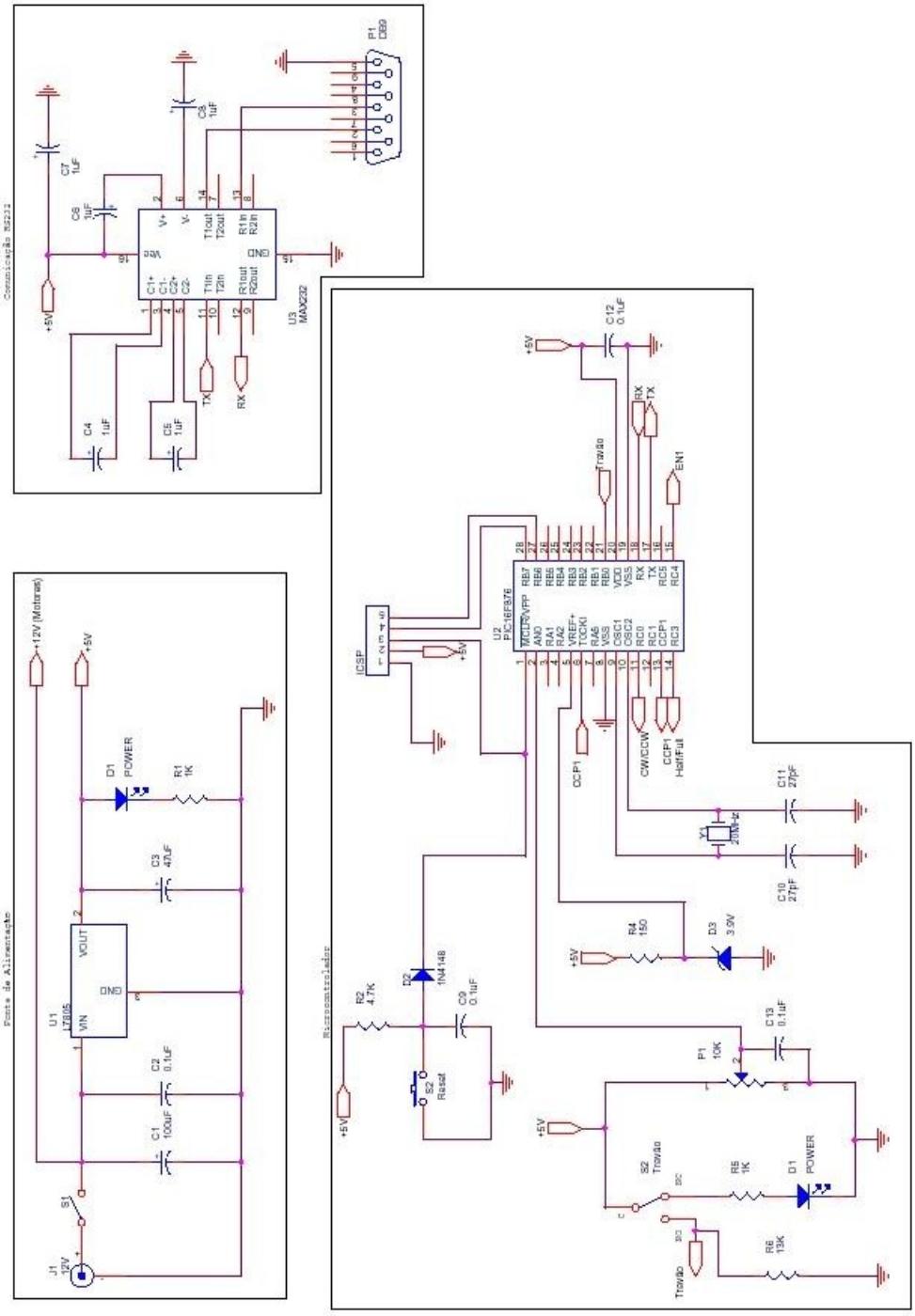


Fig. 10 – Circuito electrónico da fonte de alimentação, do microcontrolador e da comunicação RS232.

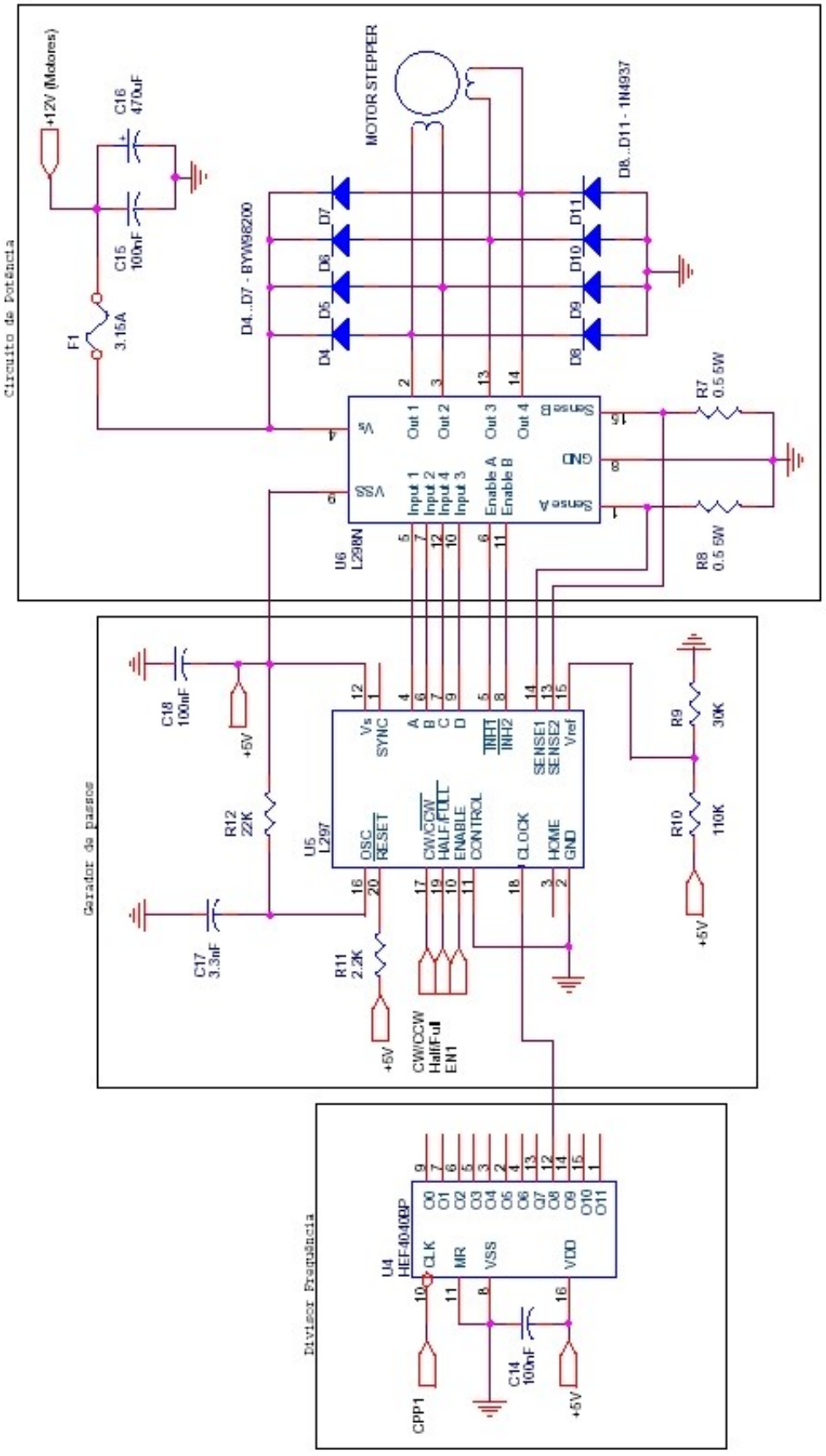


Fig. 11 – Circuito electrónico do divisor de frequência, do gerador de passos e do circuito de potência.

A placa desenvolvida, é uma solução mais elegante e menos volumosa que a solução inicial, como se pode verificar pela comparação apenas com a carta de potência utilizada na solução inicial (**Fig. 12**).

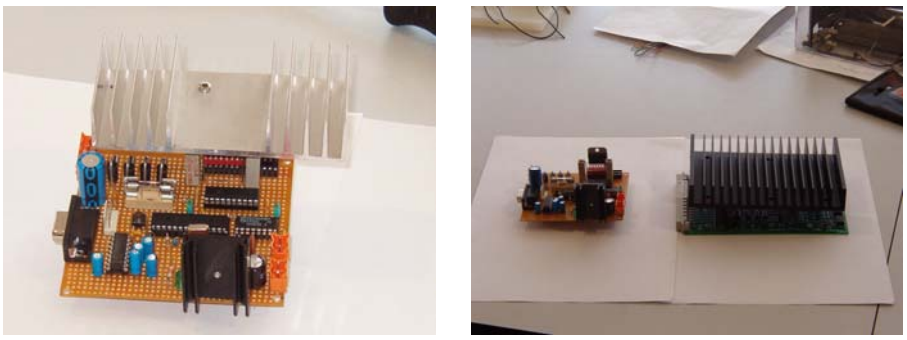


Fig. 12 – Aspecto final da placa de controlo e comparação de tamanho com carta de potência na solução inicial.

4.3.4 Descrição da Placa de Controlo

Como podemos ver pela **Fig. 13**, a placa de controlo do sistema pode ser dividida em quatro grandes grupos. Fonte de alimentação, microcontrolador e divisor de frequência, comunicação RS232 e gerador de pulsos e circuito de potência.

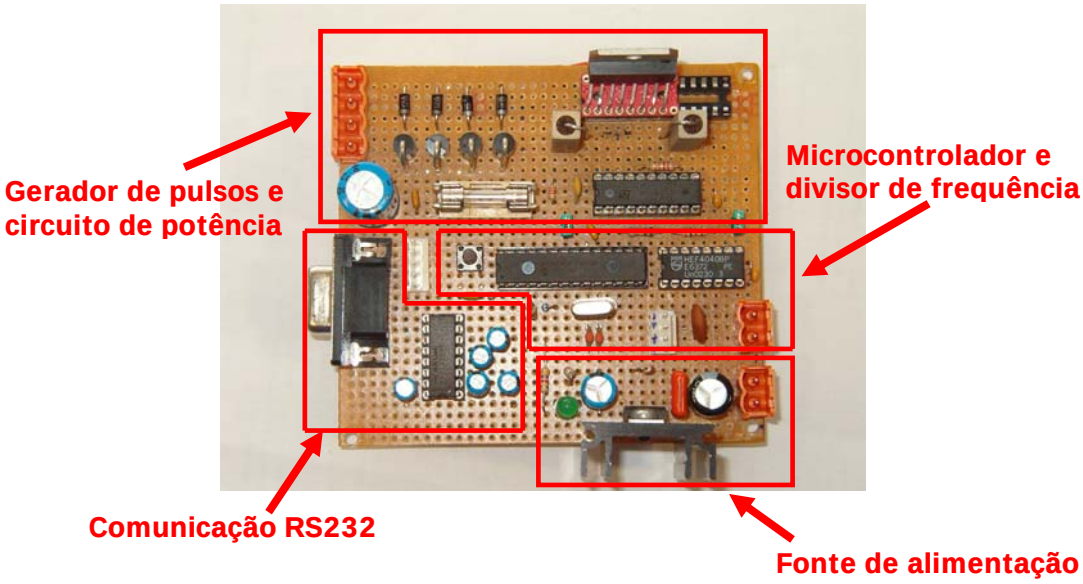


Fig. 13 – Quatro principais grupos da placa de controlo.

4.3.4.1 Fonte de Alimentação

Para o correcto funcionamento do microcontrolador PIC, bem como de toda a lógica dos restantes circuitos integrados, estes têm que ser alimentados a uma tensão de 5V. Esta tensão têm que ser o mais estável possível. Seguidamente ilustra-se a fonte de alimentação construída (**Fig. 14**).

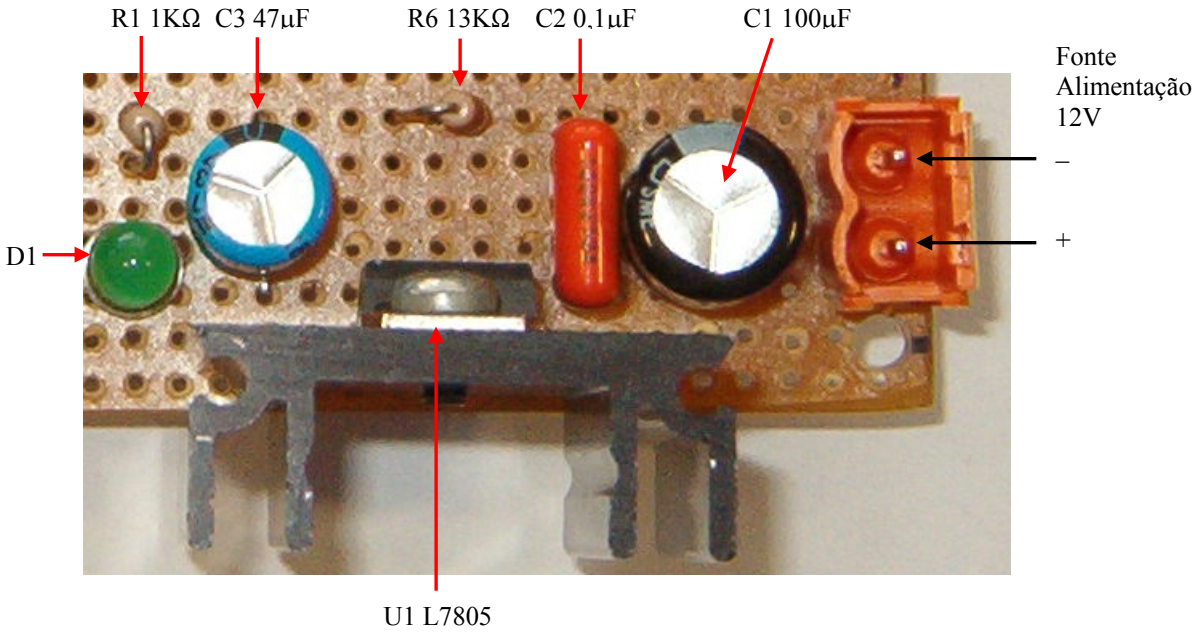


Fig. 14 – Identificação dos vários componentes – Fonte de Alimentação.

4.3.4.2 Microcontrolador e Divisor de Frequência

Podemos ver ilustrado na **Fig. 15** toda a parte do circuito do microcontrolador PIC, de notar que não são necessários muitos componentes para que este funcione correctamente. A utilização do divisor de frequência (HEF4040BP), deveu-se ao facto do gerador de PWM do PIC, não gerar frequências abaixo dos 1500Hz. Dividindo o sinal gerado pelo CCP1 por 256 obtiveram-se assim valores na ordem dos 6Hz.

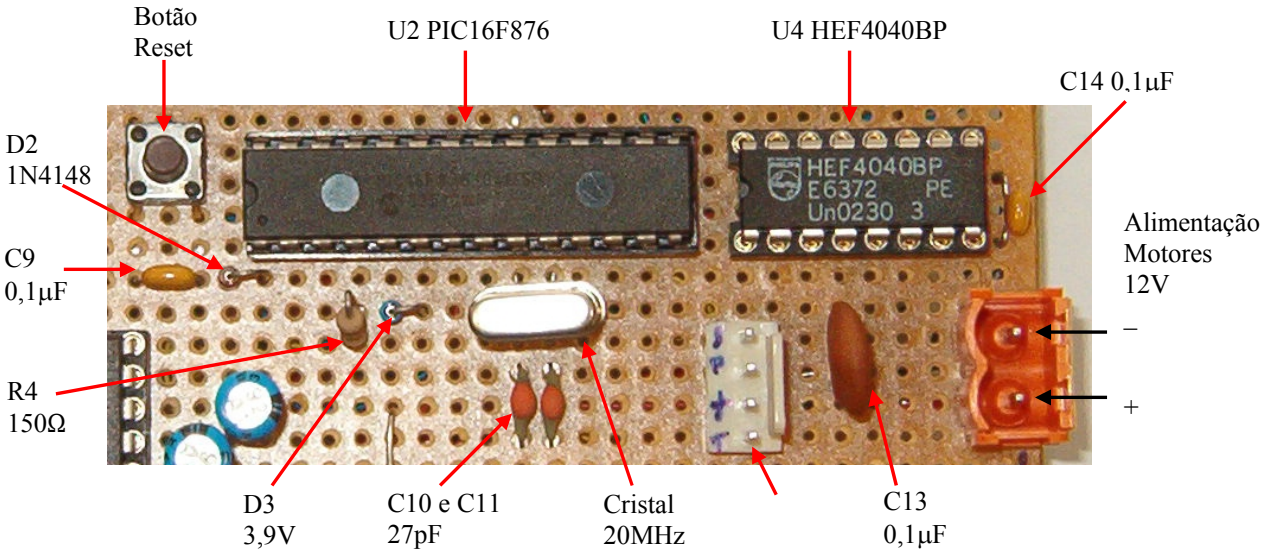


Fig. 15 – Identificação dos vários componentes – Microcontrolador e Divisor de Frequência.

4.3.4.3 Comunicação RS232

Visto que o PIC16F876 já possui porta série, foi apenas necessário normalizar os sinais TTL á saída deste. Para esse efeito utilizou-se um MAX232 (**Fig. 16**).

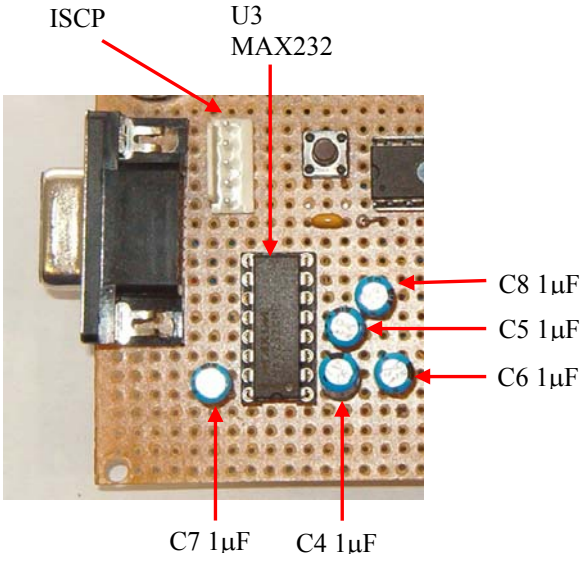


Fig. 16 – Identificação dos vários componentes – Comunicação RS232.

4.3.4.4 Gerador de Pulsos e Circuito de Potência

Para que o motor passo a passo funciona-se correctamente foi utilizado um integrado L297 para gerar as fases, que seriam depois aplicadas ao integrado de potência L298 (**Fig. 17**).

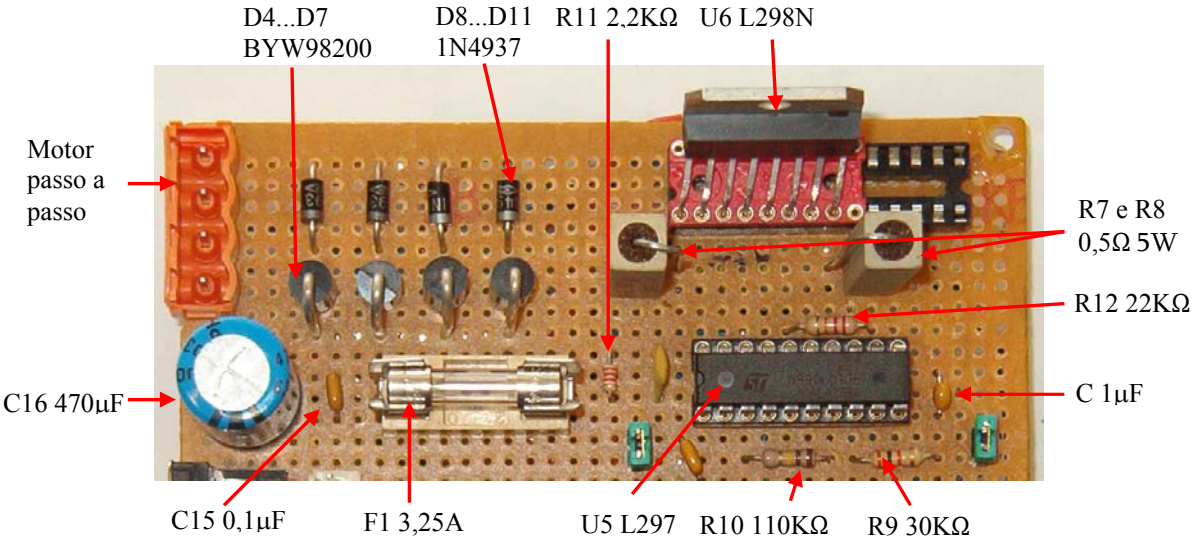


Fig. 17 – Identificação dos vários componentes – Gerador de Pulsos e Circuito de Potência.

5. Software

5.1 Introdução

Após concluído o trabalho de construção da placa de controlo do motor passo a passo, iniciou-se a fase de programação do microcontrolador PIC16F876. Como poderemos ver de seguida existem duas maneiras de o fazer, uma das quais mais vantajosa que a outra.

5.2 Como Programar um PIC?

A transferência de programas para os microcontroladores da família PIC pode ser efectuada de duas maneiras diferentes:

- Através de um dispositivo específico - o programador (porta série ou paralela);
- Ou através da linha série RS232.

5.2.1 Programação do PIC16F876 com Recurso a um Programador

Um dos métodos mais utilizados na programação de PIC's, é através da utilização de um programador especifico para programar este tipo de microcontroladores. De seguida será descrito um programador (porta paralela), desenhado inicialmente para programar o PIC16F84 por um senhor chamado BOB Blick [21], mas que funciona com grande parte dos microcontroladores da Microchip®, (inclusive com os série 18Fxxx, que têm muito mais memória e outras funções).

5.2.1.1 Programador de PIC's

De seguida ilustra-se (**Fig. 18**) um dos mais utilizados programadores (talvez até dos mais conhecidos) na programação de microcontroladores PIC.

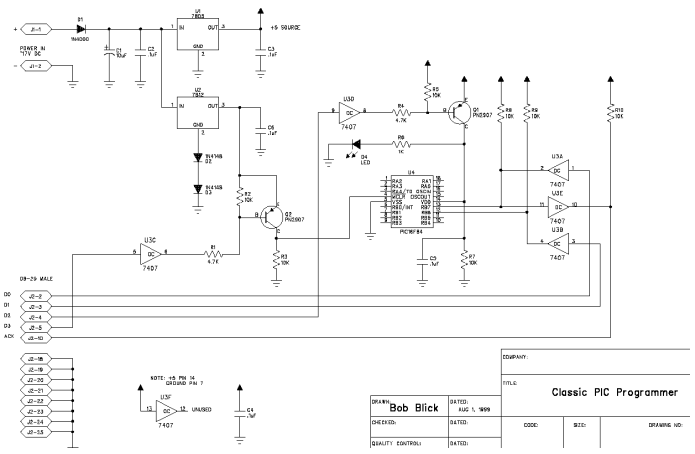


Fig. 18 – Circuito electrónico do programador de PIC's.

5.2.1.2 Software a Utilizar com o Programador

O software utilizado com o programador na programação do PIC, é o muito conhecido *IC-Prog* [22]. A seguir descrevem-se todos os passos a realizar para que funcione correctamente no *Windows 2000*®.

1. Colocar o ficheiro icprog.sys no mesmo directório que o *IC-Prog* [22] para que funcione correctamente no *Windows 2000*®;

2. Executar o programa *IC-Prog* [22], entrar no menu *Configuração* e carregar em *Opções* (**Fig. 19**).

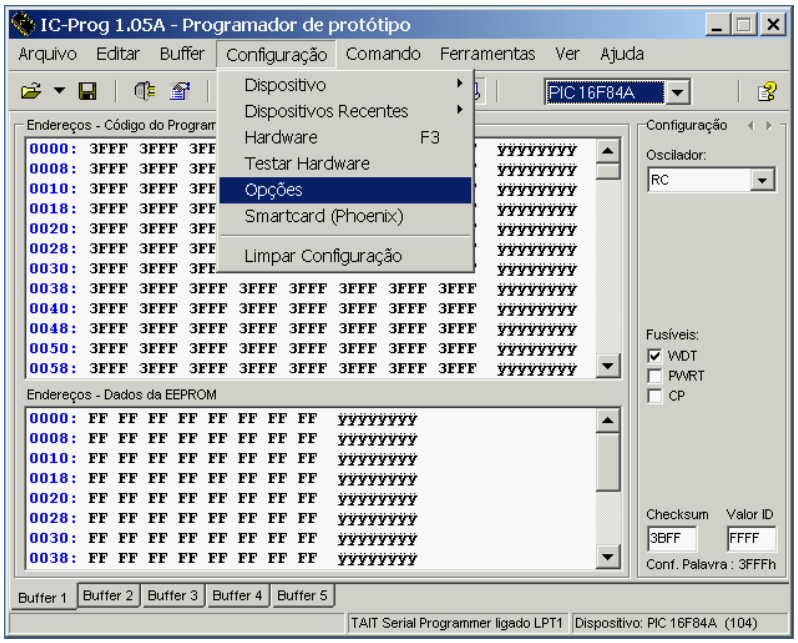


Fig. 19 – Configuração do *IC-Prog* [22].

3. Entrar na pasta *Diversos* e activar a opção *Activar Driver NT/2000/XP* (**Fig. 20**).

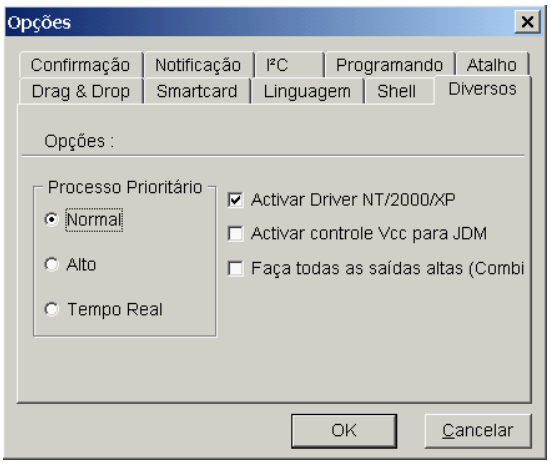


Fig. 20 – Configuração do *IC-Prog* [22] (activar driver).

Após realizar estes passos o programa reiniciará e efectuar-se-á a configuração do Hardware.

Configuração do Hardware

1. Entrar no menu *Configuração* e carregar em *Hardware* (**Fig. 21**).

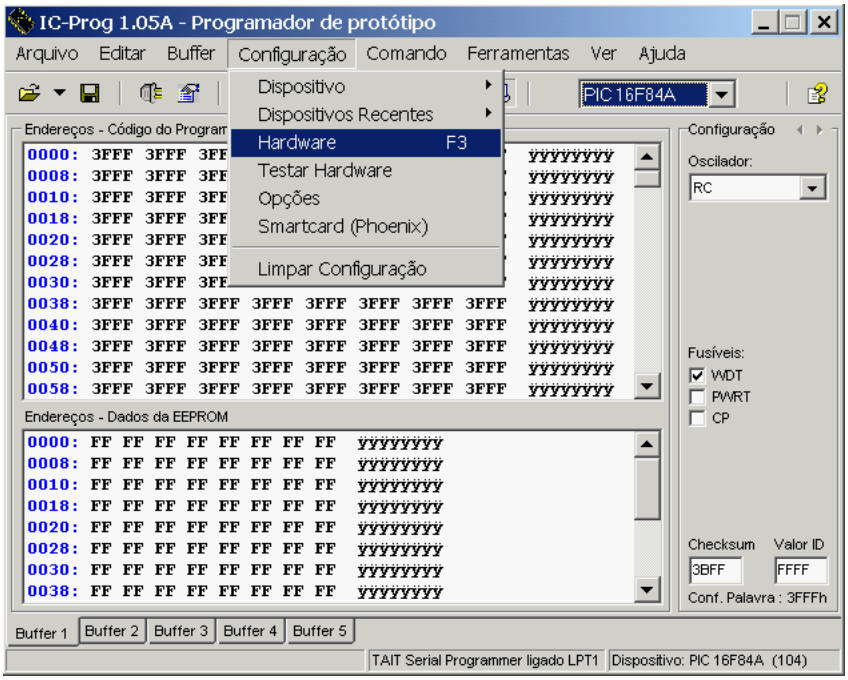


Fig. 21 – Configuração do *IC-Prog* [22] (hardware).

2. Seleccionar *TAIT Serial Programmer* em *Programador* (**Fig. 22**), *Inverter MCLR* e *Inverter VCC*. No *Interface* tanto se pode seleccionar *Directo I/O* como *Windows API*, visto que funciona com qualquer um dos dois. O *Retardo de I/O* deve ser aumentado caso o computador tenha um processador muito rápido (um valor de 10 chega para um Pentium IV a 2GHz).

Fig. 22 – Configuração do *IC-Prog* [22] (hardware - continuação).

5.2.2 Programação do PIC16F876 Através da Linha Série RS232

5.2.2.1 O que é o Bootloader?

Como alternativa ao método anteriormente visto, é possível efectuar o download de programas directamente do PC de desenvolvimento, para o PIC, através da linha série RS232, sem recurso a qualquer dispositivo adicional. Para que isso seja possível é previamente programado no PIC um pequeno programa que controla a comunicação (bootloader) com o PC e que transfere a informação recebida para a memória de programa do PIC. A programação do bootloader é efectuada com ajuda do programador (este processo só é necessário efectuar uma vez).

5.2.2.2 Como Funciona?

O bootloader funciona em conjunto com um programa que corre no PC, designado por PIC downloader 1.08 [20], e que controla o envio de um ficheiro em formato Intel Hex para o PIC. Sempre que o PIC é ligado, inicia-se a execução do bootloader. Este espera um certo tempo limite para ver se recebe informação do PC via porta RS232. Se não recebe nenhuma informação e se existir um programa válido na memória do PIC, então o bootloader inicia a execução desse programa. No caso de receber informação, inicia o processo de transferência e armazenamento de um novo programa.

5.2.2.3 Como Programar?

Para se programar o PIC16F876 (depois de ter sido previamente programado com o bootloader), basta seguir os seguintes passos:

1. Compilar o programa que se pretende transferir para o PIC usando, o Hi-Tech C [15] integrado no MPLAB [14]. O compilador produz um ficheiro em formato Intel Hex (extensão ".hex");
2. Executar o programa PIC downloader 1.08 [20] (**Fig. 23**);

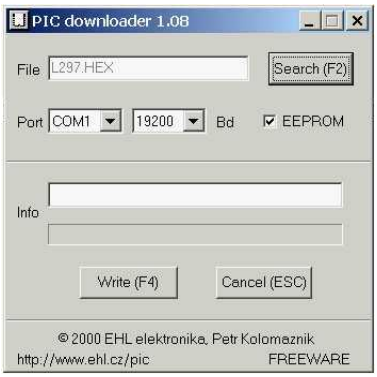


Fig. 23 – Janela do programa PIC downloader 1.08.

3. Seleccionar o ficheiro em formato Intel Hex (extensão ".hex") que se pretende enviar para o PIC através do botão "Search" ou da tecla F2;
4. Configurar a porta de comunicação (Com1 ou Com2) e o baud rate (o baud rate tem que ser igual ao do bootloader que se encontra no PIC, geralmente 19200);
5. Marcar a caixa EEPROM;
6. Iniciar o download através do botão "Write" ou da tecla F4 deverá aparecer a seguinte mensagem "Searching for bootloader" (**Fig. 24**);

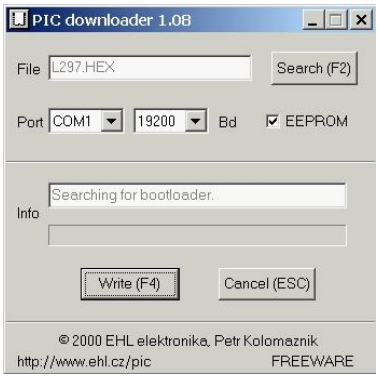


Fig. 24 – "Searching for bootloader".

7. Fazer reset do PIC (premir o botão de reset da placa do PIC) e começa a realizar-se o download (**Fig. 25**) ;

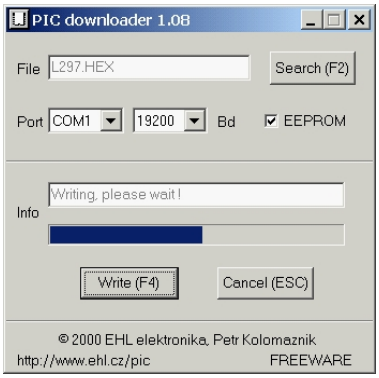


Fig. 25 – Download do ficheiro em progresso.

8. Esperar que o download termine (**Fig. 26**);

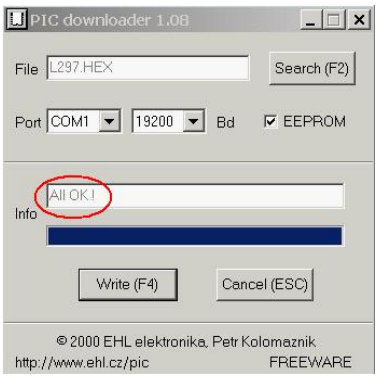


Fig. 26 – Download do ficheiro finalizado (All OK).

9. Finalmente fazer novo reset do PIC e o programa será executado passado alguns segundos (o programa é executado mal acaba o download, mas é sempre conveniente fazer o reset para assegurar que este inicia sem "bugs").

5.3 Configuração do PIC

Para o correcto funcionamento da placa de controlo é necessário proceder á correcta configuração dos pinos do PIC.

5.3.1 Configuração das I/O do PIC

Visto que alguns pinos do PIC podem exercer mais do que uma função, procede-se à correcta configuração dos mesmos, ou seja, primeiro temos que os configurar como saídas ou entradas, e depois atribuir a função desejada a cada pino.

De notar que isto é feito atribuindo valores às variáveis de registo do PIC (**Fig. 27**).

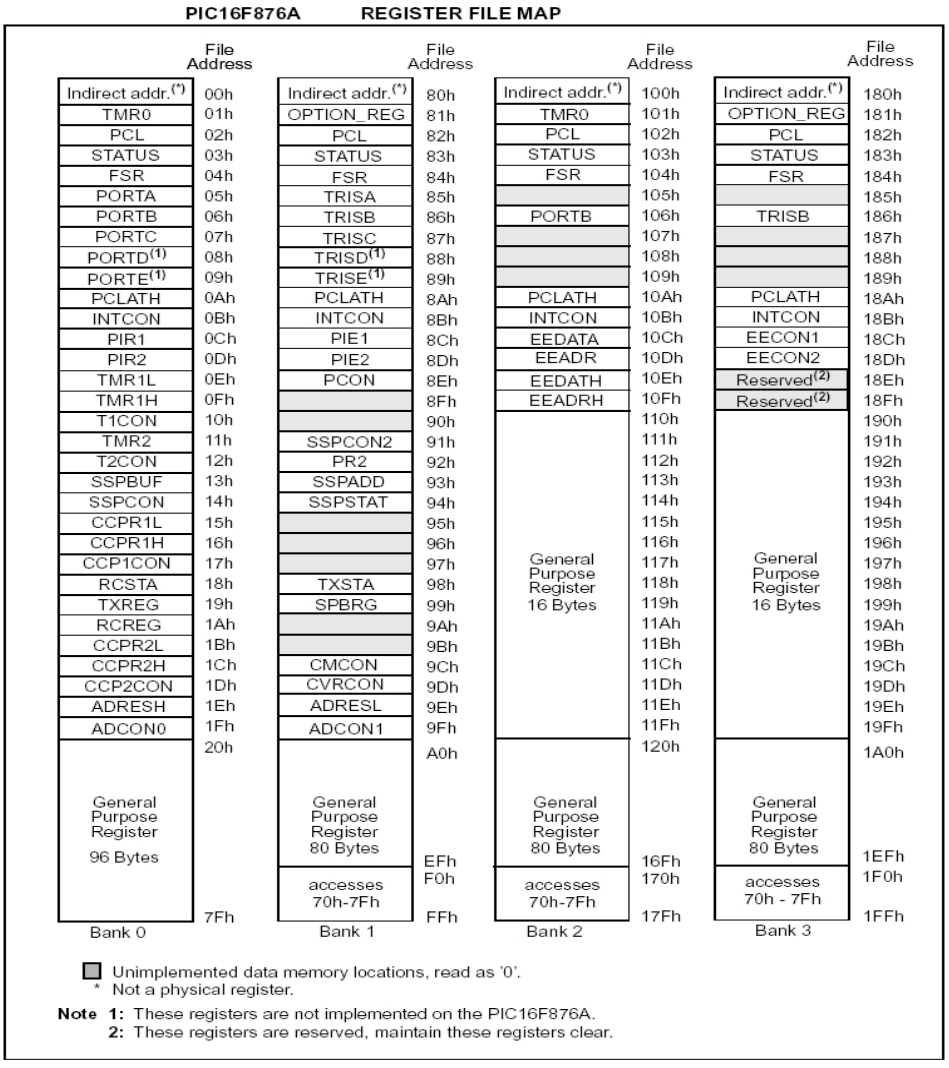


Fig. 27 – Mapa das variáveis de registo do PIC.

Os registos que controlam o funcionamento dos pinos do PIC como saídas ou entradas são os registos *TRISA*, *TRISB* e *TRISC* que correspondem respectivamente ao porto A, porto B e porto C do PIC. No exemplo que se segue podemos ver mais claramente como se efectua a configuração dos três registos (de notar, que se deve incluir sempre o ficheiro pic.h, para que assim o compilador saiba o endereço fisico de cada um dos registos na memória do PIC).

```
#include <pic.h>
TRISA=0b00011001;       /* RA0,RA3,RA4 input, RA1,RA2 e RA5, output,  (0=output, 1=input) */
TRISB=0b00000001;       /* RB0 input, RB1..RB7 outputs */
TRISC=0b10000000;       /* RC7 input, RC0..RC6 output */
```

5.3.2 Configuração da USART (Comunicação RS232)

A comunicação série é efectuada a 9600 baud, com 8 bits de dados e 1 bit de stop, modo assíncrono alta velocidade (*BRGH = 1*). De notar que para alterar a velocidade de comunicação entre o PIC e o PC basta apenas alterar o valor da variável *SPBRG* como podemos observar pela **tabela 1.**

Tabela 1 – Tabela de configuração do registo SPBRG.

BAUD RATE (K)	Fosc = 20 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-
1.2	-	-	-
2.4	-	-	-
9.6	9.615	0.16	129
19.2	19.231	0.16	64
28.8	29.070	0.94	42
33.6	33.784	0.55	36
57.6	59.524	3.34	20
HIGH	4.883	-	255
LOW	1250.000	-	0

Exemplo da configuração da USART.

```
SPBRG = 129;    /* BR=9600    (9600=129, 19200=64, 38400=32, 57600=21, 115200=10, 250000=4)  */
BRGH = 1;       /* BR high speed                  (BRGH=0, Low Speed)                        */
SPEN = 1;       /* Serial Port Enable             (SPEN=0, Serial Port Disable)              */
TXEN = 1;       /* Transmit Enable                (TXEN=0, Transmit Disable)                 */
CREN = 1;       /* Continuous Reception Enable    (CREN=0, Disables Continuous Reception)    */
RCIE = 1;       /* Enables the USART receive interrupt                                       */
```

Para se utilizarem outros valores de baudrate, o valor a colocar no registo SPBRG pode ser calculado através da seguinte formula $Baudrate = \frac{F_{osc}}{16(X+1)}$, em que X é o valor a colocar em *SPBRG.*

5.3.3 Configuração da ADC para Leitura da Posição Vertical (Inclinação)

A correcta configuração da ADC é um elemento importante, para uma correcta obtenção do valor do potenciómetro e logo da correcta posição do sensor laser.

Para se configurar a ADC é necessário configurar os valores de referência a que esta vai trabalhar. Estes poderão ser os valores de tensão de funcionamento do PIC, ou outros externos. Como se verificou que o valor máximo de tensão que existia na entrada da ADC quando o laser estava na sua posição máxima era de 3,7V então optou-se por utilizar um Vref+ externo e o Vref- interno (0V).

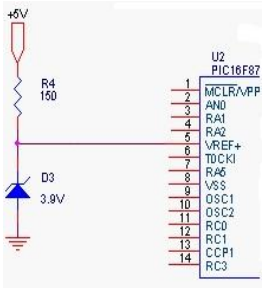


Fig. 28 – Circuito Vref +.

Através da utilização do circuito da **Fig. 28**, o valor de referência (Vref+) da ADC passou a ser igual a 3,9V, ou seja, com a utilização do Vref+ externo conseguiu-se ganhar na resolução de posicionamento do laser, permitindo assim obter um erro de posicionamento do laser de 0,27º, como se demonstra nos cálculos a seguir apresentados.

Dado que o potenciómetro utilizado têm uma resistência de 10KΩ e um percurso angular de 1080º (3 voltas, dados do fabricante), então virá:

$\frac{10000\Omega}{1080^\circ} \approx 9{,}3\Omega/^\circ$, que é a resistência por cada grau que o potenciómetro roda.

A intensidade que percorre o potenciómetro é de $i = \frac{5V}{10000\Omega} = 0{,}5mA$, como a tensão de referência da ADC (Vref+) é de 3,9V e utilizando uma resolução de 10bits, temos que $\frac{3{,}9V}{2^{10}} = 3{,}81mV$, sendo esta a variação mais pequena que a ADC consegue detectar.

Como a relação de transmissão entre o motor passo a passo e o potenciómetro é de 3/5, então:

$R = \frac{3{,}81V}{0{,}5\times10^{-3}} = 7{,}62\Omega$ que é a resistência mínima para fazer variar a ADC de um valor, isto é igual a uma variação angular do potenciómetro de $\frac{7{,}62\Omega}{9{,}3\Omega/^\circ} = 0{,}82^\circ$, que se traduz numa variação angular do motor passo a passo de $0{,}82^\circ\times\frac{5}{3} = 1{,}37^\circ \Rightarrow$ uma variação angular no laser de 0,27º.

Exemplo da programação da ADC.

```
ADCON0=0b10000001; /* Fosc/32, channel 0(RA0), enable ADC converter                    */
ADCON1=0b0101;     /* Left justified, RA0, RA1 analog, RA3 Vref+, RA2, RA4...RA7 digital  */
```

5.3.4 Configuração do PWM

Existem dois módulos, chamados CCP, que podem operar em modo "capture", "compare" ou "PWM". Tanto o módulo CCP1 como o CCP2 são iguais em funcionamento, com a excepção de possuírem registos diferentes. Para comandar o motor passo a passo, são necessárias ondas quadradas, assim sendo, apenas o módulo CCP1 foi utilizado para essa função, tendo o mesmo que ser configurado em modo "PWM – Pulse Width Modulation" (registo *CCP1CON*, bit3-0) para poder gerar as referidas ondas quadradas.

De notar ainda que os dois módulos podem funcionar de maneiras diferentes, ou seja, o CCP1 como "PWM" e o CCP2 como "capture", "compare" ou "PWM" (**Fig. 29**), visto que são módulos independentes. Para mais informações sobre os dois módulos consultar "**PIC16F87xA Data Sheet –** 28/40-pin Enhanced Flash Microcontrollers, capitulo 8" e "**PICmicro Mid-Range MCU Family**, capitulo 14".

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time-base
Capture	Compare	The compare should be configured for the special event trigger, which clears TMR1
Compare	Compare	The compare(s) should be configured for the special event trigger, which clears TMR1
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt)
PWM	Capture	None
PWM	Compare	None

Fig. 29 – Interacção dos dois módulos CCP.

Escrevendo valores diferentes no registo *PR2*, alteramos o valor da frequência do PWM. O valor a colocar em *PR2* pode ser calculado através da seguinte fórmula $\mathrm{PR2} = \frac{\mathrm{F_{osc}}}{4 \times \mathrm{PWM_{Freq}} \times \mathrm{TMR2_{prescaler}}} - 1$.

O valor do duty cycle é escrito no registo CCPR1L e calculado através da seguinte fórmula $\mathrm{Duty\ Cycle} = \frac{\mathrm{F_{osc}}}{8 \times \mathrm{PWM_{Freq}} \times \mathrm{TMR2_{prescaler}}}$.

Na **tabela 2** podemos ver os trinta e um diferentes valores implementados para a variação da velocidade do motor passo a passo. No cálculo da velocidade de rotação do motor foi usada a seguinte fórmula $\mathrm{Rpm} = \frac{60 \times 0{,}9 \times \mathrm{Frequência}}{360 \times 256}$.

Tabela 2 – Tabela de valores do PWM.

Configuração do PWM				Velocidade		
Frequência (Hz)	**Prescaler**	**PR2**	**Duty Cycle**	**Motor (rpm)**	**Sensor(rpm)**	**Sensor(º/s)**
2000	16	9B	4E	1,172	0,234	1,406
5000	16	3D	1F	2,930	0,586	3,516
10000	16	1E	F	5,859	1,172	7,031
15000	16	13	A	8,789	1,758	10,547
20000	16	E	7	11,719	2,344	14,063
25000	16	B	6	14,648	2,930	17,578
30000	16	9	5	17,578	3,516	21,094
35000	16	7	4	20,508	4,102	24,609
40000	16	6	3	23,438	4,688	28,125
45000	16	5	3	26,367	5,273	31,641
50000	4	18	C	29,297	5,859	35,156
55000	4	15	B	32,227	6,445	38,672
60000	4	13	A	35,156	7,031	42,188
65000	4	12	9	38,086	7,617	45,703
70000	4	10	8	41,016	8,203	49,219
75000	4	F	8	43,945	8,789	52,734
80000	4	E	7	46,875	9,375	56,250
85000	4	D	7	49,805	9,961	59,766
90000	4	C	6	52,734	10,547	63,281
95000	1	33	1A	55,664	11,133	66,797
100000	1	31	19	58,594	11,719	70,313
110000	1	2C	16	64,453	12,891	77,344
120000	1	28	14	70,313	14,063	84,375
130000	1	25	13	76,172	15,234	91,406
140000	1	22	11	82,031	16,406	98,438
150000	1	20	10	87,891	17,578	105,469
160000	1	1E	F	93,750	18,750	112,500
170000	1	1C	E	99,609	19,922	119,531
180000	1	1A	D	105,469	21,094	126,563
190000	1	19	D	111,328	22,266	133,594
200000	1	18	C	117,188	23,438	140,625

5.4 Programa de Controlo

Fig. 30 – Home Position.

O programa de controlo do sistema, começa primeiro por configurar o PIC, depois verifica a posição em que se encontra o laser e então posiciona-o na posição horizontal: "Home Position" (**Fig. 30**). Após ter completado todo o ciclo representado no diagrama de blocos da **Fig. 31**, o PIC entra num ciclo infinito. Cada vez que recebe uma mensagem externa é gerado um interrupt e o PIC processa a mensagem recebida e reage consoante o seu conteúdo.

De notar que todas as mensagens enviadas para a placa de controlo apenas têm 8 bits, tornando assim a comunicação fácil e de rápida implementação. Todas as mensagens a enviar têm que respeitar o formato na **tabela 3**.

Tabela 3 – Mensagens a enviar para a placa e sua descrição

Mensagem									**Descrição**
1	1	1	0	0	0	0	0		Stop imediato
0	0	0	0	0	0	0	0		Start imediato
0	0	0	x	x	x	x	x		Start com delay inicial (de 0 a 31)
0	0	1	x	x	x	x	x		Velocidade de scan (de 0 a 31)
0	1	0	x	x	x	x	x		Limite superior grosso (incrementos 10º) (de 0 a 31)
0	1	1	x	x	x	x	x		Limite superior fino (incrementos de 0.5º) (de 0 a 19)
1	0	0	x	x	x	x	x		Limite inferior grosso (incrementos 10º) (de 0 a 31)
1	0	1	x	x	x	x	x		Limite inferior fino (incrementos 0.5º) (de 0 a 19)
1	1	0	1	1	1	1	1		Configuração actual do sistema
1	1	0	0	0	0	0	0		Scan continuo

Os valores representados por x podem tomar qualquer valor.

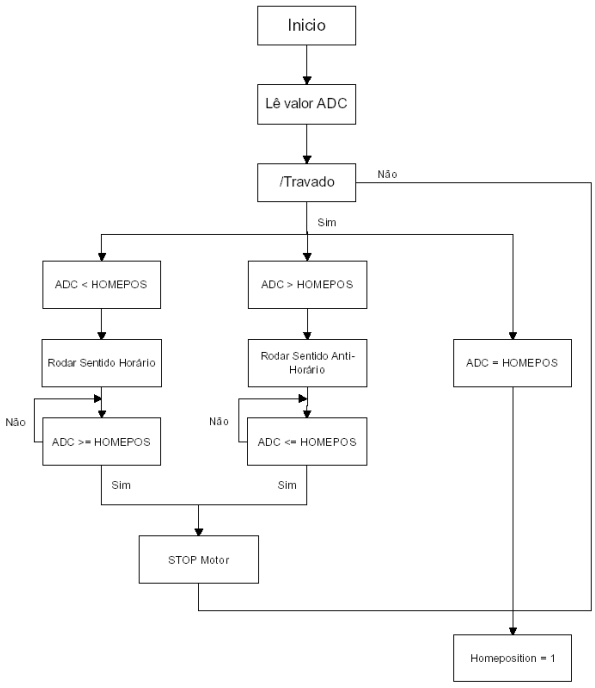


Fig. 31 – Diagrama de blocos da inicialização da placa controladora.

5.5 Aquisição de Dados

O comando do sensor laser pode ser feito de qualquer parte do mundo, uma vez que através uma ligação Telnet ao PC que se encontra no Robuter, é possível executar o programa acquire que permite criar um ficheiro .txt, que contêm informação da posição do sensor e a leitura do laser nessa posição. O processo de aquisição, tanto da posição como dos dados da leitura do laser, é feita de forma sincronizada, ou seja, cada vez que o laser recebe o valor da inclinação do laser, escreve esse valor no ficheiro criado e de seguida lê os valores do laser e escreve-os também no mesmo ficheiro. Este processo é repetido até o scan estar concluído, altura em que o ficheiro é enviado automáticamente via ftp para o PC de destino (que têm que ter instalado um ftp server) e ai visualizado através da utilização do *Matlab*®.

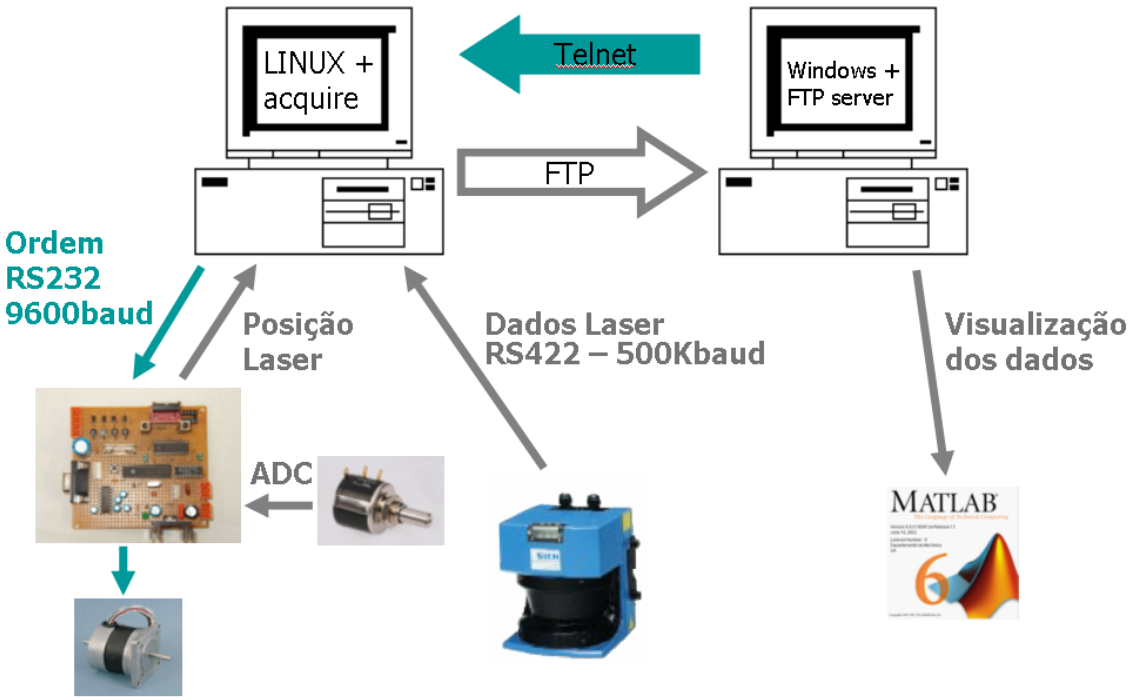


Fig. 32 – Diagrama da aquisição de dados.

6. Resultados Obtidos

Para a visualização dos resultados obtidos, foi realizado um pequeno script em *Matlab*® que lê um ficheiro .txt, separa as coordenadas da posição do sensor laser (inclinação), as distâncias aos objectos e faz por ultimo a sua representação gráfica (**Fig. 33 e 34**).

Podemos ver claramente através da análise das imagens, as potencialidades do sistema construído. De notar que a baixas velocidades notam-se imperfeições na representação de alguns objectos, isso deve-se ao facto de o movimento do motor não se realizar muito suavemente, fazendo com que o laser sofra vibrações.

Claramente, pode-se observar que por detrás de cada objecto forma-se uma espécie de sombra, impedindo assim a "visualização" por parte do laser de tudo o que se encontre nessa sombra.

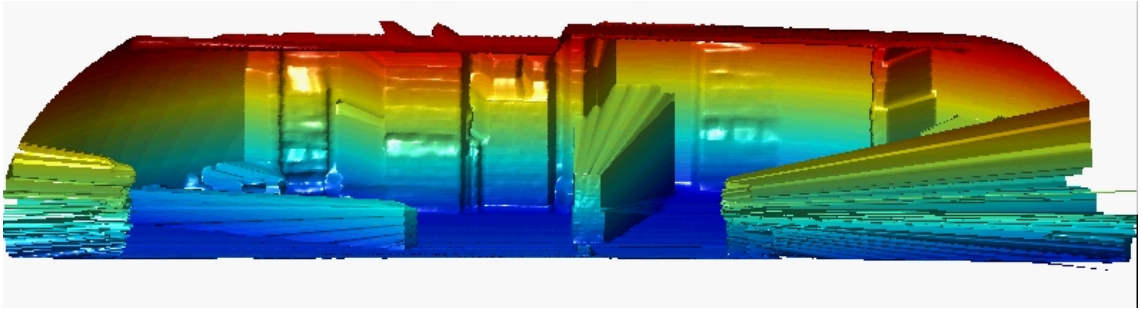


Fig. 33 – Resultado obtido a uma velocidade angular do laser de 3,5º/s.

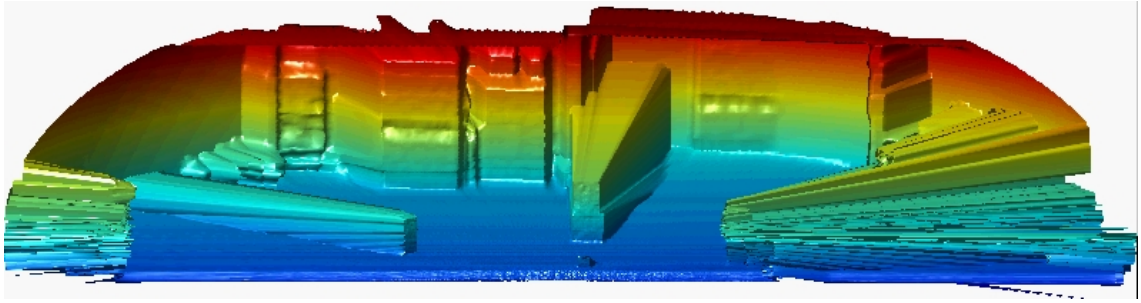


Fig. 34 – Resultado obtidos a uma velocidade angular do laser de 21,1º/s.

Fig. 35 – Imagem do ambiente real.

7. Conclusão

O trabalho elaborado ao longo deste projecto de automação revelou-se bastante aliciante, visto que foram postos em prática muitos conhecimentos adquiridos até então, mas também a capacidade de pesquisa e de invenção que um engenheiro têm que possuir para, por vezes, conseguir resolver alguns problemas com que se depara.

Todos os objectivos propostos no inicio deste projecto foram cumpridos, faltando apenas avaliar em profundidade a influência da dinâmica do movimento sobre desempenho do sensor original que, todavia, pelos exemplos obtidos não parece muito notório à primeira vista.

Futuramente seria interessante dotar a estrutura de um sistema de travão, que fosse capaz de bloquear o motor passo a passo em caso de falta de energia e impedir que o sensor embata nos fins de curso mecânicos.

8. Bibliografia

Livros:

[1] - **Mechanical Engineering Design,** J. Edward Shigley Charles R. Mischke, MacGraw-Hill.

[2] - **Apontamentos de Orgãos de Máquinas e Anteprojecto,** 2001/2003.

[3] - **PIC Microcontroller Project Book,** John Iovine, MacGraw-Hill.

[4] - **Programming and Customizing the PIC Microcontroller,** Myke Predko, MacGraw-Hill.

[5] - **BASIC for PIC microcontrollers,** Nebojsa Matic, livro online, http://www.mikroelektronika.co.yu/english/product/books/picbasicbook/00.htm.

[6] - **PIC microcontrollers for beginners,too!,** Nebojsa Matic, livro online, http://www.mikroelektronika.co.yu/english/product/books/PICbook/picbook.htm

URL's:

[7] - URL: **Sick homepage (Sensor Laser *LMS200*)**, www.sick.com, www.sick.de, www.sick.fr, www.sickoptic.com/laser.htm.

[8] - URL: **Homepage da Mitsubishi (PLC *FX_{2N}-16MR-DS* e Pulse Generator Unit *FX_{2N}-1PG*)**, www.mitsubishi-automation.com.

[9] - URL: http://pagina.netc.pt/~nf16332a/umist/smotor.html.

[10] - URL: www.ams2000.com.

[11] - URL: www.mrshp.hpg.ig.com.br/rob/m_passo.htm.

[12] - URL: **Fast acquiring and analysis of three dimensional laser range data,** http://capehorn.gmd.de:8080/.

[13] - **Autonomous Coverage Operations In Semi-Structured Outdoor Environments,** http://www.ri.cmu.edu/pub_files/pub3/batavia_parag_2002_1/batavia_parag_2002_1.pdf

[14] - URL: **Homepage do fabricante dos microcontroladores PIC. Muita informação referente á programação dos microcontroladores, exemplos, etc.,** www.microchip.com.

[15] - URL: **Homepage da Hi-TECH Software (compilador linguagem C para PIC's)**, http://www.htsoft.com/

[16] - URL: **Homepage com diversos projectos (sensors temperatura, servomotors, motores passo a passo, etc.) para PIC's, muito boa,** http://www.rentron.com/PicBasic1.htm

[17] - URL: **Homepage de fabricante de software para programação de PIC's, software muito popular e que permite programar PIC's, EEPROM, etc.,** http://www.ic-prog.com/

[18] - URL: **Homepage com muita informação sobre bootloader's e programação em C, página excelente**, http://www.microchipc.com/

[19] - URL: **Homepage do fabricante do chip's L298 e L297 utilizados na placa de controlo**, http://eu.st.com/stonline/welcome.htm

[20] - URL: **Homepage do software Pic Downloader 1.08**, http://www.ehl.cz/pic/pic_e.htm

[21] - URL: **Homepage do programador de PIC's,** http://www.bobblick.com/techref/projects/projects.html

[22] - URL: **Homepage do software IC-Prog, contém informações sobre todos os programadores suportados e respectivos esquemas**, http://www.ic-prog.net/index1.htm

ANEXOS

Anexo 1 – Programa PIC (L297main.C)

```
/*******************************************************************************************
                                        L297main.C

                                Universidade de Aveiro
                                Departamento de Mecânica
                                Miguel Matos Dias

                                3 Junho 2003
*******************************************************************************************/
#include <pic.h>
#include "c:\programa\l297\pl297.h"

#define SENTIDO   RC0     /* Sentido de rotacao motor (CW/CCW)          */
#define HFSTEP    RC3             /* HALF/FULL Step                             */
#define EN1       RC4             /* Enable bridge                              */
#define TRAVADO   RB0             /* Scanner travado                  */

bank1 unsigned int LimiteSG = 13;     /* Limite Superior Grosso 130º (Default)      */
bank1 unsigned int LimiteSF = 10;     /* Limite Superior Fino 5º (Default)          */
bank1 unsigned int LimiteIG = 4;      /* Limite Inferior Grosso 40º (Default)       */
bank1 unsigned int LimiteIF = 10;     /* Limite Inferior Fino 5º (Default)          */
bank1 long int SUP;                   /* Limite Superior                  */
bank1 long int INF;                   /* Limite Inferior                            */
bank1 unsigned int posicionar = 0;    /* Scan a posicionar para realizacao de scan  */
bank1 unsigned int Tprimeira = 0;     /* Ainda nao existe nenhuma leitura da ADC    */
bank1 unsigned int valorADC;                  /* Valor da ADC                       */
bank1 unsigned int HOMEPOS = 336; /* Valor da ADC no home position      */
bank1 unsigned int HP = 0;                    /* Home position por realizar         */
bank1 unsigned int HPSTOP = 0;
bank1 unsigned int WaitScan;
bank1 unsigned int Mens;
bank1 unsigned int MaxADC = 970;              /* Valor da ADC quando o laser esta posicionado a 270º  */
bank1 unsigned int MinADC = 20;               /* Valor da ADC quando o laser esta posicionado a 0º          */
bank1 unsigned int continuo = 0;
bank1 unsigned int PR2antigo;
bank1 unsigned int DUTYantigo;
bank1 unsigned int T2antigo;
bank1 unsigned int Vel;
bank1 unsigned int Scan = 0;
bank1 unsigned int j = 0;
bank1 unsigned int Estatico = 0;

bank1 int presT2[] = {0x04,0x05,0x06};
bank2 int velocidade[] = {0x9B, 0x3D, 0x1E, 0x13, 0xE, 0xB, 0x9, 0x7, 0x6, 0x5, 0x18, 0x15, 0x13, 0x12, 0x10,
0xF, 0xE, 0xD, 0xC, 0x33, 0x31, 0x2C, 0x28, 0x25, 0x22, 0x20, 0x1E, 0x1C, 0x1A, 0x19, 0x18};
bank3 int dutycycle[] = {0x4E, 0x1F, 0xF, 0xA, 0x7, 0x6, 0x5, 0x4, 0x3, 0x3, 0xC, 0xB, 0xA, 0x9, 0x8, 0x8, 0x7,
0x7, 0x6, 0x1A, 0x19, 0x16, 0x14, 0x13, 0x11, 0x10, 0xF, 0xE, 0xD, 0xD, 0xC};

void Home(void)
{
        stop();
        Scan = 0;
        HPSTOP = 1;
        HP = 1;
        PR2 = PR2antigo;
        CCPR1L = DUTYantigo;
        T2CON = T2antigo;
}

void GuardaValor(void)
{
        PR2antigo = PR2;
        DUTYantigo = CCPR1L;
```

```
	T2antigo = T2CON;
	PR2 = velocidade[9];
	CCPR1L = dutycycle[9];
	T2CON = presT2[2];
}

void Homeposition(void)				/* Coloca o scanner no home position */
{
	GuardaValor();
	if (valorADC > HOMEPOS)
	{
		SENTIDO = 1;			/* Rotacao do motor no sentido CCW */
		start(0);
	}
	if (valorADC < HOMEPOS)
	{
		SENTIDO = 0;			/* Rotacao do motor no sentido CW */
		start(0);
	}
	if (valorADC == HOMEPOS)		/* Ja se encontra na home position */
		HP = 1;
}

void RetornaHome(void)
{
	stop();
	HP = 0;
	wait(1000);
	Homeposition();
}

void calculo(void)				/* Calcula o valor superior e inferior para o scan */
{
long int aux, aux1, aux2, difMax_Min;

	difMax_Min = MaxADC - MinADC;
	aux = 10*LimiteSG+LimiteSF/2;
	if (aux > 270)					/* Se o limite superior for maior que 270º assume 270 */
		aux = 270;
	SUP = MinADC + (aux * difMax_Min)/270;	/* Calcula o valor superior onde o motor deve parar    */

	aux1 = 10*LimiteIG+LimiteIF/2;
	if (aux1 > 270)
		aux1 = 270;
	INF = MinADC + (aux1 * difMax_Min)/270;	/* Calcula o valor inferior onde o motor deve parar    */

	if (INF > SUP)				/* Caso o limite superior seja menor que o limite inferior */
	{
		aux2 = SUP;
		SUP = INF;			/* Limite superior toma o valor do limite inferior */
		INF = aux2;			/* Limite inferior toma o valor do limite superior */
	}

	if (INF == SUP)
		Estatico = 1;
	else Estatico = 0;
}

void analmsg(unsigned char mensagem)
{
unsigned char Cabecalho, Corpo;

	Cabecalho = (mensagem & 0b11100000) >> 5;   /* Separa o cabecalho da mensagem */
	Corpo = (mensagem & 0b00011111);		/* Separa o corpo da mensagem */

	if (Cabecalho == 0 | (Cabecalho == 6 & Corpo == 0))	/* START scan */
	{
```

```
		Mens = Cabecalho;
		if (TRAVADO == 1 & HP == 1)			/* Scanner destravado */
		{
			GuardaValor();
			calculo();
			HPSTOP = 0;
			if (INF < HOMEPOS)
				SENTIDO = 1;		/* Rotacao do motor no sentido CCW */
			else SENTIDO = 0;
			if (valorADC != INF)
				start(1);
			posicionar = 1;
			Scan = 1;
			WaitScan = Corpo;
		}
		if (TRAVADO == 0)
			printStr("\n Scanner travado!!!! Destrave Scanner primeiro! ");
		if (HP == 0)
			printStr("\n Espere por favor! O scanner ainda n s encontra no home position ");

}

if (Cabecalho == 1)		/* Velocidade de Scan*/
{
		if (Corpo < 1)
			Corpo = 1;
		if (Corpo > 31)
			Corpo = 31;
		Vel = Corpo;
		PR2 = velocidade[Corpo-1];
		CCPR1L = dutycycle[Corpo-1];
		if (Corpo <= 10)
			T2CON = presT2[2];				/* Timer2 prescaler igual a 16 */
			else if (Corpo > 10 & Corpo <= 19)
				T2CON = presT2[1];			/* Timer2 prescaler igual a 4 */
				else T2CON = presT2[0];		/* Timer2 prescaler igual a 1 */

}

if (Cabecalho == 2)		/* Limite superior grosso */
{
		if (Corpo <= 0)
			Corpo = 0;
		if (Corpo > 27)
			Corpo = 27;

		LimiteSG = Corpo;
}

if (Cabecalho == 3)		/* Limite superior fino */
{
		if (Corpo <= 0)
			Corpo = 0;
		if (Corpo > 19)
			Corpo = 19;

		LimiteSF = Corpo;
}

if (Cabecalho == 4)		/* Limite inferior grosso */
{
		if (Corpo <= 0)
			Corpo = 0;
		if (Corpo > 27)
			Corpo = 27;

		LimiteIG = Corpo;
```

```
		}

		if (Cabecalho == 5)		/* Limite inferior fino */
		{
			if (Corpo <= 0)
				Corpo = 0;
			if (Corpo > 19)
				Corpo = 19;

			LimiteIF = Corpo;
		}

		if (Cabecalho == 6 & Corpo == 31)	/* Envia configuracao do sistema */
		{
			printStr("\n Velocidade de Scan = ");
			printVal(10,Vel);
			printStr("\n Limite Superior = ");
			printVal(10,10*LimiteSG + LimiteSF/2);
			printStr("\n Limite Inferior = ");
			printVal(10,10*LimiteIG + LimiteIF/2);
		}

		if (Cabecalho == 7)		/* STOP scan */
		{
			continuo = 0;
			Scan = 0;
			Estatico = 0;
			stop();
		}
}

void lermsg(void)
{
unsigned char x, palavra;

   if (RCIF == 1)
   {
		if (OERR == 1)			/* Overrun bit error (RCREG full) */
		{
			CREN = 0;
			CREN = 1;
			printStr("\n CREN ");
		}

		else if (FERR == 1)		/* Framing error bit, stop bit is detected as a low level */
		{
			x = RCREG;				/* Discard word */
			printStr("\n FERR ");
		}

		else
		{
			palavra = RCREG;		/* Palavra recebida */

		}
   }

  if ((Scan == 0 & HP == 1) | palavra == 224)	/* So se estiver no Home Position é que analisa mensagem */
   {
			analmsg(palavra);
  }

}

void interrupt ISR(void)
{
```

```
if (RCIF == 1)                  /* Receive interrupt flag routine */
{
        lermsg();
        RCIE = 0;               /* Disables the USART receive interrupt */
        RCIE = 1;               /* Enables the USART receive interrupt */
}


if (T0IF == 1)                  /* Timer0 interrupt flag routine */
{
        if (Tprimeira == 1)
        {
                valorADC = analog(0,10);

                if (posicionar == 0 & HP == 1 & Scan == 1)
                {
                 j = j + 1;
                 if (j == 3)
                        {
                        printVal(10,valorADC);      /* Envia o valor do potenciometro */
                        printStr("\n");
                        j = 0;
                        }
                }
                if (posicionar == 0 & HP == 0 & (Scan == 1 | (Scan == 0 & Estatico == 0)))
                {
                j = j + 1;
                if (j == 3)
                        {
                          printStr("-1\n");             /* Envia o valor -1 depois do scan completo */
                          j = 0;
                        }
                }

                if (TRAVADO == 0)                       /* Se travado em qualquer altura o scaner para */
                        stop();

                if (valorADC <= MinADC+10 | valorADC >= MaxADC -10)
                {
                        stop();
                        HP = 0;
                        wait(1000);
                        Homeposition();

                        if (valorADC >= MaxADC)
                                printStr("\n Erro! - Limite maximo atingido!!");
                        if (valorADC <= MinADC)
                                printStr("\n Erro! - Limite minimo atingido!!");
                }

                if (HP == 0)
                {
                        if (SENTIDO == 1 & valorADC <= HOMEPOS)
                                Home();
                        if (SENTIDO == 0 & valorADC >= HOMEPOS)
                                Home();
                }


                if (HPSTOP == 0 & HP == 1 & posicionar == 0 & continuo == 0 & Estatico == 0)
                {
                        if (SENTIDO == 1 & valorADC <= INF)
                                RetornaHome();
                        if (SENTIDO == 0 & valorADC >= SUP)
                                RetornaHome();
                }
```

```
		if (HP == 1 & posicionar == 1 & continuo == 0)
		{
			if (valorADC <= INF & SENTIDO == 1)
			{
				stop();
				SENTIDO = 0;
				PR2 = PR2antigo;
				CCPR1L = DUTYantigo;
				T2CON = T2antigo;
				posicionar = 0;
				if (Mens == 6)
					continuo = 1;
				if (Estatico == 0)
					start(WaitScan);
			}

			if (valorADC >= INF & SENTIDO == 0)
			{
				stop();
				SENTIDO = 0;
				PR2 = PR2antigo;
				CCPR1L = DUTYantigo;
				T2CON = T2antigo;
				posicionar = 0;
				if (Mens == 6)
					continuo = 1;
				if (Estatico == 0)
					start(WaitScan);
			}
		}

		/************************/
		/*    Scan continuo     */
		/************************/

		if (HP == 1 & posicionar == 0 & continuo == 1)
		{
			if (SENTIDO == 0 & valorADC >= SUP)
			{
				stop();
				SENTIDO = 1;
				start(0);
			}
			if (SENTIDO == 1 & valorADC <= INF)
			{
				stop();
				SENTIDO = 0;
				start(0);
			}
		}
T0IF = 0;
}

if (Tprimeira == 0)		/* Le valor da ADC */
{
		T0CS = 0;			/* Internal instruction cycle clock */
		TMR0 = 57;
		T0IF = 0;
		valorADC = analog(0,10);
		Tprimeira = 1;
		if (TRAVADO == 1)
		{
			Homeposition();
		}
		if (TRAVADO == 0)			/* Ja se encontra na home position */
		{
			HP = 1;
```

```
                                }
                        }
                }
}

/*********************************************************************/
/*                          PROGRAMA PRINCIPAL                       */
/*********************************************************************/

void main(void)
{
        initPic();          /* Configura o PIC */
    while(1);
}
```

Anexo 2 – Programa PIC (PL297.C)

```
/*****************************************************************************************
                                        PL297.C

                                Universidade de Aveiro
                                Departamento de Mecânica
                                Miguel Matos Dias

                                Julho/2003
*****************************************************************************************/
#include <pic.h>

#define SENTIDO RC0     /* Sentido de rotacao motor (CW/CCW)          */
#define HFSTEP  RC3     /* HALF/FULL Step                             */
#define EN1     RC4     /* Enable bridge 1                            */
#define TRAVADO RB0     /* Scanner travado                            */
#define FreqDef 0xE     /* PR2=PWM period, 0xB corresponds to 20KHz (default)*/
#define DutyDef 0x7     /* Configura o Duty Cycle (50%) (default para 20KHz)  */


unsigned char bin2asc(unsigned char);           /* Converte um numero binario para ASCII*/

/******** Rotinas ********/
void initPic()
{

        TRISA=0b00011001;       /* RA0,RA3,RA4 input, RA1,RA2 e RA5, output,  (0=output, 1=input) */
        TRISB=0b00000001;       /* RB0 input, RB1..RB7 outputs                                    */
        TRISC=0b10000000;       /* RC7 input, RC0..RC6 output                                     */
        EN1 = 1;                /* Desliga a bridge do L298                                       */
        SENTIDO = 0;            /* Rotacao do motor no sentido do ponteiro dos relogios           */
        HFSTEP = 1;             /* Modo Half Step                                                 */
        GIE = 1;                /* Global interrupt enable bit (enables all unmasked interrupts)  */
        PEIE = 1;               /* Peripheral interrupt enable bit                                */

/*****************************************************************************************/
/*                              Programacao do PWM1 e PWM2                               */
/*****************************************************************************************/

/* The following steps configure the CCP module for PWM operation      */

/* 1º Establish the PWM period by writing to the PR2 register          */

        PR2 = FreqDef;          /* PR2=PWM period default              */
        CCPR1L = DutyDef;       /* Duty Cycle default                  */

/* 2º Establish the TMR2 prescale value and enable Timer2 by writing to T2CON  */

        T2CON = 0x06;           /* Timer2 is ON, prescale value (0x04=1,0x05=4 ou 0x06=16) */

/*****************************************************************************************/
/*                                  Programacao da Usart                                 */
/*****************************************************************************************/

        SPBRG = 129;    /* BR=9600 (9600=129, 19200=64, 38400=32, 57600=21, 115200=10      */
        BRGH = 1;       /* BR high speed                (BRGH=0, Low Speed)                */
        SPEN = 1;       /* Serial Port Enable           (SPEN=0, Serial Port Disable)      */
        TXEN = 1;       /* Transmit Enable              (TXEN=0, Transmit Disable)         */
        CREN = 1;       /* Continuous Reception Enable (CREN=0, Disables Continuous Reception) */
        RCIE = 1;       /* Enables the USART receive interrupt                             */

/*****************************************************************************************/
/*                                  Programacao da ADC                                   */
/*****************************************************************************************/
```

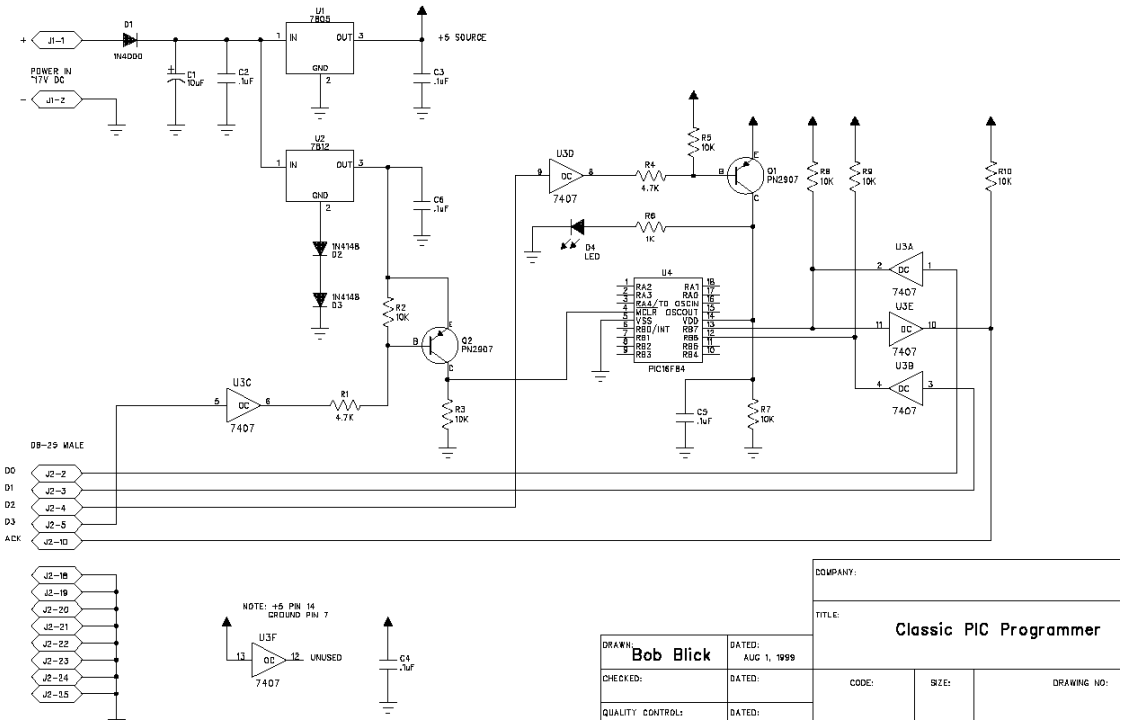
```
	ADCON0=0b10000001; /* Fosc/32, channel 0(RA0), enable ADC converter                    */
	ADCON1=0b0101;     /* Left justified, RA0, RA1 analog, RA3 Vref+, RA2, RA4...RA7 digital */

/*****************************************************************************************/
/*                                    Programacao do Timer0                              */
/*****************************************************************************************/

	T0CS = 0;              /* Internal instruction cycle clock                  */
	T0SE = 0;              /* Increment on low-to-high transition on T0CKL pin  */
	PSA = 0;               /* Prescaler is assigned to the Timer0 module        */
	PS2 = 1;               /* Prescaler 256                                     */
	PS1 = 1;
	PS0 = 1;
	T0IE = 1;              /* TMR0 overflow interrupt enable bit */
	TMR0 = 57;
}

void printStr(const char *str)
{
	while( *str != 0 ) {       /* Envia todos os chars ate ao char nulo */
		while( TXIF == 0 );
			TXREG = *str++;
	}
}

void printVal(unsigned char base, unsigned int val)
{
/* Impressao do valor em hexadecimal, o parametro "base" e 16                */
/* Impressao do valor em decimal, o parametro "base" e 10                    */

	unsigned char str[6]="     ";   /* Para base 2 de valores de 16 bits */
	/* este array deve ser inicializado */
	/* com 16 espacos - char str[17]=" (16) " */
	signed char i, num;

	i = 4;
	do {
		num = val % base;
		str[i--] = bin2asc(num);
		val /= base;
	} while( val > 0 );

	str[5]=0;
	printStr(str);
}

unsigned char bin2asc(unsigned char num)
{
/* Converte 1 digito binario em ASCII */

	num += 0x30;
	if(num > 0x39)
		num += 0x07;
	return num;
}

void wait(unsigned int value)
{
	unsigned int i,k;

	for(k=0;k<value;k++)                /* Wait value*100us          */
		for(i=0;i<400;i++)              /* Wait 100us                */
			asm("nop");

}
```

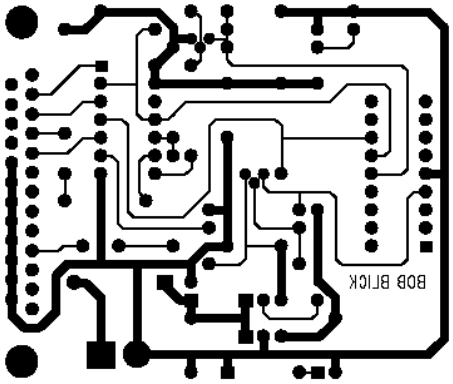
```
int analog(unsigned char chn, unsigned int resolucao)
{
	unsigned int i, leitura;

	chn &= 0x0007;		/* If (chn!=0..7) set chn 0 as default */

	if (resolucao==10)
		for(i=0;i<353;i++)		/* Wait the required acquisition time (aprox. 17.3 uS)	*/
			asm("nop");		/* Para se obter uma resolução de 10 bits		*/

	if (resolucao==9)
		for(i=0;i<324;i++)		/* Wait the required acquisition time (aprox. 16.2 uS)	*/
			asm("nop");		/* Para se obter uma resolução de 9 bits		*/

	ADGO=1;			/* Start conversion (Set GO Bit)		*/
	while(ADGO);		/* Wait for A/D conversion to complete		*/

	if (resolucao==10)
		leitura = (ADRESH << 2) + (ADRESL >> 6);	/* Forma o inteiro com resolução de 10 bits */

	if (resolucao==9)
		leitura = (ADRESH << 1) + (ADRESL >> 7);	/* Forma o inteiro com resolução de 9 bits	*/

	return(leitura);
}

void stop()
{
	CCP1M3 = 0;		/* Desactiva o gerador PWM1 do PIC */
	CCP1M2 = 0;
}

void start(unsigned int Tempo)
{
	wait(Tempo*1000);	/* Tempo de Delay em segundos		*/
	CCP1M3 = 1;		/* Activa o gerador PWM1 do PIC		*/
	CCP1M2 = 1;
}
```

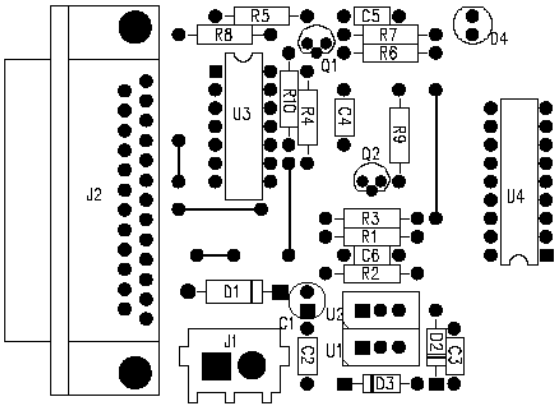
Anexo 3 – Programa PIC (PL297.H)

```
/*******************************************************************************************
                                        PL297.H

                                Universidade de Aveiro
                                Departamento de Mecânica
                                Miguel Matos Dias

                                Março/2003
*******************************************************************************************/

void initPic(void);                     /* Programa varios parametros para o funcionamento de outras funcoes */

void printStr(const char*);                     /* Envia para a porta serie a string apontada por str */

void printVal(unsigned char, unsigned int);     /* Envia para a porta serie a valor em hexadecimal ou decimal */

void wait(unsigned int);                        /* Forca o programa a esperar value*100us     */

int analog(unsigned char,unsigned int);         /* Devolve a leitura do canal chn da ADC      */

void stop(void);                                /* Desliga o motor em caso de emergencia */

void start(unsigned int);                       /* Liga o motor */
```

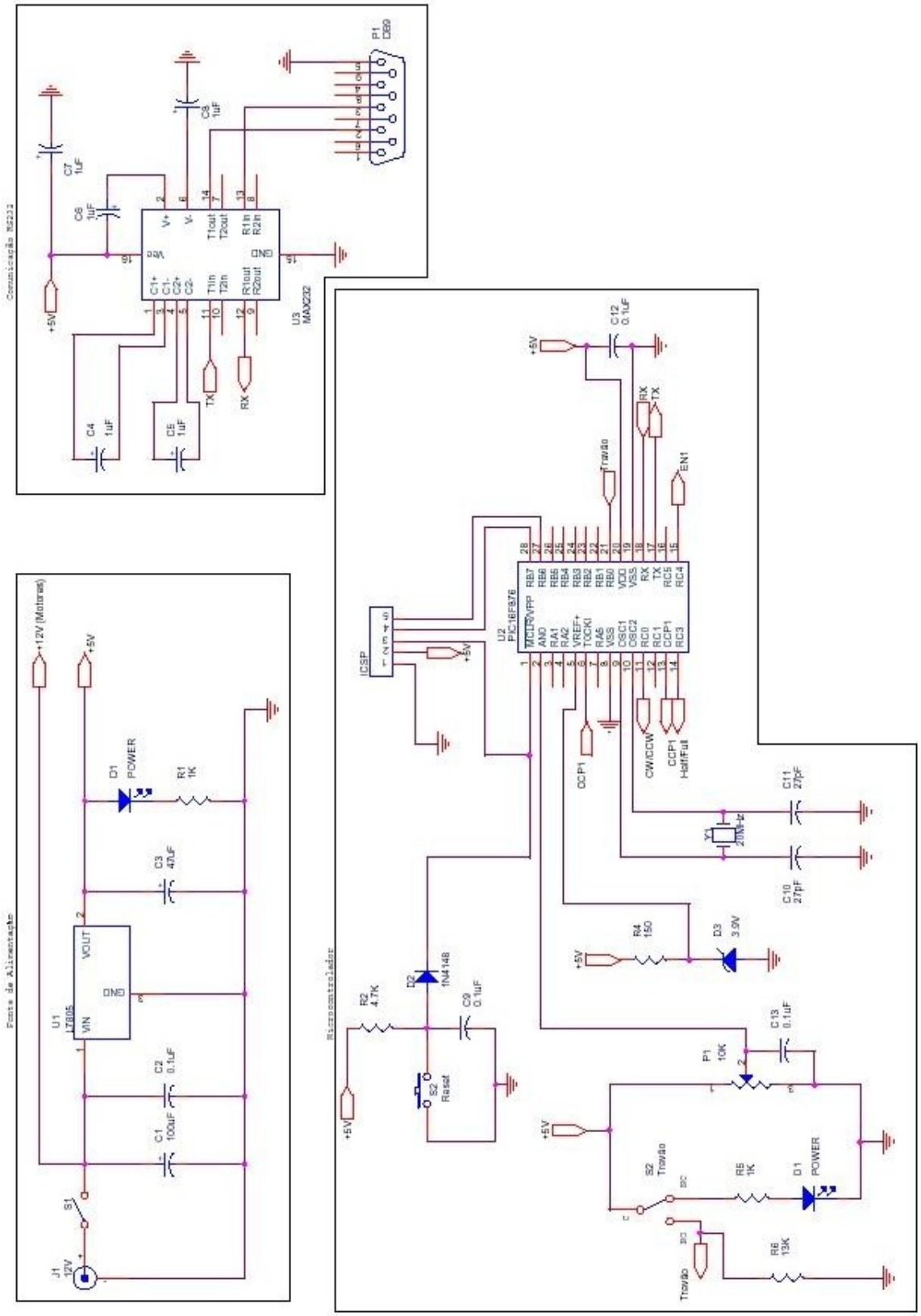
Anexo 4 – Circuito Electrónico do Programador

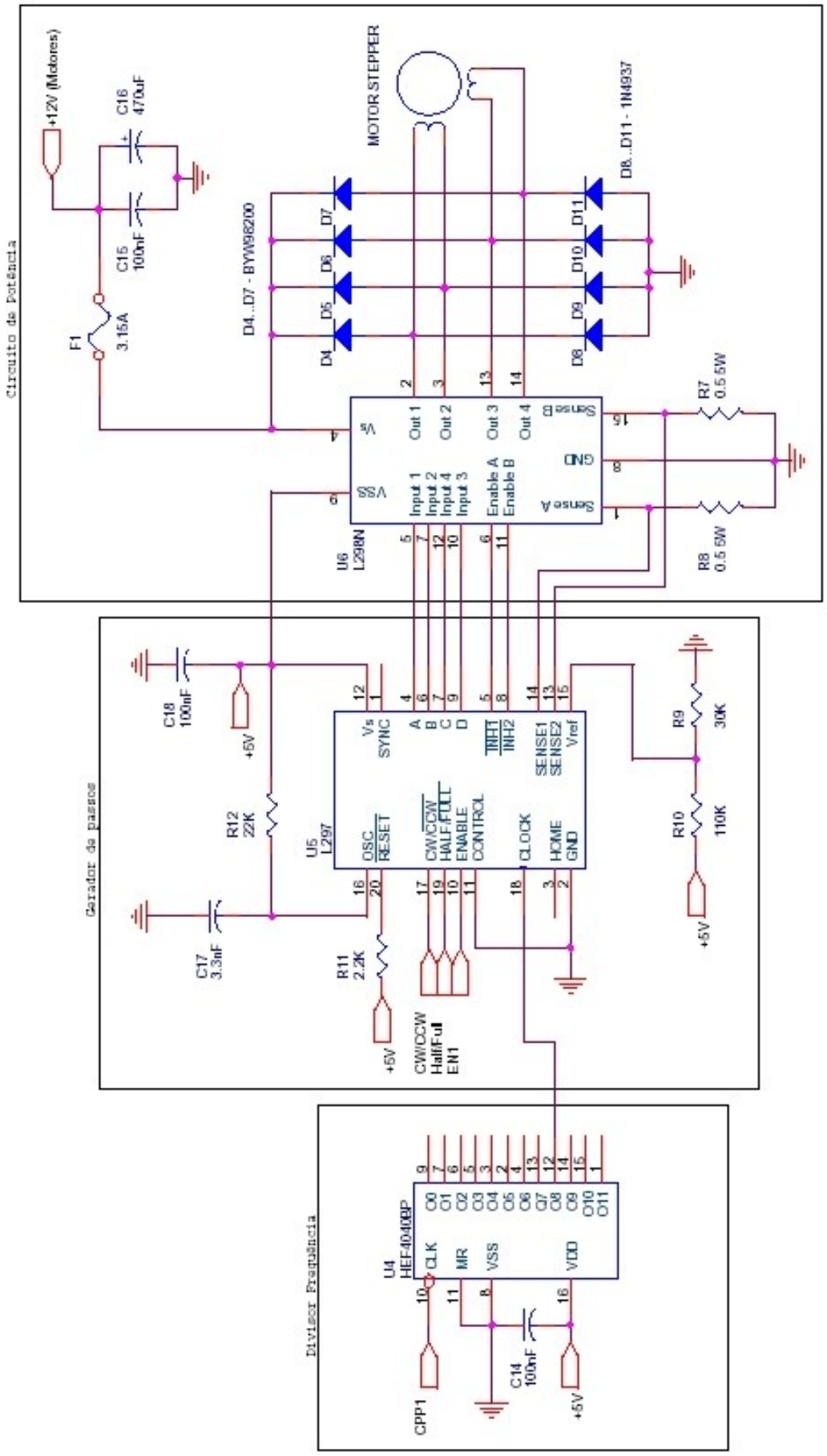






Anexo 5 – Circuito Electrónico da Placa de Controlo

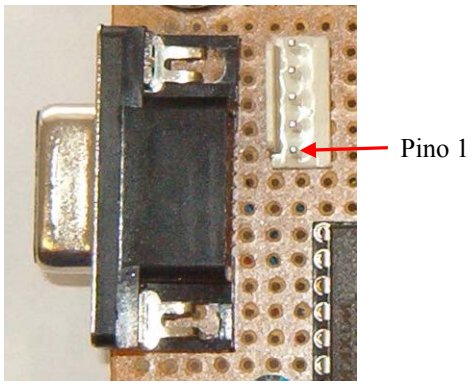



Circuito de Potência
+12V (Motores)
C16
470uF
C15
100nF
F1
3.15A
MOTOR STEPPER
D4...D7 - BYW98200
D8...D11 - 1N4937
D4
D5
D6
D7
D8
D9
D10
D11
U6
L298N
Out 1
Out 2
Out 3
Out 4
Input 1
Input 2
Input 4
Input 3
Enable A
Enable B
Sense A
Sense B
GND
Vs
Vss
R7
0.5 5W
R8
0.5 5W
Gerador de passos
U5
L297
C18
100nF
+5V
R12
22K
C17
3.3nF
R11
2.2K
R9
30K
R10
110K
SYNC
OSC
RESET
CW/CCW
HALF-/FULL
ENABLE
CONTROL
CLOCK
HOME
GND
INH1
INH2
SENSE1
SENSE2
Vref
CW/CCW
HalfFull
EN1
Divisor Frequência
U4
HEF4040BP
CLK
MR
VSS
VDD
C14
100nF
CPP1

Anexo 6 – Cabo de Comunicação PC e Cabo ICSP

Descrição Pinologia

```
Pino 1 - Massa
Pino 2 - +5V
Pino 3 - MCLR
Pino 4 - RB7
Pino 5 - RB6
```



Anexo 7 – Pinologia Ficha do Potenciómetro