Universidade do Minho
Escola de Engenharia

Sílvia Clara Mesquita da Silva Reis

Desenvolvimento de hardware e software para uma máquina protótipo de Encadernação Automática

Dissertação submetida à Universidade do Minho para obtenção do grau de Mestre em Electrónica Industrial e Computadores

Novembro de 2008

Dissertação realizada sob a orientação científica do Professor Doutor António Fernando Macedo Ribeiro, Professor Associado do *Departamento de Electrónica Industrial da Universidade do Minho* e Carlos Duarte da Costa Ventura Responsável pelo *Departamento de Engenharia na Empresa ACCO Brands Portuguesa*.

“A viagem da descoberta consiste não em achar novas paisagens, mas em ver com novos olhos”

Marcel Proust

Agradecimentos

Quero expressar o meu profundo agradecimento, a várias pessoas, que contribuíram, de modo fundamental, com o seu conhecimento e apoio, para o sucesso deste projecto:

- Ao meu orientador, Professor Doutor António Fernando Macedo Ribeiro, agradeço de forma especial, todo o apoio e incentivo para a realização do trabalho. Toda a ajuda na correcção técnica e nas sugestões apresentadas.
- Ao meu orientador na empresa, Carlos Duarte da Costa Ventura, pela sua paciência, conhecimentos práticos e experiencia técnica, que foram fundamentais na implementação do trabalho.
- Ao Jorge Sousa, da empresa *ACCO Brands Portuguesa*, pela sua enorme ajuda na obtenção dos elementos necessários e no acompanhamento que fez de todo o trabalho.
- Às empresas *ACCO Brands Portuguesa* e *ACCO Brands Americana*, por terem confiado em mim para este projecto e pela colaboração que sempre prestaram.
- À empresa *AVEL Electronica*, por toda a colaboração técnica na produção do material necessário ao desenvolvimento do trabalho. Em especial ao Paulo Maia, que colaborou mais de perto no projecto.
- Ao meu pai, pelos seus conhecimentos e experiência profissional, que se tornaram fundamentais, no enriquecimento do trabalho. À minha mãe, pelo apoio na elaboração do texto.
- Ao meu namorado, João Matos, pelo apoio incondicional e ânimo que me transmitiu ao longo deste tempo.
- À professora Clara Mesquita, pela correcção linguística e gramatical de todo o texto.
- Aos meus amigos do NEIEEE e do GAR, pela sua amizade e companheirismo, ao longo de todo o percurso universitário, e em especial aqueles que me ajudaram com a sua opinião ao longo deste trabalho.

Resumo

Não obstante a sua utilização em massa, a encadernação é ainda hoje um processo demasiado lento. Nas grandes empresas e reprografias, onde se realizam diariamente numerosas encadernações, é cada vez mais solicitada a utilização de sistemas automáticos que possibilitem a redução dos tempos e custos de produção.

A *ACCO Brands Corporation*, líder mundial no sector, tem, desde há muito, essa preocupação, procurando, ao longo da sua história, evoluir e inovar nos processos de encadernação.

Seguindo essa necessidade, em 2004, colocou no mercado a argola *clickbind*, que, para além de ser de fácil reutilização, permitiu, mais tarde, de forma não muito complexa, criar uma máquina de encadernação automática, a *GBC Pronto 3000*.

O objectivo seguinte da empresa passou a ser o de conceber uma máquina automática com capacidade para furar até 100 folhas. Surge então o projecto de desenvolvimento da *Hollowgrail* com um método de furação inovador. Com um simples carregar de botão, esta máquina é capaz de furar e encadernar o documento, sendo apenas necessária a colocação da argola adequada para o número de folhas utilizadas.

Por se tratar de uma máquina totalmente automática, são precisos vários motores para realizar a movimentação dos mecanismos. Um motor de passo e cinco motores DC (*direct current*) são capazes de prender folhas, deslocar uma mesa, furar com o auxílio de brocas e fechar a argola. Cada motor necessita de uma placa de controlo comandada por sinais digitais.

A máquina possui ainda uma grande quantidade de sensores, responsáveis por detectar fins de curso ou o estado de alguns mecanismos. Outra característica é o *display* gráfico que faz o interface entre o utilizador e a máquina, o qual para além de dar informação sobre o estado do processo de encadernação também indica quais os passos que o operador deve efectuar e eventuais falhas que possam ocorrer.

Para a realização destas funções, é necessária a utilização de dois microcontroladores, sendo um destinado ao controlo do *display* e o outro à

monitorização dos sensores e controlo de todos os motores. Para a comunicação entre eles, é utilizado o protocolo I^2C (*Inter-Integrated Circuit*).

Abstract

Even though the bookbinding process being extremely common, this is still a very slow process. In large offices and printing companies, where an enormous number of bookbindings is performed every day, it is more and more requested the use of automatic systems which allow time and cost production reductions.

The world leader ACCO Brands Corporation has this concern since long time ago, seeking throughout its history to evolve and innovate in bookbinding processes.

Pursuing this need, in 2004 ACCO placed the clickbind ring on the market, which beyond being easy reusing, allowed later on in a simple way, to create an automatic bookbinding machine, the GBC Pronto 3000.

The company next step was to develop an automatic bookbinding machine with capacity to drill up to 100 sheets. The Hollowgrail development project with an innovative drill method appears then. With a simple touch in a start button, this machine would have to be able to drill and to bind the document, only being necessary to place the adjusted ring for the correct sheets number.

Since it is a fully automatic machine several motors are required to perform all mechanism movements. One step motor and five DC motors are used to hold and drill all sheets, to move a table and to close the ring. Each motor needs a control board controlled by digital signals.

The machine uses also a large amount of sensors, to detect course ends and the state of some mechanisms. Another characteristic is the graphical display that makes the interface between the user and the machine, which beyond giving information about the bookbinding process it also indicates which steps the operator must follow and eventual failures that can occur.

For the accomplishment of these functions two microcontrollers are necessary, being one required to control the display and the other to control the sensors and motors. For the communication between them, the I^2C (*Inter-Integrated Circuit*) protocol is used.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

GBC	*General Binding Corporation*
DC	*Direct current*
I^2C	*Inter-Integrated Circuit*
UM	Universidade do Minho
CC	Corrente continua
MOSFET	*Metal Oxide Semiconductor Field Effect Transístor*
PWM	*Pulse Width Modulation*
PIC	*Peripherical Interface Controller*
NA	Normalmente Aberto
I/O	*Input/Output*
ADC	*Analog to Digital Converter*
ALU	*Arithmetic and Logic Unit*
TCU	*Time and Control Unit*
ROM	*Read Only Memory*
OTP	*One Time Programmable*
RAM	*Random Access Memory*
EEPROM	*Electrically Erasable Programmable ROM*
RISC	*Reduced Instruction Set Computer*
ICSP	*In-Circuit Serial Programming*
QEI	*Quadrature Encoder Interface*
CCP	*Capture/Compare/PWM*
EUSART	*Enhanced Universal Synchronous Asynchronous Receiver Transmitter*
RS-485	*Recommended Standard 485*
RS-232	*Recommended Standard 232*
LIN 1.2	*Local Interconnect Network*
SSP	*Synchronous Serial Port*
SPI	*Serial Peripheral Interface*
CPU	*Central Processing Unit*
MIPS	*Millions of Instructions Per Second*
IDE	*Integrated Development Environment*
ICE	*In-Circuit Emulator*
ICD	*In-Circuit Debugger*
HS	*High-Speed Crystal/Resonator*

CRT	*Cathode Ray Tube*
ELD	*Electroluminescent Display*
LCD	*Liquid Crystal Display*
LED	*Light Emitting Diodes*
OLED	*Organic LED*
PLED	*Polymer LED*
PDP	*Plasma Display Panel*
VFD	*Vacuum Flourescent Displays*
TN	*Twisted Nematic*
HTN	*High Twisted Nematic*
STN	*Super Twisted Nematic*
FSTN	*Film Compensated Super Twisted Nematic*
CSTN	*Color Super Twisted Nematic*
IPS	*In-Plane Switching*
MVA	*Multi-Domain Vertical Alignment*
TFT	*Thin-Film Transistor*
ASCII	*American Standard Code for Information Interchange*
PTMR	*PWM Time Base Registers*
PTCON0	*PWM Timer Control Register 0*
PTCON1	*PWM Timer Control Register 1*
PWMCON1	*PWM Control Register 1*
PTPER	*PWM Time Base Period Registers*
PDCx	*PWM Duty Cycle x Registers*
OVDCOND	*Output Override Control Register*
OVDCONS	*Output State Register*
SCL	*Serial Clock*
SDA	*Serial Data*
ACK	*Acknowledge*
NACK	*Not Acknowledge*
SSP	*Synchronous Serial Port*
MSSP	*Master Synchronous Serial Port*
SSPSTAT	*MSSP Status Register*
SSPCON1	*MSSP Control Register 1*
SSPCON2	*MSSP Control Register 2*
SSPADD	*MSSP Address Register*
PCB	*Printed Circuit Board*
SMT	*Surface-mount technology*

Capítulo 1
Introdução

Este capítulo aborda a história da encadernação e a sua evolução até à actualidade. Justifica as motivações para aceitar este projecto e o desafio que o mesmo representa. Identifica os objectivos do trabalho, bem como as metodologias usadas para atingir esses objectivos. O final do capítulo explica como está organizada toda a dissertação.

1.1 História da encadernação

Desde os primórdios da civilização que o Homem, para comunicar, desenhava nas paredes das suas cavernas e fazia inscrições nas rochas. Desta forma, registou e difundiu os seus conhecimentos e experiências, passando-os às gerações vindouras. Com o passar do tempo, alteraram e variaram o tipo de materiais que suportavam esses registos [1].

"Os Sumérios guardavam as suas informações em tijolo de barro. Os Indianos faziam os livros em folhas de palmeiras. Os Maias e Astecas escreviam os livros num material macio existente entre a casca das árvores e a madeira. Os Romanos escreviam em tábuas de madeira cobertas com cera" [1]. Os Chineses em livros feitos de bambu ou de seda [2].

Figura 1-1: Escrita em tijolo de barro pelos Sumérios (esq), livro indiano (centro), livro chinês (dir) [3][4]

Foram os Egípcios, no ano de 2200 A.C., que desenvolveram a tecnologia do papiro, uma planta encontrada nas margens do rio Nilo. A palavra *papiryrus*, em latim, deu origem à palavra papel. A técnica passava por unir em tiras as fibras dessa planta, formando uma superfície resistente para a escrita [1]. No Egipto, os papiros produzidos eram enrolados para facilitar o seu manuseio e guardados em caixas cilíndricas feitas de couro. Numa segunda fase, os rolos de papiro eram dobrados em páginas em forma de sanfona escritas dos dois lados. Para proteger o papiro das poeiras ou outros estragos, e o obrigar a permanecer dobrado, adaptaram-se placas no extremo das folhas. A costura lateral das dobras, em forma de sanfona, foi o passo seguinte. Esse arranjo foi o início do formato do livro tal como o conhecemos hoje, o codex. Essa primeira encadernação facilitou enormemente a conservação e a leitura dos antigos manuscritos [5].

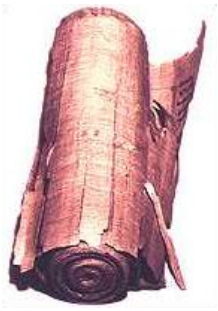


Figura 1-2: Rolo de papiro [6]

Na Grécia antiga, no ano 170 A.C., desenvolveu-se um tipo de "papel" aperfeiçoando o uso de pele de carneiro, o pergaminho. Este era mais resistente e flexível que o papiro, para além de ter a vantagem de poder ser reutilizado. A maior desvantagem era o seu elevado custo de produção e a impossibilidade de ser produzido em grandes folhas [7]. O pergaminho era cortado em folhas e encadernado, quatro folhas costuradas formavam um caderno. Mais tarde, os cadernos foram amarrados juntos, produzindo-se livros de maior espessura que eram recobertos e protegidos com couro, que se estendia até à lombada para proteger as costuras. Assim obtêm-se o livro inteiramente encadernado, tal como o conhecemos hoje [5].

Figura 1-3: Livro de pergaminho [8]

No início do século II, a China desenvolveu o papel a partir do córtex de plantas, tecidos velhos e fragmentos de rede de pesca [7]. Mas só por volta do século IX D.C. esta invenção chegou à Europa trazida pelos Árabes. O papel apresentava sobre o pergaminho a vantagem de um preço inferior e de maiores possibilidades de fabricação. Não o substituiu de imediato, mas revezou-o. Enquanto o pergaminho se destinava aos manuscritos de luxo, o papel servia para os manuscritos mais ordinários e de uso corrente, como os destinados aos estudantes da época [9].

Durante a idade média, a manufactura do livro restringia-se aos mosteiros. Os métodos orientais de encadernação tiveram um grande desenvolvimento no mundo ocidental, onde os monges (os únicos a dominar a leitura e a escrita) os escreviam e encadernavam em larga escala. Naquele tempo, o livro era olhado como algo sagrado e a maioria versava assuntos religiosos. Possuir um livro era indício de distinção [5].

"Com o advento da imprensa, no século XV, e a crescente procura e difusão do livro, tem início uma era brilhante para a encadernação" [10].

Foi Gutenberg o primeiro a combinar os "tipos móveis" com o propósito de imprimir letras e espaços na forma de páginas. Decorreram vários anos até que estes tipos fossem fabricados em larga escala para impressão. Este processo desenvolveu-se enormemente e estendeu-se rapidamente por toda a Europa, onde surgiram estabelecimentos de impressão em todas as principais cidades. A primeira impressão cadastrada, na qual foi empregue o processo dos "tipos móveis", foi uma folhinha impressa por Gutenberg, datada de 1440. O primeiro livro registado foi a *Bíblia de Gutenberg*, completada em 1456 na Alemanha [5].

Não obstante a industrialização da escrita, não existiam máquinas para encadernar livros e esse trabalho continuava a ter que ser inteiramente efectuado à mão, processo muito mais moroso que a impressão, daí a procura de encadernadores suplantar a procura de impressores. A encadernação tornou-se uma arte em que a excelência, originalidade, riqueza e perfeição do trabalho era uma forma de distinção. A obra de vários encadernadores, como Jean Grolier, Thomas Maiolus e Geoffrey Tory, tornou-se famosa e muitos exemplares dos seus trabalhos encontram-se nos maiores museus do mundo [5].

"O século XIX, com todas as transformações decorrentes do avanço da tecnologia, trouxe algumas mudanças cruciais à apresentação do livro. Estes, antes vendidos sem capa e mandados encadernar pelo proprietário, são agora trazidos a público em forma de brochura – com capas de papel, onde a possibilidade de impressão

a cores motivou um desenho gráfico mais elaborado. Aparecem também as encadernações industriais, com revestimento em tecido e ferros padronizados" [10].

Já no século XX, no final da 2ª guerra mundial, Chicago tornou-se num grande centro de empresas especializadas na impressão. Os livros e as revistas eram encadernados para serem distribuídos e, para isso, as empresas recorriam a companhias especializadas. No entanto, na maior parte dos casos, os documentos eram guardados em pastas, presos com agrafos ou clipes, não obedecendo a um processo uniforme. Devido ao alto custo e equipamento complexo usado pelas companhias de encadernação, a maioria das empresas não recorria a este método [11].

Foram Bill Lane e Ed Uihlein que viram a necessidade de uma nova metodologia de encadernação. Estes tinham como alvo as empresas, não só nos Estados Unidos como no resto o mundo. Familiarizados com equipamentos de encadernação comerciais, Bill Lane, o seu irmão John Lane e Uihlein, começaram a desenvolver equipamentos de baixo custo para empresas. Este equipamento era fácil de usar e apresentava uma aparência e qualidade quase profissional, adaptando-se a todos os tipos de documentos [11].

Esta ideia modificou o negócio da encadernação para sempre e, em 1947, a *General Binding Corporation* (GBC) começou a desenhar, produzir e vender produtos e encadernação [11].

1.2 Motivação

O grupo *ACCO Brands Corporation*, maior grupo mundial de equipamentos para escritório, aglutina várias marcas: *Quarter*, *Day-Timer*, *Kensington*, *Swingline*, *Wilson Jones* e *GBC*.

Este grupo pôs a concurso o desenvolvimento de uma máquina de furar para aplicação duma argola *clickbind*, sua patente. Apesar da imensa concorrência entre fábricas instaladas em diversos pontos do mundo, a empresa vencedora foi a *ACCO Brands Portuguesa* que se evidenciou face às outras pela inovação da sua proposta, que incluiu, além da furação, a encadernação automática. O sucesso obtido no relacionamento com alguns departamentos da Universidade do Minho (UM) em anteriores situações, levou a *ACCO Brands Portuguesa* a equacionar a possibilidade de desenvolver o projecto juntamente com um aluno da UM. Assim, foi apresentada ao

autor deste projecto a proposta acima referida, uma proposta tentadora, pois, além de ser um projecto inovador, englobava várias áreas da electrónica, como o controlo de motores DC (*direct current*) e de passo, o controlo de um *display* gráfico e a monitorização de numerosos sensores, com as mais diversas características. O facto de ser um projecto para um produto que tinha como objectivo ser produzido em série e entrar no circuito comercial tornou o desafio ainda mais aliciante.

1.3 Objectivos e Metodologia

Este trabalho tem como principal objectivo a automatização do processo de encadernação. Pretende-se que a máquina fure e encaderne de uma forma automática até 100 folhas. O processo de encadernação deverá ser efectuado em 3 passos: colocação do livro/documento a encadernar; colocação da argola apropriada e ordem de início do processo de furação e encadernação. Desde que se carrega no botão para dar início ao processo de encadernação, até ao final do ciclo com a retirada do livro já encadernado, deverão decorrer 30 a 35 segundos, na versão da máquina para os Estados Unidos, e 40 a 45 segundos, na versão europeia.

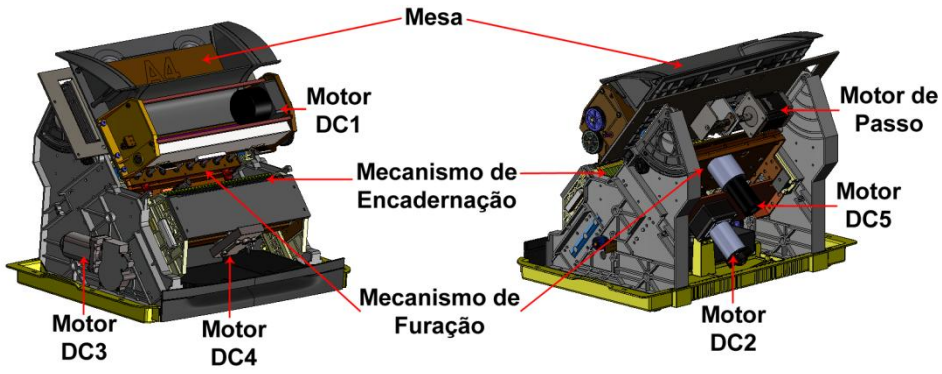


Figura 1-4: Esquema da máquina de encadernar *Hollowgrail*

A Figura 1-4 mostra o esquema da máquina de encadernar. Para realizar a furação, colocam-se primeiramente as folhas a encadernar na mesa, um sensor mecânico detecta a sua presença e a máquina fica pronta para iniciar o trabalho. É medida, então, a espessura do livro, com recurso ao motor DC1 e a dois sensores mecânicos: o motor prende as folhas, enquanto os sensores determinam qual dos três tamanhos de argola deve ser usado. Depois de colocada a argola apropriada e de premir o botão *start*, dá-se início ao processo de furação. É de notar que a porta que dá acesso à colocação da

argola, e que está inicialmente fechada, é aberta com recurso ao motor DC3. Depois de novamente fechada a porta, por intermédio do motor DC3, o motor de passo faz com que a mesa se desloque para a esquerda, até ao ponto onde se dará início à primeira furação.

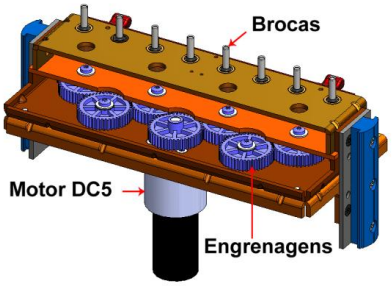


Figura 1-5: Bloco de furação

Nesse momento, o motor DC2 inicia o movimento ascendente do bloco de furação, deslocando-se em direcção às folhas. Ao atingir um nível intermédio, o motor DC5, que realiza a rotação das brocas, entra em funcionamento. Ao atingir as folhas, e depois de totalmente furadas, o motor DC2 inverte o sentido de rotação, deslocando o bloco no sentido descendente, até ao nível intermédio. Aí o motor DC5 pára e é efectuado um novo deslocamento da mesa, agora para a direita, com um espaçamento de 8.466mm, para se proceder a uma nova furação, através do movimento ascendente e descendente do bloco de furação. O processo repete-se até um total de 4 vezes, na versão da máquina para os Estados Unidos, e 6 vezes na versão europeia.

No final de todas as furações, a mesa desloca-se novamente para o centro onde se dá início ao processo de encadernação. Primeiro, o bloco de encadernação, comandado pelo motor DC3, desloca-se em direcção às folhas e, consoante a espessura do livro a encadernar, pára numa de três posições. Atingido o fim de curso, o motor DC4 faz deslocar o mecanismo de fecho do *clickbind*, até a argola fechar, afastando-se, de seguida, para a sua posição inicial, tal como todos os outros mecanismos. Nesta altura, é possível retirar o livro e a máquina está pronta para uma nova encadernação.

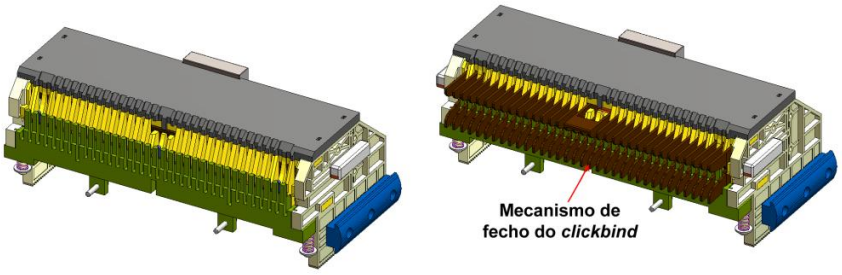


Figura 1-6: Bloco de encadernação

1.4 Organização da dissertação

A presente tese está estruturada em 6 capítulos.

O primeiro capítulo inicia com uma introdução ao tema, apresenta uma breve visão histórica sobre a evolução desta arte e fundamenta a motivação para a escolha deste trabalho em particular.

O segundo capítulo aborda a análise dos diversos elementos de encadernação e de algumas máquinas existentes, comparando-as com a máquina em desenvolvimento.

O terceiro capítulo foca o estudo dos motores, nomeadamente os motores DC e os motores de passo, aprofundando também a forma de os controlar. Este enfoque revelou-se de grande importância na escolha dos métodos de controlo adequados.

O capítulo quarto é dedicado aos microcontroladores e à sua programação. Inicialmente, realiza-se uma abordagem teórica aos microcontroladores PIC (*Peripherical Interface Controller*) e às suas ferramentas de desenvolvimento. Segue-se um estudo de *displays*, sensores, comunicação de dispositivos por I^2C (*Inter-Integrated Circuit*), desenvolve-se a forma como se realizou o seu controlo, terminando o capítulo com uma explicação detalhada sobre o funcionamento da máquina.

O quinto capítulo pretende mostrar um pouco do trabalho prático efectuado e discutir o seu desenvolvimento.

Finalmente, no capítulo 6, são tiradas conclusões, expostas as principais dificuldades e enumeradas sugestões para trabalho futuro.

Capítulo 2

Estado da arte

Neste capítulo pretende dar-se a conhecer as diferentes máquinas de encadernar que existem no mercado, bem como as suas características. Faz-se uma breve análise comparativa entre as máquinas abordadas e a máquina em desenvolvimento (*Hollowgrail*).

As características que geralmente diferenciam as diversas máquinas de encadernação são as seguintes:

- elementos de encadernação que utilizam (*stripbind*, *wirebind*, *colorcoil*, *zipbind*, *combbind* e *clickbind*) (Figura 2-1);
- função que executam (furação, encadernação ou ambas);
- forma como a executam (automática ou manual);
- passos necessários para a executar;
- frequência de execução.

Figura 2-1

2.1 Stripbind

O *stripbind* é um dos elementos de encadernação patenteado pela *General Binding Corporation* (GBC). O seu formato em forma de pente concede aos livros um aspecto profissional e elegante. Este modelo é ideal para o transporte e arquivo de livros, uma vez que permite que estes não se danifiquem se algo pesado lhes for colocado em cima. Quando abertos em cima de uma mesa não tomam uma forma lisa, o que os torna mais difíceis de fotocopiar, podendo no entanto ser lidos facilmente [12].

Figura 2-2: Exemplo de encadernação com o elemento *stripbind* [13]

Este tipo de encadernação é utilizado para livros de capa flexível e capa dura, permitindo encadernar espessuras muito maiores do que qualquer outro tipo de argola. Com um pequeno número de formatos em termos de tamanho, podem encadernar-se livros contendo até 750 folhas [12], [14]. Neste modelo existem diversas variantes que permitem escolher entre uma encadernação definitiva, que só pode ser alterada cortando o elemento de encadernação, ou uma encadernação que pode ser modificada. O facto de permitir encadernações definitivas, faz deste modelo uma encadernação segura [14]. O conjunto destas características faz com que esta opção seja muito usada em relatórios, apresentações, propostas e contratos de escritórios de advocacia, agências governamentais, universidades e gabinetes financeiros.

É formado basicamente por duas tiras plásticas, uma colocada na parte superior e outra na parte inferior das folhas (Figura 2-3), prendendo-as desta forma. A tira plástica de cima possui uns "dentes" que atravessam os furos das folhas e os encaixes da tira de baixo.

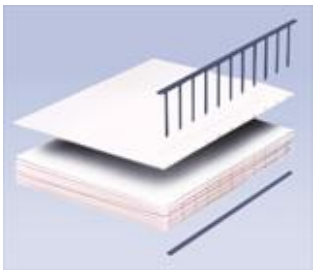


Figura 2-3: Forma de encadernar um livro recorrendo ao *stripbind* [15]

O elemento de encadernação *stripbind* pode ser dividido em dois tipos: o *velobind* e o *surebind*. O produto final resultante da encadernação com qualquer uma destas soluções é idêntico, ainda que existam algumas diferenças a nível mecânico [16].

Actualmente, existem dois tipos de *velobind*: as *hot knife strips* de 11 "dentes" (apenas existentes nos Estados Unidos e Canadá) e as *4 pin recloseable strips* (Figura 2-4). As encadernações com as *velobind hot knife strips* usam o calor para cravar os

dentes da tira plástica de cima nos furos da tira plástica de baixo. Isto promove uma encadernação permanente e segura, com um aspecto profissional. As encadernações com os *velobind 4 pin recloseable strips* podem ser modificadas, dado que os pinos, em vez de serem cravados, são dobrados para dentro da tira inferior, tornando fácil a alteração de livros e documentos. Estes elementos de encadernação proporcionam aos livros um aspecto idêntico ao obtido com os *velobind hot knife strips*, não necessitando, no entanto, de adquirir uma máquina bastante mais dispendiosa, como a utilizada com a *velobind hot knife strips* [17].

Figura 2-4: Elementos de encadernação *velobind hot knife strips* e *velobind 4 pin recloseable strips* [18]

Os elementos *surebind* possuem 10 "dentes" e, tal como os *velobind hot knife strips*, usam o calor para cravar os dentes da tira plástica superior nos furos da tira inferior. Comparando as tiras *velobind* e *surebind*, verificamos que as primeiras são ligeiramente mais estreitas e têm os furos uniformemente espaçados, ao contrário das segundas. O sistema *surebind* é compatível com o *combbind*, o que permite furar o papel recorrendo a uma máquina deste tipo. O facto do sistema *combbind* efectuar um número de furos superior aos utilizados no sistema *surebind*, não tem qualquer inconveniente dado que os furos em excesso são tapados pelas tiras plásticas [19].

Para cada tipo de *stripbind* definido até então, existem máquinas capazes de realizar a encadernação dos livros de uma forma manual ou automática. Na Tabela 2-1 podemos ver alguns dos modelos possíveis de adquirir na Europa.

Empresas	**Máquinas**	**Furos**	**Furação**		**Encadernação**		**Utilização**
			Tipo	**Capacidade** (nº folhas 80gr)	**Tipo**	**Capacidade** (nº folhas 80gr)	
ACCO	*GBC Desktop Velobinder*	4	Manual	20	Manual	200	Doméstica/pequenos escritórios
	GBC SureBind 500	10	Manual	25	Eléctrica	500	Pequenos e médios escritórios
	GBC SureBind System 1	10	Manual	22	Eléctrica	200	Pequenos e médios escritórios
	GBC SureBind System 2	10	Eléctrica	22	Eléctrica	500	Médios e grandes escritórios
	GBC SureBind System 3 Pro	10	Eléctrica	26	Eléctrica	750	Grandes escritórios e reprografias
	GBC SureBind System 4	10	-	-	Eléctrica	750	Grandes escritórios e reprografias

Tabela 2-1: Algumas máquinas de encadernar *stripbind* [20][21]

Figura 2-5: Máquina de encadernar *GBC Desktop Velobinder* (esq), *GBC SureBind 500* (centro) e *GBC SureBind System 2* (dir) [18]

2.2 Wirebind

As máquinas que usam como elementos de encadernação as argolas *wirebind* (Figura 2-6), também conhecidas por *twin loop wire*, *double loop wire*, *wire-o* ou ainda *double wire*, são a escolha de muitos arquitectos, organismos públicos, contabilistas e companhias financeiras, devido, principalmente, ao aspecto elegante e profissional que estas argolas concedem às suas encadernações [22].

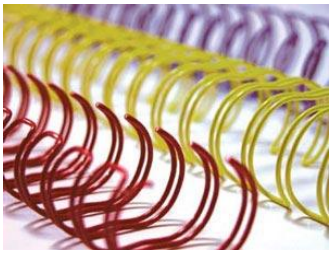


Figura 2-6: Argola *wirebind* [23]

Os elementos de encadernação *wirebind* estão acessíveis em dois formatos, um com passo 3:1, que é usado em pequenos documentos contendo entre 20 e 110 folhas, e outro com passo 2:1, para grandes documentos, com um número de folhas compreendido entre as 110 e as 250. Este último pode também ser utilizado em documentos com menos folhas [24].

A forma e o material deste tipo de argolas permitem uma rotação de 360º às encadernações. Esta característica torna este formato popular na encadernação de calendários e blocos de notas. Outra vantagem deste formato é o facto de não sofrer grandes efeitos quando expostos a temperaturas muito baixas ou muito altas, dado utilizar argolas de aço [25]. É necessário, no entanto, algum cuidado no seu manuseamento para evitar que as argolas se deformem.

A encadernação com estas máquinas é efectuada colocando as folhas, previamente furadas, na argola que tem, inicialmente, uma forma de C e, após pressionada, torna-se num elemento circular [24] (Figura 2-7).

Figura 2-7: Argola *wirebind* antes e depois da encadernação [26]

Existe um grande número de máquinas *wirebind* no mercado e são várias as empresas que as produzem. Na Tabela 2-2, faz-se uma análise a alguns dos produtos da *ACCO*, *Esselte/Leiz*, *Fellowes*, *Peach* e *Renz*.

Empresas	**Máquinas**	**Passo**	**Furação**		**Encadernação**		**Utilização**
			Tipo	**Capacidade** (nº folhas 80gr)	**Tipo**	**Capacidade** (nº folhas 80gr)	
ACCO	*GBC W15*	3:1	Manual	15	Manual	125	Doméstica/pequenos escritórios
	GBC W20	3:1	Manual	20	Manual	125	Pequenos e médios escritórios
	GBC W25E	3:1	Eléctrica	25	Manual	125	Médios e grandes escritórios
	Ibico HiTechKaro	3:1	Manual	12	Manual	125	Doméstica/pequenos escritórios
	GBC MC12	3:1 2:1	-	-	Manual	3:1 – 125 2:1 – 250	Médios e grandes escritórios
	GBC TL2900	3:1	-	-	Eléctrica	250	Grandes escritórios e reprografias
	MP2000W3	3:1	Eléctrica	16	-	-	Grandes escritórios e reprografias
Esselte	*wireBIND 500e*	3:1 2:1	Eléctrica	3:1 – 20 2:1 – 25	Manual	270	Médios e grandes escritórios
Fellowes	*Quasar*	3:1	Manual	15	Manual	130	Pequenos e médios escritórios
Peach	*Personal*	3:1	-	-	Manual	120	Doméstica/pequenos escritórios
Renz	*Eco C*	2:1	Manual	30	Manual	280	Médios e grandes escritórios
	Eco Comfort	2:1	Eléctrica	20	Manual	280	Médios e grandes escritórios
	Eco S 360	2:1	Manual	30	Manual	340	Grandes escritórios e reprografias
	Eco 360 comfort	2:1	Eléctrica	30	Manual	340	Grandes escritórios e reprografias
	SRW 360	3:1	Manual	25	Manual	120	Médios e grandes escritórios
	SRW 360 comfort	3:1	Eléctrica	30	Manual	120	Médios e grandes escritórios
	SRW comfort	3:1	Eléctrica	20	Manual	120	Médios e grandes escritórios
	WBS 340	2:1 3:1	-	-	Manual	280	Grandes escritórios e reprografias
	ECL 360	2:1 3:1	-	-	Eléctrica	280	Grandes escritórios e reprografias

Tabela 2-2: Algumas máquinas de encadernar *wirebind* [21][27][28][29][30][31][32]

Como se verifica, grande parte das máquinas permite furar e encadernar, podendo a furação ser manual ou automática, mas a encadernação é sempre manual. Em locais onde se produzem muitos livros por dia, são necessárias, normalmente, duas máquinas, uma para furação e outra para encadernação.

Figura 2-8: Máquina de encadernar *GBC W20* (esq), *GBC TL2900* (centro) e *Renz SRW 360 comfort* (dir) [18][21][32]

2.3 Colorcoil

As máquinas de encadernar que usam argolas *colorcoil* como elementos de encadernação são muito usadas por designers e nas áreas do marketing e publicidade, devido, principalmente, ao facto de existir uma enorme variedade de cores para este tipo de argola [33] (Figura 2-9). É desta característica que deriva o nome dado a este elemento, também conhecido por *spiral coil* ou *plastic coil*.

Figura 2-9: Argola *colorcoil* [18]

Os elementos de encadernação *colorcoil* estão acessíveis em três formatos, sendo o mais comum o que possui um passo de 4:1, utilizado em documentos entre 30 e 450 folhas. Deve-se, no entanto, ter em conta que documentos com mais de 200 folhas tornam-se difíceis de manusear. Argolas com passo 5:1 são também muito usadas, mas o facto de os documentos encadernados com estas argolas possuírem os furos muito próximos, faz com que mais facilmente as folhas se rasguem. Por fim, existem as argolas com um passo de 3:1, que são compatíveis com o formato 3:1 das *wirebind* e

das *clickbind*. Estas argolas são mais fáceis de inserir nas folhas já furadas, uma vez que possuem menos furos [34].

Documentos encadernados com estes elementos são capazes de rodar 360º, o que torna esta encadernação ideal em blocos de notas e calendários. As argolas não se deformam nem partem se forem dobradas ou pressionadas, tornando esta encadernação ideal em documentos muito utilizados ou que necessitem ser enviados por via postal [35]. A exposição a temperaturas elevadas é de evitar, dado que pode originar a sua deformação [33].

Figura 2-10: Livro encadernado com *colorcoil* [36]

Para encadernar insere-se manualmente a argola nos primeiros furos das folhas previamente perfuradas. De seguida, faz-se rodar a argola manualmente ou por intermédio de uma máquina apropriada, até que todos os furos sejam preenchidos. Para finalizar a encadernação, dobram-se as pontas da argola para que esta não saia do livro [37].

Entre as empresas analisadas, verificou-se que apenas a *ACCO* e a *Renz* produzem máquinas *colorcoil* que podem ser vistas na Tabela 2-3.

Empresas	**Máquinas**	**Passo**	**Furação**		**Encadernação**		**Utilização**
			Tipo	**Capacidade** (nº folhas 80gr)	**Tipo**	**Capacidade** (nº folhas 80gr)	
ACCO	*GBC C100*	4:1	Manual	12	Manual	145	Doméstica/pequenos escritórios
	GBC C200	4:1	Manual	20	Eléctrica	145	Pequenos e médios escritórios
	GBC MC10	5:1	Manual	10	Manual	450	Doméstica/pequenos escritórios
	GBC P59	5:1	Manual	15	Manual	450	Médios e grandes escritórios
	GBC EP59	5:1	Eléctrica	15	Manual	450	Médios e grandes escritórios
	GBC CC2700	4:1	-	-	Eléctrica	265	Grandes escritórios e reprografias
	GBC CI12	4:1	-	-	Eléctrica	265	Grandes escritórios e reprografias
	GBC MP2000 C4	4:1	Eléctrica	14	-	-	Grandes escritórios e reprografias
Renz	*SPB 360*	4:1	Manual	20	Eléctrica	170	Médios e grandes escritórios
	SPB 360 comfort	4:1	Eléctrica	20	Eléctrica	170	Médios e grandes escritórios
	CBS 340	4:1	-	-	Eléctrica	170	Grandes escritórios e reprografias

Tabela 2-3: Algumas máquinas de encadernar *colorcoil* [21][28][38][39]

Verifica-se que também aqui existem máquinas que permitem furar e encadernar e outras que apenas executam uma das operações. As que realizam apenas uma das funções são normalmente eléctricas, habitualmente usadas em conjunto com uma máquina complementar, em locais onde, diariamente, se realizam um elevado número de encadernações. Constata-se também que em modelos com um passo 4:1 a encadernação é maioritariamente eléctrica e que nos modelos com um passo 5:1 é sempre manual. Isso deve-se ao facto deste modelo exigir furos mais pequenos e mais próximos, fazendo com que haja uma probabilidade maior das folhas se rasgarem durante a encadernação. Estas últimas foram concebidas especialmente para o mercado ibérico que, sendo considerado um mercado de pequena dimensão, não justificava grandes investimentos no desenvolvimento de uma nova máquina eléctrica. Optou-se pela utilização de máquinas já existentes com o passo 4:1 convertidas para 5:1.

Figura 2-11: Máquina de encadernar *GBC MC10* (esq), *Renz SPB 360 comfort* (centro) e *GBC CI12* (dir) [18][21][39]

2.4 Combbind

A encadernação, recorrendo às argolas *combbind*, é um dos métodos de encadernação mais populares. Sendo o mais antigo método de encadernação existente no mercado, é mais económico que qualquer dos outros métodos de encadernação aqui descritos e possui uma maior diversidade de cores e tamanhos [40].

Também conhecidos por *Plastic Comb*, *Cerlox*, *Ibico ou GBC Binding*, estes elementos de encadernação são muito utilizados por escolas e empresas [41]. A possibilidade de permitir a impressão do logótipo, nome da empresa, ou título do relatório faz deste método de encadernação a escolha preferida em apresentações, livros e manuais [42].

Figura 2-12: Argola *combbind* [18]

As argolas *combbind* estão disponíveis em dois formatos, redondas e ovais (Figura 2-12). As primeiras são as mais populares e são capazes de encadernar de 10 a 225 folhas, não sendo, no entanto, recomendável encadernar menos de 25 folhas, pois para essas quantidades torna-se difícil o manuseamento do livro. As argolas ovais são usadas para encadernar documentos com um elevado número de folhas, podendo ultrapassar as 400 [41].

Documentos encadernados com estes elementos são capazes de uma abertura acima dos 180º, o que os torna ideais para ler e fotocopiar [40]. A exposição a temperaturas elevadas é de evitar dado que pode originar a sua deformação.

Figura 2-13: Livro encadernado com *combbind* [18]

Para encadernar, inserem-se as folhas, previamente furadas, na argola aberta com a ajuda da máquina de encadernar, procedendo de seguida ao fecho da argola. Podem ser facilmente inseridas ou removidas folhas do livro com recurso a uma máquina [41].

Todas as empresas analisadas produzem estas máquinas, e são vários os modelos existentes no mercado, estando apresentados na Tabela 2-4 apenas alguns deles.

Empresas	Máquinas	Passo	Furação		Encadernação		Utilização
			Tipo	Capacidade (nº folhas 80gr)	Tipo	Capacidade (nº folhas 80gr)	
ACCO	*GBC BindMate*	2:1	Manual	8	Manual	125	Doméstica/pequenos escritórios
	GBC P50	2:1	Manual	6	Manual	95	Doméstica/pequenos escritórios
	GBC C95	2:1	Manual	15	Manual	165	Doméstica/pequenos escritórios
	GBC C95E	2:1	Eléctrica	15	Manual	165	Doméstica/pequenos escritórios
	GBC C110	2:1	Manual	15	Manual	330	Doméstica/pequenos escritórios
	GBC C110e	2:1	Eléctrica	15	Manual	330	Pequenos e médios escritórios
	GBC C366	2:1	Manual	30	Manual	450	Médios e grandes escritórios
	GBC C366E	2:1	Eléctrica	30	Manual	450	Médios e grandes escritórios
	GBC800Pro	2:1	Eléctrica	20	Manual	450	Grandes escritórios e reprografias
	GBC PB2600	2:1	-	-	Eléctrica	425	Grandes escritórios e reprografias
	GBC 16DB	2:1	-	-	Manual	425	Grandes escritórios e reprografias
	GBC MP2000 PB	2:1	Eléctrica	20	-	-	Grandes escritórios e reprografias
Esselte	*comBIND 100*	2:1	Manual	8	Manual	145	Doméstica/pequenos escritórios
	comBIND 300	2:1	Manual	15	Manual	145	Doméstica/pequenos escritórios
	comBIND 500	2:1	Manual	25	Manual	500	Médios e grandes escritórios
	comBIND 500e	2:1	Eléctrica	28	Manual	500	Médios e grandes escritórios
Fellowes	*Starlet*	2:1	Manual	7	Manual	90	Doméstica/pequenos escritórios
	Galaxy	2:1	Manual	25	Manual	500	Médios e grandes escritórios
	Galaxy E	2:1	Eléctrica	25	Manual	500	Médios e grandes escritórios
	Pulsar 300	2:1	Manual	15	Manual	300	Pequenos e médios escritórios
	Pulsar E 300	2:1	Eléctrica	15	Manual	300	Pequenos e médios escritórios
Peach	*Professional Electric EC24*	2:1	Eléctrica	20	Manual	500	Médios e grandes escritórios
	Desktop C21	2:1	Manual	15	Manual	500	Pequenos e médios escritórios
	Personal	2:1	Manual	4	Manual	95	Doméstica/pequenos escritórios
Renz	*PBS 340*	2:1	-	-	Manual	500	Médios e grandes escritórios
	private bind	2:1	Manual	10	Manual	150	Doméstica/pequenos escritórios
	combinette combi V	2:1	Manual	25	Manual	500	Médios e grandes escritórios
	combi E	2:1	Eléctrica	25	Manual	500	Grandes escritórios e reprografias

Tabela 2-4: Algumas máquinas de encadernar *combbind* [21][28][31][43][44][45][46]

Também aqui existem máquinas que permitem a furação e a encadernação em conjunto. Embora a furação possa ser manual ou automática, a encadernação é quase sempre manual.

Figura 2-14: Máquina de encadernar *Fellowes Galaxy E* (esq), *GBC 800pro* (centro) e *Peach personal* (dir) [18][21][39]

2.5 Zipbind

O mais recente modo de encadernação é o que utiliza as argolas *zipbind* patenteadas pela *GBC*. É um tipo de encadernação que concede aos livros um aspecto elegante, resultado do novo visual destas argolas (Figura 2-15). O facto de ser compatível com qualquer sistema de encadernação *combbind*, torna-o muito utilizado por empresas que possuem um desses sistemas, mas pretendem uma aparência diferente para as suas encadernações [47].

Figura 2-15: Argola *zipbind* [48]

Este modo de encadernação é manual e está disponível em apenas dois tamanhos (para encadernar até 55 ou 85 folhas) e duas cores (preta e incolor) [49].

Para encadernar, vinca-se ligeiramente a argola, para que seja mais fácil o seu fecho, inserindo, de seguida, as folhas previamente furadas e fechando os primeiros anéis manualmente. Finalmente recorre-se ao *zipper* para fechar os restantes anéis [50].

Figura 2-16: Livro encadernado com *zipbind* [51][52]

Este modo de encadernação tem como característica, com o auxílio do *zipper*, permitir uma fácil remoção ou inserção de novas folhas. Esta característica faz deste tipo de encadernação um modelo muito utilizado em manuais que necessitem de actualizações frequentes. Outra característica destas argolas é permitir uma rotação de 360º aos seus documentos, tornando o seu uso frequente em blocos de notas ou documentos que necessitem ser fotocopiados [47].

2.6 Clickbind

Em 2004, a *GBC* introduziu no mercado um novo sistema de encadernação, o *clickbind*, também conhecido por *proclick*. Este elemento de encadernação permite que os seus documentos possam ser abertos e fechados em segundos, recorrendo, para isso, a um *zipper* como o já utilizado nas argolas *zipbind*. Esta capacidade, aliada ao seu estilo moderno, fazem desta argola ideal para muitos documentos de vendas e marketing [53].

Estas argolas possuem um passo 3:1, o que as torna compatíveis com as *wirebind* e as *colorcoil* possuidoras do mesmo passo. Desta forma, torna-se possível o uso de uma máquina *wirebind* ou *colorcoil*, para realizar a furação, sendo a encadernação efectuada com o recurso do *zipper* ou de uma máquina encadernadora. As argolas *clickbind* estão disponíveis em 4 diferentes cores (preto, branco, azul e incolor) e 3 tamanhos (45, 85 e 145 folhas). Outra vantagem destas argolas é a de permitir uma rotação de 360º aos seus documentos, facilitando assim a sua leitura e cópia [53].

Figura 2-17: Argolas *clickbind* [18]

A Tabela 2-5 mostra as máquinas actualmente no mercado para encadernar com *clickbind*.

Empresas	Máquinas	Passo	Furação		Encadernação		Utilização
			Tipo	Capacidade (nº folhas 80gr)	Tipo	Capacidade (nº folhas 80gr)	
ACCO	*GBC Clickman*	3:1	Manual	6	Manual	145	Doméstica/pequenos escritórios
	GBC C15	3:1	Manual	15	Manual	145	Médios e grandes escritórios
	GBC C15E	3:1	Eléctrica	15	Manual	145	Médios e grandes escritórios
	GBC EzClick	3:1	-	-	Manual	100	Pequenos e médios escritórios
	GBC Pronto 2000	3:1	-	-	Eléctrica	100	Grandes escritórios e reprografias
	GBC Pronto 3000	3:1	Eléctrica	20	Eléctrica	100	Grandes escritórios e reprografias

Tabela 2-5: Máquinas de encadernar *clickbind* [21][54]

Como se verifica, podem ser usadas máquinas para furar as folhas que depois são encadernadas com um *zipper* ou com uma máquina encadernadora, existindo ainda uma máquina capaz de furar e encadernar automaticamente.

Figura 2-18: Máquina de encadernar *GBC C15* (esq), *GBC EzClick* (centro) e *GBC Pronto 3000* (dir) [18][54]

A *Hollowgrail* está a ser desenvolvida para usar, como elemento de encadernação, a argola *clickbind* aqui descrita. Tem, portanto, um passo de 3:1 e capacidade para encadernar 100 folhas, tal como todas as máquinas eléctricas actualmente existentes no mercado para este elemento. Pretende-se que tanto a furação como a encadernação sejam eléctricas e prevê-se que venha a ter uma boa aceitação por parte dos grandes escritórios e reprografias.

Actualmente, no mercado, existe a *GBC Pronto 3000*, que, como se vê na Tabela 2-5, garante todas as características referidas até agora na *Hollowgrail*, com a limitação de não ser capaz de furar mais de 20 folhas. Isto é uma grande desvantagem, pois, para realizar uma encadernação de 100 folhas, é necessário proceder a 5 furações. A realização de mais passos implica despender muito mais tempo para efectuar a

encadernação. A *Hollowgrail* implementa um sistema de furação inovador, que utiliza brocas em vez de lâminas de corte, permitindo desta forma furar 100 folhas de uma só vez. Esta característica permite-lhe realizar uma encadernação com um número muito reduzido de passos e, consequentemente, num tempo substancialmente mais curto. Um dos objectivos é que esse valor ronde entre os 30 e os 40 segundos, dependendo da versão da máquina.

Outra das grandes desvantagens da *GBC Pronto 3000* reside no facto de não existir um tabuleiro onde colocar as folhas durante o processo de furação, exigindo que o operador esteja sempre presente. A *Hollowgrail* permite a eliminação de erros e uma menor ocupação do operador, por não ser necessária a sua intervenção durante a furação, de esta ser realizada de uma só vez e por não ser igualmente necessária a sua intervenção durante o processo de encadernação.

2.7 Conclusões

Comparando as máquinas aqui descritas com a *Hollowgrail*, verifica-se que, no que se refere à furação, esta é possuidora de uma capacidade muito superior, resultado do processo inovador utilizado. Todas as máquinas utilizadoras do sistema *clickbind*, ou de qualquer outro sistema aqui descrito, não são capazes de furar mais de 30 folhas, o que faz com que para encadernar mais folhas seja necessário proceder a várias furações.

Comparando o tempo e o número de operações necessárias para efectuar uma encadernação, utilizando as máquinas actualmente existentes no mercado ou a *Hollowgrail*, verifica-se que esses valores são muito mais reduzidos com a máquina agora em desenvolvimento.

Todas estas vantagens da *Hollowgrail*, em relação aos modelos existentes no mercado, levaram a que o seu desenvolvimento fosse necessário.

Capítulo 3

Motores eléctricos e seu controlo

Os motores eléctricos são, dentro do universo das máquinas eléctricas, os que têm capacidade para converter energia eléctrica em mecânica. A transformação de energia que ocorre é feita através da acção de um campo magnético.

"É graças ao trabalho de M. Faraday, J. C. Maxwell, J. Henry, P. Barlow, B. Jacobi, E. Lenz e H. Oersted que o magnetismo e a electricidade podem ser examinados como um processo único, que é a essência e o fundamento das máquinas eléctricas" [55].

Neste capítulo, abordam-se os motores usados na *Hollowgrail*, bem como o hardware utilizado no seu controlo.

Inicia-se o capítulo, com uma breve descrição dos motores DC (*direct current*), fazendo referência ao seu princípio de funcionamento, às suas principais características e aos motores utilizados na máquina em desenvolvimento.

De seguida, analisa-se o método de controlo usado, com o objectivo de obter o resultado desejado, quer no sentido de rotação, quer na velocidade de funcionamento.

Segue-se uma abordagem aos motores de passo, onde são discutidos os tipos existentes, os princípios de funcionamento, características, modos de alimentação e de accionamento; sendo igualmente abordado o modelo seleccionado para este projecto.

Termina-se este capítulo, com a análise dos diferentes modos de controlo dos motores de passo.

3.1 Motores DC

O motor DC, também conhecido como motor CC (corrente contínua), é formado por estator, rotor, colector e escovas (Figura 3-1).

Figura 3-1: Estator (esq); rotor e colector (centro); escovas (dir)

O estator é a parte do motor que envolve o rotor. É imóvel e construído de forma a permitir que o rotor gire no seu interior. No estator, produz-se um campo magnético fixo, que tem como função interagir com um campo magnético criado no rotor e, dessa forma, fazer com que ele gire. Esse campo magnético pode ser criado por ímanes ou por electroímanes. No caso dos electroímanes, o estator é formado por pólos de material ferromagnético envolvidos por um enrolamento (enrolamento de campo) que é alimentado por uma fonte de tensão contínua [56].

O rotor, também conhecido por armadura, é, como já foi referido, a parte móvel do motor. É formado por um material ferromagnético envolto num enrolamento (enrolamento de armadura) [56].

O colector, formado por várias lâminas condutoras isoladas, encontra-se no eixo do rotor e a ele estão ligados os terminais dos enrolamentos de armadura, pelo que, à medida que o rotor gira, o colector também gira [57].

Através das escovas, feitas em carbono e que fazem um contacto permanente com o colector, é conduzida a corrente para o enrolamento de armadura [58].

Em relação aos motores, cujo campo magnético do estator é produzido por electroímanes, eles diferem na forma como se faz a ligação do enrolamento de campo com o enrolamento da armadura. Esta ligação pode ser em paralelo ou em série, denominando-se motor DC *shunt* e motor DC série, respectivamente. Há motores que possuem dois enrolamentos no estator, um do tipo *shunt* e outro do tipo série, incorporando assim as características das duas máquinas simultaneamente - os denominados motores DC *compound*. Existem ainda os motores em que o enrolamento

de campo é alimentado por uma fonte de tensão independente da usada no enrolamento de armadura, sendo denominados motores DC de excitação independente.

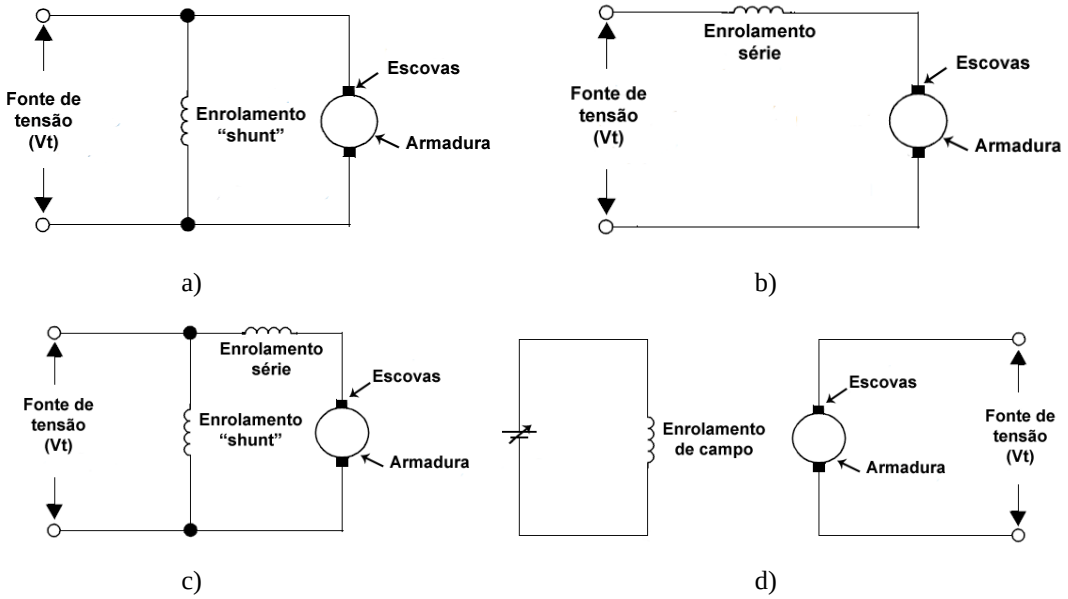


Figura 3-2: Esquema dos motores DC cujo campo magnético é criado por electroímanes: a) motor DC *shunt*; b) motor DC série; c) motor DC *compound*; d) motor DC de excitação independente [58]

No caso em que o campo magnético do estator é produzido por ímanes, os motores são denominados motores de ímanes permanentes; nestes, existe um campo magnético sempre presente, não podendo o fluxo magnético ser variado. Estes motores têm como vantagem o facto de poderem ser mais pequenos, uma vez que não possuem enrolamentos de campo, não sendo, no entanto, possível obter grandes valores de binário, pelo que apenas são utilizados em aplicações de baixa potência [59].

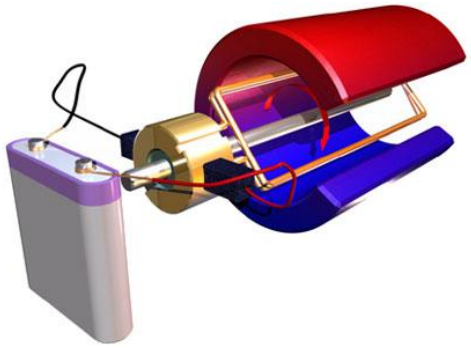


Figura 3-3: Motor DC de ímanes permanentes [60]

3.1.1 Princípio de funcionamento

Apesar de existirem diferentes modelos de motores DC, o seu princípio de funcionamento é o mesmo. Os motores utilizados neste trabalho são todos de ímanes

permanentes, pelo que o princípio de funcionamento, a seguir descrito, será baseado neste tipo de motores.

O princípio de funcionamento do motor DC é baseado no electromagnetismo: uma bobina percorrida por uma corrente produz um campo magnético, que interage com o campo dos ímanes permanentes do estator. A Figura 3-4 pretende ilustrar os diversos passos da rotação do motor. Como se vê na Figura 3-4a, surge uma força de repulsão, resultante da proximidade dos pólos norte do rotor e do estator. Quando o pólo norte se aproxima do pólo sul, como se verifica na Figura 3-4b, surge uma força de atracção. Quando o pólo norte do rotor se alinha com o pólo sul do estator, deixam de existir forças de atracção ou repulsão (Figura 3-4c). Aqui, o motor passa por um equilíbrio momentâneo e a acção do colector e das escovas faz com que haja uma inversão do sentido de circulação da corrente e, consequentemente, a alteração do sentido do campo do rotor. Este fica novamente sujeito a uma força repulsora, continuando assim o seu movimento (Figura 3-4d) [61].

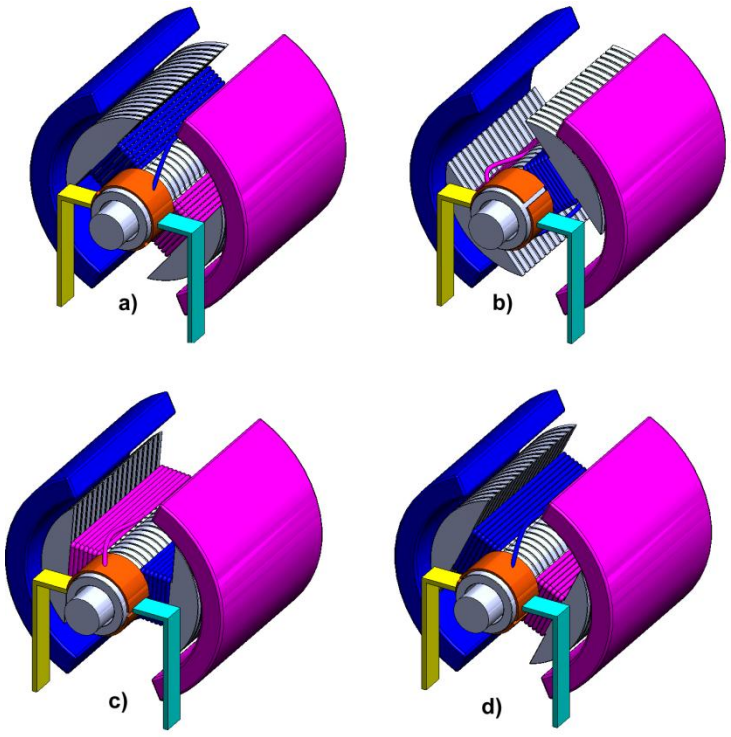


Figura 3-4: Funcionamento de um motor DC

É importante salientar que estes pontos de equilíbrio momentâneos só existem teoricamente. Na realidade, são usados sempre mais que dois pólos no rotor, o que faz com que a acção do colector e das escovas ocorra antes de atingir o equilíbrio e, desta forma, este nunca seja atingido.

3.1.2 Características

Os motores devem ser analisados tendo em conta a tensão nominal, corrente nominal, binário, velocidade e potência, sendo essencial o conhecimento das duas primeiras características para o dimensionamento dos circuitos de controlo.

Tensão nominal

A tensão nominal é o valor da tensão (em V) que presidiu ao projecto do motor e que pode ser aplicada aos seus terminais, quando este funciona nas suas condições normais, sem constituir perigo para a conservação do motor.

O motor pode ser alimentado com tensões inferiores, quando se pretende reduzir a sua velocidade ou a sua potência. Com o objectivo de vencer a inércia, pode também, ser alimentado com uma tensão superior à nominal, durante um pequeno período de tempo. Esta situação não é, no entanto, muito recomendável porque pode originar stress dielétrico (fadiga do sistema de isolamento), com eventual rotura dos isolantes, com possibilidade de originar um curto-circuito [61].

Corrente nominal

A corrente nominal é a intensidade da corrente (em A) a que o motor está sujeito, quando opera nas condições nominais, e que pode suportar sem que ponha em perigo a sua integridade. A corrente é dependente da carga aplicada ao motor, sendo reduzida para o motor a funcionar em vazio. A corrente aumenta à medida que elevamos a carga aplicada e, consequentemente, a força exigida ao motor. Deve ter-se cuidado para que não seja ultrapassada a corrente máxima indicada pelo fabricante, evitando o risco de provocar um sobreaquecimento exagerado do motor [61].

Binário

O binário, também conhecido por *torque* (em N.m), traduz a força exercida entre o rotor e o estator e pode ser determinado, aproximadamente, por:

$$\tau = k \cdot \phi_f \cdot \mathrm{I_a} \tag{3-1}$$

Sendo k uma constante, Φ_f o fluxo magnético produzido pelos enrolamentos do estator, que no caso dos motores de ímanes permanentes é constante, e I_a a corrente injectada na armadura através das escovas [57].

Velocidade

A velocidade angular do rotor (em rpm) depende da tensão que lhe é aplicada e da corrente nos seus condutores.

Sendo o fluxo magnético (Φ_f), nos motores de ímanes permanentes, constante, a tensão gerada pela armadura (E_a) depende da velocidade, como se vê na equação (3-2).

$$E_a = k \cdot \phi_f \cdot \omega_r \tag{3-2}$$

Ainda nestes motores, a tensão aplicada (V_t) pode ser dada por:

$$V_t = E_a + R_a \cdot I_a \tag{3-3}$$

Igualando as duas funções acima apresentadas, chega-se ao valor da velocidade angular do rotor (ω_r), que é apresentada na equação (3-4).

$$\omega_{r=} \frac{V_t - R_a \cdot I_a}{k \cdot \phi_f} \tag{3-4}$$

Como o valor da resistência de armadura (R_a) é muito reduzido (podendo muitas vezes ser desprezado), a velocidade depende especialmente de V_t [57].

Potência

Para produzir um binário, a uma determinada velocidade de rotação, o motor produz uma potência (em W) dada por:

$$P_{Mecanica} = \tau \cdot \omega_r \tag{3-5}$$

Esta potência, existente à saída de um motor DC, pode também ser obtida pela diferença entre o valor da potência eléctrica de entrada e da potência de perdas.

Para obter a potência eléctrica de entrada recorre-se à equação (3-6).

$$P_{Entrada} = V_t \cdot I_a \tag{3-6}$$

A potência de perdas corresponde à parte perdida por efeito de Joule na resistência dos condutores e das escovas, e é dada pela fórmula (3-7).

$$P_{Perdas} = R_a \cdot I_a^2$$

(3-7)

Assim, o que fica da potência eléctrica de entrada, corresponde à potência eléctrica convertida em potência mecânica, sendo [57]:

$$P_{Mecanica} = E_a \cdot I_a$$

(3-8)

3.1.3 Motores utilizados

São utilizados vários motores DC para realizarem a movimentação dos diversos mecanismos da máquina. Tendo em conta as dimensões da máquina, foi necessário recorrer a motores pequenos, pelo que foram escolhidos diversos motores DC de ímanes permanentes.

Quanto ao motor DC1, a sua função é deslocar o mecanismo que prende o documento a encadernar (Figura 3-5) e mantê-lo fixo durante o processo de furação. Para tal, foi escolhido o motor 827470 da *Crouzet*, já com caixa redutora (com redução de 375/4:1) implementada, que permite uma velocidade nominal de 32rpm. O motor apresenta uma tensão nominal de 24V, corrente nominal de 0.7A e 2Nm de binário [62].

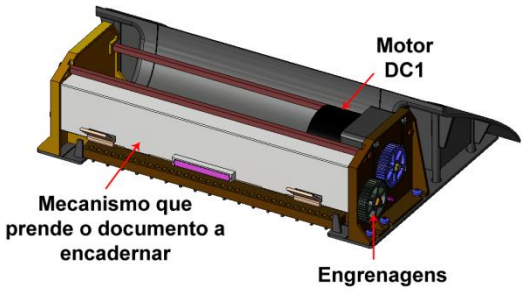


Figura 3-5: Motor DC1 acoplado ao mecanismo que prende o documento a encadernar

O motor DC2 tem como função realizar o movimento ascendente e descendente do bloco de furação. Foi escolhido para tal o motor 827470 da *Crouzet* com uma tensão nominal de 24V, corrente nominal de 2.9A, velocidade de 145rpm e binário de 1.43Nm. Este motor foi sugerido pela *Crouzet*, por possuir uma elevada velocidade, permitindo realizar a furação num tempo reduzido[63].

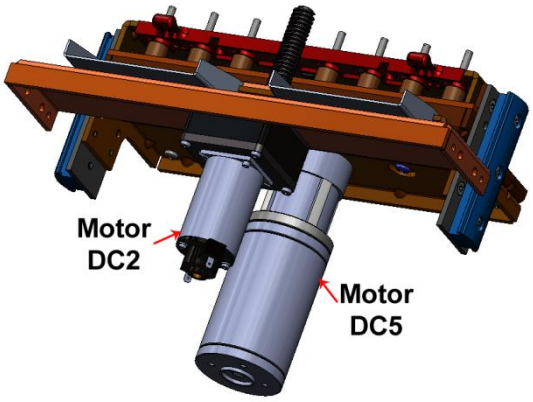


Figura 3-6: Motores DC2 e DC5 acoplados ao bloco de furação

O motor DC5 também é utilizado no processo de furação, sendo responsável pelo movimento de rotação das brocas. Para uma perfeita encadernação, efectuaram-se testes e chegou-se à conclusão que o motor deveria ter uma velocidade entre as 700 e as 1000 rpm e um binário elevado, de forma a que as brocas rodassem sempre a uma velocidade constante. Para tal, foi escolhido o motor 3863 HC da *Faulhaber* com velocidade nominal de 6106rpm, em conjunto com a caixa redutora PLG52 com uma relação de 6.31:1, obtendo-se assim a velocidade de 977rpm. O motor tem a uma tensão nominal de 24V, corrente nominal de 3.8A e binário de 0.58Nm [64].

O motor DC3 é utilizado na deslocação do mecanismo de encadernação, permitindo fazer o comando da porta para colocação do *clickbind*; é também responsável pela deslocação para a posição de encadernação, posição esta que varia em função do tamanho do *clickbind* utilizado. Foi escolhido o motor 316.2761.30.00 da *Doga*, que possui uma tensão nominal de 24V, corrente nominal de 1.7A, velocidade de 38rpm e 2Nm de binário [65].

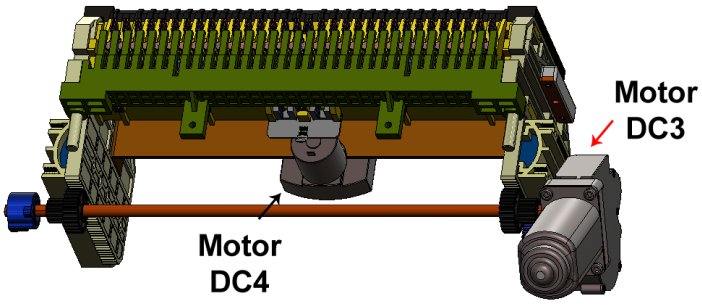


Figura 3-7: Motores DC3 e DC4 acoplados ao bloco de encadernação

Por último, temos o motor DC4 que é responsável por fechar o *clickbind*. O processo deve realizar-se de modo a que o mecanismo se desloque uniformemente para não deformar a argola. Foi usado para tal, o motor TT-38246800-94K da *Sgmada* com

uma tensão nominal de 24V, corrente nominal de 0.28A, velocidade de 54rpm e binário de 0.41Nm [66].

3.1.4 Conclusões

Depois do estudo dos motores DC e das suas características, concluiu-se que existem diversos modelos de motores DC, diferenciados pela construção do estator ou pela forma de ligação dos seus enrolamentos aos enrolamentos de armadura.

Verificou-se também que as características principais dos motores são a tensão nominal, corrente nominal, potência, binário e velocidade. Concluiu-se que o binário depende, principalmente, da corrente da armadura e a velocidade da tensão aplicada.

A especificidade da função de cada motor, levou à necessidade de seleccionar motores com diferentes características, que podem ser vistas na Tabela 3-1.

Motor	**Fabricante**	**Modelo**	**Redução**	**Tensão** (V)	**Corrente** (A)	**Velocidade** (rpm)	**Binário** (Nm)
DC1	*Crouzet*	827470	375/4	24	0.7	32	2
DC2	*Crouzet*	827470	20	24	2.9	145	1.43
DC3	*Doga*	316.2761.30.00	62	24	1.7	38	2
DC4	*Sgmada*	TT-38246800-94K	94	24	0.28	54	0.41
DC5	*Faulhaber*	3863 HC/PLG52	6.3	24	3.8	977	0.58

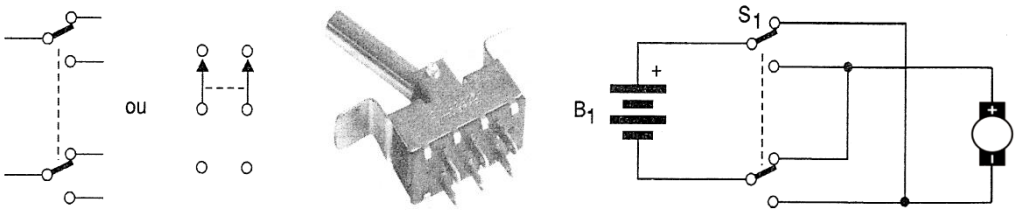
Tabela 3-1: Motores DC utilizados e suas características

3.2 Circuito de controlo dos motores DC

A variação da tensão e do sentido de circulação da corrente permitem alterar respectivamente a velocidade e o sentido de rotação do motor.

Existem diversas formas de comandar um motor DC. A mais simples, mas ao mesmo tempo a mais inviável, é recorrer a uma fonte de tensão variável, uma vez que, variando a tensão, varia a velocidade de rotação do motor. Se a fonte gerar tensões positivas e negativas é igualmente possível variar o sentido de rotação do motor [67].

Outra forma de comandar um motor DC é recorrendo a interruptores. O uso de interruptores manuais permite ligar e desligar um motor e, se estivermos perante um inversor de dois circuitos (Figura 3-8a), é possível também inverter o sentido de rotação (Figura 3-8c). Neste caso, embora sendo igualmente necessária uma fonte de tensão variável, para o controlo da velocidade, deixa de haver necessidade de tensões negativas para efectuar a inversão do sentido de rotação do motor [61].



a) b) c)

Figura 3-8: Inversor de dois circuitos: a) Símbolo; b) Aparência; c) Esquema de accionamento de um motor [61]

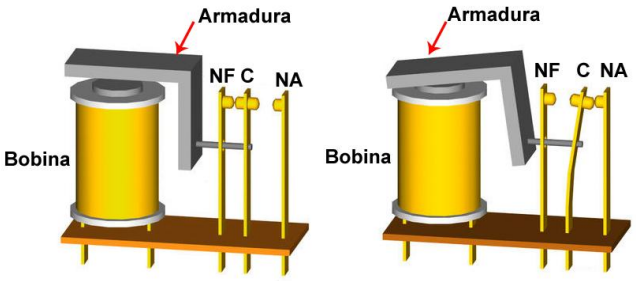
No entanto, estas soluções necessitam de intervenção humana, que pode ser evitada com a utilização de relés, transístores ou MOSFETs (*Metal Oxide Semiconductor Field Effect Transístor*).

3.2.1 Controlo do sentido de rotação

O sentido de rotação pode ser comandado de uma forma automática por intermédio de relés ou semicondutores.

Relés

Os relés são dispositivos electromecânicos (Figura 3-9), que permitem ligar ou desligar um dispositivo eléctrico e isolar o circuito de comando do circuito de potência.



a) b)

Figura 3-9: Relé electromecânico: a) repouso; b) excitado [68]

Ao ser percorrida por uma corrente, a bobina cria um campo magnético que atrai a armadura metálica, feita de material ferromagnético e, desta forma, pode ligar ou desligar um circuito externo. Assim, uma corrente e tensão muito reduzidas na bobina podem ser utilizadas para controlar correntes e tensões muito superiores. Para accionar os relés são normalmente utilizados transístores de baixa potência [61].

Com um relé de contactos simples, é possível comandar um motor para que funcione a uma determinada velocidade e num sentido. Utilizando relés de dois

contactos inversores, é possível fazer o comando do motor com inversão do sentido de rotação [61].

Deve ter-se em atenção que a comutação de correntes elevadas em cargas indutivas gera tensões e correntes elevadas que podem criar arcos eléctricos que diminuem significativamente a vida útil do relé, pelo que este método de controlo não deve ser utilizado em situações onde sejam necessárias ligações frequentes [67].

Ponte H

A outra forma estudada para o controlo de motores DC é a ponte H que recorre a transístores, não sendo, por isso, necessária a utilização de tensões negativas, ou elementos mecânicos. O seu nome deve-se à disposição dos componentes que lembram um "H", (Figura 3-10). Os interruptores representados na figura substituem os transístores, como forma simplificada de explicar o funcionamento deste circuito.

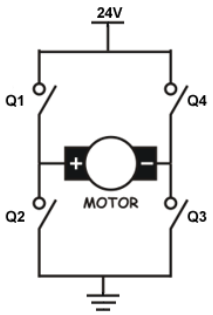
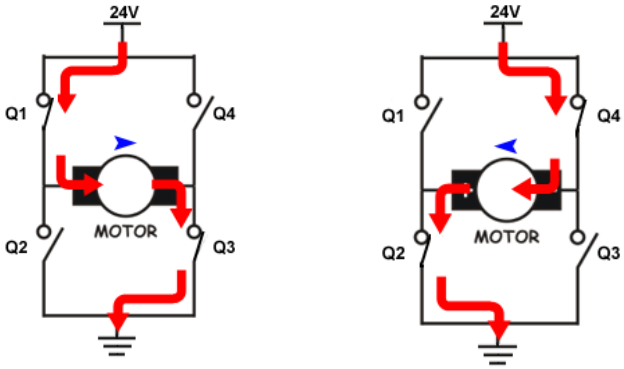


Figura 3-10: Ponte H sem passagem de corrente: motor parado [69]

Os interruptores funcionam dois a dois e, dependendo dos que estiverem ligados, o motor pode rodar para a esquerda, para a direita ou estar parado.

Com todos os interruptores desligados o motor está parado. Nesta situação, todos os interruptores estão em aberto e, por isso, não há qualquer tensão aos terminais do motor [67]. A Figura 3-10 mostra o estado da ponte para este caso.

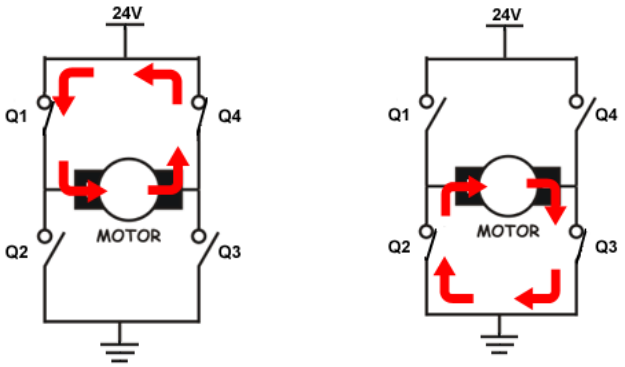
Quando se liga Q1 e Q3, e se mantém Q2 e Q4 desligados (Figura 3-11a), o motor é alimentado directamente, rodando no sentido horário. Na situação em que Q2 e Q4 estão ligados e Q1 e Q3 desligados (Figura 3-11b), o motor é alimentado com uma tensão inversa e, por isso, roda no sentido anti-horário [67].



a) b)

Figura 3-11: Ponte H com representação do sentido da corrente e sentido de rotação do motor: a) Motor roda no sentido horário; b) Motor roda no sentido anti-horário [69]

Nos casos em que se liga Q1 e Q4, mantendo Q2 e Q3 desligados (Figura 3-12a) ou se liga Q2 e Q3 e se mantém Q1 e Q4 desligados (Figura 3-12b), o motor trava forçosamente. Acontece que os terminais do motor entram em curto-circuito e toda a energia do motor é dissipada na sua resistência interna. Como ela é de valor pequeno, a energia dissipa-se rapidamente e o motor pára, o que pode provocar um sobreaquecimento do motor [67].



a) b)

Figura 3-12: Ponte H com representação do motor parado forçosamente: a) Interruptores superiores fechados; b) Interruptores inferiores fechados [69]

O efeito *shoot-through* acontece ao accionar Q1 e Q2 (Figura 3-13a) ou Q3 e Q4 (Figura 3-13b), o que provoca um curto-circuito na bateria, gerando uma corrente muito elevada que destrói os semicondutores [67].

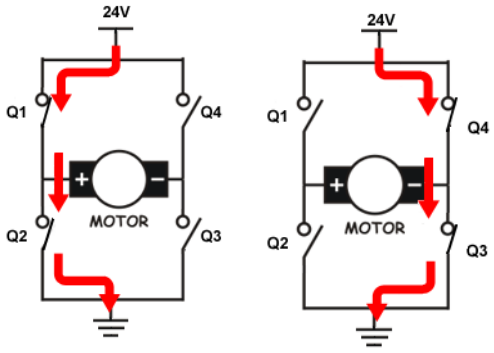


Figura 3-13: Ponte H com a representação do efeito *shoot-through*: a) Fecham simultaneamente os interruptores Q1 e Q2; b) Fecham simultaneamente os interruptores Q3 e Q4 [69]

Uma das formas de evitar este efeito é não permitir que os interruptores liguem simultaneamente. Isto pode ser feito por hardware utilizando um controlador, por software introduzindo um *dead-time* entre as comutações dos interruptores superiores e inferiores ou utilizando os dois métodos em conjunto.

3.2.2 Controlo da velocidade

Como se viu anteriormente, para variar a velocidade de um motor DC, é necessário variar a tensão que lhe é aplicada. Existem duas tecnologias para controlar a velocidade de um motor: o controlo linear e o controlo por PWM (*Pulse Width Modulation*).

Controlo linear

Como o próprio nome indica, neste tipo de controlo a tensão aplicada ao motor varia de uma forma linear. A forma mais simples de o realizar é através de um reóstato, como se pode ver na Figura 3-14 [70].

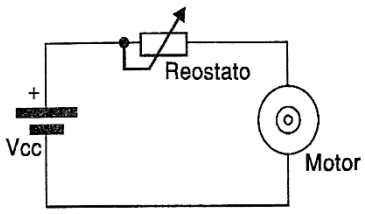


Figura 3-14: Controlo linear realizado com um reóstato [70]

Quando a resistência do reóstato é mínima, a tensão no motor é máxima assim como a sua velocidade. Com o aumento da resistência do reóstato, a tensão aplicada ao motor diminui e com isso diminui a velocidade do motor [70].

A utilização dos reóstatos tem, no entanto, limitações. Como a corrente que percorre o reóstato é a corrente do motor, ele deve ser capaz de controlar correntes relativamente intensas e, como tal, ter uma elevada capacidade de dissipação do calor. Desta forma, este método só deve ser utilizado para motores de pequenas potências [70].

Outra forma de realizar o controlo linear é usando transístores bipolares ou MOSFETs a funcionar na sua região linear. Com uma aplicação com transístores bipolares, como a da Figura 3-15, pode controlar-se a corrente no motor através de uma corrente muito pequena aplicada na base do transístor [70].

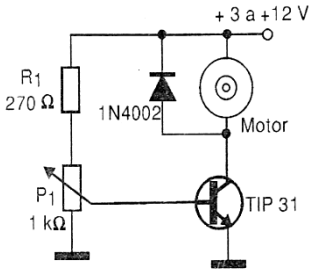


Figura 3-15: Controlo linear usando um transístor bipolar [70]

O controlo linear, mesmo com transístores bipolares ou MOSFETs, apresenta um problema, a inércia. Como mostra a Figura 3-16, os motores, quando alimentados linearmente com uma tensão que parte do zero, não arrancam imediatamente, nem mantêm uma velocidade que acompanha linearmente o valor da tensão. É preciso que o motor rode até que, em determinado momento, vença a inércia e com isso arranque já com uma certa velocidade. Isto faz com que o arranque não seja suave [70].

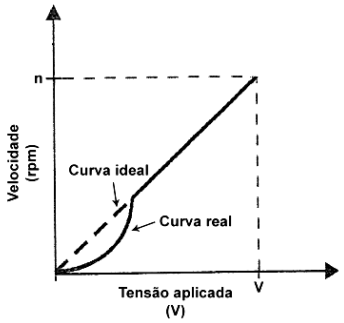


Figura 3-16: Gráfico da curva real e ideal da velocidade no controlo linear [70]

Controlo PWM

O controlo por PWM, ao contrário do controlo linear, não controla a tensão no motor de forma constante, mas actua sobre a média da tensão que lhe é aplicada. Este controlo consiste em ligar e desligar o motor a uma frequência elevada, de forma a que

rode a uma velocidade proporcional à relação que é dada pelo intervalo de tempo ligado (T_{on}) e o período do impulso (T) (Figura 3-17). Essa relação denomina-se *duty cycle* (D) e é dada pela equação (3-9) [67].

$$D = \frac{T_{on}}{T} \cdot 100 \tag{3-9}$$

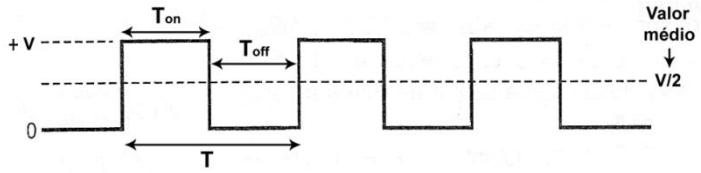


Figura 3-17: Sinal de PWM [71]

A Figura 3-18 mostra três casos possíveis de controlo por PWM. Em todas as situações é usada a mesma frequência. A Figura 3-18a mostra o PWM com um *duty cycle* próximo dos 100%, estando o motor a rodar praticamente à velocidade máxima, porque é alimentado com uma tensão próxima da nominal. A Figura 3-18b mostra o caso em que o T_{on} é igual ao T_{off} e, por isso, a tensão aplicada ao motor é metade da tensão nominal e, consequentemente, a velocidade é igualmente metade da nominal. Por fim temos uma situação em que a velocidade é muito baixa, dado que T_{off} é muito maior que T_{on} (Figura 3-18c) [67].

a) b) c)

Figura 3-18: Sinais de PWM com diferentes *duty cycles*: a) $T_{on}>T_{off}$; b) $T_{on}=T_{off}$; c) $T_{on}<T_{off}$ [67]

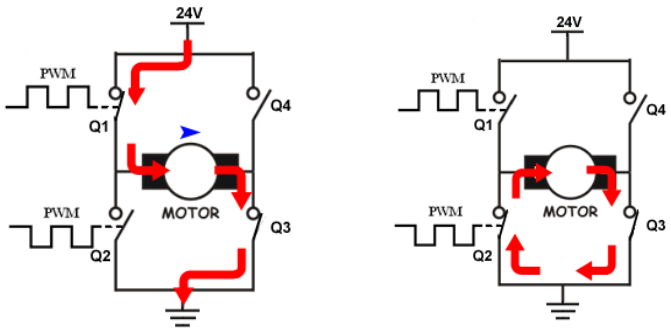
O facto de o motor receber sempre a tensão máxima através dos impulsos permite que o binário seja mantido independentemente da velocidade. Esta característica permite tirar o motor da imobilidade, mesmo quando se pretendem velocidades muito pequenas. Outra vantagem deste controlo é o maior rendimento que apresenta. Quando o transístor está desligado a corrente é nula, não havendo, por isso, dissipação de calor. Quando o transístor liga e atinge a saturação, a corrente é máxima, sendo a resistência praticamente nula, tal como a potência dissipada. A situação em que se verifica maior dissipação de potência é na transição dos estados, mas mesmo aí essa dissipação é menor que a verificada no controlo linear [71].

A desvantagem do controlo PWM está no facto da comutação rápida verificada nos semicondutores, que podem ligar e desligar milhares de vezes por segundo, poder dar origem a instabilidades no circuito. A transição rápida de estado destes componentes gera transitórios e sinais de altas frequências que são responsáveis por interferências electromagnéticas. Estas interferências podem afectar outros equipamentos que se encontrem nas proximidades ou que têm as linhas de alimentação como transmissoras. Para evitar estas interferências, pode ser necessária a utilização de filtros ou outros recursos que evitem a sua propagação [71].

3.2.3 Solução implementada

Até aqui foram referidas diversas formas de controlar os motores, tanto a nível da variação do sentido de rotação, como da sua velocidade. No projecto em desenvolvimento, os cinco motores DC têm que ser controlados e a escolha dos métodos a utilizar tem que ter em conta a sua função na máquina. No caso dos motores DC1, DC2, DC3 e DC4, é necessário variar tanto a velocidade como o sentido de rotação. O motor DC5 apenas precisa ser ligado e desligado.

Desta forma, decidiu-se utilizar nos quatro primeiros motores um controlo do sentido de rotação, através da ponte H, com semicondutores, sendo a variação da velocidade conseguida por PWM. Para aplicar uma onda quadrada ao motor, fazendo-o rodar no sentido horário, é necessário que em cada período ele funcione segundo o apresentado nas Figura 3-11 e Figura 3-12, ou seja, em cada período existe um T_{on} no qual o motor roda no sentido horário e um T_{off} no qual pára. Para obter este resultado, aplica-se o sinal de PWM a Q1, o seu inverso a Q2 e mantém-se Q3 fechado, como se pode ver na Figura 3-19.



a) b)

Figura 3-19: PWM aplicado na ponte H: a) durante T_{on}; b) durante T_{off} [69]

Durante T_{on}, o Q1 e Q3 encontram-se fechados e Q2 e Q4 estão em aberto, estando o motor alimentado directamente (Figura 3-19a). Quando Q1 é aberto (T_{off}), Q2 é fechado e, como Q3 se mantém fechado, a corrente existente, devido à indutância do motor, é descarregada através da sua bobina, travando o motor momentaneamente (Figura 3-19b).

Para que o motor rode no sentido inverso, segue-se o mesmo princípio de funcionamento, tendo em conta que se aplica o sinal de PWM a Q4, o seu inverso a Q3 e mantêm-se Q2 fechado.

No motor DC5, por apenas ser necessário um controlo *on/off* utilizou-se um relé electromecânico de contactos simples.

Determinados os métodos de controlo a usar, foi necessário decidir entre adquirir um controlador para o motor ou implementá-lo.

No caso mais simples, em que o controlo é feito por um relé, adquiriu-se um relé de 24VDC que possui um circuito inversor com contactos para uma corrente nominal de 16A e implementou-se o circuito da Figura 3-20.

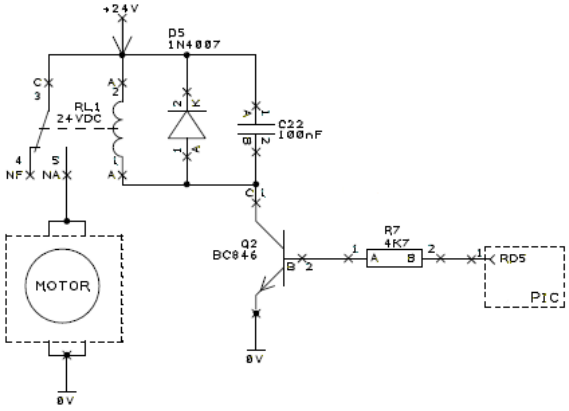


Figura 3-20: Esquemático do circuito de accionamento do motor DC5

Quando o sinal do PIC (*Peripherical Interface Controller*) for a "1" tem-se uma tensão positiva na base do transístor que o faz entrar em condução, alimentando, desta forma, a bobina do relé e assim fazendo ligar o contacto NA (Normalmente Aberto) que conduz a corrente para o motor. A função do díodo e do condensador é proteger o transístor da tensão produzida, quando se desliga o relé.

Para a escolha da ponte H, fez-se um estudo de mercado para ver se, entre as pontes H já implementadas, há alguma que satisfaça os limites de tensão e corrente dos motores. Pretende-se, desta forma, obter uma ponte para 24V, 3A e frequência de PWM de 20kHz (valor elevado mas não audível).

O integrado A3968 da *Allegro* incorpora duas pontes H que suportam, cada uma, uma corrente de 650mA, uma tensão de 30V e uma frequência de PWM de 25.4kHz [72].

O MC33886 da *Freescale Semiconductor* implementa uma ponte H que suporta uma corrente de 5A e uma tensão de 40V, funcionando o seu PWM a uma frequência de 10kHz [73].

Estes dois integrados foram rejeitados por não satisfazerem os requisitos mínimos; no primeiro, a corrente surge como uma limitação e, no segundo, a frequência de PWM é inferior à pretendida.

O integrado LMD18201T da *National Semiconductor* é constituído por uma ponte H que suporta uma corrente de 3A e uma tensão de 55V [74].

O L298, fabricado pela *STMicroelectronics*, implementa duas pontes H, cada uma delas suportando uma corrente de 2A, uma tensão de 46V e uma frequência de PWM de 25kHz. Neste é possível ligar em paralelo as duas pontes H, ficando, desta forma, o integrado a suportar uma corrente de 4A [75].

Verificou-se que estes dois integrados satisfaziam as especificações pretendidas. Foi, portanto, necessário decidir qual deles usar. Inicialmente, a escolha recaiu sobre o LMD18201T, pelo facto de a ponte ser constituída por MOSFETS que são controlados por tensão e permitem uma frequência de comutação maior que a dos transístores bipolares. O elevado custo deste componente tornou, no entanto, inviável a continuação da sua utilização, concluindo-se, então, que a opção mais adequada era o L298. Na análise anterior, verificou-se que este integrado teria a desvantagem de ser construído com transístores bipolares; facto que, no entanto, não impediu o sucesso dos objectivos. Desta forma, implementou-se o circuito descrito na Figura 3-21, que é composto pelo circuito integrado L298, díodos rápidos, para proteger o integrado das tensões geradas nos enrolamentos do motor, e condensadores para filtrar ruídos.

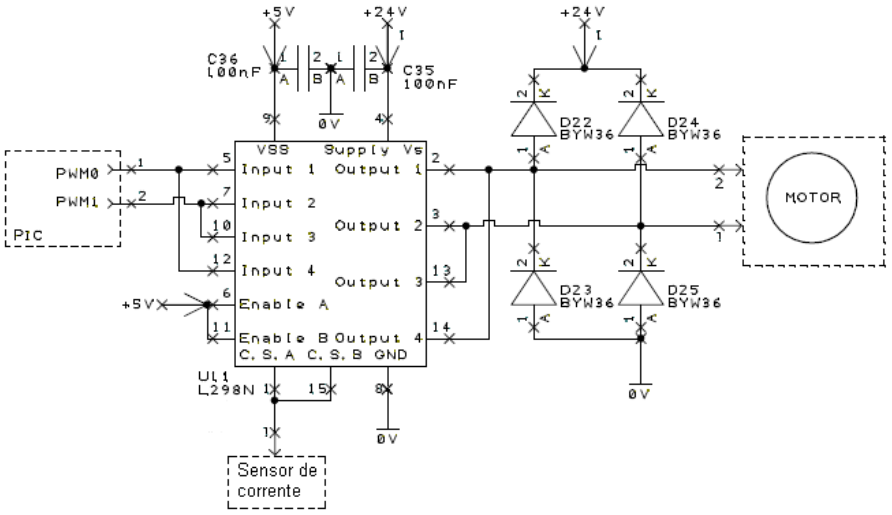


Figura 3-21: Esquemático do circuito de accionamento dos motores DC1 a DC4

O integrado L298 "incorpora duas pontes H para tensões e correntes elevadas, tendo sido desenvolvido para aceitar lógica TTL e comandar cargas indutivas como relés, solenoides, motores DC e de passo." [75]

Na Figura 3-22 pode ver-se o esquemático deste integrado que é composto por quatro entradas (In1, In2, In3 e In4) responsáveis pelo comando dos transístores; duas entradas *enable* (EnA e EnB), que são capazes de habilitar ou desabilitar o motor, independentemente dos sinais de entrada; quatro saídas para alimentação dos motores (Out1, Out2, Out3 e Out4) e duas saídas, que permitem obter indicação sobre o seu consumo (Sense A e Sense B).

Verifica-se que na base de cada transístor, existem portas lógicas que os controlam, de modo a que os sinais enviados para os transístores superiores e inferiores sejam complementares. Assim, quando um está em condução, o outro fica bloqueado, evitando o efeito *shoot-through* e permitindo que o sinal de PWM seja aplicado ao motor.

Como a corrente pretendida é superior à suportada por cada ponte, foi necessário ligar as pontes em paralelo. Foram ligadas todas as entradas e saídas da ponte A às correspondentes entradas e saídas da ponte B. Desta forma, tem-se In1 e In4 ligados, bem como In2 e In3. O accionamento do motor, quer no sentido directo, quer no sentido inverso, é feito com o En ligado. Para o sentido directo é ainda necessário ligar ao par In1-In4 o sinal de PWM e colocar o par In2-In3 à massa, para o sentido inverso liga-se o PWM ao par In2-In3 e o par In1-In4 à massa. Tendo o mesmo sinal de entrada no par

In1-In4 e no par In2-In3, tem-se uma paragem forçada do motor. Quando se desliga o En, obtém-se a paragem do motor qualquer que sejam os valores nas restantes entradas.

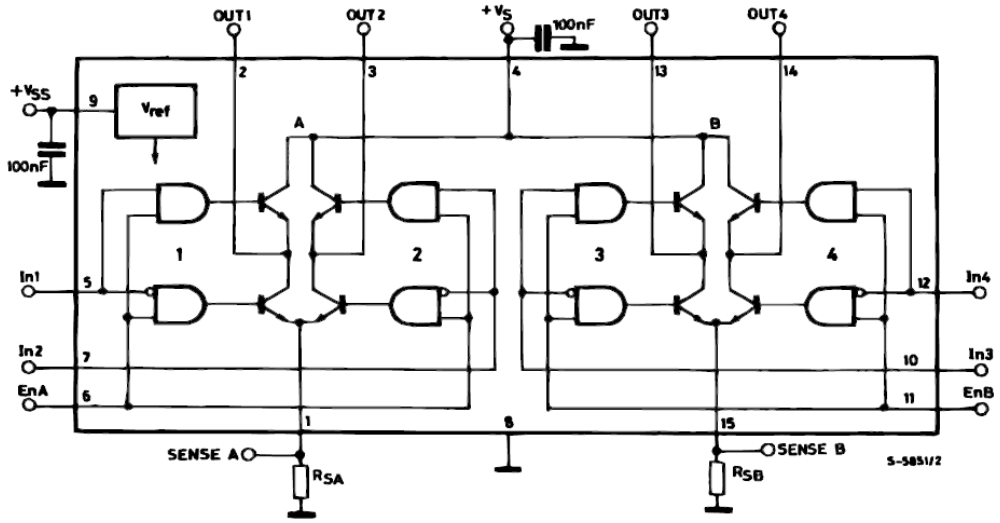


Figura 3-22: Esquemático do integrado L298 [75]

A protecção do integrado deve ser feita com díodos rápidos com um $t_{rr} \leq 200ns$ e uma $I_n > 1A$ [75]. Para tal, utilizaram-se os díodos BYW36 que suportam estas características [76].

3.2.4 Conclusões

Depois do estudo sobre a forma de controlar um motor DC, concluiu-se que, para variar o seu sentido de rotação, é necessário variar o sentido da corrente e, para isso, pode ser usado um relé ou uma ponte H construída com semicondutores. Chegou-se à conclusão que o uso da Ponte H aporta várias vantagens sobre os relés, já que permite a utilização de altas frequências e, consequentemente, um fácil controlo da velocidade do motor por PWM. Como utiliza semicondutores, não está sujeita à fadiga eléctrica. Tem como desvantagem o facto de ser um componente de elevado custo.

Constatou-se também que a variação da velocidade depende da tensão aplicada e pode ser feita por variação linear (controlo linear) ou impulsos (controlo PWM). Verificou-se que uma das dificuldades do controlo da velocidade dos motores ocorre quando se pretende fazer o arranque do motor de uma forma suave e progressiva. Esta situação só é possível utilizando PWM.

Para o projecto em causa optou-se pela utilização do controlo do motor através da ponte H e PWM, nas situações em que havia necessidade de variar a velocidade e o sentido de rotação. Para tal, utilizou-se o integrado L298, obtendo uma solução simples

e de baixo custo para realizar o controlo de motores com corrente nominal até 4A. No motor onde apenas se pretendia um controlo *on/off*, a utilização de um relé electromecânico mostrou ser eficaz e extremamente simples.

3.3 Motores de passo

Os motores de passo convertem energia eléctrica, na forma de impulsos, em energia mecânica, na forma de movimento rotacional discreto, ou seja, cada impulso de tensão aplicada corresponde a um passo do motor. Esta característica impede-os de ter uma rotação contínua como acontece com os motores DC, mas se os impulsos forem aplicados de uma forma contínua e muito rápida, tem-se uma rotação aparentemente contínua [57].

Apesar das inúmeras diferenças entre os motores de passo e os outros modelos de motores eléctricos, o princípio básico de funcionamento mantém-se. Pela passagem de uma corrente, as bobinas criam campos magnéticos que interagem estabelecendo forças que movimentam as partes móveis do motor. Têm também em comum o facto de serem formados por um estator e um rotor, sendo o estator a parte estacionária do motor que envolve o rotor e este a parte central e móvel do motor [77].

A principal diferença entre estes motores e os outros modelos está no modo de excitação das suas bobinas, que permite que se desloquem um determinado número de passos, segundo um ângulo de passo e parem com os seus eixos posicionados de forma precisa. Esta característica torna-os ideais para serem utilizados em sistemas de controlo em malha aberta, possibilitando a eliminação de sensores e com isso a redução de custos [78]. Como desvantagem têm o facto de serem lentos e pouco potentes [77].

É importante também que estes motores possuam uma relação *torque*/inércia superior à dos outros modelos, para garantirem uma resposta rápida aos impulsos, permitindo rápidas paragens, arranques e inversões no sentido de rotação [79].

Todas estas características fazem deles dispositivos muito utilizados em drives de discos flexíveis, impressoras, fotocopiadoras, robôs e todo o tipo de máquinas e equipamentos onde seja necessário recorrência e precisão de movimentos [80].

3.3.1 Tipos de motores de passo e seu funcionamento

Um motor de passo é formado por um estator que possui ranhuras com duas ou mais bobinas individuais e um rotor sem enrolamentos. Dependendo do tipo de rotor, os motores de passo podem ser classificados como motores de íman permanente, de relutância variável ou híbridos [80].

Motor de íman permanente

O rotor deste tipo de motor é cilíndrico e formado por um íman permanente. O estator é, normalmente, formado por dois enrolamentos pelos quais se faz a alimentação do motor. Desta forma, tem-se um campo magnético sempre presente no rotor e, quando se alimenta as bobinas do estator, forma-se outro campo magnético, que interage com o campo magnético do rotor. Dependendo da polaridade do campo magnético gerado no enrolamento e do campo magnético do rotor, é criada uma força de atracção ou repulsão [81].

Na Figura 3-23, está representado um motor de passo a efectuar uma volta. Como se verifica, o estator é formado por dois enrolamentos e o rotor possui dois pólos magnéticos. Aplicando ao enrolamento 1, uma corrente com o sentido descrito na Figura 3-23a, surge, no pólo superior do estator, um pólo sul e, consequentemente, um pólo norte no inferior. Isto faz com que o rotor se alinhe conforme a mesma figura.

O próximo passo do motor é conseguido, removendo a corrente do enrolamento 1 e aplicando-a ao enrolamento 2, no sentido descrito na Figura 3-23b. Surge então um campo magnético que aponta para o pólo do estator, à esquerda, fazendo o rotor girar no sentido anti-horário, de forma a que o seu pólo sul fique apontado para o pólo norte do estator.

O processo continua, removendo desta vez a corrente do enrolamento 2 e aplicando-a novamente ao enrolamento 1, mas agora com sentido contrário, como se vê na Figura 3-23c. Surge, portanto, um campo magnético que faz com que o rotor gire novamente no sentido anti-horário, para que o seu pólo sul fique apontado para o pólo norte do estator.

Em seguida, conforme a Figura 3-23d, remove-se a corrente do enrolamento 1 e aplica-se novamente ao enrolamento 2, o que resulta num novo passo do rotor ainda para a direita.

O último passo do motor é efectuado depois de remover a corrente do enrolamento 2 e aplicá-la ao enrolamento 1, regressando assim o rotor à posição inicial apresentada na Figura 3-23a [82].

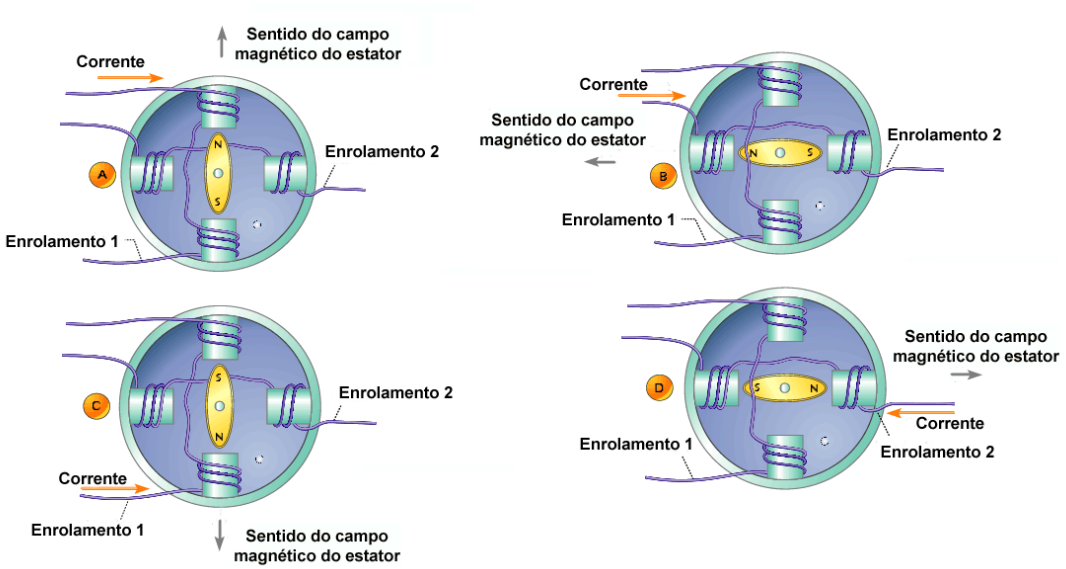


Figura 3-23: Sequência de funcionamento de um motor de passo de íman permanente [82]

A construção do íman permanente do rotor de um motor de passo é consideravelmente diferente da apresentada na Figura 3-23. É desejável aumentar o número de pólos do rotor de forma a produzir um ângulo de passo menor [83].

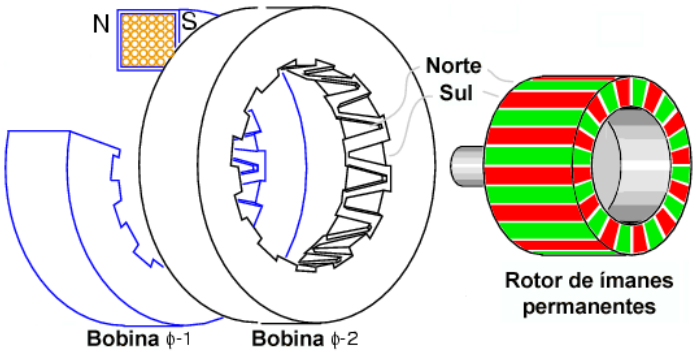


Figura 3-24: Motor de passo de íman permanente de 2 enrolamentos e 24 pólos [83]

Uma característica destes motores é a capacidade de manter o *holding torque* indefinidamente, quando o rotor está parado. Quando nenhum sinal é aplicado aos enrolamentos, uma pequena força é estabelecida entre o rotor e o estator, é o chamado binário residual ou *detent torque* [84].

Os motores de ímanes permanentes requerem menos potência que os outros motores de passo, para obter o mesmo binário; possuem uma construção mecânica simples, excepto quando se pretendem passos pequenos; apresentam um baixo custo, mas têm velocidades elevadas limitadas [85].

Motor de relutância variável

Neste tipo de motor de passo, o rotor tem o aspecto de uma roda dentada (Figura 3-25), com os dentes em ferro macio [78]. O facto de tanto o rotor como o estator serem estruturas de pólos salientes; de só existirem enrolamentos no estator; de o rotor, para além de não possuir enrolamentos também não ser formado por ímanes permanentes; de não serem necessárias escovas nem um colector, faz destes tipos de motores os possuidores da estrutura electromagnética mais simples de entre as máquinas eléctricas [79].

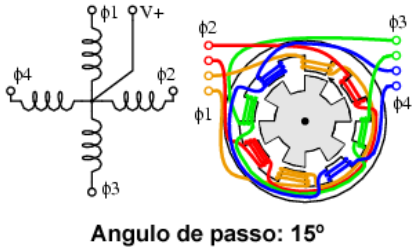


Figura 3-25: Esquema de um motor de passo de relutância variável [83]

Os dentes do rotor e do estator são feitos de forma a que exista uma variação de relutância que é mínima, quando se encontram alinhados, sendo este o ponto de equilíbrio. Assim, através da alimentação de cada uma das fases, o rotor movimenta-se de forma a reduzir o entreferro entre as partes, variando a relutância. O número de pólos do estator tem que ser sempre superior aos do rotor, pois se assim não fosse, o arranque seria impossível sempre que os pólos do estator e do rotor estivessem alinhados [79].

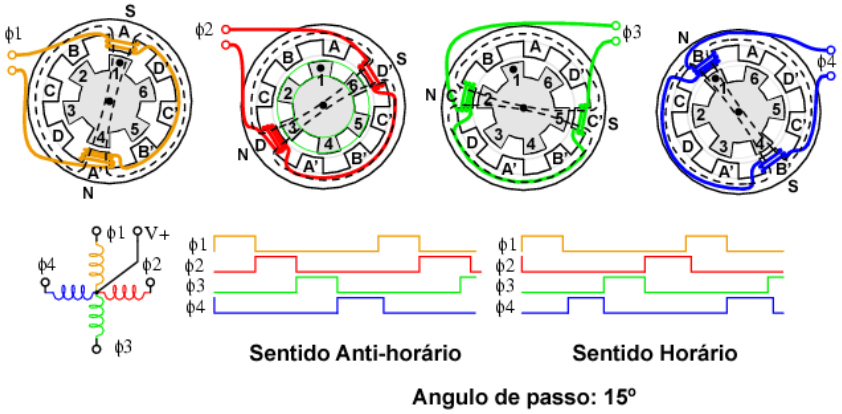


Figura 3-26: Sequência de funcionamento de um motor de passo de relutância variável [83]

Na Figura 3-26, os enrolamentos do estator são alimentados com corrente contínua. Alimentando a fase φ1, os pólos magnéticos do estator A e A' exercem atracção sobre os pólos do rotor 1 e 4, respectivamente, deslocando-se o rotor até ao seu alinhamento total. Quando a alimentação é comutada para a fase φ2, os pólos do estator

D e D’ passam a exercer atracção magnética sobre os pólos do rotor 3 e 6, continuando o movimento do rotor até que se verifique um novo alinhamento dos pólos. Segue-se nova comutação da alimentação, desta vez para a fase φ3, passando a atracção magnética a verificar-se entre os pólos C e C’ do estator e 2 e 5 do rotor. Mais uma vez, é comutada a alimentação, de tal modo que a atracção magnética passa a dar-se entre os pólos do B e B‘ do estator e os pólos do rotor 1 e 4. Obtém-se, assim, um movimento contínuo do rotor no sentido anti-horário [79].

Como o rotor não possui magnetização própria, a alteração do sentido da corrente nas bobinas não altera o sentido de rotação do motor. Assim, a posição relativa dos pólos do rotor em relação ao enrolamento alimentado do estator e a sequência de alimentação dos seus enrolamentos possibilitam a mudança do sentido de rotação [79].

Os motores de passo de relutância variável possibilitam velocidades elevadas e passos pequenos. Por outro lado, não possuem binário residual, a sua construção é mais complexa que a dos motores de íman permanente e possuem uma razão binário/tamanho inferior à dos outros tipos de motores [85].

Motor híbrido

O motor híbrido combina características do motor de passo de relutância variável e do motor de passo de íman permanente, de forma a produzir ângulos de passo menores [83].

O seu estator é idêntico ao do motor de passo de íman permanente, possuindo normalmente duas bobinas enroladas em pólos alternados. O rotor é cilíndrico, formado por duas secções com muitos dentes, como o de relutância variável, e magnetizado, como o de ímanes permanentes (Figura 3-27) [58].

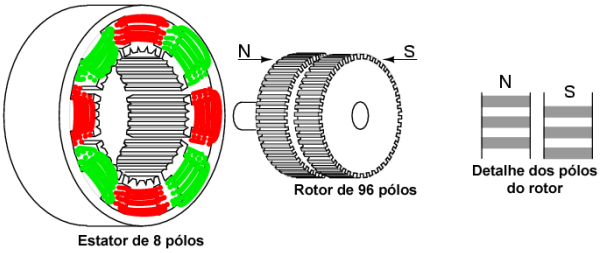


Figura 3-27: Motor de passo híbrido [83]

A Figura 3-28 mostra um motor onde os 48 dentes de cada secção do rotor estão desfasados uns dos outros. Os dentes nos 8 pólos do estator correspondem aos 48 dentes do rotor, com excepção dos dentes que faltam no espaço entre os pólos. Desta forma, um pólo do rotor pode alinhar-se com um pólo do estator em 48 posições distintas.

Como os dentes do pólo sul estão desfasados dos dentes do pólo norte pela metade de um dente (Figura 3-28), o rotor pode alinhar com o estator em 96 posições distintas. Os pólos do estator estão divididos em duas fases, φ1 e φ2, que, como se pode ver na Figura 3-28, estão desfasadas uma da outra por ¼ de dente. Isto leva a que o rotor se desloque com passos de ¼ de dente, quando se alimentam alternadamente as fases. O desfasamento entre os pólos, em conjunto com o sentido da corrente, definem o sentido de rotação do motor [83].

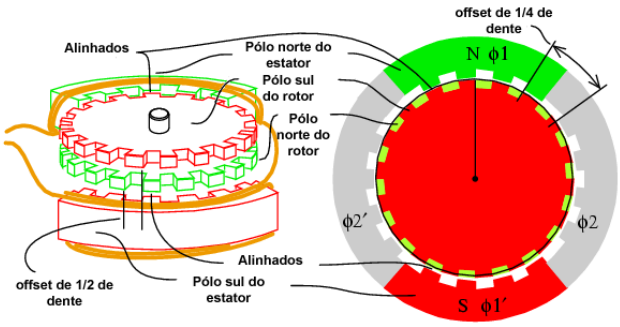


Figura 3-28: Esquemático de um motor de passo híbrido [83]

Ainda na Figura 3-28, verifica-se que, no topo do rotor de ímanes permanentes, se encontra o pólo sul e, na base, o pólo norte. Alimentando a fase φ1 do estator de forma a ter o pólo norte no topo e o pólo sul na base, obtêm-se os dentes do pólo norte do estator alinhados, no topo, com os dentes do pólo sul do rotor, e os dentes do pólo sul do estator alinhados, na base, com os dentes do pólo norte do rotor. É com esta fase alimentada e φ2 desligada que se inicia o movimento do motor apresentado na Figura 3-29a [83].

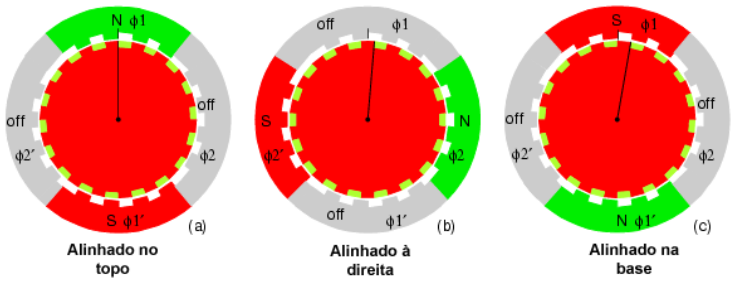


Figura 3-29: Sequência de funcionamento de um motor de passo híbrido [83]

Na Figura 3-29b, φ1 encontra-se desligada e φ2 alimentada, o que resulta numa deslocação do rotor de ¼ de dente, para que o pólo sul do rotor fique alinhado com o pólo norte do estator (φ2) e o pólo norte do rotor com o pólo sul do estator (φ2'). Na Figura 3-29c, φ1 é novamente alimentada, mas agora reversamente e φ2 é desligada. Desta forma, o rotor desloca-se mais ¼ de dente para que o pólo sul do rotor fique

alinhado com o pólo norte o estator (φ1') e o pólo norte do rotor com o pólo sul do estator (φ1). Até este momento, o rotor moveu-se ½ dente. Com φ1 desligada e φ2 alimentada reversamente, o rotor deslocar-se-ia mais ¼ de dente resultando, por isso, em ¾ de dente. Para completar uma volta, alimentar-se-ia novamente φ1 e desligar-se-ia φ2 [83].

3.3.2 Características

Os motores de passo devem ser analisados tendo em conta a tensão e a corrente nominais, o ângulo de passo, a velocidade de rotação, o binário de retenção e binário dinâmico.

Tensão e corrente nominais

A tensão e corrente nominais são os valores da tensão (em V) e da corrente (em A) que o motor pode suportar sem que ponha em perigo a sua conservação. Os seus valores são geralmente fornecidos pelo fabricante do motor. Com recurso a estes valores, é possível calcular a resistência do enrolamento e a potência eléctrica [77].

Ângulo de passo

O ângulo de passo é o deslocamento angular (em º) correspondente a um passo, pelo que os movimentos do rotor serão efectuados em múltiplos do ângulo de passo [57]. Os valores mais usuais para o ângulo de passo dos motores são 0.72, 1.8, 2.0, 2.5, 5.0, 7.5 e 15 [87].

Velocidade de rotação

A velocidade de rotação de um motor de passo é proporcional à frequência com que se aplicam impulsos aos enrolamentos do estator. Comparados com outros modelos de motores, a velocidade de rotação dos motores de passo é bastante reduzida [77].

Binário de retenção

O binário de retenção, também conhecido por *holding torque* (em Nm), é o binário necessário para mover o rotor de um passo, quando este se encontra parado, estando os enrolamentos alimentados [57].

Binário dinâmico

Para este binário normalmente são definidas duas curvas que são obtidas através de um gráfico que representa o binário x velocidade. A curva de binário *pull in* indica

qual o binário que é possível fornecer à carga sem que o motor sofra a perda de qualquer passo. A curva de binário *pull out* indica o binário que é possível fornecer à carga, quando o motor é acelerado lentamente até à velocidade de funcionamento desejada [57].

3.3.3 Modos de alimentação

A alimentação de um motor de passo consiste na alimentação sucessiva dos vários enrolamentos do estator [86]. A forma de ligação dos enrolamentos do estator está, por isso, intimamente ligada ao modo de alimentação dos mesmos. Já foi visto como se ligam os enrolamentos do estator no motor de relutância variável, assim como o método pelo qual se realiza a sua alimentação. Os motores de ímanes permanentes e híbridos podem ter os seus enrolamentos estatóricos ligados de diversas maneiras e, dependendo do modo como são ligados, a alimentação diz-se unipolar, bipolar ou bifilar.

Unipolar

Os motores de passo unipolares possuem uma derivação central em cada uma das bobinas, que leva a que o número de fases seja o dobro do número de bobinas [87] (Figura 3-30). Estes motores podem ter cinco ou seis fios.

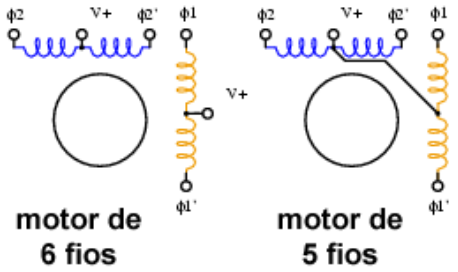


Figura 3-30: Esquema dos motores de passo unipolares de cinco e seis fios [83]

A tensão é aplicada sequencialmente entre a derivação central e os extremos de cada bobina, como se vê na Figura 3-30. A derivação central é sempre alimentada positivamente, sendo os terminais das bobinas ligados sequencialmente à massa [87]. Como o sentido da corrente em cada metade do enrolamento é oposto, o campo magnético produzido também o é [57].

Bipolar

Os motores de passo bipolares são constituídos por bobinas sem derivação central, o que os torna mais simples, Figura 3-31 [88].

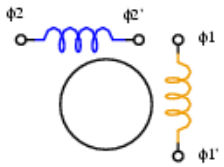


Figura 3-31: Esquema dos motores de passo bipolares [83]

Nestes motores, a corrente percorre todo o enrolamento, e o campo magnético é invertido, quando o sentido da corrente no enrolamento também o for. O facto de ser necessário inverter a polaridade de cada par de pólos faz com que sejam necessários circuitos de drive mais complexos [88].

Bifilar

Os motores de passo bifilares possuem dois conjuntos idênticos de bobinas em cada pólo (

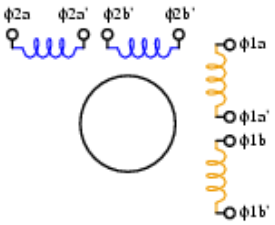


Figura 3-32), que são enrolados juntos, mas em sentidos opostos, como se fossem uma única bobina [86].

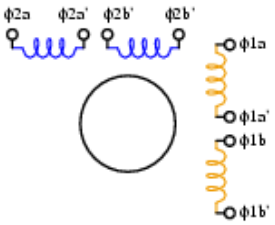


Figura 3-32: Esquema dos motores de passo bifilares [83]

Os fios destes motores são ligados de modo a que as suas bobinas fiquem ligadas como nos motores unipolares ou bipolares [88].

Para usar um motor bifilar como unipolar, devem ligar-se em série os dois enrolamentos existentes em cada par de pólos, usando o ponto de conexão como derivação central (Figura 3-33) [88].

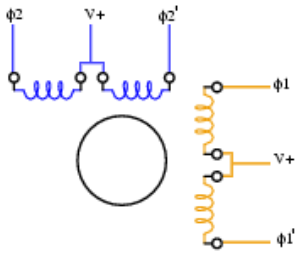
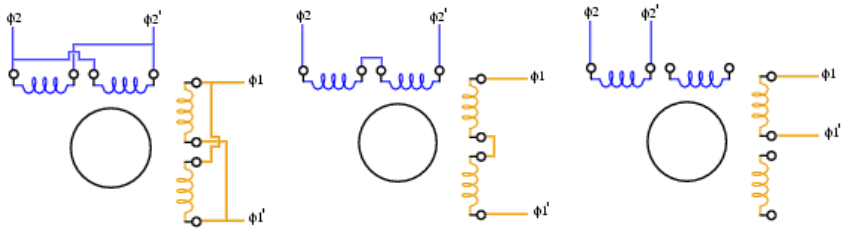


Figura 3-33: Motor bifilar com tipo de ligação unipolar [83]

Para usar um motor bifilar como bipolar, podem ligar-se em paralelo, em série ou usando apenas um único enrolamento por fase, os dois enrolamentos existentes em cada par de pólos (Figura 3-34) [88].



a) b) c)

Figura 3-34: Motor bifilar com tipo de ligação bipolar: a) paralelo; b) série; c) um enrolamento [83]

Comparando as três formas de ligação bipolares dos motores bifilares, verifica-se que a ligação em série concede uma maior indutância, permitindo correntes de operação inferiores e um melhor desempenho a baixas velocidades. A ligação em paralelo permite uma indutância inferior, o que requer uma corrente superior. Mas atinge um melhor desempenho a grandes velocidades, visto garantir um binário superior (Figura 3-35). A alimentação com apenas um único enrolamento por fase reduz o binário para baixas velocidades e a corrente requerida, visto apenas metade das bobinas disponíveis serem utilizadas [89][90].

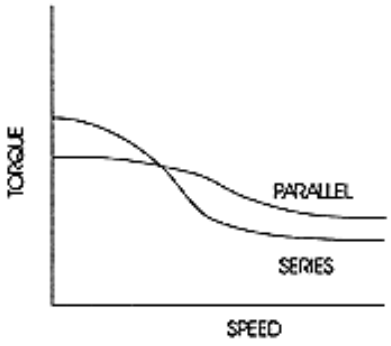


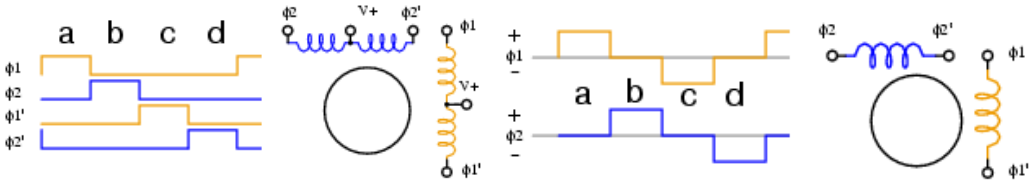
Figura 3-35: Gráfico *torque* x velocidade para as ligações bifilares série e paralelo [90]

3.3.4 Modos de accionamento

O desempenho de um motor de passo também depende muito do método de accionamento utilizado. O modo de accionamento pode ser *wave excitation*, *full step*, *half step* ou *microstepping*. Estes accionamentos podem ser feitos tanto em motores unipolares como bipolares. No modo bipolar, é necessário ter em conta que o sentido da corrente nos enrolamentos pode ser positivo ou negativo, produzindo, respectivamente, um campo magnético positivo ou negativo. No modo unipolar, conduz-se alteradamente cada uma das metades do enrolamento, produzindo também alternadamente um campo magnético positivo ou negativo [57].

Wave excitation

Neste tipo de accionamento, apenas um enrolamento (no modo bipolar) ou meio (no modo unipolar) é alimentado de cada vez [57]. A Figura 3-36a representa o caso unipolar e a Figura 3-36b o caso bipolar.



a) b)

Figura 3-36: Formas de onda aplicadas aos motores funcionando no modo *wave excitation*: a) motor unipolar; b) motor bipolar [83]

Realizando as sequências apresentadas na Figura 3-36, num motor de passo com ângulo de passo de 90º, o motor dá uma volta, como mostra a Figura 3-37.

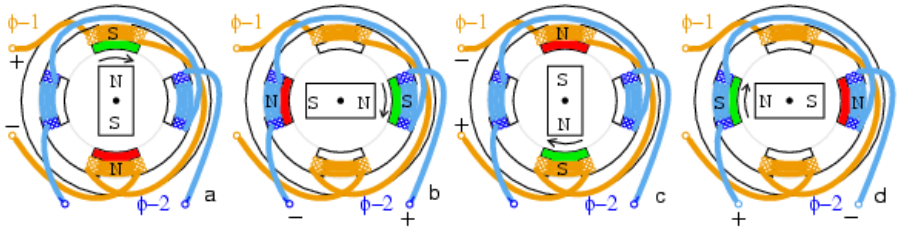
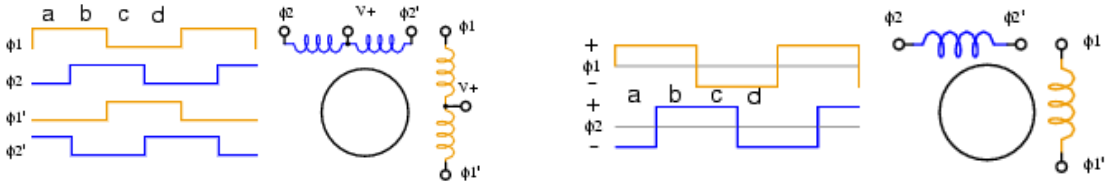


Figura 3-37: Sequência do motor accionado em modo *wave excitation* [83]

Full step

Neste tipo de accionamento, os enrolamentos são alimentados simultaneamente, dando-se a alteração das polaridades num enrolamento de cada vez [57]. A Figura 3-38a representa o caso unipolar e a Figura 3-38b o caso bipolar.



a) b)

Figura 3-38: Formas de onda aplicadas aos motores funcionando no modo *full step*: a) motor unipolar; b) motor bipolar [83]

Tal como para o modo *wave excitation*, a realização das sequências bipolar ou unipolar, apresentadas na Figura 3-38, levam ao mesmo resultado, que, neste caso, é o representado na Figura 3-39.

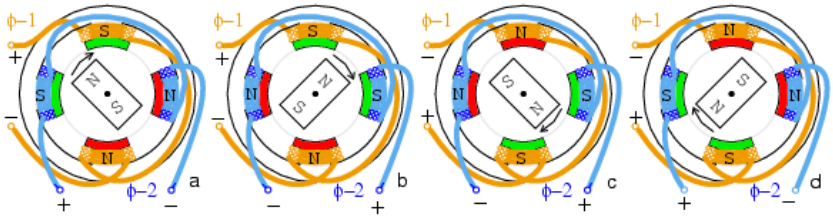
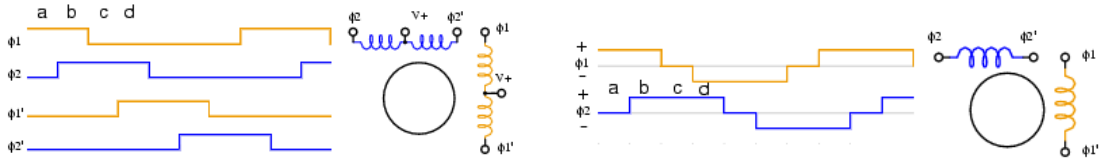


Figura 3-39: Sequência do motor accionado em modo *full step* [83]

Pelo facto de haver sempre duas bobinas alimentadas em simultâneo, o binário dinâmico e o *holding torque* são superiores em cerca de 30% em relação aos existentes no modo *Wave excitation*, onde apenas se alimenta uma bobina [57].

Half step

O modo de accionamento *half step* combina os modos de accionamento *wave excitation* e *full step*. O que acontece é que, num passo, estão alimentados dois enrolamentos e, no passo seguinte, apenas um, voltando, de seguida, a estar dois enrolamentos alimentados, e assim sucessivamente [57]. A Figura 3-40a representa o caso unipolar e a Figura 3-40b o caso bipolar.



a) b)

Figura 3-40: Formas de onda aplicadas aos motores funcionando no modo *half step*: a) motor unipolar; b) motor bipolar [83]

No *half step*, a realização das sequências bipolar ou unipolar, apresentadas na Figura 3-40, levam ao mesmo resultado (Figura 3-41).

Este modo de accionamento permite que o rotor desloque meio passo de cada vez (Figura 3-41), mas exige que a realização de uma volta necessite do dobro dos impulsos, quando comparado com os outros modos de accionamento referidos [87].

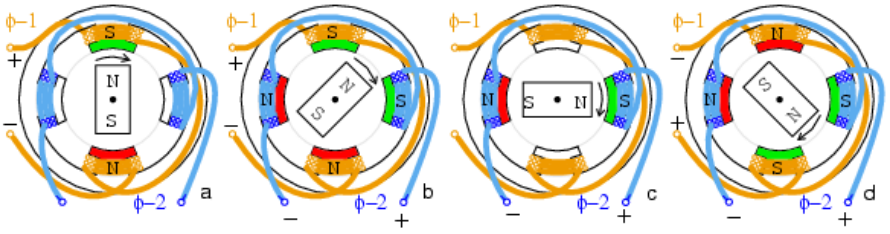


Figura 3-41: Sequência do motor accionado em modo *half step* [83]

O facto de estarem alimentadas uma ou duas bobinas, alternadamente, resulta em dois valores diferentes de *holding torque* que é superior, quando se tem as duas bobinas alimentadas. O facto de serem dados passos mais pequenos proporciona um movimento mais suave para baixas velocidades [87].

Microstepping

Este modo de accionamento permite obter ângulos de passo extremamente pequenos com recurso a uma sequência complexa de alimentação das fases, baseada em aproximações digitais de curvas sinusoidais, de onde resultam interpolações que correspondem a posições de equilíbrio [86].

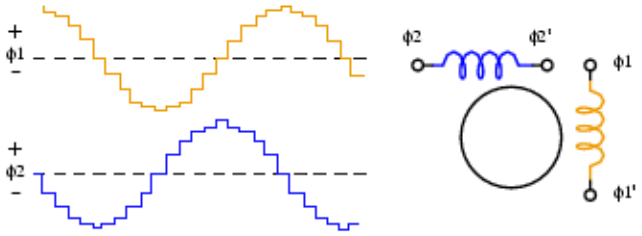


Figura 3-42: Formas de onda aplicadas aos motores bipolares funcionando no modo *microstepping* [87]

Como o deslocamento do rotor é ainda menor, a suavidade do movimento para baixas velocidades é melhorada. Podem ser obtidos 500 micro passos, por passo, com *drivers* de alta resolução [87].

3.3.5 Motor utilizado

A necessidade de mover a mesa com elevada precisão, levou ao uso de motores de passo. Escolheu-se o motor ST5918M1008 da *Nanotec* (Figura 3-43), que é um motor híbrido e de alimentação bifilar.

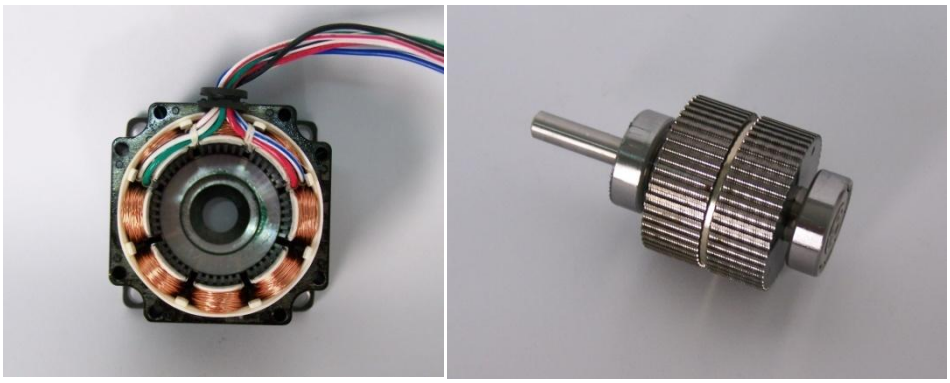


Figura 3-43: Estator (esq) e rotor (dir) do motor ST5918M1008

De entre os modos de alimentação que os motores bifilares permitem, seleccionou-se o bipolar com ligação em paralelo, por permitir um binário mais elevado. Apresenta uma tensão nominal de 6.9V, uma corrente/fase de 1.41A, um ângulo de passo de 1.8º, um *holding torque* de 1.48Nm e um *detent torque* de 0.04Nm [91].

3.3.6 Conclusões

Após o estudo dos motores de passo e das suas características, concluiu-se que existem diferentes modelos de motores de passo que se diferenciam pela construção do rotor, denominando-se motor de passo de íman permanente, de relutância variável e híbrido.

Constatou-se que podem ser alimentados de forma unipolar, bipolar ou bifilar, obtendo-se um binário superior no modo bipolar e no modo bifilar com ligação em paralelo.

Podem ter um modo de accionamento do tipo *wave excitation*, *full step*, *half step* ou *microstepping*, sendo o modo *full step* o que permite um maior binário por ter sempre dois enrolamentos excitados e o modo *microstepping*, o que permite uma maior suavidade do movimento para baixas velocidades.

As principais características destes motores são a corrente e tensão nominais, o ângulo de passo, a velocidade de rotação e os binários dinâmico e de retenção.

Concluiu-se que a sua utilização é especialmente aconselhada em situações de elevada recorrência e precisão de movimentos.

No presente projecto, seleccionou-se o modelo ST5918M1008 da *Nanotec* por apresentar um ângulo de passo reduzido e um elevado binário.

3.4 Circuito de controlo dos motores de passo

O sistema electrónico de controlo de motores de passo é feito com recurso a um controlador e um *driver*, (Figura 3-44) [87].

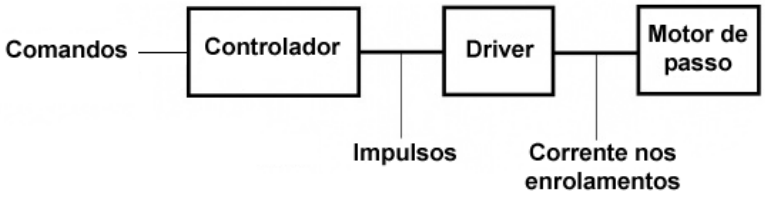


Figura 3-44: Circuito de controlo de motores de passo

O controlo do motor de passo é feito, alimentando cada enrolamento do motor individualmente, num tempo e ordem específicos, através do circuito de *driver*. O sincronismo é executado pelo controlador, cujas entradas recebem comandos apropriados para o número de passos, velocidade e sentido de rotação pretendidos [92].

3.4.1 Controlador

O controlador ou *indexer* é capaz de gerar os sinais para indicação do número de passos e direcção para o *driver*, podendo executar igualmente outras funções de controlo, como aceleração e desaceleração [90].

Os impulsos no controlador podem ser produzidos por software, usando um microprocessador, ou por hardware [92].

Os controladores por hardware possuem, normalmente, um circuito de passo controlado e um circuito sequenciador. O primeiro produz a sequência de impulsos à velocidade desejada para o motor, impulsos esses que devem ter uma duração constante para um correcto accionamento. Aplicado ao sequenciador, cada impulso faz com que o motor dê um passo. O circuito sequenciador tem por finalidade sequenciar os impulsos de forma a que o motor rode no sentido pretendido [77].

Os controladores por software são cada vez mais usados nos dias de hoje, uma vez que os microprocessadores são frequentemente utilizados para substituir parte do hardware, tornando os sistemas mais compactos, mais rápidos e de menor custo [88].

3.4.2 Drivers

O bloco de *driver* é responsável por ligar e desligar a corrente em cada bobina do motor e controlar a sua direcção. Ele é ligado directamente às bobinas do motor de passo e à tensão de alimentação do motor e é controlado por um sistema digital, o controlador, que determina quando os interruptores devem ligar ou desligar [88].

Os circuitos de *driver* variam conforme se comanda um motor de passo de relutância variável, de íman permanente, unipolar ou bipolar, ou híbrido também nestes dois modos [88]. Em relação aos de ímanes permanentes e híbridos, é possível usar um *driver* bipolar para controlar um motor unipolar, não sendo, no entanto, permitido o inverso, devido à derivação central das bobinas [92].

Motor de relutância variável

O *driver* de um motor de relutância variável possui, para cada enrolamento do motor, um bloco de comando em semicondutores, que está representado por um quadrado na Figura 3-45, funcionando como um interruptor. O seu comando é feito por sinais vindos do controlador.

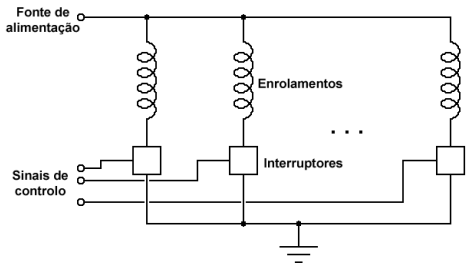


Figura 3-45: *Driver* de um motor de relutância variável [88]

Os enrolamentos dos motores de passo são cargas indutivas e, por isso, a sua corrente não pode ser ligada e desligada instantaneamente sem que surjam tensões muito elevadas. Quando um interruptor é fechado, permitindo que a corrente atravesse o enrolamento, o resultado é um aumento suave da corrente. Quando o interruptor é aberto, o resultado é um pico de tensão que pode danificar o interruptor. Uma forma simples de proteger os interruptores é colocar, em paralelo com o enrolamento do motor, um díodo (Figura 3-46) [88].

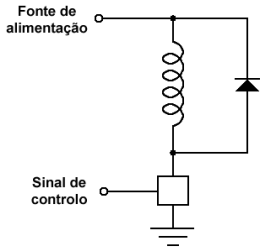


Figura 3-46: Esquema de protecção dos enrolamentos com um doído [88]

Devem ser usados díodos rápidos, para que à alteração do estado dos interruptores haja uma resposta atempada. O díodo deve permitir que a corrente atravesse os enrolamentos do motor e apenas conduz, por breves instantes, no momento em que o interruptor é aberto e são criadas tensões elevadas na bobina [88].

Motor de íman permanente ou híbrido unipolar

O *driver* de um motor unipolar possui, para cada metade de enrolamento do motor, um bloco de comando em semicondutores, que está representado por um quadrado na Figura 3-47. O seu comando é igualmente feito por sinais vindos do controlador.

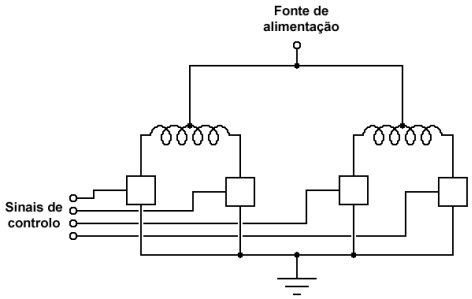


Figura 3-47: *Driver* de um motor de íman permanente ou híbrido unipolar [88]

Nestes motores, tal como nos de relutância variável, também é necessário o uso de díodos rápidos, como forma de protecção dos enrolamentos. Neste caso, são necessários quatro díodos, sendo os dois díodos extras essenciais, dado que cada enrolamento possui uma derivação central ligada a uma tensão de alimentação [88].

Motor de íman permanente ou híbrido bipolar

O *driver* de um motor bipolar é mais complexo que os outros apresentados anteriormente. Como não possui derivação central, para inverter a direcção do campo magnético produzido pelos enrolamentos, é necessário inverter a corrente nos enrolamentos, sendo necessária a utilização de uma ponte H, como a usada nos motores

DC. Na Figura 3-48, os interruptores são igualmente representados por quadrados e comandados por sinais vindos do controlador.

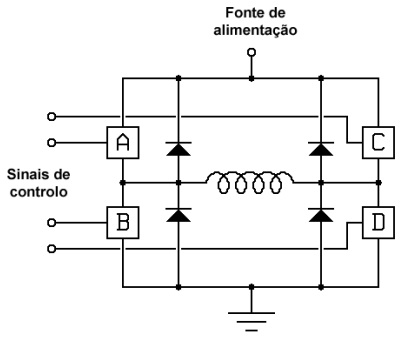


Figura 3-48: *Driver* de um motor de íman permanente ou híbrido unipolar [88]

Nestes motores, o uso de díodos rápidos, como forma de protecção dos enrolamentos, também é estritamente necessário.

3.4.3 Solução implementada

Até agora foram explicadas as formas de controlar um motor de passo. Verificou-se que é necessário o uso de um controlador e de um *driver*. Um estudo do mercado mostrou que existem circuitos integrados que possuem as duas funções e outros que tem apenas uma delas.

Como já foi referido, o motor utilizado é do tipo híbrido, com modo de alimentação bifilar. Como os seus enrolamentos foram ligados de forma a obter-se um motor bipolar paralelo, o *driver* pretendido, para além de permitir esta característica, deve ser capaz de satisfazer os valores de tensão (6.9V) e de corrente (1.41A).

O integrado MC3479 da *ON Semiconductor* incorpora tanto o circuito de controlo como o de *driver*, suportando uma corrente de 350mA por fase e uma tensão entre os 7.2V e os 16.5V. Possui entradas para definir o sentido de rotação, escolher o modo de accionamento e *clock* [93]. Como se verifica, este integrado não pode ser utilizado, visto que tanto a tensão como a corrente não estão dentro dos limites desejados.

O L297 é um circuito integrado da *STMicroelectronics* que implementa apenas um controlador. São aplicados sinais para definir o modo de accionamento, o sentido de rotação e o *clock*. Este integrado tem que ser utilizado em conjunto com um circuito de *driver* de duas pontes H. Um dos integrados indicados para ser usado com o L297 é o

L298, já usado para controlar os motores DC. Verifica-se ainda que este integrado é capaz de suportar os limites de tensão e corrente impostos pelo motor de passo [75][94].

Assim, escolheu-se o conjunto L297/L298 e implementou-se o circuito representado na Figura 3-49. Também aqui são utilizados díodos rápidos para proteger o integrado das tensões geradas nos enrolamentos do motor, e condensadores para filtrar ruídos.

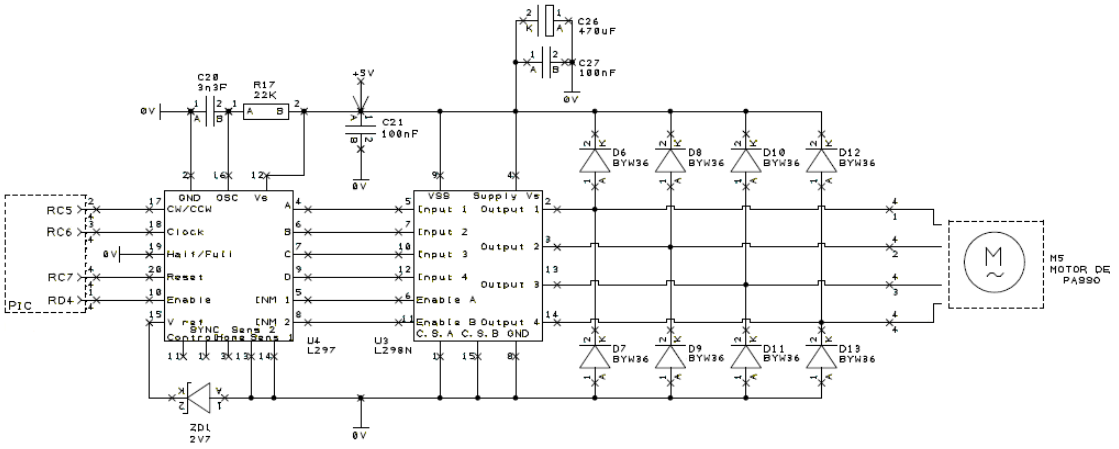


Figura 3-49: Esquemático do circuito de accionamento do motor de passo

O L297 é um controlador que gera quatro saídas para um *driver* de um motor de passo bipolar de duas fases ou unipolar de quatro fases. Este circuito apenas necessita de um sinal de *clock*, direcção e modo de accionamento, para gerar os sinais de controlo necessários para o circuito de potência. O seu esquemático pode ser visto na Figura 3-50.

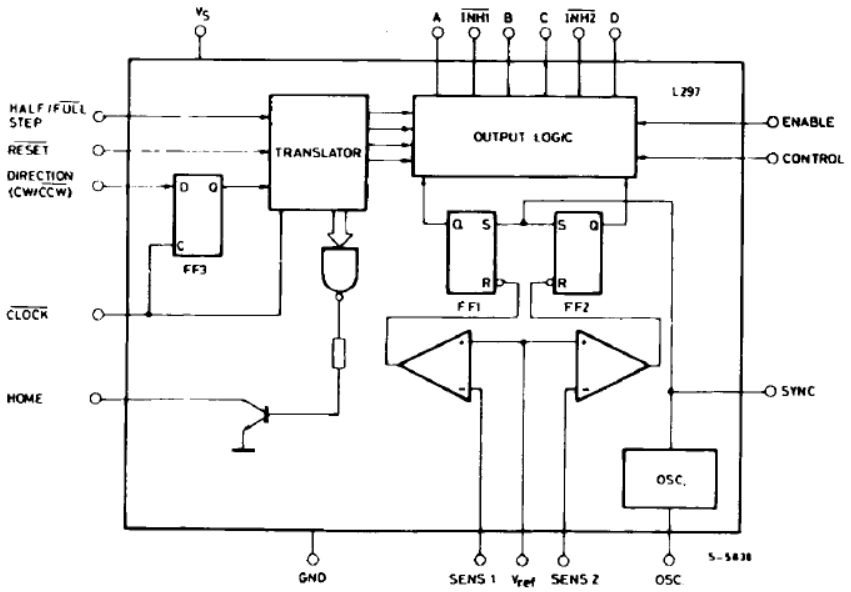


Figura 3-50: Esquemático do integrado L297 [94]

As funções principais são realizadas pelo bloco *translator* que gera a sequência de fases para o motor (saídas ABCD) e o *chopper* que regula a corrente nos enrolamentos do motor. A sequência de fases do motor pode ter diferentes configurações, como se verificou no subcapítulo dos modos de accionamento, sendo o modo determinado pela entrada *Half/Full*. Os dois sinais de *inhibit*, *Inh1* e *Inh2*, são ligados directamente às entradas de *Enable* do integrado L298 e destinam-se a aumentar a velocidade de decaimento da corrente, quando uma bobina é desernegizada. Estes sinais são gerados nos modos *half step* e *wave excitation*. O sinal *Enable* permite habilitar ou desabilitar o motor de passo e o *Direction* determina qual o seu sentido de rotação. As duas entradas *Sens* verificam o consumo dos motores e comparam-no com o valor de V_{REF}.

Para accionar um motor de passo deve ser seguido o diagrama temporal da Figura 3-51. Inicialmente, deve ser efectuado o *Reset*; são, de seguida, dados os sinais de direcção (*CW/CCW*) e modo de accionamento (*Half/Full*); por fim, dá-se um impulso de *Clock*.

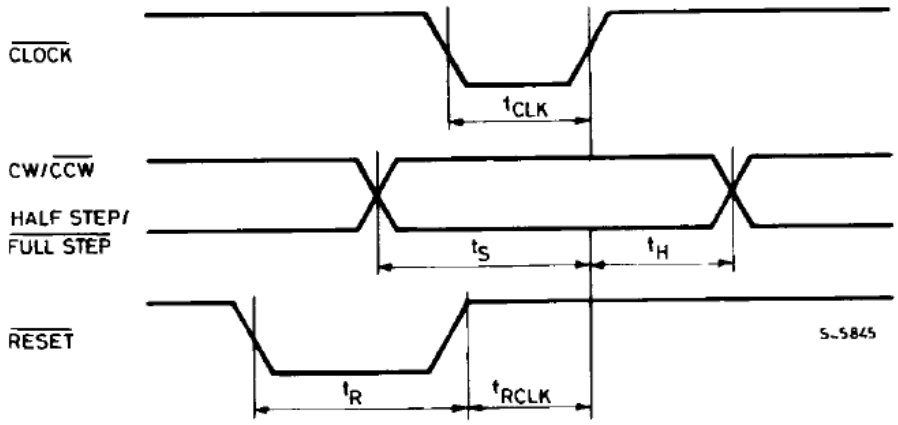


Figura 3-51: Diagrama temporal do L297 [94]

Ligado ao L297, encontra-se o circuito integrado L298 (Figura 3-22), onde cada meia ponte é responsável por comandar um dos enrolamentos do motor de passo. O seu princípio de funcionamento encontra-se explicado no ponto referente ao controlo dos motores DC. A protecção dos integrados deve ser feita com díodos rápidos com um $t_{rr} \leq 200ns$ e uma $I_n > 2A$ [75], tendo-se utilizado os díodos BYW36 que suportam estas características [76].

3.4.4 Conclusões

Concluiu-se que o comando de um motor de passo deve ser feito com um controlador e um *driver*. O controlador pode ser implementado por hardware ou software. O circuito de *driver* está intimamente ligado ao tipo de motores e ao seu modo de alimentação.

No projecto em causa, optou-se por um controlador em hardware em conjunto com uma ponte H. Esta opção deveu-se ao facto da ponte H ser a mesma já utilizada para o controlo dos motores DC, o que reduz o custo do componente.

Capítulo 4

Desenvolvimento de hardware e software

O desenvolvimento de projectos em automação passa, pela construção de um *hardware* e, muitas vezes, pelo seu controlo por *software*.

Neste capítulo abordam-se os microcontroladores, dando maior ênfase aos seleccionados para o trabalho. Referem-se as ferramentas de desenvolvimento utilizadas, linguagem de programação, compilador e programador usado, e identifica-se o *hardware* associado ao microcontrolador. Segue-se um estudo sobre *displays* e sensores, assim como do seu *hardware* e *software*; desenvolve-se o *software* utilizado no controlo do *hardware*, referido no capítulo anterior. Termina-se explicando a sequência de passos da máquina e o código utilizado para a sua realização, bem como o modo de comunicação utilizado.

4.1 Microcontroladores

Os microcontroladores são compostos por um microprocessador, memórias e diversos periféricos, tais como portas de I/O (*Input/Output*), ADC (*Analog to Digital Converter*), *timers*, protocolos de comunicação, entre outros.

O microprocessador responsável pelo processamento de dados e instruções do sistema é constituído pela ALU (*Arithmetic and Logic Unit*), registos de dados e TCU (*Time and Control Unit*) [95]. Existem dois tipos de memórias num microcontrolador, a memória de código, que pode ser do tipo ROM (*Read Only Memory*), OTP (*One Time Programmable*) ou *Flash* e a memória de dados que pode ser do tipo RAM (*Random Access Memory*) ou EEPROM (*Electrically Erasable Programmable ROM*). Os pinos de I/O, dado tratarem-se de pinos bidireccionais, podem ser configurados como entradas, permitindo que o programa responda a sinais externos, ou como saídas,

podendo ser levados a “0” ou a “1” [96]. Os módulos de ADC permitem que aos microcontroladores sejam ligados sensores ou outros dispositivos externos com sinais analógicos, uma vez que é capaz de os converter em valores digitais entendidos por estes [97]. Os *timers* podem funcionar como temporizadores ou contadores, medindo tempo ou eventos, respectivamente. Os protocolos de comunicação permitem que se estabeleça comunicação do microcontrolador com outros microcontroladores ou com outros componentes do circuito.

Tanto o microprocessador, como alguns dos seus periféricos, necessitam de uma base de tempo, o *clock*, que é gerada com um oscilador. O oscilador pode ser obtido com um cristal de quartzo, um ressonador cerâmico, um oscilador RC ou através do oscilador interno. O cristal de quartzo além de possuir a gama de valores mais abrangente, é de todos o mais preciso, tendo o inconveniente de ser o mais caro. O oscilador interno é o mais económico, uma vez que já vem incorporado, mas é muito pouco preciso [98].

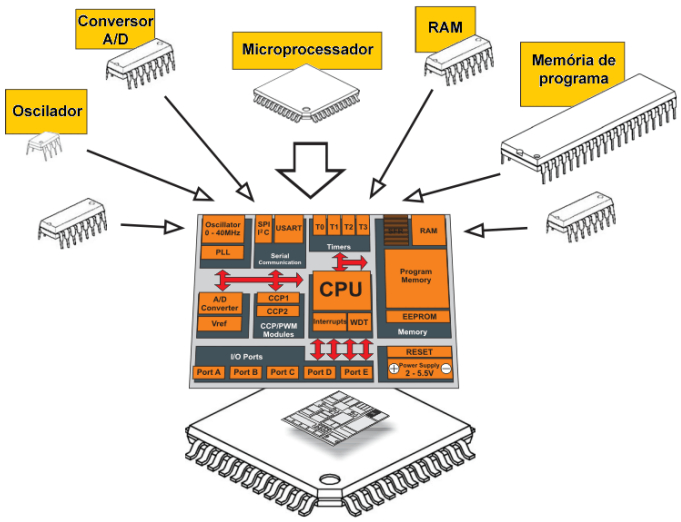


Figura 4-1: Microcontrolador e seus periféricos [97]

4.1.1 Microcontroladores utilizados

No capítulo anterior explicou-se o princípio de funcionamento dos motores utilizados, assim como do hardware necessário para o seu controlo. Verificou-se que, tanto no caso dos motores DC (*Direct current*), como do motor de passo, o hardware de controlo precisa ser comandado por sinais digitais. A monitorização de todos os sensores e o controlo do *display* gráfico têm igualmente que ser efectuados por software, pelo que surgiu a necessidade de utilizar microcontroladores.

Para seleccionar um microcontrolador capaz de suprir estas necessidades, foi necessário pesquisar as diversas marcas existentes no mercado e depois seleccionar o microcontrolador mais apropriado.

Foram analisadas algumas marcas como a *Microchip*, *Atmel* e *Intel*. A escolha pela *Microchip* e pelos seus microcontroladores PIC (*Peripherical Interface Controller*) deveu-se a um conjunto de factores, nomeadamente ao número de funcionalidades que os microcontroladores PIC apresentam, à experiência já adquirida em trabalhos anteriores, à facilidade na obtenção de amostras, à existência de compiladores livres, à quantidade de programas-exemplo disponíveis gratuitamente, à existência de um fórum da *Microchip*, onde é possível esclarecer qualquer tipo de dúvida, bem como ao facto da empresa que irá produzir a placa de comando da máquina trabalhar habitualmente com esta marca.

Depois de escolhida a marca, o próximo passo era escolher, de entre os microcontroladores PIC, aquele que melhor se adequava às necessidades. Para tal, consultou-se o site da *Microchip* e seguiu-se o conselho do fabricante para o controlo de motores DC e motores de passo, optando-se pelo PIC18F4431. Pesou também na escolha o facto de possuir um periférico especializado para o controlo de motores que inclui um módulo com 8 canais de PWM (*Pulse Width Modulation*). Com o desenvolvimento do trabalho e o deparar de um enorme número de sensores que era necessário monitorizar, houve necessidade de recorrer a uma outra solução. Não sendo possível deixar de parte o PIC18F4431, dado o módulo de PWM que ele possui ser essencial para o controlo dos quatro motores DC, decidiu usar-se dois microcontroladores PICs, sendo um destinado ao controlo dos motores e monitorização dos sensores e um outro destinado ao controlo do *display* gráfico. Para esta função, escolheu-se o PIC18F2520 por possuir uma memória de código que permite gravar as imagens a serem utilizadas no *display* e por já ter sido utilizado, anteriormente, noutros trabalhos.

4.1.2 Características do PIC 18F4431

O PIC18F4431 é um PIC de 8 bits da família PIC18F que, tal como todos os microcontroladores PIC desta família, possui uma arquitectura baseada na arquitectura *Harvard* com um *instruction set* RISC (*Reduced Instruction Set Computer*) [99].

Possui uma memória de código do tipo *Flash* de 16kBytes, que permite que os dados sejam escritos e apagados várias vezes. A memória de dados compreende 256Bytes de memória EEPROM e 768Bytes de memória RAM [100].

Integra tecnologia ICSP (*In-Circuit Serial Programming*), que lhe concede a possibilidade de ser programado depois de colocado na placa de circuito impresso, o que permite reduzir o tempo de desenvolvimento, aumentar a eficiência na produção e reduzir custos em situações de upgrades [99].

O PIC 18F4431 possui, como já foi referido, um módulo de 8 canais de PWM de 14-Bit com saídas complementares; um módulo de *motion feedback*, que pode funcionar com três canais de *Input Capture* ou QEI (*Quadrature Encoder Interface*); um ADC com 9 canais de 10-Bit e uma velocidade de 200k amostras por segundo; dois módulos CCP (*Capture/Compare/PWM*); um módulo EUSART (*Enhanced Universal Synchronous Asynchronous Receiver Transmitter*), que suporta RS-485 (*Recommended Standard 485*), RS-232 (*Recommended Standard 232*) e LIN 1.2 (*Local Interconnect Network*); um módulo SSP (*Synchronous Serial Port*), que pode funcionar com SPI (*Serial Peripheral Interface*) ou I^2C (*Inter-Integrated Circuit*) e 4 *timers*, sendo um de 8bits e os restantes de 16bits. É composto por 40 pinos, 36 deles para I/O e 4 para alimentação. O sinal de *clock* pode ser obtido com um cristal, um ressonador cerâmico, um oscilador RC ou através do oscilador interno, que é capaz de gerar dois sinais de *clock* diferentes, um de 8MHz e outro de 31KHz. O uso de um cristal permite obter uma velocidade máxima de CPU (*Central Processing Unit*) de 40MHz com 10 MIPS (*Millions of Instructions Per Second*) [100].

4.1.3 Características do PIC 18F2520

O PIC18F2520, tal como o PIC18F4431, é um PIC de 8 bits da família PIC18F, possui, portanto, uma arquitectura baseada na arquitectura *Harvard* com um *instruction set* RISC [99].

A memória de código é do tipo *Flash* de 32kBytes e a memória de dados compreende 256Bytes de memória EEPROM e 1536Bytes de memória RAM. Possui ainda a tecnologia ICSP, tal como o PIC18F4431 [101].

O PIC 18F2520 é um integrado de 28 pinos, sendo 25 deles para I/O e 3 para alimentação do integrado. Possui ainda um ADC com 10 canais de 10-Bit, dois módulos CCP, um módulo EUSART que suporta RS-485, RS-232 e LIN 1.2, um módulo MSSP

que pode funcionar com SPI ou I^2C e 4 *timers*, sendo um de 8bits e os restantes de 16bits. O oscilador para obter o tempo base de execução das operações pode ser conseguido da mesma forma que para o PIC18F4431. Também neste PIC a velocidade máxima de CPU é de 40MHz com 10 MIPS [101].

4.1.4 Conclusões

Concluiu-se que os microcontroadores PIC têm uma estrutura básica semelhante, apresentando cada um dos PIC, algumas funções específicas.

Relativamente ao controlo de motores DC, constatou-se que não havia muitas alternativas. Por isso, tendo em conta a sua capacidade para controlar diversos motores em simultâneo, foi seleccionado o modelo PIC18F4431.

Seleccionou-se o modelo PIC18F2520, para realizar o controlo do *display* gráfico, uma vez que a sua memória de código é suficientemente elevada, para permitir a gravação das imagens a apresentar no *display*.

4.2 Ferramentas de desenvolvimento

Escolhidos os PIC a utilizar, foi necessário definir que linguagem de programação usar, assim como o seu compilador e programador. Como plataforma de desenvolvimento, a *Microchip* possui o MPLAB IDE (*Integrated Development Environment*) [96].

4.2.1 MPLAB IDE

O MPLAB IDE é uma ferramenta gratuita que corre num PC e ajuda na escrita, edição, *debug* e programação do código nos microcontrolador PIC e dsPIC da *Microchip* [96].

O MPALB IDE inclui um servidor de componentes de software livres que permitem a detecção de erros e um rápido desenvolvimento de aplicações [102]. O *Programmer Editor*, incluído no MPLAB IDE, reconhece a construção da programação do compilador, permitindo que o código fonte tenha diferentes cores, como forma de assegurar uma correcta sintaxe. O *Project Manager* permite organizar os vários ficheiros *source*, *header* e bibliotecas utilizados nas aplicações. O MPLAB SIM permite

testar o código, mesmo sem o *hardware* concluído, simulando as entradas como modo de testar a resposta do *firmware* a sinais externos. Com o *hardware* funcional é possível realizar o *debug* em tempo real na aplicação, com recurso ao MPLAB ICE (*In-Circuit Emulator*) ou MPLAB ICD (*In-Circuit Debugger*) 2. Estes dispositivos, assim como os programadores PICSTART Plus ou MPLAB PM3, permitem também realizar a programação do microcontrolador [96]. A alteração de ferramentas é bastante simples, assim como a passagem de um simulador como o MPLAB SIM para o MPLAB ICD2 ou MPLAB ICE, uma vez que o MPLAB IDE tem o mesmo interface para todas as ferramentas [102].

4.2.2 Linguagem de programação

Os microcontroladores PIC podem ser programados em *Assembly*, *Basic*, *C* ou *Pascal*. O *Assembly* é uma linguagem de baixo nível, onde a cada instrução se faz corresponder uma outra executada pelo microcontrolador. Tem como vantagens tornar o código mais rápido e optimizado, aumentando, desta forma, a sua eficiência. Por outro lado, é uma linguagem complexa e de baixa portabilidade, ou seja, ao mudar de microcontrolador, o *instruction set* pode ser diferente, tendo, por exemplo, mais instruções de forma a melhorar a performance do microcontrolador. As linguagens de programação de alto nível, como o *Basic* e o *Pascal*, têm como vantagens o facto de serem escritas de uma forma facilmente entendida, de possuírem grande portabilidade e de permitirem simplificar operações aritméticas e condicionais. Como o desenvolvimento de programas se torna mais simples, diminui o tempo de desenvolvimento e a probabilidade de surgirem erros, obtendo-se, no entanto, um código menos compacto e mais lento [103]. Por último, a linguagem *C* é considerada de nível médio, uma vez que possui tanto características do *Assembly*, sendo capaz de trabalhar com instruções ao nível do bit e de endereços, como características de linguagens de alto nível [95].

Optou-se, então, pela linguagem *C* tendo em conta as vantagens aqui referidas, a experiência obtida durante o curso e dado tratar-se de uma linguagem largamente usada, havendo, por isso, muita informação. Esta escolha implicou a procura de um compilador.

4.2.3 Compilador MPLAB C18

Um compilador é uma ferramenta de software capaz de traduzir cada instrução do programa para código máquina [95], obtendo-se, desta forma, um ficheiro em formato hexadecimal que pode ser enviado para o microcontrolador. Ao contrário do assembler (compilador de linguagem *Assembly*), o compilador não realiza a tradução linha a linha [104].

O compilador escolhido foi o MPLAB C18 C Compiler da *Microchip*, trata-se de um *cross-compiler* que corre num PC e produz código que pode ser executado pelos microcontroladores PIC da família PIC18F. O facto de ser integrado com o MPLAB IDE, concede-lhe todas as características deste *software* [104].

O MPLAB C18 é capaz de optimizar código, utilizando rotinas que foram usadas numa função, aplicando-as noutras funções. É capaz de reformular ou eliminar código que nunca será executado, partilhar fragmentos comuns de código entre muitas funções e identificar os dados e registos, que são usados de forma ineficaz, optimizando o processo [104].

4.2.4 MPLAB ICD 2

O envio do ficheiro hexadecimal para o microcontrolador PIC é feito através de um programador. A *Microchip*, como já foi referido, possui diversos programadores que trabalham com o MPLAB IDE, tais como o MPLAB ICE, MPLAB ICD 2, MPLAB PM3 e o PICkit 2 [102].

A escolha recaiu sobre o MPALB ICD2, porque pode ser utilizado como programador ou *debugger*. Possui tecnologia ICSP e ICD, tecnologia esta que permite, após programação do PIC, realizar o *debug* em tempo real e examinar em detalhe o decorrer do programa, utilizando a função *debug* do MPLAB IDE [102].

O MPLAB ICD2 conecta-se ao PIC, através de um cabo de 6 condutores, com a numeração apresentada na Figura 4-2.

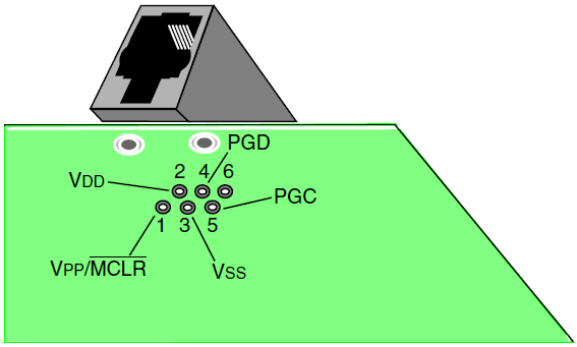


Figura 4-2: Conector MPALB ICD 2 [105]

A Figura 4-3 mostra as ligações do MPLAB ICD2 ao PIC. Verifica-se que apenas 5 condutores são utilizados. É necessário usar uma resistência de *pull-up* com valor compreendido entre 1kΩ e 10kΩ entre V_{PP}/MCLR e V_{DD}, para evitar *resets* intempestivos [105].

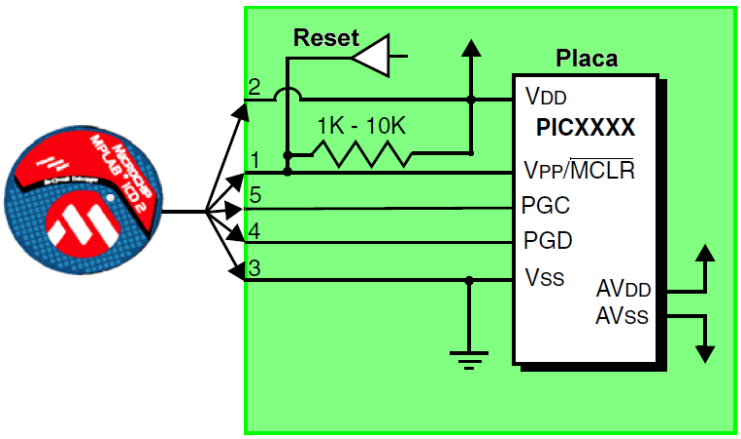


Figura 4-3: Ligação do MPLAB ICD 2 à aplicação [105]

4.2.5 Conclusões

Constatou-se que a *Microchip* possui uma gama alargada de ferramentas, que vai desde uma plataforma de desenvolvimento, a vários compiladores, *debuggers* e programadores.

A plataforma de desenvolvimento é o MPLAB IDE que auxilia na escrita, *debug* e programação do código, nos microcontrolador PIC.

A utilização do PIC da família PIC18F, programados em linguagem C, levou à escolha do compilador MPLAB C18.

Constatou-se ainda que o facto do MPLAB IDE ter o mesmo interface para todas as ferramentas, permite a utilização de qualquer *debugger* ou programador. A opção

pelo MPLAB ICD2 deveu-se, essencialmente, às suas elevadas funcionalidades e baixo custo.

4.3 Hardware associado aos microcontroladores

Os microcontroladores PIC18F4431 e PIC18F2520 têm associado *hardware* essencial para o seu correcto funcionamento.

Na Figura 4-4, pode ver-se o *hardware* do PIC18F4431, onde se observam quatro entradas para alimentação do PIC (VSS, VDD, AVSS e AVDD), tendo associados condensadores cerâmicos de 100nF, para filtrar o ruído e electrolíticos de 10µF, para estabilizar a tensão à entrada do PIC. Para o sinal de *clock*, optou-se por uma frequência de 20MHz com o oscilador no modo HS (*High-Speed Crystal/Resonator*), utilizando-se um cristal de 20MHz em conjunto com dois condensadores de 15pF, inseridos nas duas entradas de *clock* (OSC1 e OSC 2). Para a programação da PIC, são utilizadas três entradas (MCLR, PGC e PGD) em conjunto com os sinais de alimentação.

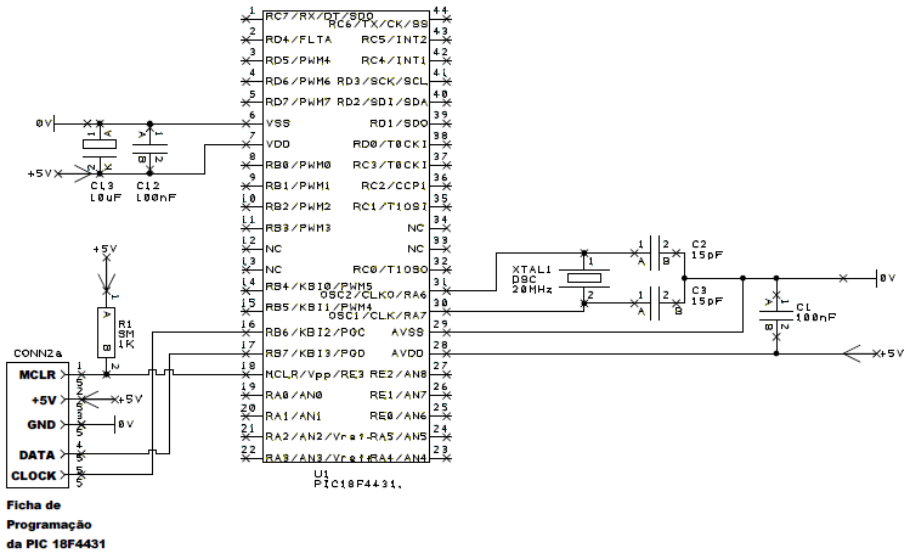


Figura 4-4: Esquemático do hardware do PIC18F4431

No PIC18F2520, como se vê na Figura 4-5, o *hardware* é idêntico ao utilizado no PIC18F4431, tendo como única diferença o facto de apenas serem necessárias três entradas para alimentação do PIC (2 entradas VSS e uma VDD), pelo que apenas são utilizados 2 condensadores, um cerâmico de 100nF e um electrolítico de 10µF.

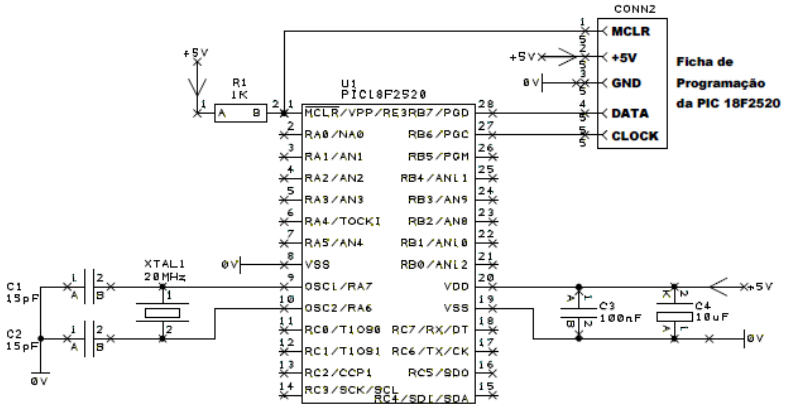


Figura 4-5: Esquemático do hardware do PIC18F2520

4.3.1 Conclusões

Concluiu-se que todos os microcontroladores PIC têm que ser alimentados com uma tensão VDD de 5V. Para reduzir ruídos e estabilizar a tensão na PIC, devem ser utilizados condensadores cerâmicos e electrolíticos, respectivamente.

Constatou-se ainda que a frequência de oscilação de *clock* do PIC depende do cristal, sendo associados igualmente condensadores, cujo valor deve estar de acordo com o recomendado no *data sheet* do PIC.

4.4 Displays

Neste projecto é necessário um *display*, para fazer o interface entre o utilizador e a máquina de encadernar *Hollowgrail*. O *display* deve ser capaz de indicar ao utilizador os passos a efectuar, assim como informá-lo do estado da máquina e de possíveis erros que surjam durante o processo de encadernação. Pretende-se um *display* gráfico que permita a projecção de imagens e texto, de tamanho a rondar os 60x40mm, aspecto agradável e baixo custo. Desta forma, fez-se um estudo sobre as diversas tecnologias de *displays*, para encontrar a solução que melhor se adequa ao presente projecto.

4.4.1 Tipos de displays e seu princípio de funcionamento

Foram várias as tecnologias de *display* analisadas: CRT (*Cathode Ray Tube*), ELD (*Electroluminescent Display*), *flip-dot*, incandescentes, LCD (*Liquid Crystal Display*), LEDs (*Light Emitting Diodes*), OLED (Organic LED) e PLED (*Polymer*

LED), tubos de Nixie, PDP (*Plasma Display Panel*) e VFD (*Vacuum Flourescent Displays*) [106].

Os CRT são, de entre as tecnologias de *displays*, a mais antiga. A sua ampla utilização em osciloscópios, televisores e computadores deve-se ao alto brilho, grande definição, amplo ângulo de visão e baixo custo. O facto de serem frágeis e pesados, bem como a necessidade de tensões de operação muito elevadas, tornam-no uma opção inviável para este projecto [106].

Nos *displays flip-dot*, os pixéis são um disco preto de um lado e fluorescente do outro. Esta característica leva à dificuldade de obter *displays* de pequenas dimensões, uma vez que torna complicada a obtenção de pixéis de dimensões reduzidas. Por outro lado, só necessita de energia na transição da imagem, o que permite que esta seja retida, mesmo quando desligado. Tudo isto, aliado ao seu baixo custo e grande durabilidade, torna-os ideais em grandes *outdoors*, mas impossibilita a sua utilização neste projecto [106].

Os *displays* de LEDs são constituídos por LEDs com formato de *pixéis* ou segmentos, que são alimentados quando se pretender apresentá-los. O facto de serem fontes pontuais de luz exige a montagem de diversos LEDs, quando se pretende formar imagens [106]. Os *displays* de LEDs, com formato de segmentos, utilizam-se em calculadoras e *displays* alfanuméricos, enquanto que os que possuem formato de *pixeis* são, normalmente, utilizados em grandes painéis informativos [107]. Atendendo à definição e dimensão pretendida para o *display*, esta solução não é a mais indicada.

Outros *displays* que foram desde início excluídos, por não serem capazes de projectar imagens, foram os *displays* incandescentes, tubos de Nixie e VFD. Todos estes *displays* são formados por um tubo de vidro contendo no seu interior filamentos. Nos *displays* incandescentes e VFD, estes tomam normalmente a disposição de sete segmentos. Os tubos de Nixie contêm dez ou mais cátodos individuais com o formato de números, símbolos ou letras [106].

a) b) c)

Figura 4-6: Tecnologias de *displays*: a) *flip-dot* [108]; b) tubos de Nixie [109]; c) LED [110]

Com o objectivo de escolher o *display* indicado, fez-se uma análise mais aprofundada dos *displays* com viabilidade de serem utilizados no presente projecto.

Display electroluminescente

Os ELD são *displays* do tipo emissivo, que emitem, portanto, a sua própria luz, pelo que podem ser vistos em ambientes com elevada luminosidade [111].

Estes *displays* são constituídos por uma camada de fósforo que emite luz, quando na presença de um campo eléctrico intenso. Essa camada encontra-se no meio de duas placas isolantes e de uma matriz de eléctrodos horizontais e verticais (Figura 4-7). Uma tensão aplicada aos eléctrodos provoca a emissão de luz na área de intercepção (*pixel*) [111].

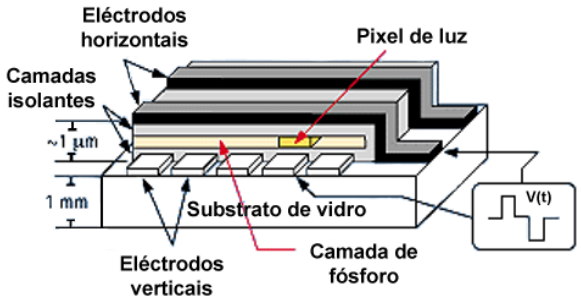


Figura 4-7: Estrutura do *display* electroluminescente [111]

Estes *displays* são, normalmente, monocromáticos e as placas de fósforo utilizadas produzem luz laranja, amarela ou verde [106]. Com o desenvolvimento da tecnologia associada às películas luminescentes, surgiram os *displays* EL multicolores, transparentes e de vidro [112].

Figura 4-8: *Displays* electroluminescentes [112]

As vantagens estão na capacidade de suportarem condições adversas como frio, calor, vento, poeiras, vibrações e luz solar. Além disso, obtêm-se imagens com elevado brilho e contraste, possibilitando ainda um amplo ângulo de visão [111]. Como desvantagens, têm o facto de necessitar de tensões elevadas, na ordem dos 150V. Estes *displays* são difíceis de fabricar a cores e em dimensões elevadas [106].

LCD

Os LCD são *displays* reflectivos, ou seja, não produzem luz, necessitando de uma fonte externa de iluminação [106].

Os LCDs são formados, como o próprio nome indica, por cristais líquidos. Estas substâncias têm a particularidade de poderem existir num estado transitório entre o sólido e o líquido, designado por estado líquido cristalino, que combina as propriedades dos estados sólido e líquido (Figura 4-9). Nos sólidos, a estrutura molecular tem ordenação posicional e orientacional, ao contrário dos líquidos que não têm qualquer tipo de ordenação. Um cristal líquido tem alguma ordenação orientacional, não tendo, no entanto, qualquer ordenação posicional [113].

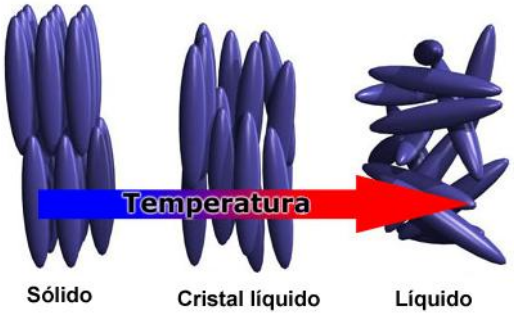
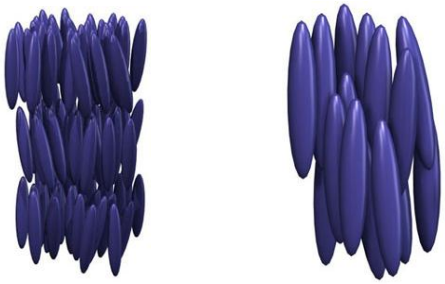


Figura 4-9: Comparação da estrutura molecular entre um sólido, um cristal líquido e um líquido [113]

Os cristais líquidos, dependendo da temperatura e da natureza da substância, podem ser do tipo esmético ou nemático. O primeiro, obtém-se a uma temperatura inferior, estando mais próximo do estado sólido, com moléculas compactadas em diversas camadas (Figura 4-10a). O cristal líquido nemático apresenta uma disposição unidimensional e aspecto menos viscoso (Figura 4-10b) [114].



a) b)

Figura 4-10: Tipos de cristais líquidos: a) esméticos; b) nemáticos [113]

Os LCDs usam cristais líquidos nemáticos e podem ser encontrados em diversas tecnologias, sendo a mais comum a TN (*Twisted Nematic*). Como se verifica pela observação da Figura 4-11, os cristais líquidos são colocados entre duas placas de vidro, que contêm polarizadores nas suas superfícies externas e eléctrodos nas internas. Uma fonte de luz fluorescente, habitualmente designada por *backlight*, é responsável pela emissão dos raios. Os polarizadores controlam os raios de luz que atravessam os cristais líquidos. Como as linhas de polarização de um estão dispostas perpendicularmente às do outro, as moléculas dos cristais líquidos são forçadas a um movimento rotacional, direccionando os raios de luz da mesma forma [115].

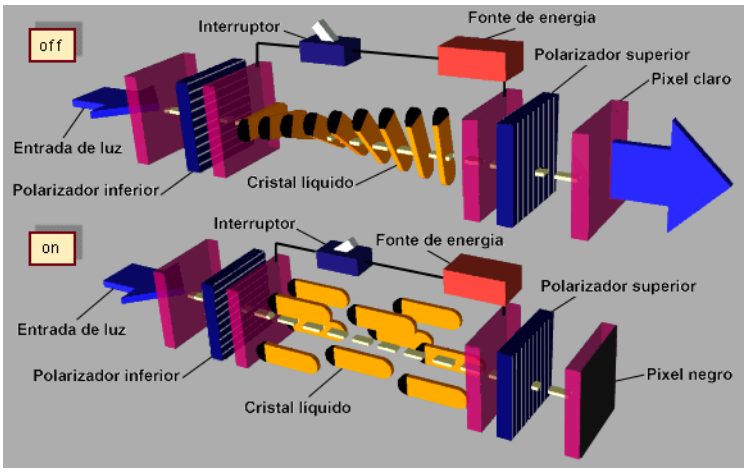
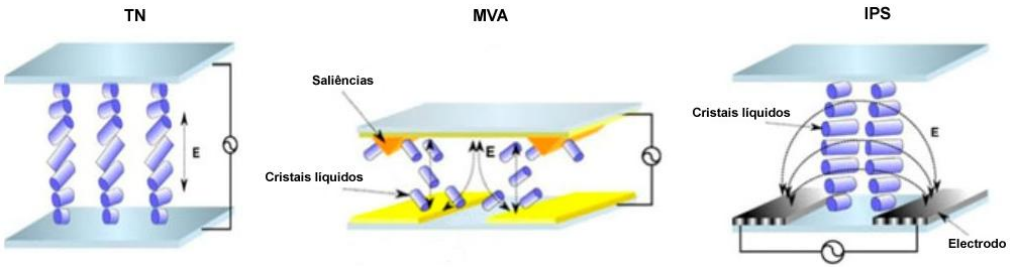


Figura 4-11: Funcionamento do LCD [116]

Enquanto não é aplicada tensão, a luz, ao atravessar o primeiro polarizador, é linearmente polarizada, sofrendo o movimento rotacional acima descrito. De seguida, atravessa o vidro e o polarizador frontal, atingindo finalmente o observador, que vê uma zona clara. Aplicando tensão entre os eléctrodos, surge um campo eléctrico que provoca o alinhamento vertical dos cristais líquidos na direcção do campo. Neste caso, o observador vê uma zona escura, dado que os cristais líquidos não provocam o movimento rotacional da luz, impedindo-a de atravessar o polarizador frontal [117].

Como forma de compensar limites de ângulos de visão da tecnologia TN, surgiram as tecnologias HTN (*High Twisted Nematic*), STN (*Super Twisted Nematic*), FSTN (*Film Compensated Super Twisted Nematic*) e CSTN (*Color Super Twisted Nematic*). Na tecnologia de *displays* TN, as moléculas de cristais líquidos, bem como os raios de luz, são sujeitos a uma rotação de 90º, originando um ângulo de visão reduzido. Esta tecnologia apresenta apenas uma cor, caracteres pretos sobre um fundo cinzento, tendo, no entanto, a vantagem de ter um custo mais reduzido. A tecnologia HTN é baseada numa rotação superior, aproximadamente 110º, oferecendo, desta forma, um ângulo de visão superior e um maior contraste. Na tecnologia STN, entre os polarizadores e os cristais líquidos, existe uma determinada inclinação, que faz com que o seu princípio de funcionamento, se diferencie, do existente na tecnologia TN. Este passa agora a ser baseado no princípio de dupla refracção. A posição dos polarizadores, a distância entre eles e a dupla refracção criada permitem obter diferentes cores de fundo, sendo a mais comum a verde. Pode encontrar-se esta tecnologia também com fundo prata e caracteres azuis ou pretos. Quando em fundo verde, os caracteres são de cor violeta ou preta. Esta tecnologia permite uma rotação entre os 180º e os 270º, o que leva a uma qualidade visual muito boa. O contraste encontra-se ao nível do obtido com a tecnologia TN. Na tecnologia FSTN, adiciona-se uma membrana ao *display* STN que permite obter uma imagem com o fundo branco, conseguindo-se, desta forma, um contraste e ângulo de visão superiores, sendo, no entanto, uma tecnologia cara. A tecnologia CSTN usa luz branca e filtros coloridos para obter a cor desejada. Cada pixel do *display* é formado por três *sub-pixeis* que possuem um filtro vermelho, verde e azul. Controlando cada *sub-pixel* separadamente, obtém-se a cor desejada [118].

Todas estas tecnologias derivam da TN, possuindo uma estrutura básica idêntica. Existem também as tecnologias IPS (*In-Plane Switching*) e MVA (*Multi-Domain Vertical Alignment*), que apresentam diferenças em relação à tecnologia TN e lhes concedem também melhores ângulos de visão e uma gama mais alargada de cores. Os IPS possuem dois eléctrodos para cada pixel, colocados em paralelo na mesma placa de vidro, permitindo um melhor ângulo de visão e cor. Nos MVA, os cristais líquidos são colocados perpendicularmente às placas de vidro que têm saliências em forma de pirâmide, para controlar a rotação dos cristais líquidos, concedendo-lhes um ângulo de visão mais amplo, maior contraste e respostas mais rápidas [119].



a) b) c)

Figura 4-12: Tecnologias de LCDs: a) TN; b) MVA; c) IPS [120]

Outra forma de caracterizar os LCDs é tendo em conta o modo como o brilho dos *pixeis* é controlado. A forma mais simples e com menor custo é designada por matriz passiva e consiste numa grelha de fios horizontais e verticais, onde a intersecção define um pixel. Um *pixel* é activado, quando é aplicada uma tensão a uma das colunas e a massa a uma das linhas. O cruzamento da linha e da coluna no *pixel* origina a tensão necessária para realizar a rotação dos cristais líquidos no *pixel*. Esta solução também apresenta desvantagens: um tempo de resposta lento, que provoca uma renovação lenta da imagem e um controlo de tensão impreciso, que permite que os pixéis adjacentes possam também ser influenciados, provocando uma imagem deturpada. A outra forma de controlar o brilho dos *pixeis* designa-se por matriz activa e recorre a um transístor TFT (*Thin-Film Transistor*), para alimentar separadamente os *pixeis*. Consegue-se, assim, que a corrente de alimentação dos *pixeis* seja inferior, podendo ser ligada ou interrompida mais rapidamente, obtendo-se desta forma uma resposta mais rápida [115].

Inicialmente, os LCDs eram aplicados essencialmente em relógios e máquinas de calcular, tendo, com o tempo, vindo a ser largamente utilizados em novas aplicações, como aparelhos de televisão, computadores, câmaras digitais, telemóveis, entre outros.

Os LCD têm como vantagens a possibilidade de serem fabricados em tamanho reduzido, a baixo custo e poderem gerar imagens monocromáticas ou coloridas de qualquer formato ou tamanho, com excelente definição. Além disso, possuem um baixo consumo e são alimentados com tensões muito reduzidas [106].

As desvantagens estão no facto de necessitarem de uma alimentação alternada e uma iluminação posterior ou frontal, sendo que na iluminação posterior é necessário ter em conta o seu custo, consumo e vida útil. Outros problemas que estes *displays* acarretam são o facto de serem frágeis, terem um ângulo de visão relativamente pequeno e o funcionamento limitado a uma reduzida gama de temperaturas, obrigando à utilização de circuitos de compensação, para que as imagens não sejam afectadas [106].

OLED e PLED

Os OLED e os PLED são *displays* emissivos. A principal diferença está no processo de produção e nos materiais utilizados. O OLED usa pequenas moléculas de material orgânico e o PLED usa moléculas de polímeros. Estes materiais são formados por várias camadas, que são depositadas em substratos de vidro ou flexíveis e baseiam o seu funcionamento em pequenas moléculas orgânicas, ou de polímeros que emitem luz quando são atravessadas por corrente [106].

A estrutura típica dos OLED (Figura 4-13) consiste em três camadas orgânicas entre os eléctrodos. A camada orgânica adjacente ao cátodo é a camada de transporte de electrões e a adjacente ao ânodo é a camada de transporte de lacunas. A camada de emissores orgânicos encontra-se no meio e, como o próprio nome indica, é constituída por vários tipos de materiais orgânicos que produzem luz vermelha, verde e azul [121].

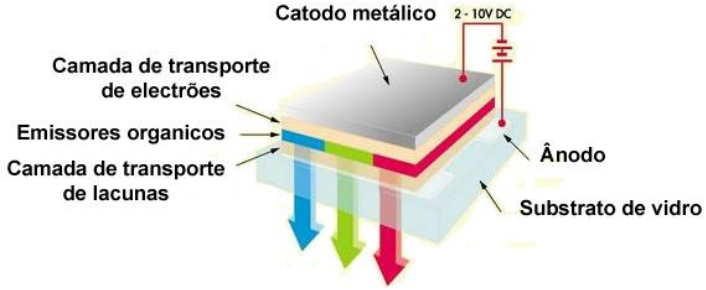


Figura 4-13: Estrutura dos OLED [122]

A principal vantagem deste *display* está no facto de poder ser fabricado com substratos flexíveis, o que, para além de ser algo inédito, possibilita que os *dispays* possam ser enrolados ou mesmo dobrados. O amplo ângulo de visão, a óptima qualidade da cor e a melhor velocidade de resposta, quando comparada com a dos LCD, leva a que esta tecnologia comece a ser bastante popular em muitas aplicações. As desvantagens prendem-se com o facto de possuírem uma vida limitada e um consumo elevado [106].

Figura 4-14: OLED de substrato flexível [123]

Embora seja uma tecnologia muito recente, começa já a ser bastante usada em aplicações comerciais de pequena dimensão como em ecrãs para telemóveis e mp3, auto-rádios e câmaras digitais [124].

Plasma

Os plasmas são *displays* emissivos [106], formados por muitas cavidades pequenas, contendo xénon e néon entre duas placas de vidro. Longos eléctrodos são também colocados entre as duas placas, dos dois lados das cavidades. Os eléctrodos de endereço colocam-se atrás das cavidades, ao longo da placa de vidro posterior. Os eléctrodos de *display*, colocados sobre uma camada de óxido de magnésio, são cobertos por um material isolante sobre o qual é aplicada a placa de vidro frontal [125].

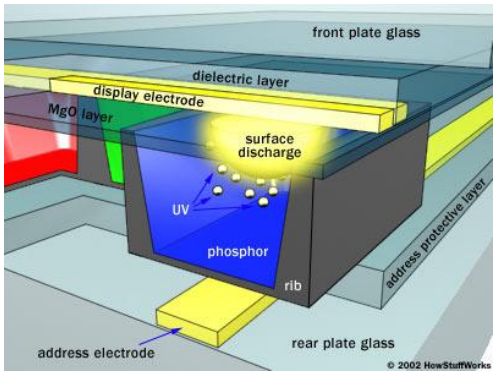


Figura 4-15: Estrutura de um *sub-pixel* do plasma [125]

Ambos os conjuntos de eléctrodos estendem-se ao longo do ecrã, formando uma grelha. Os eléctrodos de *display* são dispostos horizontalmente e os de endereço verticalmente [125].

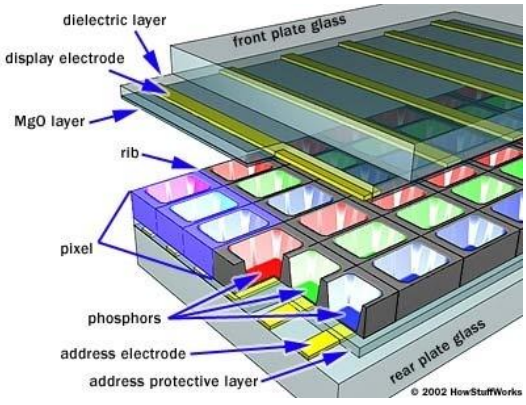


Figura 4-16: Estrutura de um plasma [125]

Quando se aplica uma tensão aos eléctrodos, flui uma corrente através do gás na cavidade, provocando a libertação de fotões ultravioleta que, ao interagirem com o fósforo que reveste o interior da cavidade, origina a emissão de luz. Cada *pixel* é

constituído por três cavidades separadas (*sub-pixel*), cada uma com a sua coloração, formando tríades RGB. A mistura da luz radiada pelos *sub-pixeis* dá como resultante a radiação de uma cor para o pixel, podendo abranger todo o espectro luminoso através do controlo da intensidade de cada *sub-pixel* [125].

A grande vantagem destes *displays* reside na possibilidade de construir ecrãs de grandes dimensões com materiais extremamente finos. O facto de cada pixel ser iluminado individualmente, faz com que a imagem seja muito brilhante e o ângulo de visão elevado [125].

Como desvantagens tem-se o facto de possuírem uma vida útil curta (5000 a 20000 horas), necessitarem de elevadas tensões de alimentação (na ordem dos 150V), poderem aparecer manchas ou desgaste localizado em áreas que operam por demasiado tempo, não possuírem grandes resoluções e terem preços elevados [106] [125].

4.4.2 Display utilizado

De entre as diversas soluções estudadas, optou-se pelos LCDs, tendo em conta que a sua utilização intensiva em muitas aplicações, faz com que o seu custo no mercado seja baixo, quando comparado com as outras alternativas.

O *display* escolhido foi o LCD module SF12L64DLGB-E da *SDEC*. É do tipo FSTN, possui 128x64 *pixeis*, tem cor de polarização cinzenta e de *backlight* azul. O módulo inclui o controlador RA8863 da *RAiO* que é compatível com o T6963C da *Toshiba*.

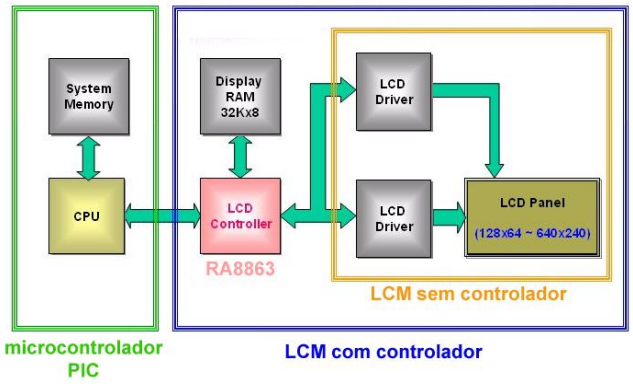


Figura 4-17: LCD e controlador RA8863 [126]

A Figura 4-17 apresenta um diagrama de blocos de um LCD, onde se destaca o painel LCD, o seu controlador e o microcontrolador PIC. O PIC é programado de acordo com instruções específicas do controlador que depois as descodifica e transmite ao LCD. Este controlador, tal como o T6963C, foi projectado para ser usado com

drivers de controlo e memória de dados (Figura 4-17). Possui 256Bytes de memória ROM e uma RAM externa de 64kBytes. O interface com o PIC é feito através de um barramento de 8 bits de dados e linhas de controlo, para realizar a leitura ou a escrita.

4.4.3 Hardware

O circuito necessário para o controlo do *display* é bastante simples. É constituído pelo PIC18F2520, que é responsável por enviar para o controlador RA8863 as instruções necessárias para o seu funcionamento, e um potenciómetro para regular a tensão de alimentação do LCD (V_{EE}). A tensão para alimentação do controlador RA8863 e dos LEDs usados para luz de *backlight* é de 5V.

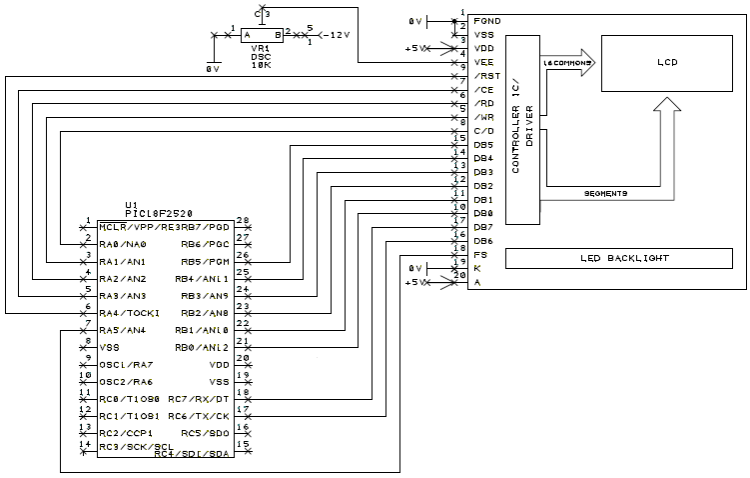


Figura 4-18: Esquemático do hardware associado ao LCD

Como se verifica, o barramento entre o PIC e o controlador RA8863 é constituído pelas linhas de controlo de escrita (/WR), leitura (/RD), *enable* (/CE), comando ou dado (C/D), *reset* (/RST), tamanho da letra (FS) e por 8 bits de dados (DB0:DB7).

4.4.4 Software

Depois de definido o circuito para controlar o *display*, é importante ter conhecimento da lista de comandos do controlador e dos algoritmos necessários à escrita de caracteres ou imagens.

Começa-se por inicializar o controlador RA8863, sendo, para tal, necessário fazer o *reset* e configurar alguns comandos, tais como o *mode set*, *control word set* e *display mode*, como se pode ver na Figura 4-19.

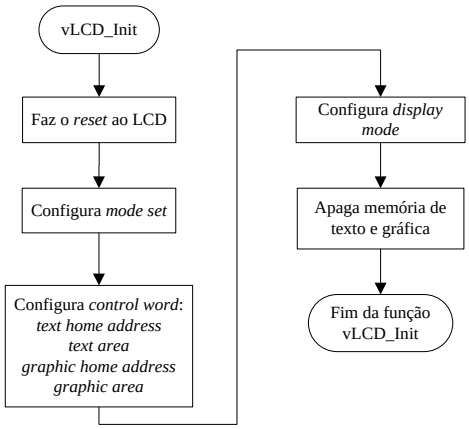


Figura 4-19: Algoritmo da função vLCD_Init()

O *mode set* é o comando responsável por definir o modo como o texto e os gráficos são apresentados, podendo ser uma combinação dos dados segundo a lógica OR, EXOR ou AND (Figura 4-20).

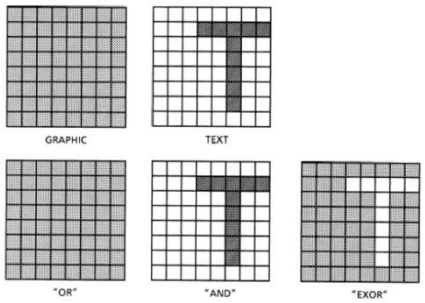


Figura 4-20: Combinação do texto e dos gráficos segundo a lógica OR, AND e EXOR [127]

Caso não se pretenda usar gráficos, pode seleccionar-se o modo TEXT ATRIBUTE, que funciona apenas com texto e possui os dados definidos na memória gráfica. Neste comando também é possível definir se os caracteres de texto vêm do gerador de caracteres da ROM (*Internal character generator mode*) ou da RAM (*External character generator mode*). A Tabela 4-1 apresenta o código para cada uma das funções do comando *mode set*.

Função	**Código**
OR	1000X000
EXOR	1000X001
AND	1000X011
TEXT ATRIBUTE	1000X100
Internal character generator	10000XXX
External character generator	10001XXX

Tabela 4-1: Funções do comando *mode set* e seus códigos

Para este trabalho definiu-se o modo OR em conjunto com o *Internal character generator*.

No *control word*, é importante definir o *text home address*, o *graphic home address*, o *text area* e o *graphic area*. Para definir estes comandos, é necessário ter em conta que estamos perante um LCD de 128x64 *pixeis* onde o tamanho da letra é de 8x8 *pixeis*.

O *text home address*, endereço da memória RAM onde o texto começa a ser guardado, foi inicializado em 0x0000H. O *text area* representa o número de colunas que é possível obter para um determinado tamanho de letra, sendo o seu cálculo realizado da seguinte forma:

$$text\ area = \frac{pixeisX}{caracterPixeisX} = \frac{128}{8} \tag{4-1}$$

Para definir o *graphic home address*, é preciso primeiro calcular o tamanho de memória necessária para guardar o texto apresentado no *display* (*text size*), obtendo-se esse valor através da equação (4-2).

$$text\ size = \frac{pixeisX}{caracterPixeisX} \cdot \frac{pixeisY}{caracterPixeisY} = \frac{128}{8} \cdot \frac{64}{8} = 128\ Bytes \tag{4-2}$$

Apesar de só serem necessários 128Bytes, reservou-se 1kBytes, o que permite, caso necessário, reduzir a dimensão dos caracteres. Desta forma, inicializou-se em 0x0400H o *graphic home address*, que representa o endereço da memória RAM onde as imagens começam a ser guardadas. O *graphic area* é calculado da mesma forma que o *text area*, uma vez que, ao nível das colunas, não há variação no número de *pixeis* utilizados para o texto e para os gráficos. O *graphic size* é calculado da seguinte forma:

$$graphic\ size = \frac{pixeisX}{caracterPixeisX} \cdot pixeisY = \frac{128}{8} \cdot 64 = 1024\ Bytes \tag{4-3}$$

A configuração destes comandos implica a colocação do código e do endereço (nos casos do *text* e *graphic home address*) ou do código e do número de colunas (nos casos do *text* e *graphic area*), como se vê na Tabela 4-2.

Função	**Código**	**D1**	**D2**
Text home address	01000000	Endereço LSB	Endereço MSB
Text area	01000001	Número de colunas	00H
Graphic home address	01000010	Endereço LSB	Endereço MSB
Graphic area	01000011	Número de colunas	00H

Tabela 4-2: Funções do comando *control word set*

O *display mode* é o comando responsável por definir o estado do modo de texto e do modo gráfico, assim como o estado do cursor e se este é contínuo ou intermitente. A Tabela 4-3 apresenta as funções referidas e os códigos correspondentes.

Função	Código
Cursor off	1001XX00
Cursor on, blink off	1001XX10
Cursor on, blink on	1001XX11
Text on, graphic off	100101XX
Text off, graphic on	100110XX
Text on, graphic on	100111XX

Tabela 4-3: Funções do comando *display mode*

Como se pretende utilizar texto e imagens no LCD, escolheu-se o modo *text on, graphic on*, sem o uso de cursor (*cursor off*).

Conhecidos os comandos necessários para inicializar o controlador RA8863, verifica-se que, para definir o *mode set* e o *display mode*, é necessário escrever apenas um código de 1Byte no controlador, ao contrário do comando *control word set* onde é necessário escrever, para além do código de 1Byte, 2Bytes contendo o endereço ou o número de colunas. Desta forma, tornou-se essencial definir algoritmos capazes de escrever um comando, um dado ou um endereço no controlador RA8863.

Na escrita ou na leitura do barramento de dados do controlador, é importante seguir uma ordem e um tempo específicos para activar e desactivar as entradas de controlo do *display*. Na Figura 4-21, pode ver-se o diagrama temporal onde estas especificações estão representadas e na Tabela 4-4 encontram-se os valores correspondentes.

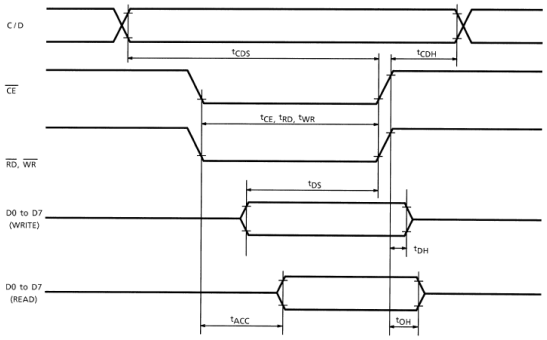


Figura 4-21: Diagrama temporal da escrita ou leitura no LCD

Tempos	Símbolo	Min	Tip	Max	Unidades
C/D Set Up Time	tCDS	100	-	-	ns
C/D Hold Time	tCDH	10	-	-	ns
/CE, /RD, /WR Pulse Width	tCE,tRD,tRW	80	-	-	ns
Data Set Up Time	tDS	80	-	-	ns
Data Hold Time	tDH	40	-	-	ns
Acess Time	tACC	-	-	150	ns
Output Hold Time	tOH	-	-	50	ns

Tabela 4-4: Tempos do diagrama temporal

A escrita no controlador deve ser precedida por um teste à capacidade de execução de um comando, escrita ou leitura de um dado. Para tal, é necessário ler e testar o bit0 e o bit1 do barramento de dados do controlador, através da função *vLCD_BusyCheck()*, cujo algoritmo é o seguinte:

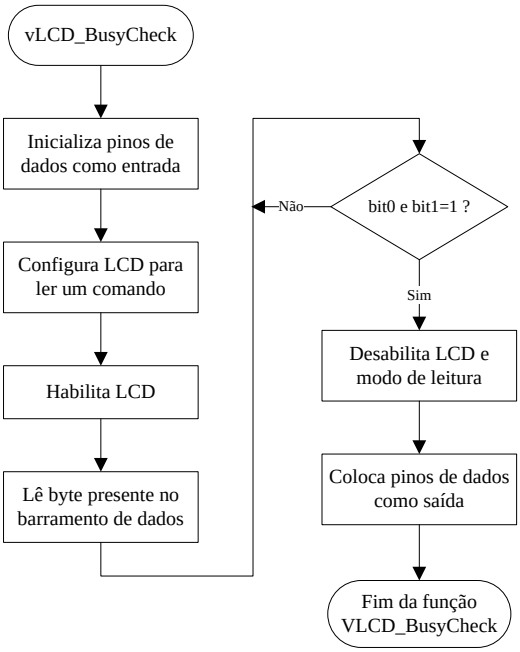


Figura 4-22: Algoritmo da função *vLCD_BusyCheck()*

A função *vLCD_Write(hInstruction, bCD)* (Figura 4-23) permite escrever um comando ou um dado no controlador do LCD. Tem como parâmetros de entrada a instrução que se pretende escrever e uma variável que indica se ela é um comando ou um dado.

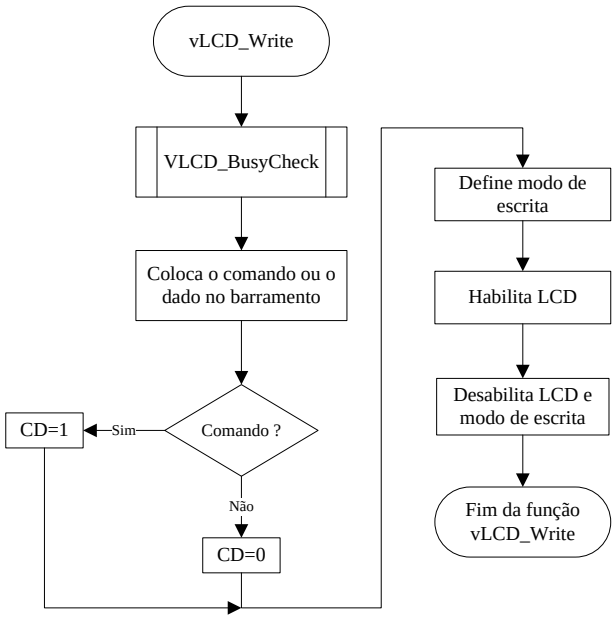


Figura 4-23: Algoritmo da função *vLCD_Write(hInstruction, bCD)*

A função *vLCD_WriteAddr(hAddr, hCmd)* (Figura 4-24) permite escrever um endereço no controlador do LCD, como o usado nos comandos *text* e *graphic home address*. Os seus parâmetros de entrada são o endereço e uma instrução de comando com o código relativo ao endereço.

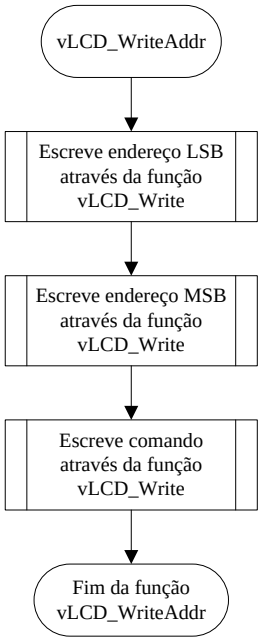


Figura 4-24: Algoritmo da função *vLCD_WriteAddr(hAddr, hCmd)*

A função *vLCD_WriteChar(hCaracter)*, que pode ser vista na Figura 4-25, escreve um caracter no ecrã do LCD.

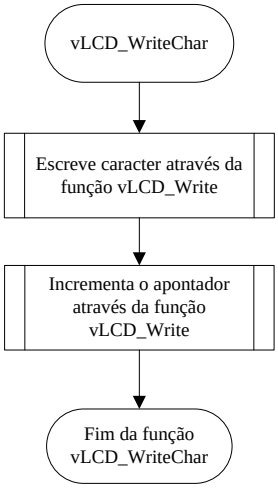


Figura 4-25: Algoritmo da função *vLCD_WriteChar(hCaracter)*

No RA8863, um caracter não tem o mesmo valor do ASCII (*American Standard Code for Information Interchange*), ou seja, para escrever o caracter ‘a’ do ASCII no RA8863 faz-se ‘a’-20h. A escrita de um caracter deve ser seguida do incremento,

permanência ou decremento do apontador, que indica a posição de escrita na memória, Tabela 4-5.

Função	Código
Data write and address pointer increment	11000000
Data write and address pointer decrement	11000010
Data write and address pointer nonvariable	11000100

Tabela 4-5: Funções para escrita de um dado

Para terminar a inicialização do *display*, Figura 4-19, é ainda necessário apagar as memórias de texto e gráfica, de forma a não aparecerem caracteres indesejáveis no ecrã, é o que fazem as funções *vLCD_ClearText()* e *vLCD_ClearGraphic()*.

Nestas funções, bem como nas de escrita de texto ou imagens no *display*, é necessário configurar o comando *address point*, que indica o endereço da memória onde se pretende iniciar a escrita. O seu formato pode ser observado na Tabela 4-6.

Função	Código	D1	D2
Address point	00100100	Endereço LSB	Endereço MSB

Tabela 4-6: Formato do comando *address point*

Sendo assim, as funções *vLCD_ClearText()* e *vLCD_ClearGraphic()* recorrem ao comando *address point*, para colocar o apontador na primeira posição de cada memória, correndo, de seguida, todas as suas posições e limpando o ecrã.

Depois de inicializado, o *display* está pronto para apresentar texto ou imagens. A apresentação de uma frase no *display*, com os caracteres fornecidos pelo próprio controlador, pode ser feita seguindo o algoritmo da Figura 4-26.

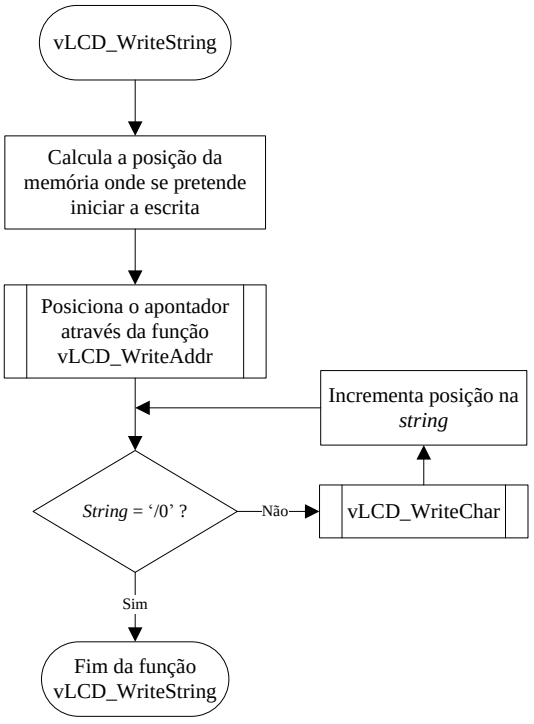


Figura 4-26: Algoritmo da função *vLCD_WriteString(iX, iY, sPhrase)*

A função *vLCD_WriteString(iX, iY, sPhrase)* tem como parâmetros de entrada a coluna e a linha do *display* onde se inicia a escrita, bem como a frase que se pretende escrever. Recorrendo à coluna (X) e à linha (Y), é possível calcular, pela equação (4-4), a posição inicial de escrita na memória.

$$address = text\ home\ address + Y \cdot text\ area + X \tag{4-4}$$

Para apresentar imagens no *display*, é necessário obter o seu valor em linguagem decimal, tendo-se feito um programa que converte uma imagem no formato *bitmap* em linguagem decimal. Na Figura 4-27, pode ver-se o algoritmo utilizado.

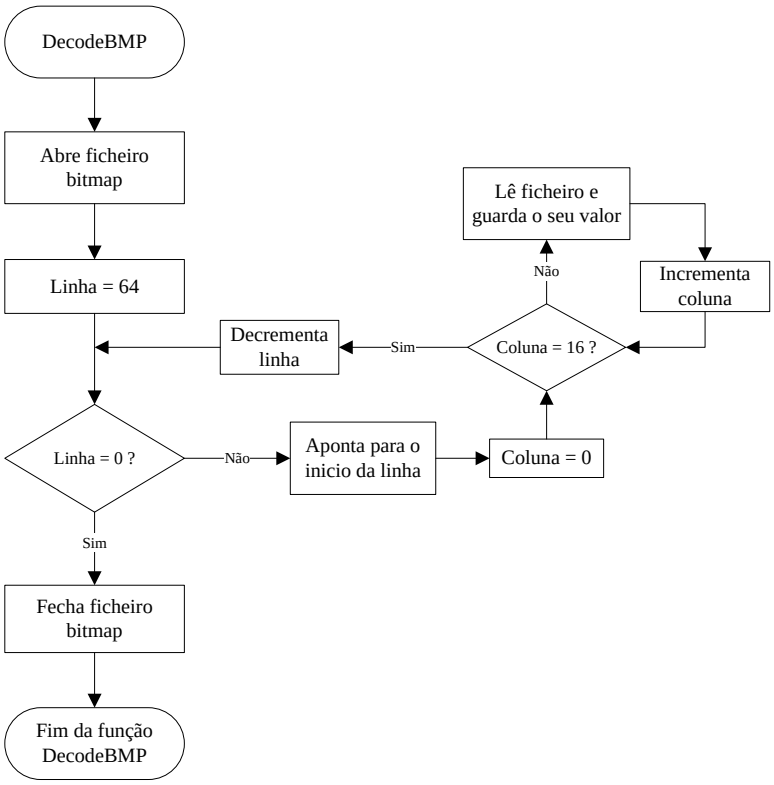


Figura 4-27: Algoritmo para converter um *bitmap* em linguagem decimal

Guardaram-se os valores decimais da imagem no ficheiro bitmap.h e desenvolveu-se uma função, *vLCD_WriteBitmap(iX, iY, iBitmapX, iBitmapY, vBitmap)*, capaz de os converter numa nova imagem para ser apresentada no *display*. Esta função tem como parâmetros de entrada a coluna e a linha onde se pretende começar a desenhar a imagem, o número de colunas e linhas da imagem, e um *array* onde estão guardados os valores decimais da imagem. A posição inicial de escrita na memória é também calculada pela equação (4-4). Na Figura 4-28, pode ver-se o seu algoritmo.

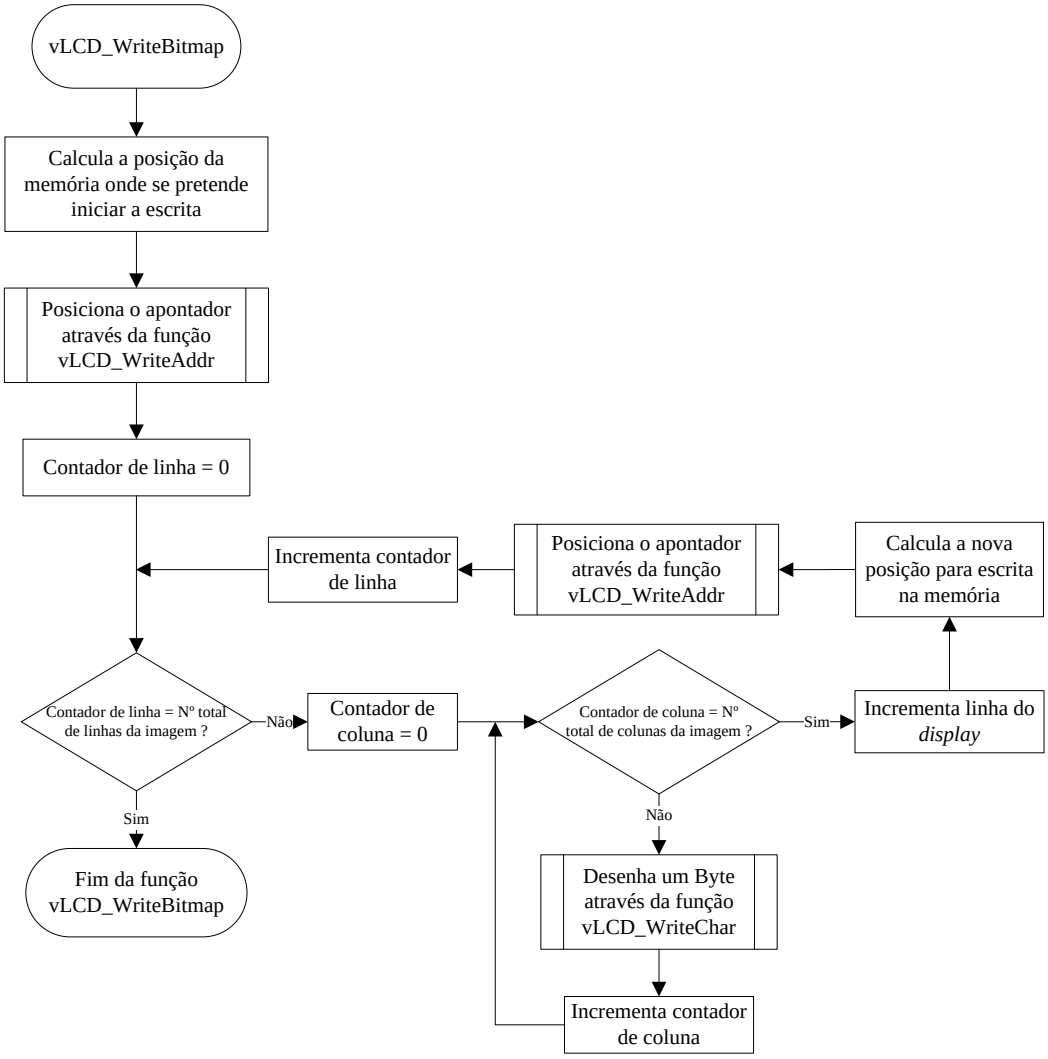


Figura 4-28: Algoritmo da função *vLCD_WriteBitmap(iX, iY, iBitmapX, iBitmapY, vBitmap)*

Em virtude do tipo de letra pretendido pelo fabricante da máquina não ser standard e possuir um design específico, obrigou a que cada caracter fosse tratado como uma imagem. Na Figura 4-29 está representado o tipo de letra a utilizar.

Figura 4-29: Tipo de letra utilizada

Recorreu-se ao Excel para converter cada um dos caracteres em linguagem decimal. Ao desenho inicial com 0 e 1, fez-se corresponder um outro, cujo valor da célula é dado por:

$$valor\ da\ célula \cdot 2^{nº\ coluna} \tag{4-5}$$

Somaram-se todos os valores de uma linha, obtendo-se um valor que corresponde ao valor decimal da linha, que foi guardada num *array* no ficheiro bitmap.h. Desenvolveram-se as funções *vLCD_WritePixel(iX, iY)* e *vLCD_WriteWord(iX, iY, sWord)*, para poder apresentar os diferentes caracteres no *display*.

A primeira função escreve um pixel no ecrã e tem como parâmetros de entrada a coluna e a linha onde se pretende escrever. Na Figura 4-30, pode ver-se o seu algoritmo.

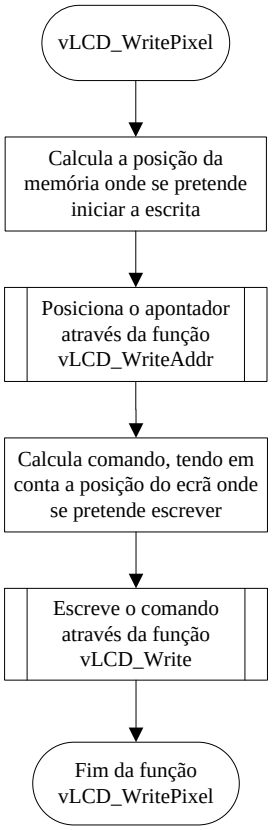


Figura 4-30: Algoritmo da função *vLCD_WritePixel(iX, iY)*

O facto de o controlador funcionar com 8bits no tratamento das imagens, faz com que seja necessário recorrer a um comando, quando se pretende desenhar ou apagar um dos 8bit, Tabela 4-7.

Função	**Código**
bit reset	11110XXX
bit set	11111XXX
bit 0	1111X000
bit 1	1111X001
bit 2	1111X010
bit 3	1111X011
bit 4	1111X100
bit 5	1111X101
bit 6	1111X110
bit 7	1111X111

Tabela 4-7: Funções para desenhar ou apagar um bit

Este comando necessita ser calculado e depende do número de *pixeis* do caracter (*caracterPixeisX*) e do local onde se pretende escrevê-lo. O cálculo utilizado pode ser visto na equação (4-6). Utilizou-se o código *bit set*, uma vez que se pretende escrever um bit e não apagá-lo.

$$Comando = bitSet \bigvee\left(caracterPixeisX - 1 - (X\%caracterPixeisX)\right) \tag{4-6}$$

A segunda função (Figura 4-31) escreve uma frase no ecrã com o tipo de letra definido para este trabalho. Como parâmetros de entrada, tem-se, novamente, a posição do ecrã, onde se pretende escrever, e a correspondente frase.

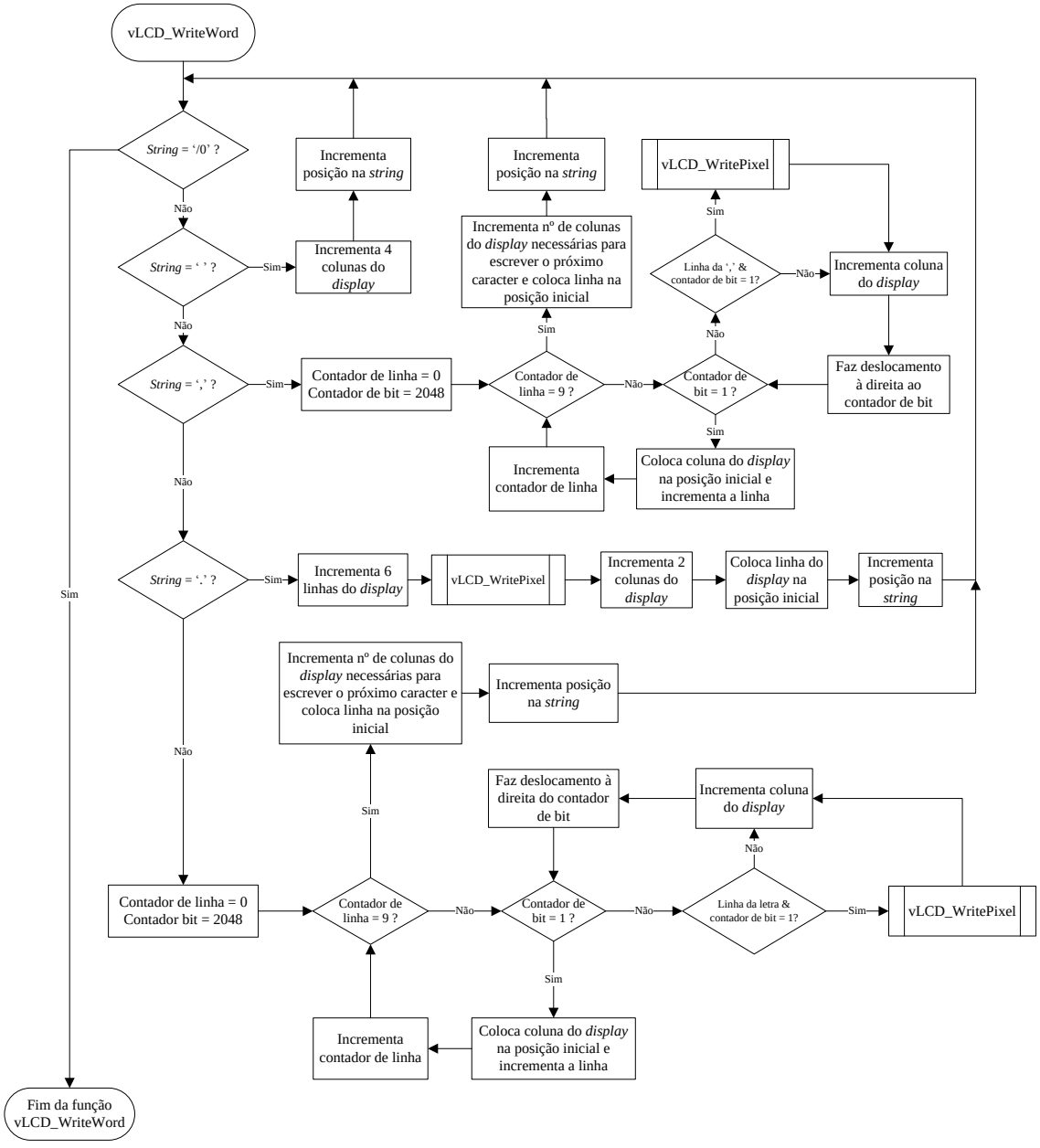


Figura 4-31: Algoritmo da função *vLCD_WriteWord(iX, iY, sWord)*

São várias as imagens a apresentar no *display*. Encontram-se definidas na função *vLCD_Words()* e podem ser vistas no Anexo II. Cada imagem apresenta um estado da máquina ou informa a operação que o utilizador deve realizar.

Na *main* do ficheiro lcd.c chamam-se as funções aqui descritas. Primeiramente, inicializam-se as variáveis, o LCD, o I^2C e as interrupções. De seguida, num ciclo infinito, apaga-se a memória de texto e a gráfica e desenha-se a imagem no LCD. Na Figura 4-32, pode ver-se o algoritmo utilizado.

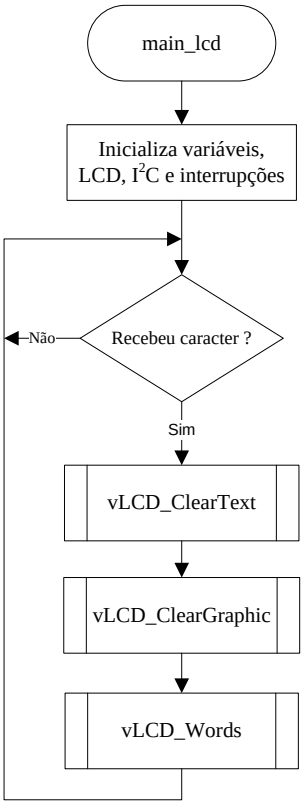


Figura 4-32: Algoritmo da *main* do ficheiro lcd.c

4.4.5 Conclusões

Concluiu-se que existe uma grande variedade de *displays*, com algumas características específicas que os diferenciam e os tornam mais recomendados para determinadas aplicações.

Neste caso, optou-se pelos LCDs tendo em conta o seu baixo custo, para a dimensão e definição desejadas.

Para fazer o controlo do *display*, foi necessário desenvolver um hardware e um software apropriado, tendo em conta a necessidade de comandar o controlador do LCD.

4.5 Sensores

Sensores são dispositivos que alteram o seu comportamento pela acção de uma grandeza física, respondendo directa ou indirectamente, com um sinal que é transmitido a um dispositivo de medição ou controlo. Nem sempre é possível obter sensores que meçam directamente uma grandeza, utilizando-se, nestes casos, um dispositivo intermédio capaz de alterar as suas propriedades, como a resistência, a capacidade ou a indutância e, desta forma, converter a grandeza a medir noutra já mensurável por um sensor, trabalhando assim os sensores de forma indirecta [128][129].

4.5.1 Tipos de sensores e seu princípio de funcionamento

Há uma imensa variedade de sensores que detectam uma vasta gama de formas de energia, podendo ter as seguintes classificações: digitais ou analógicos, passivos ou activos, podem ainda classificar-se, pelo tipo de sinal que transformam ou pela sua aplicação. Podem ser utilizados em diversas áreas como a medicina, indústria e robótica.

Os sensores digitais são utilizados em aplicações onde se pretende fazer contagens ou detectar a presença ou ausência de uma peça. Os sensores analógicos são mais complexos, mas permitem a obtenção de mais informação sobre um determinado processo, uma vez que a saída pode variar gradualmente dentro dos limites da escala [130].

Os sensores passivos requerem uma fonte de energia exterior, ao contrário dos activos [131].

Tendo em conta o tipo de sinal que transformam, os sensores podem ser mecânicos, eléctricos, ópticos, magnéticos, térmicos, acústicos, entre outros [131].

Analisando as diferentes aplicações dos sensores, eles podem ser classificados como de temperatura, de luz, de velocidade, de fluxo, de posição, entre outros [128].

No corrente trabalho são necessários sensores para detectar os fins de curso e o estado de alguns mecanismos da máquina, pelo que se optou por fazer uma análise aos diferentes sensores de posição, verificando-se que podem ser mecânicos, magnéticos ou ópticos.

Sensores mecânicos

Sensores mecânicos são sensores que quando sujeitos a uma força abrem ou fecham um contacto. Utilizam-se para detectar se foi ou não atingida uma posição, um fim de curso ou um obstáculo. Os sensores mecânicos mais conhecidos são os *micro-switch* (Figura 4-33), que existem em formatos, dimensões e sensibilidades muito diferentes [132].

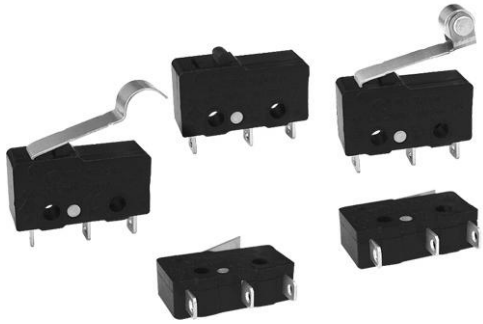


Figura 4-33: Vários formatos de *micro-switch* [133]

Sensores magnéticos

Sensores magnéticos, são sensores que alteram o seu comportamento quando sujeitos a um campo magnético. O sensor magnético mais comum é o *reed-switch* (Figura 4-34), que é formado por lâminas ferromagnéticas dentro de um invólucro de vidro com gás inerte, que tem como função evitar a oxidação dos contactos. Estas lâminas quando sujeitas a um campo magnético movimentam-se, fazendo abrir ou fechar um circuito. Outro modelo de sensor magnético é uma simples bobina, sujeita à passagem de um objecto que produza ou altere o valor do campo magnético. Essa alteração pode ser amplificada e utilizada para controlar circuitos externos. Existem ainda os sensores de efeito de *Hall*, que são o resultado da variação da resistência num semicondutor em função do campo magnético presente [132].

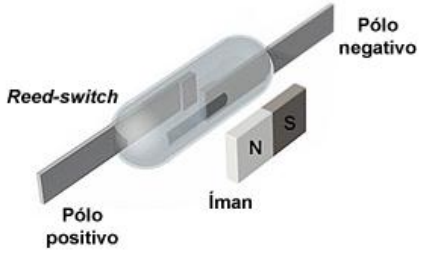


Figura 4-34: *Reed-switch* [134]

Sensores ópticos

Os sensores ópticos utilizam a alteração das características de alguns elementos, quando expostos à luz. Os mais vulgares utilizam fotodíodos ou fototransistores (Figura

4-35) que alteram a sua condutibilidade, quando expostos a radiações de determinadas frequências. Normalmente, são utilizadas frequências incluídas no espectro das radiações infra-vermelhas. Os fototransistores têm, geralmente, maior sensibilidade e maior tempo de resposta que os fotodíodos [132].

Figura 4-35: Fototransistor [135]

4.5.2 Hardware

Tanto o processo de furação como o de encadernação necessitam recorrer a diversos sensores, visto serem vários os mecanismos onde é importante detectar posições, fins de curso e estados.

O processo de furação é precedido pela detecção, indicação e confirmação do tamanho do documento a encadernar e da argola a utilizar.

A detecção de folhas na mesa é realizada por um *micro-switch*, como se verifica na Figura 4-36. Denominado por *sensor_paper*, este sensor atinge o nível lógico "1" na presença de folhas.

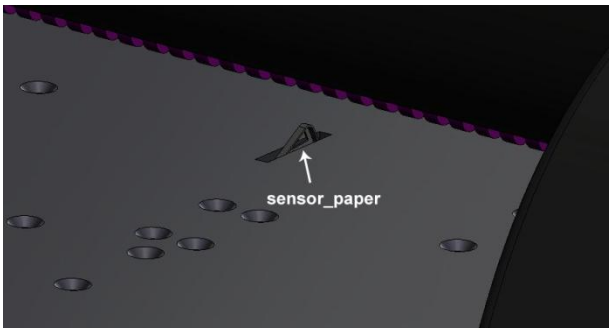


Figura 4-36: Sensor utilizado na detecção de folhas para encadernar

O interface entre a máquina e o utilizador é realizado pelo *display* gráfico e cinco *tact-switch* (Figura 4-37). O início de todo o processo é ditado pelo *sensor_start*. O *sensor_stop* permite interromper o processo, qualquer que seja o seu estado. Os sensores *sensor_ok*, *sensor_right* e *sensor_left* permitem navegar no *display* e confirmar as indicações que ele fornece.

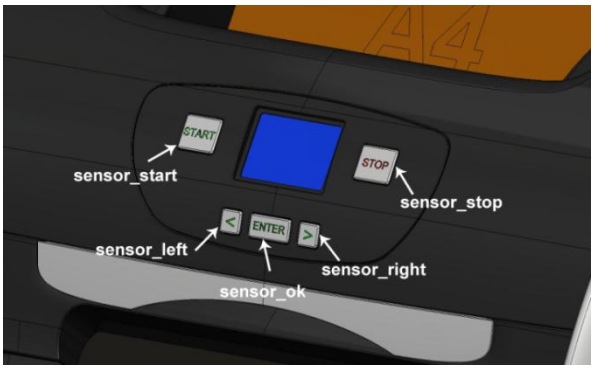


Figura 4-37: Sensores utilizados para o comando da máquina

No capítulo do estado da arte referente às argolas *clickbind*, verificou-se que estas existem em três tamanhos, para serem aplicadas em função do número de folhas que se pretende encadernar. Sendo assim, o tamanho do documento é essencial para seleccionar a argola mais adequada a cada encadernação. Utilizaram-se dois *micro-switch*, representados na Figura 4-38 por *sensor_sizeClickA* e *sensor_sizeClickB*, para identificar o tamanho do documento. Na Tabela 4-8, estão definidos os níveis lógicos dos sensores para os três tamanhos de documentos possíveis.

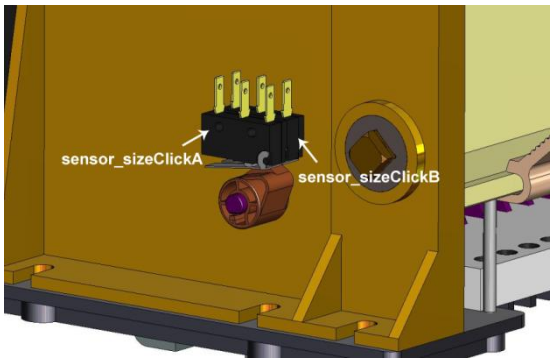


Figura 4-38: Sensores utilizados para identificar o tamanho do documento a encadernar

Tamanho do documento	**sensor_sizeClickA**	**sensor_sizeClickB**
Pequeno	0	0
Médio	0	1
Grande	1	1

Tabela 4-8: Estado dos sensores para os diferentes tamanhos do documento

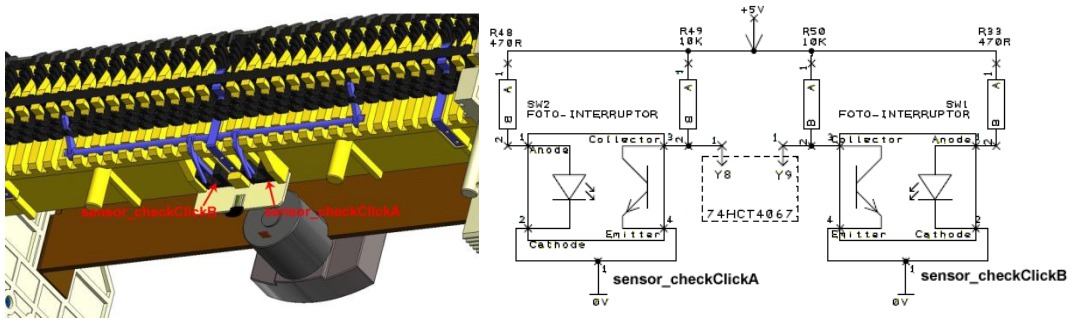
Na detecção e confirmação do tamanho da argola, utilizam-se também dois sensores, representados na Figura 4-39 por *sensor_checkClickA* e *sensor_checkClickB*. Os níveis lógicos que os sensores podem apresentar, em função do tamanho da argola, são os apresentados na Tabela 4-9.

Tamanho do *clickbind*	**sensor_checkClickA**	**sensor_checkClickB**
Ausência	0	0
Pequeno	0	1
Médio	1	0
Grande	1	1

Tabela 4-9: Estado dos sensores para os diferentes tamanhos da argola

Quando a argola é aplicada no mecanismo de fecho do *clickbind*, faz rodar um dispositivo (Figura 4-39a) que interrompe o feixe existente num ou nos dois sensores ópticos, em função da rotação verificada. Esta rotação depende da dimensão do *clickbind*.

Os dois sensores ópticos são constituídos por um LED infra-vermelhos e um fototransístor (Figura 4-39b). As resistências são utilizadas como forma de reduzir a corrente, tanto no LED como no fototransístor. Para o LED foi utilizada uma resistência de 470Ω, obtendo-se uma corrente na ordem dos 6mA, que é suficiente para que o seu feixe obtenha o alcance desejado.



a) b)

Figura 4-39: Sensores utilizados na detecção e confirmação do tamanho da argola: a) Desenho mecânico; b) Esquemático

O processo de furação é constituído por dois passos fundamentais, o movimento do bloco de furação em direcção às folhas e o deslocamento horizontal da mesa.

O primeiro deslocamento da mesa é realizado em direcção ao sensor *sensor_stepLeft* (Figura 4-40), que é um *micro-switch* cujo nível lógico é "1", quando é alcançado. Este é o ponto onde é feita a primeira furação e considerado como referência para a medição dos deslocamentos da mesa.

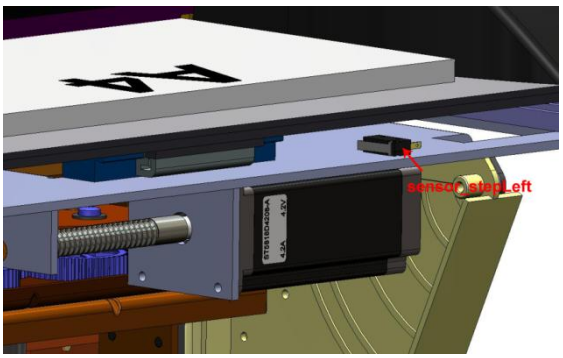


Figura 4-40: Sensor utilizado como ponto de referência para os deslocamentos da mesa

Os próximos passos do processo de furação são os do movimento ascendente e, posteriormente, descendente do bloco de furação. Neste processo, foi necessário definir

três posições: o local onde se inicia o processo, um ponto intermédio utilizado para minimizar o tempo de furação e o ponto que indica que o furo está concluído. Como mecanicamente não foi possível indicar estas três posições apenas com dois sensores, utilizaram-se três *micro-switch* que, quando pressionados, passam para o nível lógico "1". Estes sensores estão representados na Figura 4-41 e Figura 4-42 por *sensor_drillPositionA, sensor_drillPositionB* e *sensor_drillPositionC*.

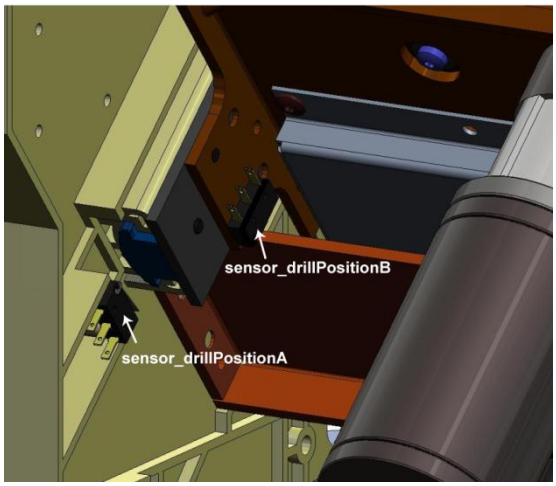


Figura 4-41: Sensores utilizados no ponto inicial e intermédio do processo de furação

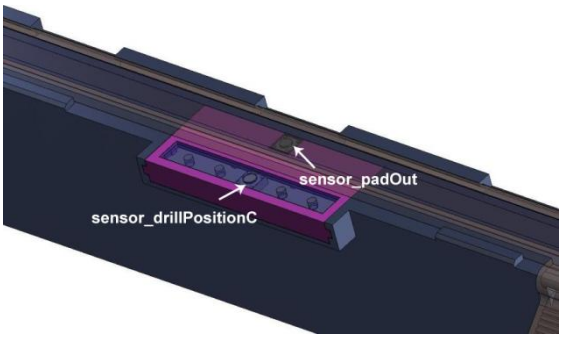


Figura 4-42: Sensores utilizados no ponto final do processo de furação e na verificação da presença do batente das brocas

O mecanismo de encadernação encontra-se sempre posicionado em uma de cinco posições. As duas primeiras posições, porta aberta e porta fechada, são utilizadas na colocação do *clickbind*, ainda antes de se dar início ao processo de furação. As restantes são usadas na encadernação e definem o local onde esta se efectua, dependendo do tamanho da argola. Utilizaram-se, para tal, três micro-switch, representados na Figura 4-43 e designados por *sensor_closeA, sensor_closeB* e *sensor_closeC*, que, dependendo dos seus níveis lógicos, como se vê na Tabela 4-10, definem a posição do mecanismo de encadernação.

Posição do mecanismo de encadernação	sensor_closeA	sensor_closeB	sensor_closeC
Porta Aberta	0	0	1
Porta Fechada	0	1	0
Encadernação de *clickbind* grande	1	0	0
Encadernação de *clickbind* médio	1	1	0
Encadernação de *clickbind* pequeno	1	1	1

Tabela 4-10: Estado dos sensores para as diferentes posições do mecanismo de encadernação

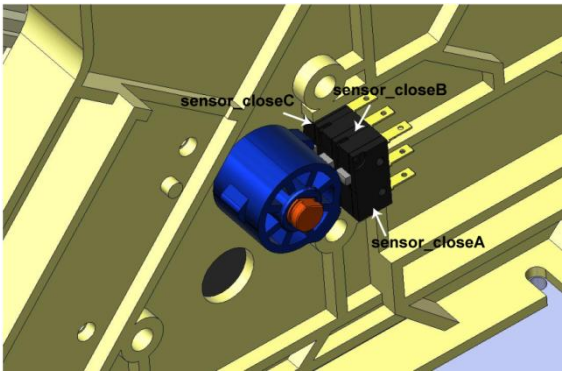
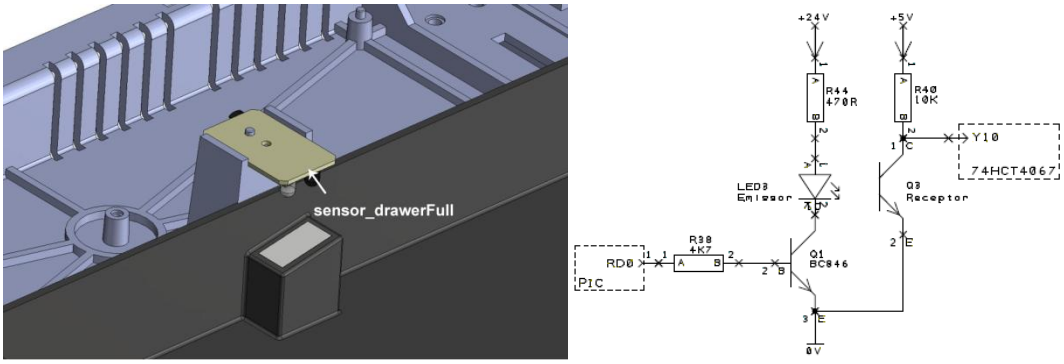


Figura 4-43: Sensores utilizados no posicionamento do mecanismo de encadernação

Fazem também parte da máquina três sensores responsáveis por analisar o estado de algumas peças essenciais para dar início a todo o processo.

Na gaveta onde são depositados os restos de papel resultantes da furação, utilizou-se um sensor óptico, representado na Figura 4-44 por *sensor_drawerFull*, para verificar o estado da gaveta. Este sensor utiliza um feixe de luz infravermelha emitida por um LED, que alcança o fototransistor, depois de reflectido por uma película reflectora, caso a gaveta não se encontre cheia. Quando o feixe não atinge o fototransistor, é sinal que foi interrompido devido à existência de restos de papel no seu percurso, sinal de que a gaveta se encontra cheia. Para o comando do LED, foi utilizado um sistema por impulsos com *duty cycle* de 10%, de forma a que o feixe tenha o alcance desejado, sem serem atingidos os limites de potencia do LED.



a) b)

Figura 4-44: Sensor de gaveta: a) Desenho mecânico; b) Esquemático

O processo não pode ser iniciado sem que as brocas e o seu batente estejam bem colocados, tendo-se utilizado um *micro-switch* junto de cada um desses mecanismos para efectuarem essa verificação. O sensor das brocas está representado na Figura 4-45 por *sensor_drillOut* e seu batente na Figura 4-42 por *sensor_padOut*. Estes dois sensores encontram-se, normalmente, com nível lógico "1", ficando a "0", quando se verifica a ausência de brocas ou do batente.

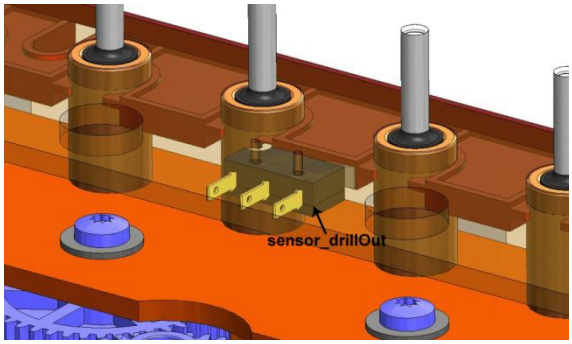


Figura 4-45: Sensor utilizado na verificação da presença das brocas

Existe ainda o *sensor_portOpen*, que tem como função verificar o estado da porta que dá acesso às brocas. Tal como o *sensor_stop*, interrompe o processo de encadernação a qualquer momento, caso a porta seja aberta, activando, para tal, uma interrupção externa. Este procedimento é imprescindível, dado o risco que esta situação pode representar para o utilizador. O *sensor_portOpen* é um *micro-switch* que é pressionado pela porta, conforme se pode ver na Figura 4-46. Este sensor encontra-se normalmente com nível lógico "0", passando a "1", quando a porta é aberta.

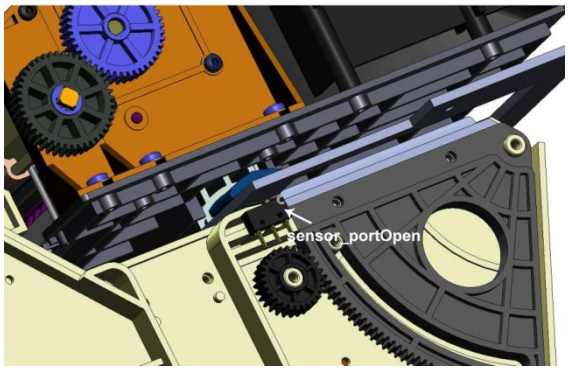


Figura 4-46: Sensor utilizado na verificação do estado da porta que dá acesso às brocas

Finalmente, tem-se os sensores de corrente no motor DC1 e DC4, designados, respectivamente, por *dc1_current* e *dc4_current*. O sensor *dc1_current* permite verificar, através do aumento da corrente, quando o mecanismo que prende o documento a encadernar se encontra a pressioná-lo. O sensor *dc4_current* permite verificar, quando o mecanismo de fecho do *clickbind* terminou o seu movimento de

abertura ou fecho. O circuito integrado L298, utilizado no controlo dos motores DC, tem uma saída de corrente através da qual se pode obter uma tensão proporcional à corrente no motor. Este sinal pode ser amplificado e comparado, de seguida, com um valor predefinido correspondente à intensidade do motor antes de bloquear. Assim, sempre que a intensidade ultrapassar esse valor significa que o mecanismo se encontra a pressionar o documento e, consequentemente, a saída do circuito muda de estado, passando a nível lógico "1".

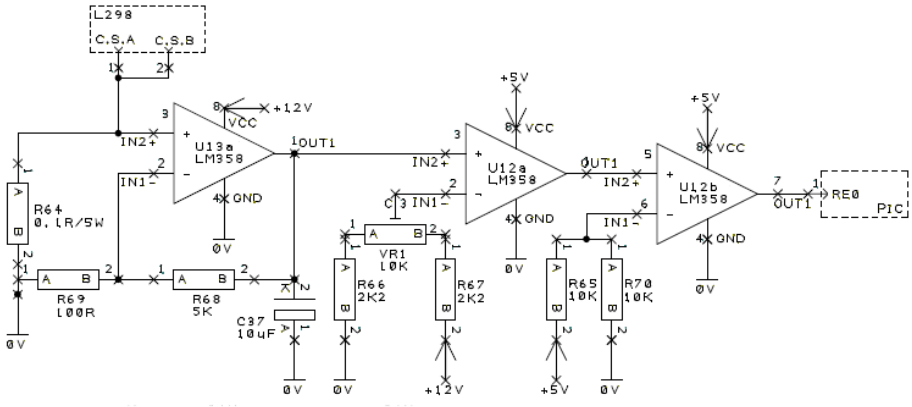


Figura 4-47: Hardware associado ao sensor de corrente do motor DC1

O elevado número de sensores utilizados levou à necessidade de usar um multiplexer de 16:1. Na Figura 4-48, pode ver-se a ligação de cada um destes sensores ao multiplexer ou ao PIC18F4431.

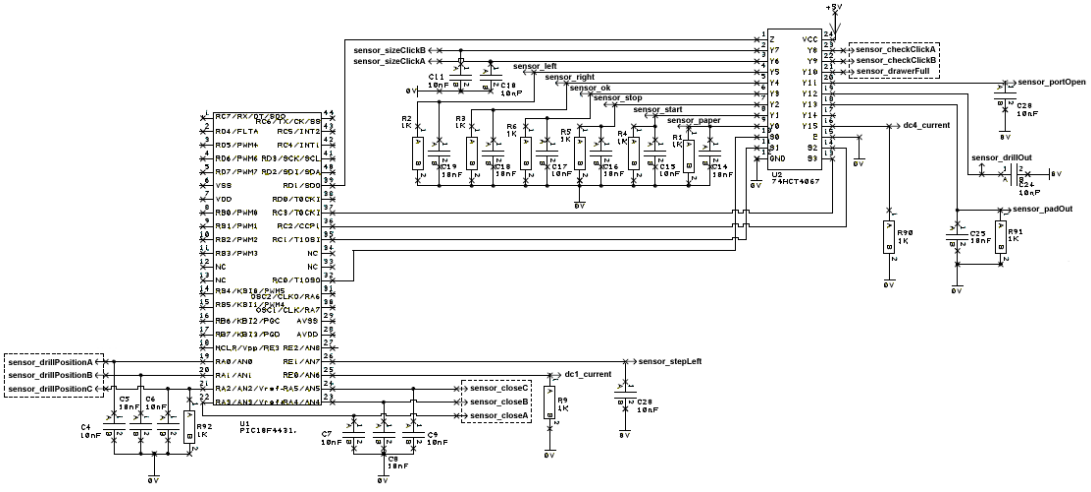


Figura 4-48: Hardware de controlo dos sensores

4.5.3 Software

A verificação do estado de cada sensor é extremamente importante para o correcto acompanhamento da evolução do processo de encadernação. O estado de todos

os sensores é guardado no *array sensor[]* de 27 posições, onde cada posição corresponde a um sensor. Foram realizadas três funções para fazer a monitorização de todos os sensores: *vSENSOR_ControlTris()*, *vSENSOR_Control(iNumber)* e *vSENSOR_Read()*.

As duas primeiras são responsáveis por enviar os sinais de comando para o *multiplexer*, sendo que uma inicializa os pinos I/O para controlo do *multiplexer* como saída (*vSENSOR_ControlTris()*) e a outra indica ao *multiplexer* qual o sensor que se pretende monitorizar (*vSENSOR_Control(iNumber)*).

A função *vSENSOR_Read()* (Figura 4-49) monitoriza os sensores ligados ao *multiplexer* e ao microcontrolador PIC, guardando os seus valores no *array sensor[]*. Além disso, é responsável por impedir que a máquina entre em funcionamento, após uma interrupção causada pelo pressionar do botão *stop* ou pela abertura da porta que dá acesso às brocas. Esta característica é obtida através da criação de uma *flag*, que é activada com o início da interrupção e assim se mantém, enquanto os sensores não voltarem ao seu estado inicial. Para uma permanente monitorização dos sensores, esta função é constantemente chamada.

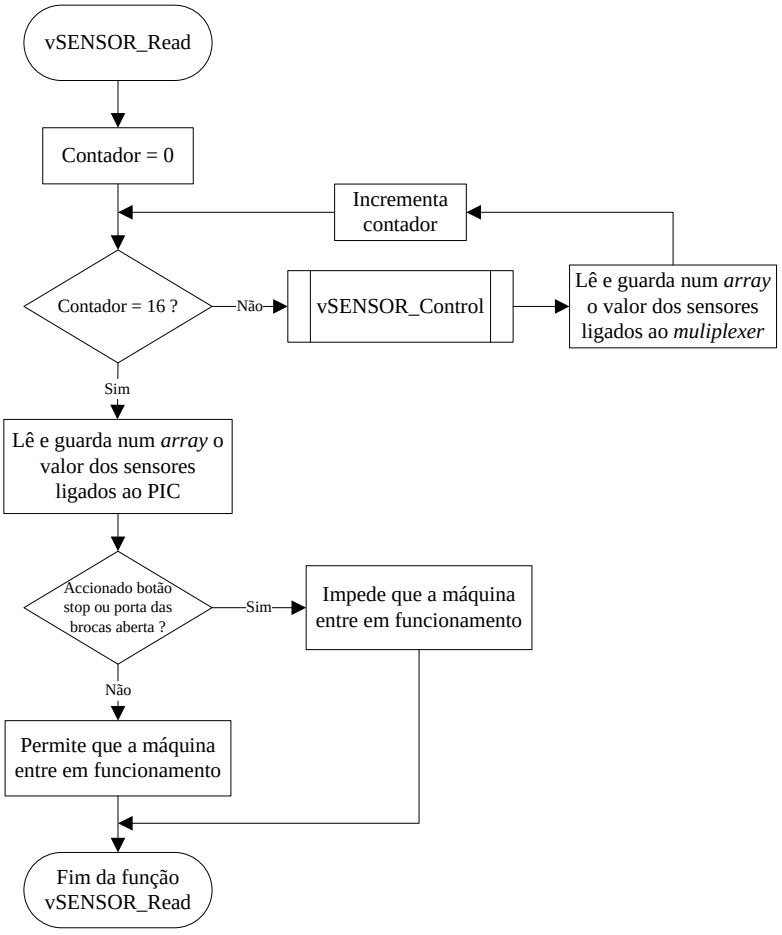


Figura 4-49: Algoritmo da função *vSENSOR_Read()*

A função *iSENSOR_Error()*, cujo o algoritmo pode ser visto na Figura 4-50, analisa o estado do sensor da gaveta, das brocas e do seu batente. Só depois de verificados estes sensores é que a máquina está pronta para iniciar a encadernação.

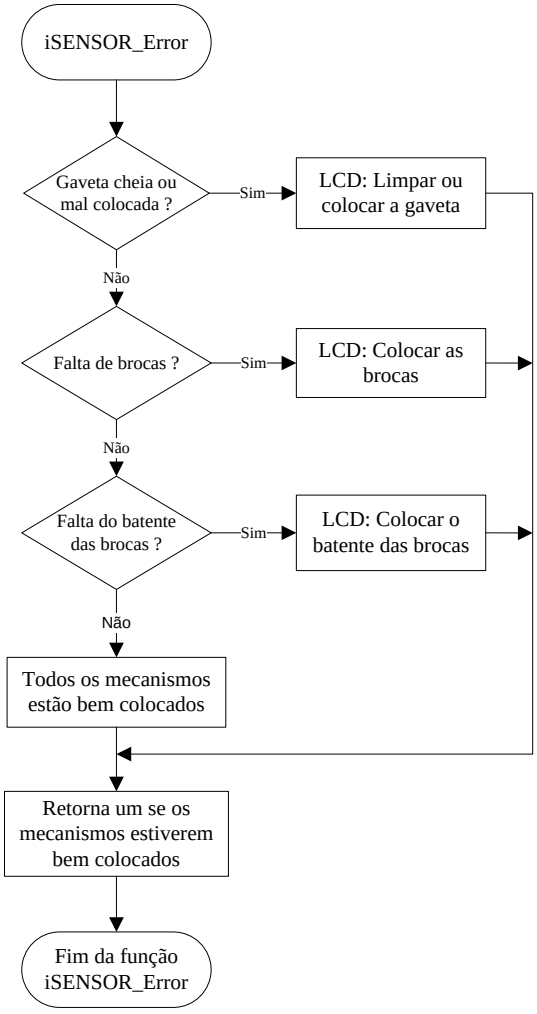


Figura 4-50: Algoritmo da função *iSENSOR_Error()*

A gaveta é o primeiro sensor a ser monitorizado. Caso ela esteja cheia ou não esteja presente, é enviada uma mensagem para o *display* com a indicação de que deve ser limpa ou colocada. A análise do estado do sensor da gaveta é feito com recurso à função *iSENSOR_DrawerFull()*, que controla a emissão dos impulsos efectuados pelo LED e lê o sinal recebido pelo fototransistor, retornando esse valor. Na Figura 4-51, pode ver-se o seu algoritmo.

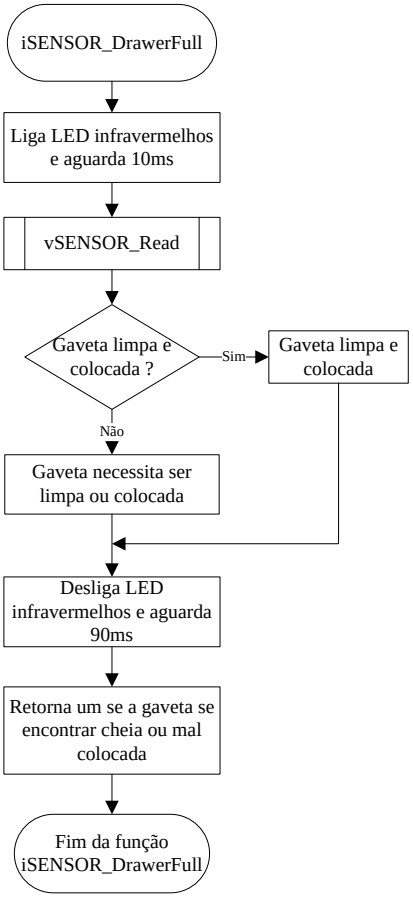


Figura 4-51: Algoritmo da função *iSENSOR_DrawerFull()*

Não existindo erros com a gaveta, passa-se a analisar o sensor das brocas, que verifica se elas estão ou não colocadas. Caso não estejam, é enviada uma mensagem para o *display* com a informação de que as brocas devem ser colocadas.

Não se verificando erros na gaveta, nem no sistema de brocas, passa-se a analisar o sensor do batente das brocas. Caso este não esteja colocado, é enviada uma mensagem para o *display* com a informação de que o batente deve ser aplicado.

Em relação ao sensor de corrente, é importante saber quando a corrente ultrapassa o limite definido. Com a utilização do *hardware* explicado anteriormente, obtém-se um sinal que indica se essa corrente foi ou não ultrapassada. A função *bDC_Current(iMotorNumber)*, lê esse valor e realiza o seu controlo, de forma a verificar se o motor está ou não parado. As leituras são feitas com intervalos de 0.1s e a função retorna um valor que indica se a corrente foi ultrapassada durante três leituras consecutivas. Desta forma, conseguem eliminar-se erros que poderiam ser provocados por variações intempestivas do valor da corrente. Esta função tem como variável de entrada o número do motor que se pretende analisar e o seu algoritmo é apresentado na Figura 4-52.

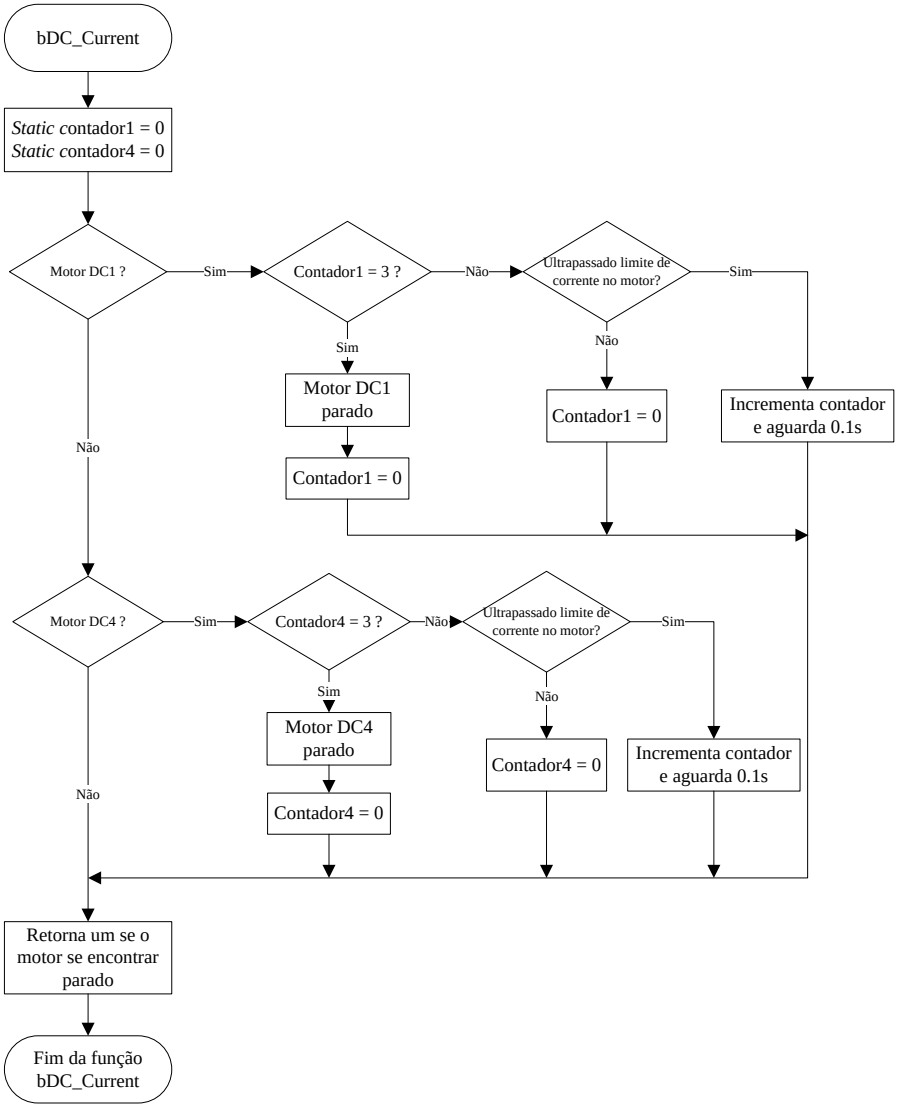


Figura 4-52: Algoritmo da função *bDC_Current(iMotorNumber)*

4.5.4 Conclusão

Concluiu-se que a necessidade de detectar fins de curso e o estado de alguns mecanismos da máquina, levou ao uso de sensores de posição mecânicos e ópticos. À excepção dos sensores para verificar o tamanho da argola, do sensor da gaveta e dos sensores de corrente, em todos os outros foram utilizados sensores mecânicos do tipo *micro-switch*, dado o seu baixo custo, simplicidade na aplicação e fiabilidade.

Para o sensor de verificação do tamanho da argola e para o sensor de gaveta, utilizaram-se sensores ópticos; para os sensores de corrente, utilizam-se resistências de precisão, nas saídas do circuito integrado L298 existentes para essa função.

A monitorização dos 22 sensores existentes na máquina é realizada por software.

4.6 Software para controlo dos motores

Os circuitos de controlo dos motores DC e do motor de passo, estudados no capítulo anterior, têm que ser comandados por sinais digitais. Em relação aos motores DC, nos casos em que se pretende inversão do sentido de rotação e variação de velocidade, é necessário aplicar um sinal de PWM à ponte H, tal como explicado anteriormente. No caso em que apenas é necessário fazer o comando *on/off* do motor, é utilizado um sinal que comanda um relé que liga e desliga o motor. Nos motores de passo, é necessário aplicar às entradas do seu controlador os sinais necessários para o motor efectuar um determinado número de passos a uma velocidade e sentido de rotação definidos.

4.6.1 Motor DC

O módulo de *power control PWM*, presente no PIC18F4431, permite o controlo de quatro motores DC. As oito entradas de PWM funcionam aos pares, de modo a que cada par seja responsável por comandar um motor. Decidiu-se controlar o motor DC1 através das entradas PWM0 e PWM1, o motor DC2 através das entradas PWM2 e PWM3, o motor DC3 através das entradas PWM4 e PWM5 e o motor DC4 através das entradas PWM6 e PWM7.

A definição do período e do *duty cycle* do PWM passa pela configuração de vários registos. A base de tempo do PWM é formada por um *timer* de 12bit com funções de *prescaler* e *postscaler*, encontra-se definida no registo PTMR (*PWM Time Base Registers*) e é configurada nos registos PTCON0 (*PWM Timer Control Register 0*) e PTCON1 (*PWM Timer Control Register 1*) [100]. Estes dois últimos registos, juntamente com os registos PWMCON0 (*PWM Control Register 0*) e PWMCON1 (*PWM Control Register 1*), são configurados na função *vPWM_Init(hPtcon0,hPtcon1,hPwmcon0,hPwmcon1)* (Figura 4-53), cujas variáveis de entrada são os valores desses mesmos registos.

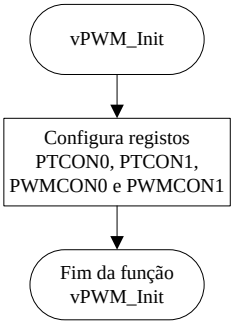


Figura 4-53: Algoritmo da função *vPWM_Init(hPtcon0,hPtcon1,hPwmcon0,hPwmcon1)*

Na Figura 4-54, encontra-se o formato do registo PTCON0, sendo o *postscaler* e o *prescaler* definidos nos bits PTOPS3:PTOPS0 e PTCKPS1:PTCKPS0, respectivamente. Nos bits PTMOD1:PTMOD0 escolhe-se o modo de operação, que pode ser *Free-Running*, *Single-Shot*, *Continuous Up/Down Count* ou *Continuous Up/Down Count with interrupts for double PWM updates* [100].

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0
bit 7							bit 0

Figura 4-54: Registo PTCON0 [100]

O *postscaler* e *prescaler* permitem diminuir a frequência de PWM, estando a diferença no facto do *prescaler* actuar sob a frequência de oscilação e do *postscaler* sob a interrupção de PWM [100].

No modo *Free-Running*, a contagem do PTMR é efectuada até que o seu valor iguale o período de PWM, presente no registo PTPER (*PWM Time Base Period Registers*), altura em que ocorre uma interrupção e é efectuado o *reset* ao PTMR reiniciando a contagem. Um *postscaler* diferente de 1:1 permite reduzir a frequência com que as interrupções ocorrem.

No modo *Single-Shot*, a contagem do PTMR é iniciada aquando da activação do PTEN. Quando valor do registo PTMR iguala o do PTPER, o PTMR é reiniciado e o PTEN é posto a zero como forma de o parar. O *postscaler* não tem qualquer interferência neste modo de temporização.

No modo *Continuous Up/Down Count*, o PTMR conta de forma crescente até que o seu valor atinja o de PTPER, altura em que o PTMR inicia uma contagem decrescente. Neste modo uma interrupção é gerada cada vez que o registo PTMR é reiniciado e inicia uma nova contagem. O *postscaler* pode ser usado, neste modo, para reduzir a frequência das interrupções.

O modo *Continuous Up/Down Count with interrupts for double PWM updates* é idêntico ao *Continuous Up/Down Count*, com a particularidade de que a interrupção é gerada cada vez que o PTMR é reiniciado ou iguala o valor do PTPER.

Na Figura 4-55, encontra-se o formato do registo PTCON1, onde PTEN permite habilitar ou desabilitar o PTMR e o PTDIR define se a contagem é efectuada de forma crescente ou decrescente [100].

R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
PTEN	PTDIR	—	—	—	—	—	—
bit 7							bit 0

Figura 4-55: Registo PTCON1 [100]

Na Figura 4-56, pode ver-se o formato do registo PWMCON0, onde os bits PWMEN2:PWMEN0 permitem definir quais os pinos de PWM activos e os bits PMOD3:PMOD0 permitem escolher entre o modo independente ou complementar, para cada par de pinos de I/O do PWM [100].

U-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
—	PWMEN2	PWMEN1	PWMEN0	PMOD3	PMOD2	PMOD1	PMOD0
bit 7							bit 0

Figura 4-56: Registo PWMCON0 [100]

Na Figura 4-57, pode ver-se o formato do registo PWMCON1, onde os bits SEVOPS3:SEVOPS0 permitem definir o *postscaler* do PWM *special event trigger*, o bit SEVTDIR define se o *special event trigger* ocorre, quando o PTMR conta de forma crescente ou de forma decrescente, o bit UDIS permite habilitar ou desabilitar actualizações do PTPER e do PDCx (*PWM Duty Cycle x Registers*) e o bit OSCNY permite escolher, para o caso das saídas activadas manualmente, entre um modo assíncrono ou um modo sincronizado com o PTMR.

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0	SEVTDIR	—	UDIS	OSYNC
bit 7							bit 0

Figura 4-57: Registo PWMCON1 [100]

Optou-se por um *postscaler* e um *prescaler* de 1:1, pelo modo de operação *Free-Running*, por um PTMR com contagem de forma crescente, por utilizar todos os pinos como saídas de PWM em modo independente, pela possibilidade de habilitar ou desabilitar actualizações de PTMR e PDCx e por uma sincronização entre as saídas activadas manualmente e o PTMR.

O período é definido no registo PTPER (quatro LSB do PTPERH e oito bits do PTPERL), para tal desenvolveu-se a função *vPWM_Period()* que pode ser vista na Figura 4-58.

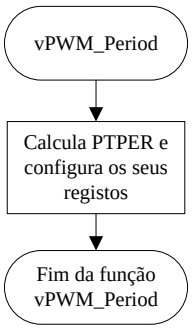


Figura 4-58: Algoritmo da função *vPWM_Period()*

O cálculo do PTPER depende da frequência de PWM (F_{PWM}), do valor de *prescaler* (PTMRPS) e da frequência de oscilação (F_{OSC}) e pode ser dada pela equação (4-7) [100].

$$PTPER = \frac{F_{OSC}/4}{F_{PWM} \cdot PTMRPS} - 1 \tag{4-7}$$

O *duty cycle* é definido no registo PDCx de 14bits (seis LSB do PDCxH e oito bits do PDCxL). Existem quatro registos com o *duty cycle* do PWM, um para cada par de pinos de I/O do PWM. O valor em cada registo de *duty cycle* determina o tempo em que a saída de PWM está ao nível lógico "1".

No modo *Free-Running*, como se pode ver pela Figura 4-59, a saída de PWM está activa no início do período e é levada a "0", quando o valor no registo PTMR igualar o valor no registo PDCx. Um novo valor de *duty cycle* é actualizado, quando o valor no registo PTMR igualar o valor do registo PTPER.

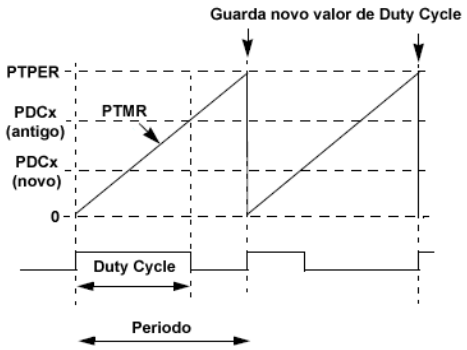


Figura 4-59: *Duty cycle* no modo *Free-Running* [100]

A função *vDC_Mode(iMotorNumber, bDirection, iPerDutyCycle)* (Figura 4-60) define o valor do *duty cycle* e o sentido de rotação para cada motor, tendo, portanto, estes valores como variáveis de entrada.

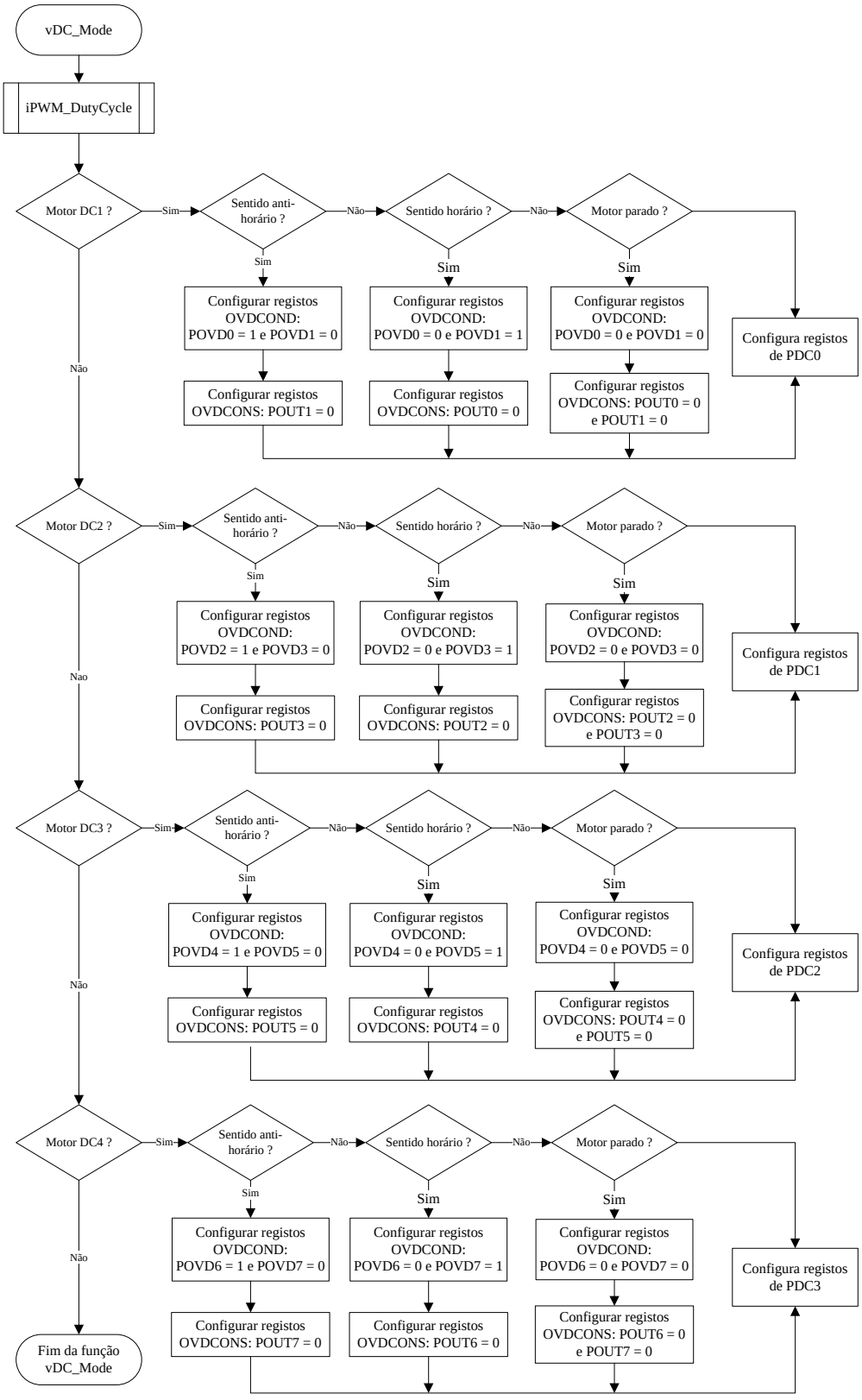


Figura 4-60: Algoritmo da função *vDC_Mode(iMotorNumber, bDirection, iPerDutyCycle)*

Esta função recorre à função *iPWM_DutyCycle(iPerDutyCycle)* (Figura 4-61), para fazer o cálculo do *duty cycle*, tendo como variável de entrada a sua percentagem.

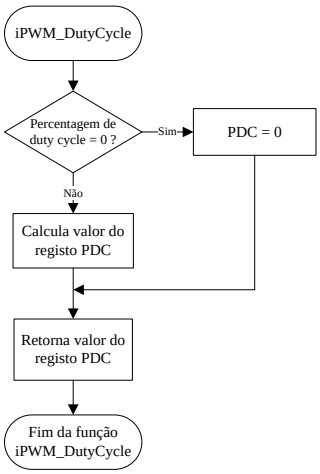


Figura 4-61: Algoritmo da função *iPWM_DutyCycle(iPerDutyCycle)*

Inicialmente, faz-se corresponder a percentagem a um valor dependente do período que se designa por DC_{PWM}. De seguida, é necessário calcular o valor do registo que é dado pela equação (4-8).

$$PDC_X = \frac{DC_{PWM}}{T_{OSC} \cdot PTMRPS} \tag{4-8}$$

Onde T_{OSC} representa o período de oscilação e PTMRPS o valor de *prescaler*.

O sentido de rotação na função *vDC_Mode(iMotorNumber, bDirection, iPerDutyCycle)* é controlado através dos registos OVDCOND (*Output Override Control Register*) e OVDCONS (*Output State Register*). Estes registos permitem escolher uma saída inactiva, activa ou um sinal de PWM, em cada pino de I/O do PWM a operar no modo independente [100].

A Figura 4-62 apresenta o formato do registo OVDCOND, onde POVD7:POVD0 determinam quais as saídas de PWM que são activadas manualmente. Na Figura 4-63, pode ver-se o formato do registo OVDCONS, onde POUT7:POUT0 determinam o estado do pino de I/O do PWM, quando uma saída de PWM é activada manualmente. Quando um dos bits POVD estiver activo, o correspondente bit POUT não tem qualquer influência na saída de PWM. Por outro lado, quando um bit de POVD estiver inactivo a saída de PWM é ditada pelo valor presente em POUT [100].

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
POVD7	POVD6	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0
bit 7							bit 0

Figura 4-62: Registo OVDCOND [100]

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
bit 7							bit 0

Figura 4-63: Registo OVDCONS [100]

Desta forma, é possível, ligando as saídas de PWM às entradas do integrado L298, fazer com que cada motor rode no sentido horário ou anti-horário. O sentido horário obtém-se quando o PWM é aplicado ao par In1-In4, pela activação do bit POVD e o par In2-In3 está a nível lógico "0", pela inactivação dos bits POVD e POUT. O sentido anti-horário obtém-se, quando o PWM é aplicado ao par In2-In3 e o par In1-In4 está a nível lógico "0".

Para efectuar o controlo do relé, utilizou-se o pino RD5 configurado como saída. Quando se pretende ligar o relé, activa-se este pino; colocando-o ao nível lógico "0", quando se pretender desligá-lo.

4.6.2 Motor de passo

Para o motor de passo, utilizaram-se quatro pinos configurados como saída, o RC5:RC7 e o RD4. Cada um destes pinos está ligado a uma das entradas do integrado L297. O pino RD4 está ligado à entrada *Enable*, o RC7 ao *Reset*, o RC6 ao *Clock* e o RC5 ao *CW/CCW*.

A inicialização do motor de passo é feita através da equação *vSTEP_Init()*, cujo algoritmo está apresentado na Figura 4-64. Esta função é responsável por habilitar o motor e fazer o seu *reset*.

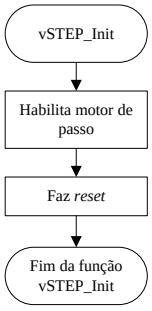


Figura 4-64: Algoritmo da função *vSTEP_Init()*

Estando o motor pronto para iniciar o seu funcionamento, escolhe-se o sentido de rotação pretendido.

A realização de um passo pelo motor é feita com recurso à função *vSTEP_Clock(itime)* (Figura 4-65), que tem como variável de entrada o tempo entre impulsos de *clock*, o que define a velocidade do motor de passo.

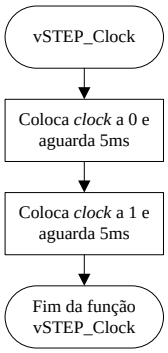


Figura 4-65: Algoritmo da função *vSTEP_Clock(itime)*

4.6.3 Conclusão

Concluiu-se que o módulo de PWM existente no PIC18F4431 foi extremamente importante no desenvolvimento deste trabalho, tendo em conta a sua capacidade para controlar, simultaneamente, quatro motores DC. Constatou-se que é formado por diversos registos, que permitem definir o período, o *duty cycle* e o sentido de rotação do motor. Possui ainda um *timer* que funciona como base tempo do PWM e que, em conjunto com os registos do período e do *duty cycle*, são os principais responsáveis por definir o PWM do motor.

O controlo do relé foi efectuado através de uma das saídas do PIC de uma forma extremamente simples, dado tratar-se de um dispositivo *on/off*.

O controlo do motor de passo revelou-se bastante simples, em parte devido ao uso de um controlador por hardware, o L297.

4.7 Descrição do funcionamento

Neste capítulo, pretende-se explicar o funcionamento da máquina e de todos os passos que ela tem que percorrer, desde o momento em que é ligada, até ao momento em que se retira o documento já encadernado.

Na Figura 4-66 pode ver-se a primeira parte do algoritmo utilizado, onde se constata que é necessário definir os pinos de entrada e de saída e inicializar todas as variáveis. Em relação aos motores, inicializa-se o PWM, através da função *vPWM_Init(hPtcon0, hPtcon1, hPwmcon0, hPwmcon1),* define-se o período de PWM, através da função *vPWM_Period(),* e mantêm-se os motores parados, através da função *vDC_Mode(iMotorNumber, bDirection, iPerDutyCycle).* É igualmente importante

inicializar a interrupção externa e defini-la como de alta prioridade. Segue-se um *reset* à máquina, através da função *vINTERRUPT_Restart()*, colocando todos os motores em posição de iniciar o processo de encadernação. Apresenta-se, entretanto, no LCD a frase "GBC Welcome".

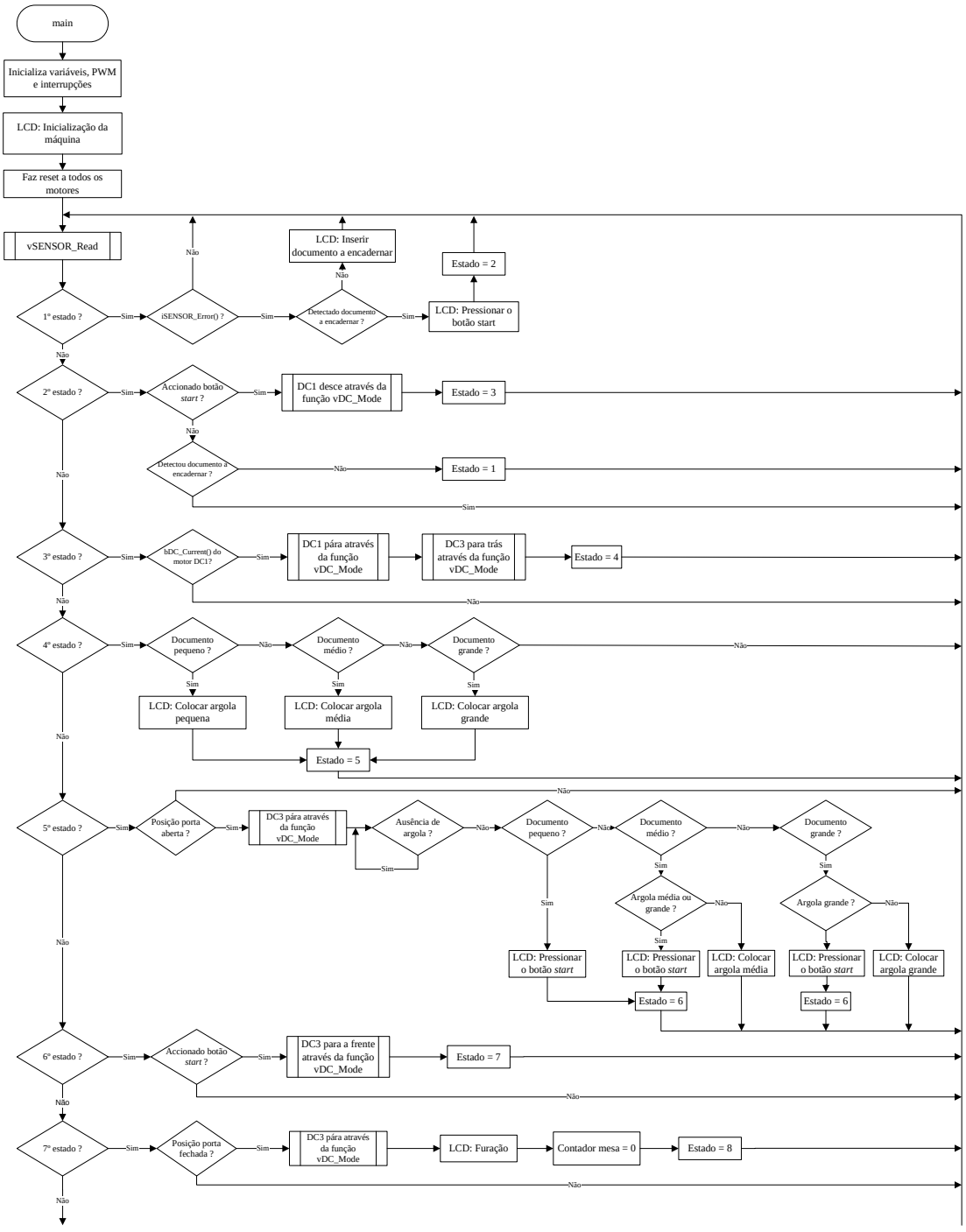


Figura 4-66: Primeira parte do algoritmo - detecção, indicação e confirmação do tamanho do documento e da argola para encadernar

Inicia-se, neste momento, um ciclo infinito, onde se efectuam todos os passos necessários para a realização de uma encadernação. No primeiro estado deste ciclo,

verifica-se a eventual existência de situações que impeçam o início da encadernação, como gaveta cheia ou mal colocada, ausência de brocas e do seu batente, através da função *iSENSOR_Error()*. Quando estas situações estão regularizadas, verifica-se a presença do documento a encadernar através do sensor *sensor_paper*. Enquanto este não for activado, apresenta-se no LCD a frase "ready, insert book", quando o *sensor_paper* for activado, passa-se ao estado seguinte, aparecendo no LCD a frase "press start". Neste estado, verifica-se se o botão *sensor_start* é activado. Enquanto não o for é permanentemente verificado se o livro a encadernar ainda se encontra no local, caso não esteja, volta-se ao estado anterior. Quando o *sensor_start* é activado, o motor DC1 entra em funcionamento, deslocando o mecanismo que prende as folhas em direcção ao documento, ao mesmo tempo que é apagada a informação existente no LCD. Está-se assim no 3º estado, onde se verifica se as folhas estão presas através da função *bDC_Current(iMotorNumber)*. No momento em que isso acontece, o motor DC1 pára, mantendo-se a aplicação de alguma tensão, para prender as folhas. Ainda neste estado o motor DC3 entra em funcionamento, deslocando o mecanismo de encadernação para trás em direcção à posição de porta aberta. Antes de verificar o *sensor_close* no 5º estado, é verificado o tamanho do documento a encadernar, através dos *sensor_sizeClick*. No caso de ser pequeno, aparece no LCD a mensagem "place spine below, small size", se for médio "place spine below, medium size" e se for grande surge "place spine below, large size". Após esta verificação no 4º estado, passa-se ao 5º estado onde é testada se a porta está aberta. Se isso se verificar, o motor DC3 pára, testa a presença do *clickbind* no mecanismo de encadernação e inicia uma comparação entre o tamanho da argola inserida e a pretendida. Tendo em conta que um documento pode ser encadernado com uma argola de tamanho superior, mas nunca de tamanho inferior, no caso de se pretender encadernar um documento pequeno, qualquer tamanho de argola pode ser usado para o efeito, aparecendo portanto no LCD "ready to start binding, press start". Para um documento médio, se a argola colocada for média ou grande, aparece a mesma frase no LCD, mas caso seja pequena, surge a frase "please place, larger spine". Para um documento grande acontece o mesmo, mas neste caso, esta última frase surge se a argola for pequena ou média. Para qualquer um dos casos acima descritos, quando a argola para encadernar é a apropriada, o utilizador deve carregar novamente no botão *sensor_start*. Quando este for accionado, o motor DC3 entra novamente em funcionamento, deslocando o mecanismo de encadernação agora para a frente até à posição de porta fechada, parando no momento em que atinge essa posição.

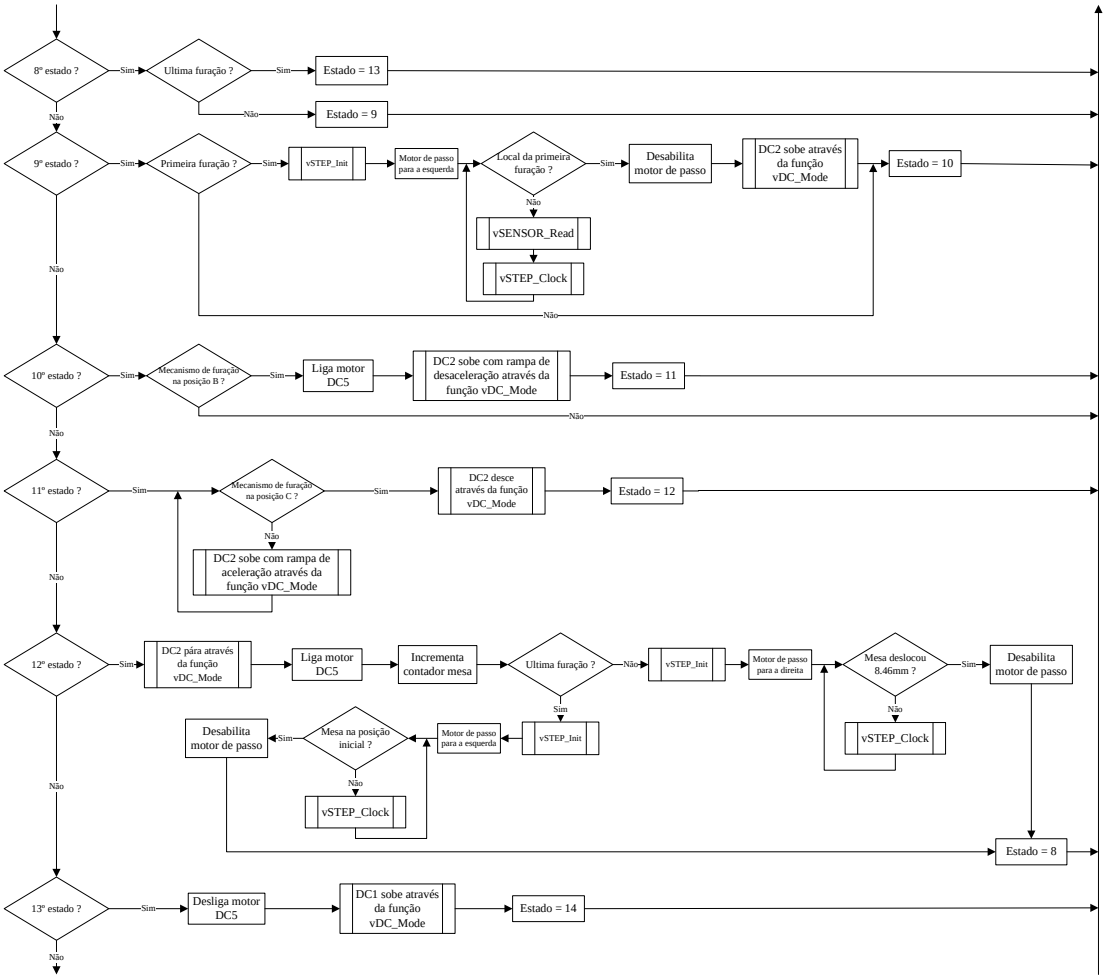


Figura 4-67: Segunda parte do algoritmo - Processo de furação

Começa então o processo de furação (Figura 4-67), apresentando no LCD a frase "punching". No 8º estado, verifica-se em que etapa está o processo de furação. Se estiver na primeira etapa, inicializa-se o motor de passo através da função *vSTEP_Init()*, indica-se o seu sentido de rotação e coloca-se o motor a rodar até a mesa atingir o sensor *sensor_stepleft*, altura em que se desabilita o motor de passo e inicia-se o movimento do motor DC2, que faz deslocar o bloco de furação. Este ao atingir o sensor *sensor_drillPositionB*, diminui de velocidade, seguindo uma rampa de desaceleração para as brocas entrarem nas folhas com alguma suavidade. Ao mesmo tempo, o motor DC5, que realiza a rotação das brocas, entra em funcionamento. Ao atingir as folhas, e enquanto o sensor *sensor_drillPositionC* não for activado, o bloco de furação realiza uma rampa de aceleração, para facilitar a furação. Quando o sensor *sensor_drillPositionC* for activado, o motor DC2 inverte o sentido de rotação até o bloco de furação atingir novamente o sensor *sensor_drillPositionB*, altura em que o motor DC5 desliga. Até que seja atingida a última etapa de furação, o motor de passo

desloca a mesa para a direita 8.46mm, de seguida o motor DC2 entra, novamente, em funcionamento conduzindo o bloco de furação em direcção às folhas, seguindo os mesmos passos. Na última etapa da furação, a deslocação do motor de passo dá-se para a esquerda até a mesa atingir a posição central. É importante novamente salientar que o número de etapas é de 4 ou 6, conforme se trate da máquina na versão para os USA ou para a Europa. No 13º estado, o motor DC1 inicia um movimento ascendente do mecanismo que prende as folhas, libertando-as, só parando quando for detectada uma sobrecorrente pela função *bDC_Current(iMotorNumber)*, sinal de que o mecanismo atingiu o topo. Nesse momento, o motor DC1 pára, entrando novamente em funcionamento o motor DC3.

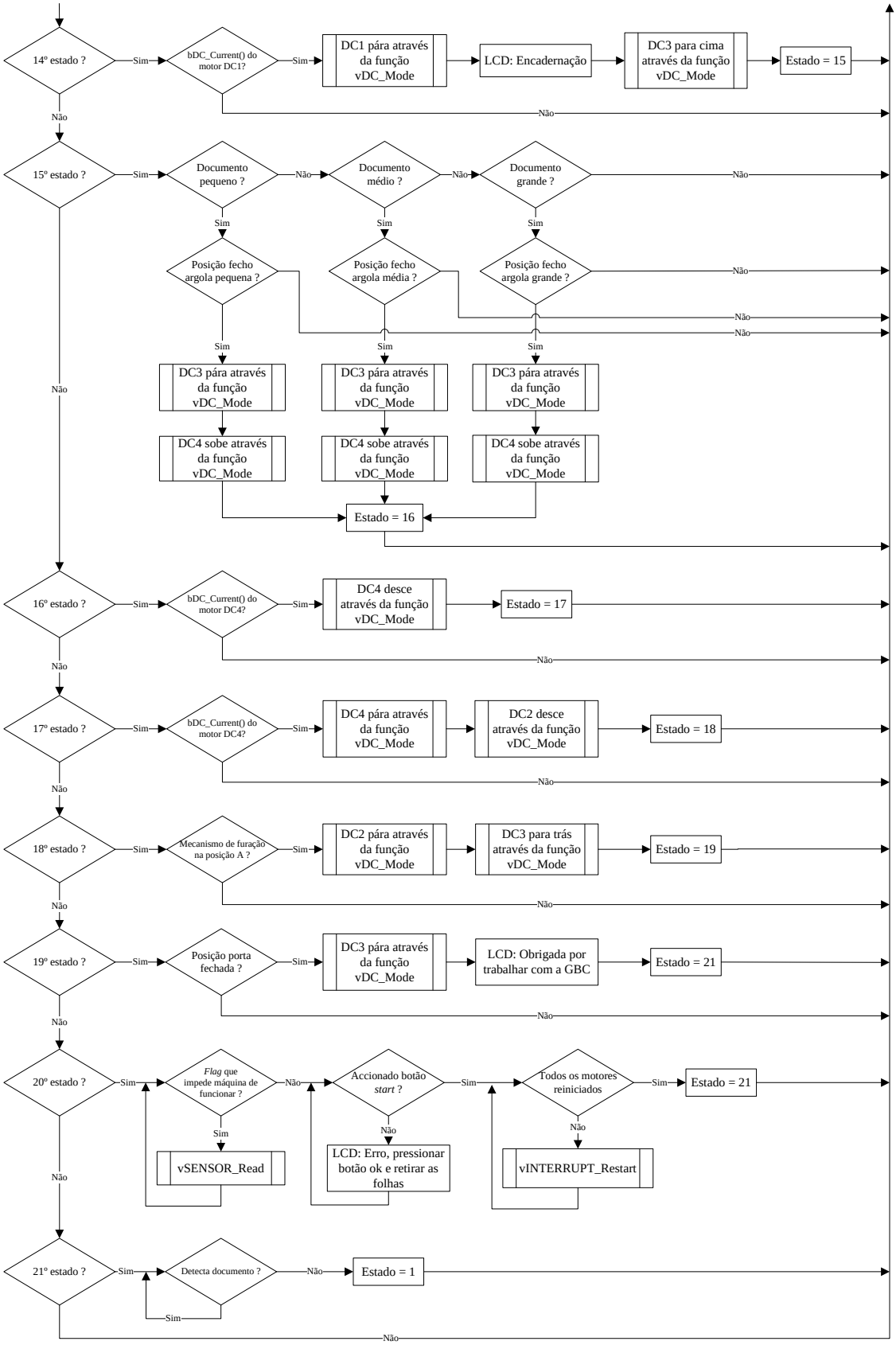


Figura 4-68: Terceira parte do algoritmo – Processo de encadernação

Começa então o processo de encadernação (Figura 4-68), aparecendo no LCD a frase “binding”. O deslocamento do mecanismo de encadernação pelo motor DC3 depende da argola com que se pretende encadernar, uma vez que quanto maior for a argola, mais afastado da margem do documento se dá o processo de encadernação. Ao atingir a posição correcta, o motor DC3 pára, entrando em funcionamento o motor DC4 accionando o mecanismo que fecha a argola. Na operação seguinte o motor DC4 inverte o sentido de rotação colocando o mecanismo na posição inicial. Seguidamente, o bloco de furação, posicionado no *sensor_drillPositionB*, por acção do motor DC2, desloca-se em direcção ao *sensor_drillPositionA*. Ao atingi-lo faz com que o motor DC3 desloque o mecanismo de encadernação para a posição de porta fechada, apresentando no LCD a informação “thank you for binding with gbc”. Retirado o documento já encadernado, o sensor *sensor_paper* detecta a ausência de folhas e o processo volta ao inicio, estando a máquina pronta para uma nova encadernação.

Como referido no capítulo sobre sensores, o processo de encadernação pode ser parado a qualquer momento, bastando para tal, que o botão *sensor_stop* seja pressionado ou que a porta que dá acesso às brocas seja aberta. Esta situação origina uma interrupção que resulta na execução da função *vSTOP_Interrupt()*, cujo algoritmo é apresentado na Figura 4-69.

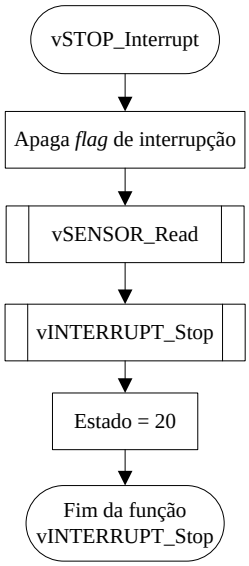


Figura 4-69: Algoritmo da função *vSTOP_Interrupt()*

Nesta função, primeiramente apaga-se a *flag* de interrupção através do bit INT1IF do registo INTCON3. Recorre-se à função *vSENSOR_Read*, para analisar o estado dos sensores que originaram a interrupção, activando ou não uma *flag* que impede a máquina de voltar a entrar em funcionamento. Param-se todos os motores,

através da função *vINTERRUPT_Stop* (Figura 4-70), e define-se que no regresso à *main* se siga para o 20º estado.

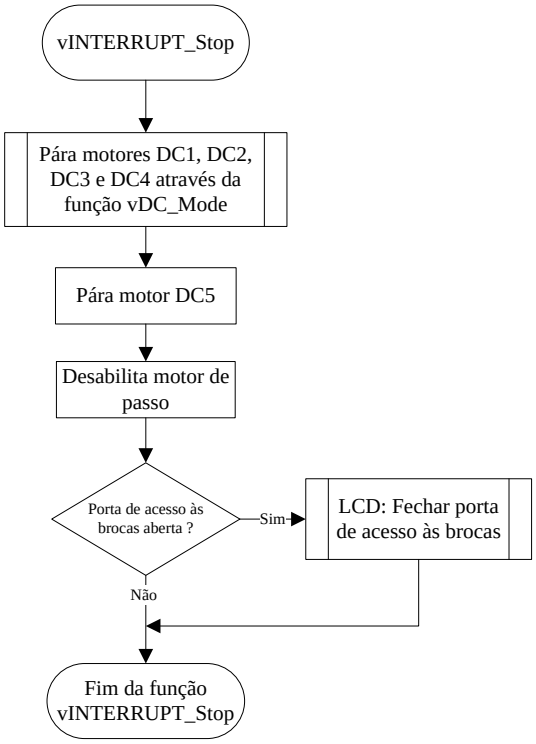


Figura 4-70: Algoritmo da função *vINTERRUPT_Stop*

No 20º estado da *main*, (Figura 4-68) é testada a *flag* que impede que a máquina regresse ao funcionamento. Logo que seja libertado o botão stop, ou quando a porta de acesso às brocas é fechada, aparece no ecrã do LCD a frase "*Error during process, please press ok to reset machine and remove sheets*". Isto deve-se ao facto destas ocorrências obrigarem a máquina a iniciar o processo de encadernação, devendo para tal ser feito o *reset* à máquina, através da função *vINTERRUPT_Restart()* e retirado o documento, estando a máquina pronta para uma nova encadernação.

A função *vINTERRUPT_Restart()*, apresentada na Figura 4-71, permite fazer um *reset* à máquina, colocando todos os seus motores prontos para iniciar uma nova encadernação. Como nem sempre se sabe a posição exacta do motor, utiliza-se um ponto de referência que, para o motor DC3, é a posição de porta aberta e, para o motor de passo, é o *sensor_stepLeft*.

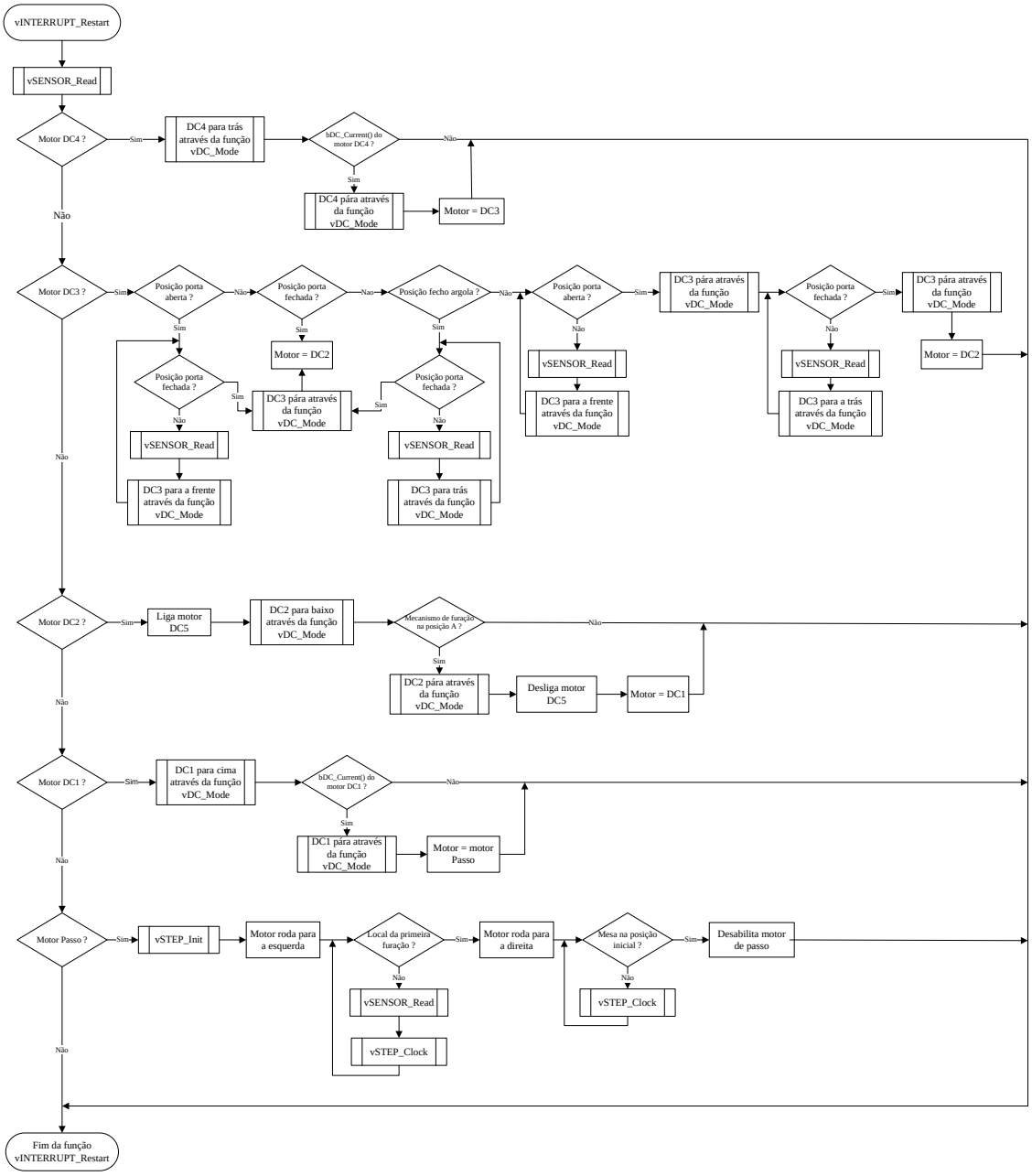


Figura 4-71: Algoritmo da função *vINTERRUPT_Restart()*

4.7.1 Conclusões

Verificou-se que a realização de uma encadernação pode ser dividida em três partes fundamentais: detecção, indicação e confirmação do tamanho do documento e da argola a encadernar; processo de furação e processo de encadernação. Cada uma destas partes é composta por vários passos, que só podem ocorrer quando se verifica a acção precedente.

Da primeira parte (detecção, indicação e confirmação) salientam-se os seguintes passos:

- Inicializa-se as variáveis, os motores e o I^2C
- Faz-se um *reset* à máquina
- Verifica-se as situações que impedem o início da encadernação
- Verifica-se a presença do documento a encadernar
- Pressiona-se o botão *start*
- O mecanismo que prende as folhas desloca-se na sua direcção, através do motor DC1
- Verifica-se o tamanho do documento a encadernar
- O mecanismo de encadernação desloca-se até à posição de porta aberta, através do motor DC3
- Verifica-se a presença do *clickbind* no mecanismo de encadernação e compara-se o seu tamanho com o do *clickbind* pretendido
- Pressiona-se novamente o botão *start*
- O mecanismo de encadernação desloca-se até à posição de porta fechada, através do motor DC3

Da segunda parte (processo de furação) salienta-se os seguintes passos:

- Na primeira etapa da furação desloca-se a mesa, através do motor de passo, para a posição da primeira furação
- Desloca-se o bloco de furação, através do motor DC2, até à posição B
- Liga-se o motor DC5, e o motor DC2 segue o seu movimento até à posição C
- Afasta-se o bloco de furação das folhas, através do motor DC2, até atingir novamente a posição B
- O motor DC5 desliga
- A mesa desloca-se 8.46mm para a direita e dá-se uma nova furação
- Depois de realizada a última etapa da furação, o motor de passo desloca a mesa para a posição central
- O motor DC1 liberta as folhas

Da terceira parte (processo de encadernação) salientam-se os seguintes passos:

- O mecanismo de encadernação desloca-se até uma das três posições de fecho da argola, através do motor DC3
- O mecanismo de fecho do *clickbind* fecha a argola, através do motor DC4

- O mecanismo de fecho do *clickbind* abre-se, regressando à posição inicial, através do motor DC4
- Desloca-se o bloco de furação, através do motor DC2, até à posição A
- O mecanismo de encadernação desloca-se até à posição de porta fechada, através do motor DC3
- Retira-se o documento e a máquina está pronta para uma nova encadernação.

Em qualquer um destes passos a máquina pode ser parada, pressionando o botão *stop* ou abrindo a porta que dá acesso às brocas. Este acto implica o reiniciar da encadernação.

4.8 I^2C

O uso de dois microcontroladores PIC levou à necessidade de usar um protocolo de comunicação. Foram analisados alguns dos protocolos suportados pelos PIC, como o série, RS485, I^2C e SPI. O facto de o autor estar familiarizado com os protocolos série e I^2C fez com que a escolha recaísse sobre esses. A preferência do I^2C deveu-se, essencialmente, ao facto de ser possível ligar posteriormente, se necessário, mais dispositivos no barramento. O facto de suportar menores distâncias do que o protocolo série, é irrelevante para este projecto.

4.8.1 Introdução teórica

O protocolo I^2C foi desenvolvido em 1980 pela *Philips Semiconductors*. É constituído por dois fios, o SCL (*Serial Clock*) e SDA (*Serial Data*). O SCL é a linha de *clock*, usada para sincronizar todos os dados transferidos no barramento e o SDA é a linha de dados. Estas duas linhas são ligadas a todos os dispositivos no barramento de I^2C. Tanto o SCL como o SDA são linhas *open drain*, ou seja, o circuito integrado pode levar a saída a "0", mas não a "1". Para que as linhas possam ir ao nível lógico "1", é necessário colocar uma resistência de *pull-up* aos 5V [136].

Os dispositivos no barramento I^2C são *masters* ou *slaves*. O *master* comanda o SCL e é o único capaz de iniciar uma transferência. Embora possa existir mais que um *master*, o normal é existir um *master* e diversos *slaves*. Tanto o *master* como o *slave*

podem transferir dados no barramento, mas a transferência é sempre controlada pelo *master* [136].

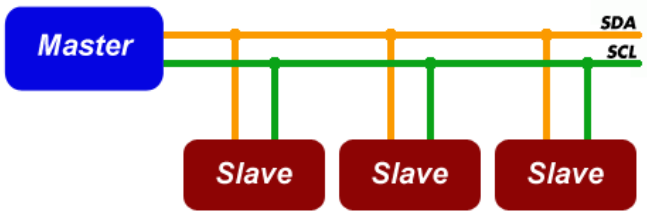


Figura 4-72: Barramento I^2C

Quando o *master* pretende comunicar com o *slave*, começa por enviar uma sequência de *start* para o barramento I^2C. Uma sequência de *stop* é enviada, quando se pretende terminar a comunicação. Estas sequências são especiais, dado serem as únicas em que o SDA está autorizado a alterar o seu estado, estando o SCL a nível alto (

Figura 4-73). Durante o processo de transferência de dados, o SDA deve manter-se estável e não sofrer alterações enquanto o SCL estiver a nível alto [136].

Este tipo de barramento é capaz de suportar 128 ou 1024 dispositivos, dependendo se usa endereços de 7 ou 10bits, sendo os mais usuais os de 7bits [136].

O *clock* funciona a 100kHz em modo *standard*, podendo trabalhar a 400kHz em modo *High-Speed* [136].

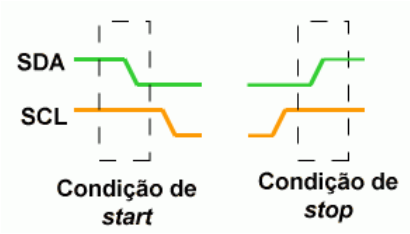


Figura 4-73: Condições de *start* e *stop* no protocolo I^2C [137]

Na Figura 4-74, pode ver-se como se realiza a escrita de dois bytes pelo *master* no *slave*. A transmissão verifica-se quando o *master* inicia a condição de *start*, sendo, de seguida, enviado um endereço para o *slave*, onde os sete MSBs contêm o endereço do *slave* e o LSB especifica se é referente a uma leitura (LSB=1) ou uma escrita (LSB=0). Ao 9º impulso de *clock*, o *master* liberta a linha SDA e o *slave* pode enviar o ACK (*Acknowledge*), que é de nível lógico "0" caso o *slave* receba o endereço correcto. Assumindo que o ACK foi recebido, o *master* envia os bytes de dados. Ao 9º impulso de *clock*, depois de enviados os dados, o *slave* responde com um novo ACK. No final do último byte, o *slave* envia um NACK (*Not Acknowledge*) para indicar que não devem ser enviados mais bytes, altura em que o *master* inicia uma condição de *stop* [138].

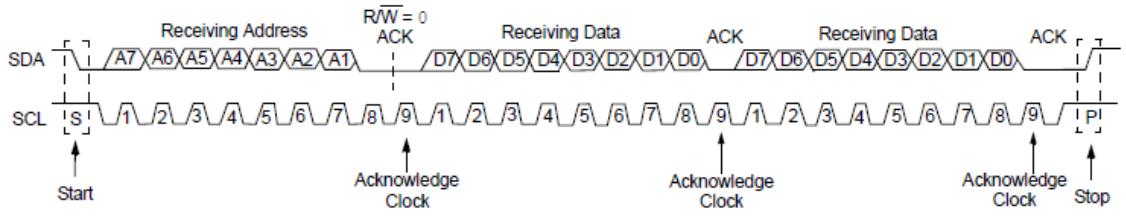


Figura 4-74: Operação de escrita

Na Figura 4-75, pode ver-se uma operação de leitura, que é similar à de escrita. Neste caso, o LSB do byte de endereço é colocado a nível lógico "1". Depois de recebido o byte de endereço, o *slave* envia um ACK e mantém a linha de *clock* a "0", garantindo, desta forma, o tempo necessário para preparar os dados a serem enviados para o *master*. Quando o *slave* está pronto, liberta o SCL e o *master* recebe os dados do *slave*. Ao 9º impulso de *clock*, se o *slave* receber do *master* um ACK, prepara o próximo byte a ser transmitido, se receber um NACK, a transmissão está completa [138].

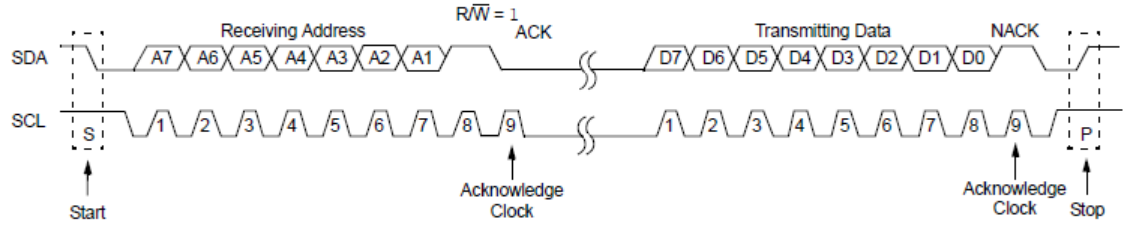


Figura 4-75: Operação de leitura

4.8.2 Software

O PIC18F4431 possui um módulo SSP (*Synchronous Serial Port*) e o PIC18F2520 um módulo MSSP (*Master Synchronous Serial Port*). Estes módulos permitem a comunicação série do microcontrolador com outros microcontroladores ou periféricos, por SPI ou I^2C.

Neste projecto, utilizou-se o PIC18F4431 como *master*, o PIC18F2520 como *slave* e a comunicação entre eles por I^2C. No *master*, o SCL foi definido no pino de I/O RD2 e o SDA no RD3. No *slave*, o pino de I/O do SCL é o RC3 e do SDA é o RC4. Como o PIC18F4431 apenas implementa funções de *slave*, a implementação do *master* teve que ser feita por software. A *Microchip* possui duas bibliotecas que implementam algumas funções para simplificar a programação do I^2C quer por hardware (I2C) quer por software (SW_I2C).

Pretende-se que o *master* seja capaz de enviar um caracter para o *slave* e que este, ao recebê-lo, transmita ao LCD a informação correspondente. Desta forma, implementou-se no PIC18F4431 a função *vI2C_SendCaracter(hAddr, hCaracter)* (Figura 4-76) que tem como variáveis de entrada o endereço do *slave*, para onde se pretende enviar o caracter e o próprio caracter.

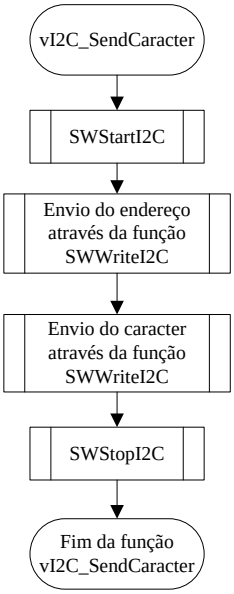


Figura 4-76: Algoritmo da função *vI2C_SendCaracter(hAddr, hCaracter)*

Pela Figura 4-76 pode ver-se que são utilizadas funções da bibliteca SW_I2C para implementar esta função. A função *SWStartI2C()* permite gerar a condição de *start* para dar início à comunicação I^2C, a função *SWWriteI2C(data_out)* permite enviar um byte para o *slave* e a função *SWStopI2C()* permite gerar a condição de *stop* para finalizar a comunicação I^2C. Tal como se verificou no estudo da comunicação I^2C, no primeiro byte envia-se o endereço do *slave* e a indicação se a operação é de leitura ou de escrita, enquanto no segundo byte são enviados os dados.

No *slave*, a inicialização do I^2C é feita através da função *OpenI2C(sync_mode, slew)*, que inicializa os registos SSPSTAT (*MSSP Status Register*), SSPCON1 (*MSSP Control Register 1*) e SSPCON2 (*MSSP Control Register 2*). Tem como variáveis de entrada o modo do MSSP, que corresponde aos bits SSPM3:SSPM0 do registo SSPCON1 e um bit para controlo do *slew rate*, que corresponde ao bit SMP do registo SSPSTAT. Optou-se pelo modo *slave* com um endereço de 7bits e por habilitar o controlo do *slew rate* para o modo *high-speed*. É ainda importante inicializar o registo SSPADD (*MSSP Address Register*) onde se define o endereço do *slave*.

A recepção do caracter pelo *slave* origina uma interrupção, que resulta na execução da função *vI2C_SlaveInterrupt()*. Para tal, é importante activar a interrupção do I^2C através do bit SSPIE do registo PIE1, habilitar a prioridade das interrupções através do bit IPEN do registo RCON, definir a interrupção de I^2C como de alta prioridade através do bit SSPIP do registo IPR1 e habilitar todas as interrupções de alta prioridade através do bit GIEH do registo INTOCON.

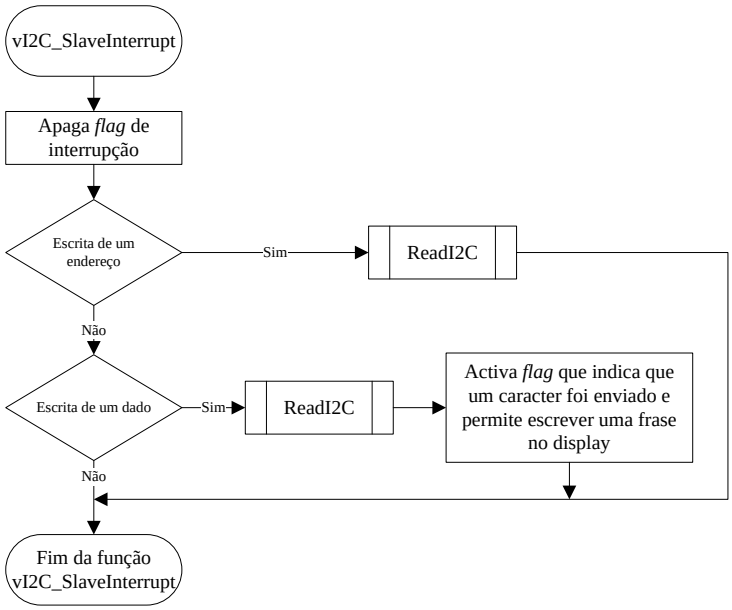


Figura 4-77: Algoritmo da função *vI2C_SlaveInterrupt()*

Na função *vI2C_SlaveInterrupt()*, cujo algoritmo é apresentado na Figura 4-77, o primeiro passo é apagar a *flag* de interrupção através do bit SSPIF do registo PIR1. De seguida, é analisado o registo SSPSTAT para verificar o estado do *slave*.

Na Figura 4-78 pode ver-se o formato deste registo, de onde se destaca o bit D/A, que indica se o byte recebido ou transmitido foi um dado ou um endereço, o bit P que indica se ocorreu uma condição de *stop*, o bit S, que indica se ocorreu uma condição de *start*, o bit R/W, que indica se o *master* está a ler ou a escrever no *slave* e o bit BF indica se o registo SSPBUF (registo no qual os dados são lidos ou escritos) se encontra ou não vazio. Há ainda que salientar que o estado do bit P é inverso ao do bit S, excepto na primeira inicialização, situação em que os dois bits estão no nível lógico "0".

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P(1)	S(1)	R/$\overline{W}$(2,3)	UA	BF
bit 7							bit 0

Figura 4-78: Registo SSPSTAT [100]

Tendo em conta que neste projecto o *slave* apenas realiza a função de leitura, os seus estados podem ser de escrita de um endereço ou de um dado pelo *master*.

No primeiro estado, os bits do registo SSPSTAT apresentam os valores da Figura 4-79.

SMP	CKE	D/$\overline{A}$	P	S	R/W	UA	BF
		0	0	1	0		1
bit 7							bit 0

Figura 4-79: Estado do SSPSTAT na escrita de um endereço no *slave*

Verifica-se que ocorreu uma condição de *start* e que o *master* escreveu um endereço no *slave*. Verifica-se ainda que o *buffer* está cheio contendo o byte de endereço, sendo, portanto, necessário ler o registo SSPBUF, para que o bit BF seja apagado. A leitura do registo é feita através da função *ReadI2C()* da biblioteca I2C.

No segundo estado, os bits do registo SSPSTAT apresentam os valores da Figura 4-80.

SMP	CKE	D/$\overline{A}$	P	S	R/W	UA	BF
		1	0	1	0		1
bit 7							bit 0

Figura 4-80: Estado do SSPSTAT na escrita de um dado no *slave*

A diferença entre esta situação e a anterior está no bit D/A, já que neste caso, indica que o último byte recebido pelo *slave* foi um dado. Neste caso, também é necessário fazer uma leitura do registo SSPBUF, sendo que o valor obtido nessa leitura indica qual o byte enviado pelo *master*. Esse byte pode ser utilizado para saber qual a informação que o *master* pretende que seja apresentada no LCD.

4.8.3 Conclusões

Constatou-se que neste protocolo o *master* é o responsável por comandar toda a comunicação, controla o sinal SCL e inicia qualquer transferência com o *slave*. Não obstante este facto, ao *slave* também é permitido transferir dados para o *master*.

Na comunicação I2C a trama de escrita do *master* para o *slave* é composta por uma condição de *start*, um byte de endereço, um ou mais bytes de dados e uma condição de stop, sendo que, o *slave* responde à recepção de cada byte com um ACK.

A comunicação por I^2C permitiu a ligação dos dois microcontroladores PIC, através de dois condutores (SCL e SDA). O PIC18F4431 foi utilizado como *master* e o PIC18F2520 como *slave*.

Concluiu-se que este método de comunicação é bastante simples e eficaz e que as bibliotecas de I^2C *Microchip* disponíveis, são uma grande ajuda no desenvolvimento do software de controlo.

Capítulo 5
Discussão

Neste capítulo, pretende-se mostrar um pouco do trabalho prático efectuado e discutir o seu desenvolvimento. Apresenta-se o primeiro protótipo da *Hollowgrail*, as placas desenvolvidas e resultados obtidos em testes funcionais.

5.1 Desenvolvimento do protótipo de encadernação

Um produto como o aqui descrito passa por muitas fases antes de chegar a uma versão final, preparada para entrar em produção industrial. O desenvolvimento passa por uma fase inicial em que o produto é desenhado e simulado, recorrendo a programas informáticos; numa fase posterior começam a ser construídos protótipos que vão evoluindo até que se chegue a uma versão final.

Os primeiros protótipos devem concentrar-se na verificação das características funcionais do equipamento face ao inicialmente previsto, deixando, para uma fase posterior, pormenores relacionados com cuidados a ter com a montagem e cablagem.

A *Hollowgrail* não fugiu à regra. O estudo sobre a viabilidade de fabricar uma máquina com as características da *Hollowgrail* começou em 2005, tendo a empresa decidido avançar com o projecto em 2007, estando previsto o inicio da produção em 2009. Ainda antes de começar a trabalhar com a empresa *ACCO Brands*, já alguns protótipos tinham sido construídos, essencialmente na parte mecânica, quer pela empresa Americana, quer pela Portuguesa (Figura 5-1).

Figura 5-1: Protótipo realizado pela *ACCO Brands* Americana

Como se pode verificar pela imagem, ainda havia muito trabalho a desenvolver antes de o produto atingir uma versão comercializável. Durante a participação do autor neste projecto, houve uma grande evolução, estando os actuais protótipos já próximos de uma versão final (Figura 5-2). No entanto, há ainda alguns pontos a melhorar, nomeadamente na selecção dos motores, brocas e algumas peças mecânicas, que apresentam algumas fragilidades e originam demasiados atritos.

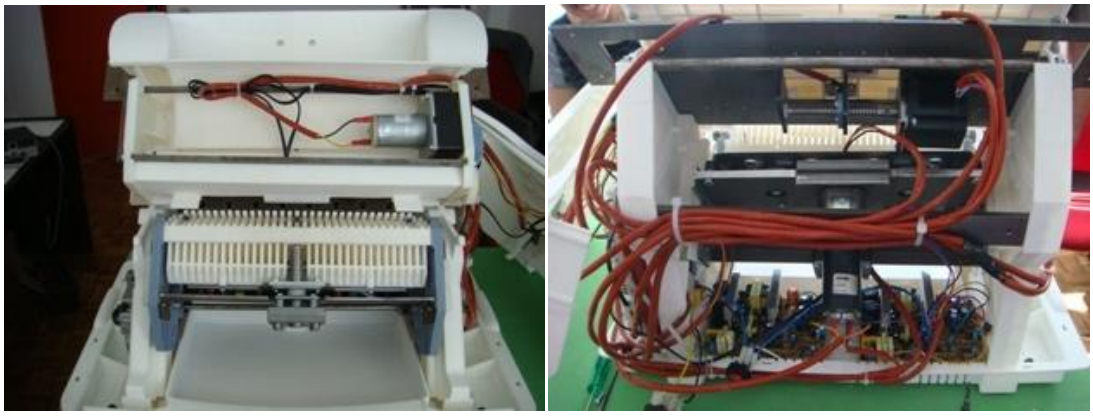


Figura 5-2: Protótipo realizado pela *ACCO Brands Portuguesa* com a colaboração do autor

Ao nível da electrónica, foram desenvolvidas duas placas para o controlo da máquina. Uma responsável por comandar o LCD (*Liquid Crystal Display*) (Figura 5-3) e, por isso, colocada na parte frontal da máquina, e outra para efectuar o controlo dos motores e sensores (Figura 5-4), que se encontram posicionados em pontos específicos da máquina.

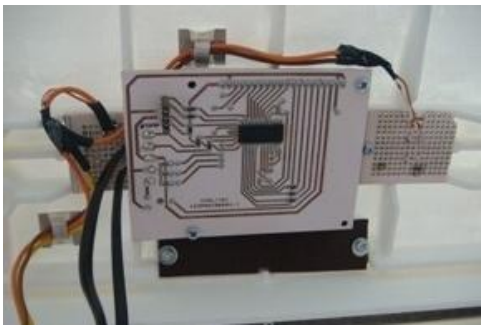


Figura 5-3: Vista da retaguarda da placa de controlo do LCD

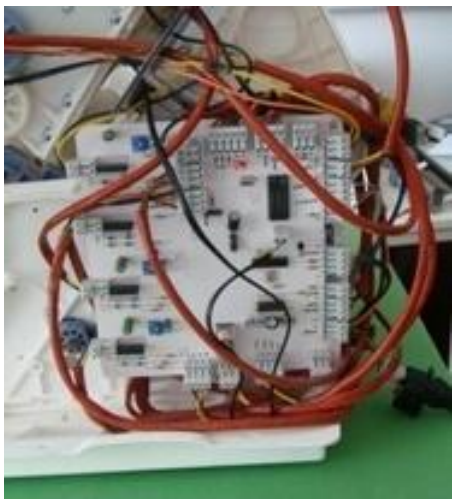


Figura 5-4: Vista frontal da placa de controlo dos motores e sensores

Na Figura 5-5, podem ver-se os PCB (*Printed Circuit Board*) utilizados. De referir que se trata de um protótipo, havendo, portanto, ainda muito a ser melhorado, tal como reduzir o tamanho da placa, seleccionar quais os componentes que deverão ser SMT (*Surface-mount technology*) ou não, fazer uma melhor distribuição dos componentes e alterar a forma de conexão das ligações aos motores e sensores.

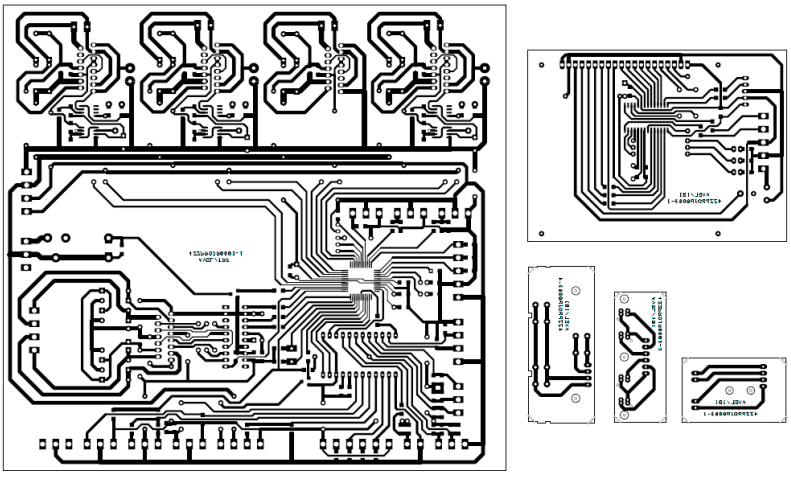


Figura 5-5: PCBs da placa de controlo dos motores e sensores, do LCD e de alguns sensores

5.2 Testes

Ao longo da implementação, foram efectuados testes de funcionalidade e de consumo, com vista a detectar desvios relativamente aos valores previstos. Os processos foram inicialmente testados em separado, primeiro a furação e, posteriormente, a encadernação. No final, juntaram-se estes dois processos.

O processo de furação foi cuidadosamente testado, começando com um número reduzido de folhas e apenas duas brocas, tendo sido realizadas várias furações. O motor DC2 foi programado para deslocar o bloco das brocas a uma velocidade elevada, até chegar perto das folhas, altura em que a velocidade é diminuída, gradualmente, para entrar nas folhas suavemente, diminuindo para tal, a tensão de alimentação aplicada ao motor. Com apenas duas brocas, este método de furação resultou sem muitos problemas, apesar de o tempo gasto ser demasiado longo, tendo em conta o objectivo para duração de uma encadernação.

Com o aumento do número de brocas e do número de folhas, os problemas começaram a surgir, devido ao facto da carga requerida aos motores aumentar. O motor DC5 passou a não ser capaz de manter a rotação, o que fez com que houvesse maior dificuldade a furar, tendo, para isso, contribuído também a fraca qualidade das brocas, que se degradavam com alguma facilidade. Estas dificuldades estão a ser ultrapassadas, por um lado, com a selecção de brocas de melhor qualidade e, por outro, com o envolvimento de um fabricante de motores, a *Crouzet*, que procura desenvolver um motor, para aquela aplicação, que necessita de velocidade e binário elevados, mantendo uma reduzida dimensão.

A necessidade de aplicar pouca tensão ao motor DC2 para o mecanismo se deslocar a uma velocidade menor, também não beneficiou a furação, visto o binário ter diminuído e, por consequência, o motor não ter força suficiente para empurrar as brocas depois de estas atingirem as folhas. Verificou-se ainda que, mesmo aumentando gradualmente a tensão, os resultados não melhoravam satisfatoriamente, tendo-se concluído que isso era motivado pelas brocas não estarem a furar convenientemente. Espera-se que este problema fique ultrapassado com a alteração do motor DC5.

Em relação ao processo de encadernação, não surgiram tantos problemas. Verificou-se que a estrutura do mecanismo de fecho da argola não era suficientemente resistente, provocando o indesejável dobrar de alguns componentes durante o fechar da argola, como resultado a argola fechava bem nos extremos, mas não tão facilmente no

centro. Esta dificuldade foi ultrapassada com o reforço da estrutura do mecanismo de fecho da argola.

Da realização de alguns testes verificou-se que com aproximadamente 50 folhas, o processo de furação demorava cerca de 50 a 55 segundos, o de encadernação rondava os 10 segundos e a fusão dos dois processos, aproximadamente 60 a 70 segundos para a versão americana. Atendendo aos objectivos definidos conclui-se que os tempos precisam ser reduzidos.

Por se tratar de um protótipo, os atritos existentes entre as peças são a principal razão para estes tempos tão elevados, espera-se que com a utilização de outros materiais de prototipagem, mais próximos do produto final, este problema seja reduzido.

Outro problema prende-se com o mecanismo comandado pelo motor de passo, o qual é relativamente lento, sendo uma das razões o esforço exercido para deslocar a mesa. Neste caso também se espera melhorias com a alteração dos materiais do protótipo, podendo no entanto, ser necessário encontrar um outro motor de passo com mais binário e velocidade, para diminuir ao máximo o tempo dispendido entre furações.

Ainda no processo de furação e com o objectivo de melhorar os tempos, pode vir a optar-se por utilizar um *encoder* no motor DC2. Apesar desta hipótese ter sido anteriormente posta de parte devido ao preço que acarreta, sabe-se que reduz bastante o tempo de furação, permitindo que sejam efectuados deslocamentos mais pequenos na furação.

Actualmente, já se encontra em desenvolvimento um novo protótipo onde, para além de alteradas algumas peças mecânicas, estão já a ser utilizados outros motores.

Capítulo 6
Conclusões e trabalho futuro

6.1 Conclusões

O principal objectivo deste trabalho consiste na automatização de uma máquina de encadernar, onde o processo de encadernação deve ser efectuado em 3 passos: colocação do documento a encadernar; colocação da argola apropriada e ordem de início do processo de furação e encadernação. A máquina deve ser capaz de furar e encadernar até 100 folhas, em 30-35 segundos, na versão da máquina para os Estados Unidos e 40 a 45 segundos na versão europeia.

Na solução proposta, salienta-se o uso de cinco motores DC e um motor de passo, para realizar o deslocamento dos vários mecanismos da máquina, em conjunto com vários sensores, responsáveis por detectar os seus estados e fins de curso e um *display* gráfico para fazer o interface com o utilizador.

Dos cinco motores DC, quatro necessitam de um controlo para variação de velocidade e inversão do sentido de rotação, enquanto que o quinto apenas necessita de um controlo *on/off*. Para os primeiros casos, o uso da variação de velocidade por PWM em conjunto com uma ponte H para inverter o sentido de circulação da corrente, revelou-se uma solução simples e de baixo custo, tendo para tal contribuído o módulo de PWM existente no PIC18F4431. No motor onde apenas se pretendia um controlo *on/off*, a utilização de um relé electromecânico mostrou-se eficaz e extremamente simples, necessitando apenas do controlo de uma saída do PIC 18F4431, para ligar ou desligar o motor.

Para o motor de passo, optou-se por uma alimentação bifilar com ligação bipolar em paralelo, que exigiu o uso de uma ponte H como *driver*, para permitir a inversão do sentido de rotação do motor. O controlador utilizado permitiu que a programação deste motor fosse extremamente simples.

A utilização de um LCD gráfico no interface com o utilizador mostrou-se um elemento de relevada importância no design da máquina, permitindo a apresentação de imagens gráficas que indicam ao utilizador os passos a efectuar, informa-o do estado da máquina e de possíveis erros que surjam durante o processo de encadernação. O hardware associado ao LCD mostrou-se bastante simples, sendo formado, essencialmente, por um barramento de dados e linhas de controlo. O desenvolvimento do software implicou um estudo aprofundado do *data sheet* do controlador T6963C, tornando-se este, em conjunto com os inúmeros fóruns existentes sobre *displays*, muito importantes para compreender o funcionamento do controlador.

Os diversos sensores utilizados mostraram-se eficazes na detecção dos fins de curso, bem como do estado de alguns mecanismos. Os sensores mecânicos revelaram-se uma solução de baixo custo, capaz de efectuar com sucesso a detecção dos mecanismos, os sensores ópticos mostraram-se uma boa alternativa, principalmente, em locais onde a sua utilização não era adequada. A utilização do sensor de corrente revelou-se também muito importante na monitorização da corrente no motor, mostrando-se uma boa alternativa aos sensores de fins de curso convencionais na detecção do estado dos mecanismos, quando não é fácil encontrar um local para a sua colocação.

É importante ainda referir que os microcontroladores PIC utilizados conseguiram responder às expectativas e demonstraram ser uma solução eficaz no controlo da máquina de encadernar. O facto de permitir a programação em diferentes linguagens tornou-se importante, uma vez que foi possível a utilização de linguagem C, simplificando e, consequentemente, diminuindo o tempo necessário à programação. A utilização de dois PIC também não acarretou problemas, sendo o I^2C uma solução acessível e que permite a comunicação entre vários dispositivos.

Concluiu-se que o objectivo principal foi cumprido, ainda que nem todos os requisitos propostos para a máquina estejam totalmente satisfeitos. Automatizou-se com sucesso a maioria dos passos, estando já identificado o percurso a seguir para a correcção dessas tarefas.

O processo de furação, onde se verificou que o motor DC5 não possui potência suficiente para realizar a rotação simultânea de oito brocas, principalmente, com o aumento do número de folhas, precisa ainda de alguns ajustes. É de notar que este problema se encontra actualmente a ser resolvido, tendo sido já adquirido um motor com uma potência superior e aplicado no novo protótipo.

O processo de encadernação encontra-se a funcionar, e não obstante todas as dificuldades que surgiram no desenvolvimento do projecto, conseguiu-se sempre encontrar soluções viáveis que as ultrapassassem. Encontra-se actualmente já desenvolvido um protótipo num estado mais avançado, faltando realizar alguns acertos. O trabalho tem evoluído favoravelmente o que leva a concluir que, no tempo previsto, a máquina se encontre em perfeitas condições para entrada em produção.

Há ainda a salientar que este projecto foi um desafio muito estimulante, permitiu adquirir novos conhecimentos, aprofundar e pôr em prática muitos outros adquiridos durante o curso.

Foi possível abranger áreas muito diversas, como o estudo e comando de motores DC, motores de passo, microcontroladores, sensores e *displays*, bem como o software e hardware necessários ao controlo destes dispositivos.

O contacto adquirido com as empresas e o mundo do trabalho foi extremamente enriquecedor, a possibilidade de transportar para a prática os conhecimentos teóricos acumulados ao longo do curso foi uma oportunidade fundamental para a transição entre a academia e a vida empresarial. Foi importante a dimensão prática e económica subjacentes ao projecto, bem como o contacto com a programação de *timings* e a disciplina de empresa. O facto da empresa promotora pertencer a um grupo multinacional permitiu adquirir também uma experiência e dimensão muito relevante.

Por se tratar de uma máquina a ser construída para comercialização em grande escala, todos os detalhes têm uma importância fundamental nos custos associados à produção, de forma a que este projecto obtenha um eventual sucesso. Percebeu-se que devem ser considerados, não só o preço dos componentes, como a forma como vão ser montados e interligados e todos os tempos associados a cada operação de produção. Importa realçar que a forma de realizar pequenas tarefas, que normalmente parecem totalmente irrelevantes, podem fazer toda a diferença entre a viabilidade ou não de um projecto.

Foi submetido o pedido de patente para este sistema aguardando-se uma resposta.

Sendo esta máquina de encadernação completamente inovadora, e sem concorrência no mercado, a empresa *ACCO Brands* pretende colocar este produto no mercado a uma escala mundial.

6.2 Trabalho futuro

Todos os produtos necessitam de um contínuo processo de desenvolvimento, por forma a melhorar as suas funções e atrair novos mercados. Este produto não é excepção e vai continuar a necessitar de melhoramentos.

As intervenções a ser realizadas na Hollowgrail:

Algumas das sugestões para trabalho futuro já se iniciaram. Neste momento decorre o desenvolvimento de um novo protótipo com o objectivo de melhorar a sua funcionalidade. Uma das intervenções passa pela alteração de alguns motores e de algumas peças mecânicas. Outra intervenção está a acontecer na alteração de alguns dos materiais utilizados. Está também a equacionar-se a utilização de um *encoder* no motor DC2, por forma a reduzir o tempo dispendido durante a furação, se os custos associados a esse componente conseguirem ser diluídos pelo volume de máquinas comercializadas.

Uma vez que a empresa pretende vender este produto à escala mundial, torna-se necessário traduzir o texto das imagens gráficas para outras línguas, como alemão, espanhol e francês, para permitir a selecção da língua por parte do utilizador.

É importante ainda realizar vários testes, mesmo em situações adversas, com o intuito de testar todas as funcionalidades da máquina e encontrar possíveis bugs.

Após a decisão final sobre os motores a utilizar, devem ser efectuados testes ao consumo dos motores, de forma a escolher uma fonte de alimentação adequada.

Por se tratar de um projecto que se pretende rentável, este implica um estudo pormenorizado de todas as fases de fabrico de forma a ser possível minimizar os tempos de montagem e reduzir os custos de produção.

Ainda com vista ao controlo destes custos, deverão ser avaliadas junto dos fabricantes de cablagens, as diversas alternativas de cablagens para os sensores.

O desenvolvimento de novos produtos tendo por base o trabalho na *Hollowgrail*:

Pode ser proposto que o trabalho base aplicado no desenvolvimento da Hollowgrail, seja utilizado para outras máquinas de encadernar, preparadas para outro tipo de argolas, como é o caso das *wirebind* e *zipbind*, cuja base de funcionamento não difere muito das *clickbind*.

Referencias

[1] http://www.usp.br/espacoaberto/arquivo/2002/espaco24out/vaipara.php?materia=0varia
CALDEIRA, Cinderela. “Do papiro ao papel manufacturado”
Consultado em Abril/Maio de 2008

[2] http://www.sinonet.com.br/cultura/cultura_ler.asp?idioma=1&cadid=77&cat=7
“A Origem dos Livros Chineses e Tipografia”
Consultado em Abril/Maio de 2008

[3] http://www.schoyencollection.com/scribes.htm
“Scribal activity, school and learning”
Consultado em Maio de 2008

[4] http://en.wikipedia.org/wiki/Bookbinding
“Bookbinding - Wikipedia”
Consultado em Maio de 2008

[5] http://wagnersilva.com.br/portfolio/sites/site_prodamec/historia.html
Consultado em Abril/Maio de 2008

[6] http://www.ancientegypt.co.uk/writing/explore/scroll.html
“Papyrus scroll”
Consultado em Outubro de 2008

[7] http://www.ufac.br/informativos/ufac_imprensa/2003/10out_2003/artigo985.html
FERREIRA, Dalmir. “Escritores homenageiam o livro”
Consultado em Abril/Maio de 2008

[8] http://www.metmuseum.org/explore/Jde/jdesplash.htm
“The Hours of Jeanne d'Evreux”
Consultado em Maio de 2008

[9] http://www.unices.com.br/html2/modules/news/article.php?storyid=99
LIMA, Ilane; AZEREDO, Rosany. “O livro e seus principais suportes: papiro, pergaminho e papel”
Consultado em Abril/Maio de 2008

[10] http://www.escritoriodolivro.org.br/historias/encadernacao.html
Consultado em Abril de 2008

[11] ACCO Brands. “Document Finishing Evolution - 60 Years in the Making”. 2007

[12] http://ezinearticles.com/?Choosing-A-Binding-System---Six-Reasons-To-Consider-Velobind&id=917764
MCRITCHIE, Jeff. “Choosing a Binding System - Six Reasons to Consider Velobind”

Consultado em Maio de 2008

[13] https://www.laminator.com/Product_Details~ProductID~332.htm
"GBC VeloBind SureBind V800PRO Binding Machine from LAMINATOR.COM"
Consultada em Maio de 2008

[14] http://ezinearticles.com/?GBC-Velobind-Essentials&id=971438
MCRITCHIE, Jeff. "GBC Velobind Essentials"
Consultado em Maio de 2008

[15] http://www.probinding.com/SuppliesByColor.aspx?SubCategoryID=3
"VeloBind Supplies, Strips"
Consultado em Maio de 2008

[16] http://ezinearticles.com/?GBC-SureBind-An-Introduction-to-Strip-Binding-Systems&id=843528
MCRITCHIE, Jeff. "GBC SureBind an Introduction to Strip Binding Systems"
Consultado em Maio de 2008

[17] http://ezinearticles.com/?Understanding-the-Difference-Between-the-Four-Different-Types-of-GBC-Velobind-Strips&id=843539
MCRITCHIE, Jeff. "Understanding the Difference Between the Four Different Types of GBC Velobind Strips"
Consultado em Maio de 2008

[18] http://uk.accoimages.com/imagesList/default.aspx?s=3&c=F&b=44
"ACCO ImageBank"
Consultado em Maio de 2008

[19] http://ezinearticles.com/?What-is-the-Difference-Between-Velobind-and-Surebind?&id=763709
MCRITCHIE, Jeff. "What is the Difference Between Velobind and Surebind?"
Consultado em Maio de 2008

[20] http://www.acco.co.uk/business_machines/submenu.asp?subsection=510&category=2
"ACCO UK - Strip Binders"
Consultado em Maio/Outubro de 2008

[21] http://www.gbcconnect.com/subcategory.aspx?parent=Bind&sub=Equipment&product=Finishers
"GBC - Finishers"
Consultado em Maio/Outubro de 2008

[22] http://ezinearticles.com/?Five-Tips-for-Binding-Documents-with-Wire-O&id=932042
MCRITCHIE, Jeff. "Five Tips for Binding Documents with Wire-O"
Consultado em Maio de 2008

[23] http://www.boundtoimpress.com.au/shop/
"Bound to Impress, Online Orders"
Consultado em Maio de 2008

[24] http://ezinearticles.com/?An-Introduction-to-Twin-Loop-Wire-Binding&id=914477
MCRITCHIE, Jeff. "An Introduction to Twin Loop Wire Binding"
Consultado em Maio de 2008

[25] http://ezinearticles.com/?Choosing-a-Binding-System---Four-Reasons-to-Choose-Twin-Loop-Wire-Binding&id=915114
MCRITCHIE, Jeff. “Choosing a Binding System - Four Reasons to Choose Twin Loop Wire Binding”
Consultado em Maio de 2008

[26] http://www.probinding.com/SuppliesMixed.aspx?SubCategoryID=4
“Wire Binding Supplies”
Consultado em Maio de 2008

[27] http://www.acco.co.uk/business_machines/submenu.asp?subsection=9&category=2
“ACCO UK - Wire Binders”
Consultado em Maio/Outubro de 2008

[28] http://www.gbcconnect.com/subcategory.aspx?parent=Bind&sub=Equipment&product=Punches
“GBC - Punches”
Consultado em Maio/Outubro de 2008

[29] http://www.esselte.co.uk/enGB/Categories/wireBIND_Machine.html
“Esselte - wireBIND 500e”
Consultado em Maio/Outubro de 2008

[30] http://www.fellowes.com/Fellowes/site/products/productssubcategory.aspx?loc=left&Name=WIRE_COIL_BINDING
“Binding Machines - Wire”
Consultado em Maio/Outubro de 2008

[31] http://www.peach.info/product_profile/bin_products.html
“Peach Office”
Consultado em Junho/Outubro de 2008

[32] http://www.renz-germany.de/a_homepage_englisch/frame_index_eng.htm
“Renz – Wire comb binding systems – for professional presentation”
Consultado em Maio/Outubro de 2008

[33] http://ezinearticles.com/?Book-Binding-with-Spiral-Coil&id=967589
MCRITCHIE, Jeff. “Book Binding with Spiral Coil”
Consultado em Maio/Junho de 2008

[34] http://ezinearticles.com/?Understanding-The-Different-Pitches-Of-Color-Coil-Binding&id=891151
MCRITCHIE, Jeff. “Understanding the Different Pitches of Color Coil Binding”
Consultado em Maio/Junho de 2008

[35] http://ezinearticles.com/?Choosing-a-Binding-System---Six-Reasons-to-Choose-Spiral-Coil-Binding&id=917599
MCRITCHIE, Jeff. “Choosing a Binding System - Six Reasons to Choose Spiral Coil Binding”
Consultado em Maio/Junho de 2008

[36] http://www.gbcconnect.com/skugroup.aspx?pid=9665000G&parent=Bind&sub=Supplies&product=Spines

"ColorCoil"
Consultado em Maio de 2008

[37] http://ezinearticles.com/?Six-Tips-for-Binding-Documents-With-Spiral-Coil&id=1026382
MCRITCHIE, Jeff. "Six Tips for Binding Documents with Spiral Coil"
Consultado em Maio/Junho de 2008

[38] http://www.acco.co.uk/business_machines/submenu.asp?subsection=511&category=2
"ACCO UK - Coil Binders"
Consultado em Junho/Outubro de 2008

[39] http://www.renz-germany.de/a_homepage_englisch/frame_index_eng.htm
"RENZ - spiral binding systems"
Consultado em Junho/Outubro de 2008

[40] http://ezinearticles.com/?Choosing-a-Binding-System---Five-Reasons-to-Choose-Plastic-Comb-Binding&id=915046
MCRITCHIE, Jeff. "Choosing a Binding System - Five Reasons to Choose Plastic Comb Binding"
Consultado em Maio/Junho de 2008

[41] http://ezinearticles.com/?Plastic-Comb-Binding-Essentials&id=971468
MCRITCHIE, Jeff. "Plastic Comb Binding Essentials"
Consultado em Maio/Junho de 2008

[42] http://ezinearticles.com/?A-Brief-Introduction-To-Plastic-Comb-Binding&id=914251
MCRITCHIE, Jeff. "A Brief Introduction to Plastic Comb Binding"
Consultado em Maio/Junho de 2008

[43] http://www.acco.co.uk/business_machines/submenu.asp?subsection=7&category=2
"ACCO UK - Comb Binders"
Consultado em Junho/Outubro de 2008

[44] http://www.esselte.co.uk/enGB/Categories/comBIND_Machines.html
"Esselte - comBIND Machines"
Consultado em Junho/Outubro de 2008

[45] http://www.fellowes.com/Fellowes/site/products/ProductsSubCategory.aspx?loc=center&Name=PLASTIC_COMB_BINDING
"Binding Machines - Plastic Comb"
Consultado em Junho/Outubro de 2008

[46] http://www.renz-germany.de/a_homepage_englisch/frame_index_eng.htm
"RENZ - Plastic Comb Binding Systems - for professional presentations"
Consultado em Junho/Outubro de 2008

[47] http://ezinearticles.com/?Choosing-a-Binding-System---Four-Reasons-to-Choose-GBC-Zipbind&id=915089
MCRITCHIE, Jeff. "Choosing a Binding System - Four Reasons to Choose GBC Zipbind"
Consultado em Junho de 2008

[48] http://www.epinions.com/offc-Supplies-All-Gbc_ZipBind_Editable_Spines_3_8_Dia_55-Sheet_Capacity_Black_25_Pieces_Box
"Compare prices on GBC ZipBind Editable Spines 3/8" Dia. 55-Sheet Capacity Black 25 Pieces/Box"
Consultado em Outubro de 2008

[49] http://ezinearticles.com/?An-Introduction-To-GBC-ZipBind-Binding&id=914279
MCRITCHIE, Jeff. "An Introduction to GBC ZipBind Binding"
Consultado em Junho de 2008

[50] http://ezinearticles.com/?How-to-Bind-Documents-With-Zipbind&id=1026042
MCRITCHIE, Jeff. "How to Bind Documents with Zipbind"
Consultado em Junho de 2008

[51] http://www.gbcconnect.com/skugroup.aspx?pid=4100501&parent=Bind&sub=Supplies&product=Spines
"ZipBind"
Consultado em Junho de 2008

[52] http://www.amazon.com/Plastic-ZipBind-Binding-90-Page-Capacity/dp/B0000AQOMR
"Amazon.com: GBC(R) Plastic ZipBind™ Binding Spines"
Consultado em Junho de 2008

[53] http://ezinearticles.com/?An-Introduction-to-GBC-Proclick-Binding&id=914262
MCRITCHIE, Jeff. "An Introduction to GBC Proclick Binding"
Consultado em Junho de 2008

[54] http://www.acco.co.uk/business_machines/submenu.asp?subsection=509&category=2
"ACCO UK - Click Binders"
Consultado em Junho/Outubro de 2008

[55] KOSTENKO, M. P; PIOTROVSKI, L. M. "Máquinas eléctricas I". Editorial Mir Moscú, 1975. Introducción, pág.11.

[56] http://pt.wikipedia.org/wiki/Motor_de_corrente_cont%C3%ADnua
"Máquina de corrente contínua"
Consultado em Julho de 2008

[57] AFONSO, João L. "Sebenta de Máquinas Eléctricas".

[58] http://ww1.microchip.com/downloads/en/AppNotes/00905a.pdf
"Brushed DC Motor Fundamentals"
Consultado em Julho de 2008

[59] CHAPMAN, Stephen J. "Electric Machinery Fundamentals". McGraw-Hill, 2º Edição, 1991. Cap.6, pág.340-343.

[60] http://www.dorlingkindersley-uk.co.uk/nf/ClipArt/Thumbnail/0,,239022_,00.html
"Dorling Kindersley Clip Art image page"
Consultado em Julho de 2008

[61] BRAGA, Newton C. "Electrónica Básica para Mecatrónica". Editora Saber. Cap.2

[62] http://www.crouzet.com/ngcc/pdf/ENG_NGCC_LD.pdf

"D.C. motors and geared motors from 1 to 11 W"
Consultado em Outubro de 2007

[63] CROUZET. "Data Sheet 82740". 2008

[64] http://www.elmeqmotor.es/produits-mdp-2/catalogue/fiche.asp?idGamme=8&aff=car&url=http://www.elmeqmotor.es/produits-mdp-2/catalogue/liste.asp?idGamme=3&idVersion=10126
3863HC/PLG52 - Motoreductor - Catalogue MDP
Consultado em Janeiro de 2007

[65] DOGA. "Catalogo Doga" 2007

[66] http://www.sgmada.com/zl/pdf/TT38246800.pdf
"TT-38246800 Series"
Consultado em Maio de 2007

[67] http://www.riobotz.com.br/Tutorial%20RioBotz.pdf
MEGGIOLARO, Marco; MAIMON, Felipe; et. al. "Tutorial em Robôs de Combate"
Consultado em Agosto de 2008

[68] http://commons.wikimedia.org/wiki/Image:Rele_2.jpg
"Image:Rele 2.jpg"
Consultado em Agosto de 2008

[69] http://www.mcmanis.com/chuck/robotics/tutorial/h-bridge/index.html
"H-Bridge Theory & Practice -- Chuck's Robotics Notebook"
Consultado em Agosto de 2008

[70] BRAGA, Newton C. "Electrónica Básica para Mecatrónica". Editora Saber. Cap.5

[71] BRAGA, Newton C. "Electrónica Básica para Mecatrónica". Editora Saber. Cap.6

[72] http://www.allegromicro.com/en/Products/Part_Numbers/3968/3968.pdf
"A3968 - Dual Full-Bridge PWM Motor Driver"
Consultado em Novembro de 2007

[73] http://www.farnell.com/datasheets/73945.pdf
"MC33886 - 5.0A H-Bridge"
Consultado em Novembro de 2007

[74] http://cache.national.com/ds/LM/LMD18201.pdf
"LMD18201 - 3A, 55V H-Bridge"
Consultado em Novembro de 2007

[75] http://www.st.com/stonline/products/literature/ds/1773/l298.pdf
"L298 - DUAL FULL-BRIDGE DRIVER"
Consultado em Novembro de 2007

[76] http://www.datasheetcatalog.org/datasheet/GeneralSemiconductor/mXyzqxyq.pdf
"BYW32 THRU BYW36"
Consultado em Novembro de 2007

[77] BRAGA, Newton C. "Electrónica Básica para Mecatrónica". Editora Saber. Cap.7

[78] http://ww1.microchip.com/downloads/en/AppNotes/00822a.pdf

"Stepper Motor Microstepping with PIC18C452"
Consultado em Agosto de 2008

[79] http://members.fortunecity.com/camusp/relatorios/pmc527Acionamentosparamecatronica/MotordeRelutanciaChaveada.doc
"Motor de Passo de Relutância Variável"
Consultado em Agosto de 2008

[80] DEL TORO, Vicent. "Fundamentos de máquinas eléctricas". Prentice Hall do Brasil, 1994. Cap.11, pág.406-408

[81] http://www.freescale.com/files/microcontrollers/doc/app_note/AN2974.pdf
"Quick Start for Beginners to Drive a Stepper Motor"
Consultado em Agosto de 2008

[82] http://www.embedded.com/story/OEG20030410S0057
SIMON, Dan. "Get Your Motor Running"
Consultado em Agosto de 2008

[83] http://www.allaboutcircuits.com/vol_2/chpt_13/5.html
"Stepper motors"
Consultado em Agosto de 2008

[84] http://zone.ni.com/devzone/cda/ph/p/id/287
"Types of Stepper Motors (detailed) - Developer Zone - National Instruments"
Consultado em Agosto de 2008

[85] MACHADO, J. A. Tenreiro. "Motores Passo a Passo - Controlo e Modos de Funcionamento". Porto 1994. Cap.2, pág.3-9.

[86] MACHADO, J. A. Tenreiro. "Motores Passo a Passo - Controlo e Modos de Funcionamento". Porto, 1994. Cap.3, pág.15-26.

[87] http://pasta.ebah.com.br/download/motor-de-passo-pdf-5479
SOUZA, Paulo. "Motor de Passo"
Consultado em Setembro de 2008

[88] http://www.cs.uiowa.edu/~jones/step/types.html
JONES, Douglas. "Control of Stepping Motors"
Consultado em Agosto de 2008

[89] http://www.answers.com/topic/stepper-motor
"Stepper motor: Definition from Answers.com"
Consultado em Agosto de 2008

[90] http://www.ams2000.com/stepping101.html
"Step Motors Reference Guide"
Consultado em Setembro de 2008

[91] http://en.nanotec.com/steppermotor_st5918.html
"Stepping motor - ST5918- Nema 23 - Nanotec"
Consultado em Janeiro de 2008

[92] http://hades.mech.northwestern.edu/wiki/index.php/Stepper_Motor_Circuits

"Stepper Motor Circuits - Mechatronics Wiki"
Consultado em Setembro de 2008

[93] http://www.onsemi.com/pub_link/Collateral/MC3479-D.PDF
"MC3479 - Stepper Motor Driver"
Consultado em Janeiro de 2008

[94] http://www.st.com/stonline/products/literature/ds/1334/l297.pdf
"L297 - STEPPER MOTOR CONTROLLERS"
Consultado em Janeiro de 2008

[95] PEREIRA, Nino. "Curso de Programação em C". 2008

[96] http://ww1.microchip.com/downloads/en/DeviceDoc/51519B.pdf
"MPLAB IDE – User's guide"
Consultado em Outubro de 2007

[97] http://www.mikroe.com/en/books/picmcubook/ch0/
"mikroElektronika - PIC Microcontrollers - Introduction: World of microcontrollers"
Consultado em Setembro de 2008

[98] http://dqsoft.blogspot.com/2008_08_01_archive.html
"DQSoft: Agosto 2008"
Consultado em Outubro de 2008

[99] http://ww1.microchip.com/downloads/en/DeviceDoc/DS-39630d.pdf
"8-bit PIC® Microcontroller Solutions"
Consultado em Setembro de 2008

[100] http://ww1.microchip.com/downloads/en/DeviceDoc/39616C.pdf
"Data Sheet PIC18F2331/2431/4331/4431"
Consultado em Outubro de 2007

[101] http://ww1.microchip.com/downloads/en/DeviceDoc/39631D.pdf
"Data Sheet PIC18F2420/2520/4420/4520"
Consultado em Outubro de 2007

[102] http://ww1.microchip.com/downloads/en/DeviceDoc/51549d.pdf
"Integrated development environment"
Consultado em Setembro de 2008

[103] http://www.best-microcontroller-projects.com/article-programming-a-microcontroller.html
"C, Pascal or BASIC for microcontroller programming"
Consultado em Setembro de 2008

[104] http://ww1.microchip.com/downloads/en/DeviceDoc/51295E.pdf
"MPLAB C18 - C Compiler Getting Started"
Consultado em Outubro de 2007

[105] http://ww1.microchip.com/downloads/en/DeviceDoc/51331C.pdf
"MPLAB ICD 2 – In Circuit Debugger User's Guide"
Consultado em Outubro de 2007

[106] BRAGA, Newton C. "Comparação de *displays*". Saber Electrónica, Editora Saber Ltda, Ano 44, nº 422. Março, 2008. pág.18-22.

[107] http://en.wikipedia.org/wiki/Light-emitting_diode
"Light-emitting diode"
Consultado em Setembro de 2008

[108] http://www.mckennabrothers.co.uk/flipdot.html
"McKenna Brothers - Flip Dot Signs"
Consultado em Outubro de 2008

[109] http://en.wikipedia.org/wiki/Nixie_tube
"Nixie tube"
Consultado em Outubro de 2008

[110] http://technabob.com/blog/2007/05/07/metro-retro-led-dot-clocks/
"Metro retro led dot clocks on"
Consultado em Novembro de 2008

[111] http://www.devicelink.com/mem/archive/98/10/011.html
"Electroluminescent Technology"
Consultado em Setembro de 2008

[112] http://www.planarembedded.com/products/el/
"Planar Embedded: Electroluminescent Display Technology, Transparent Displays"
Consultado em Setembro de 2008

[113] http://barrett-group.mcgill.ca/teaching/liquid_crystal/LC02.htm
"BRG - Introduction to Liquid Crystals"
Consultado em Novembro de 2008

[114] http://ericafb.sites.uol.com.br/Quimica/Texto6.htm
"Mundo da Keka"
Consultado em Novembro de 2008

[115] http://paginas.terra.com.br/negocios/ineplast/prosystem/monitores_lcd.htm
"LCD - Display de Cristal Líquido"
Consultado em Setembro de 2008

[116] http://www.britannica.com/EBchecked/topic-art/343093/59986/A-twisted-nematic-cell-In-the-off-state-in-the
"Twisted Nematic cell - liquid crystal display - Britannica Online Encyclopedia"
Consultado em Novembro de 2008

[117] http://fluidos.usp.br/atualprof/ATufaile_arquivo3_IMFCx07.pdf
"Princípio de funcionamento de um LCD-TN"
Consultado em Novembro de 2008

[118] http://www.altadox.com/lcd/knowledge/lcd_liquid_type.htm
"LCD Liquid Types"
Consultado em Novembro de 2008

[119] http://www.presentationtek.com/2008/01/25/lcd-liquid-crystal-display-technology-overview/

"LCD - Liquid Crystal Display - Technology Overview"
Consultado em Setembro de 2008

[120] http://www.cmo.com.tw/opencms/cmo/technology/Production_Process/?__locale=en
"Chi Mei Optoelectronics - Production Process"
Consultado em Setembro de 2008

[121] http://www.sigmaaldrich.com/catalog/search/TablePage/19353440
"OLED and PLED Materials"
Consultado em Setembro de 2008

[122] http://www.scienceofspectroscopy.info/edit/index.php?title=OLED
"OLED - The Science of Spectroscopy"
Consultado em Setembro de 2008

[123] http://ourdigitallife.wordpress.com/2008/06/25/19/
"Nem LCD, nem Plasma.. OLED!"
Consultado em Novembro de 2008

[124] http://en.wikipedia.org/wiki/Organic_light-emitting_diode
"Organic light-emitting diode"
Consultado em Setembro de 2008

[125] http://www.cc.gatech.edu/classes/AY2005/cs7470_fall/papers/how_plasma_works.html
"Howstuffworks - How Plasma Displays Work"
Consultado em Setembro de 2008

[126] http://www.mitsutech.com/RA8863.html
"RA8863_LCM.jpg"
Consultado em Setembro de 2008

[127] SDEC. "LCD Module Description: SF12L64DLGB-E". 2007

[128] http://www.micropic.com.br/noronha/Informatica/HARDWARE/Sensores%20.pdf
"Sensores"
Consultado em Outubro de 2008

[129] COUTO, Carlos. "Sebenta de Instrumentação e medidas"

[130] http://www.engprod.ufjf.br/epd_automacao/EPD030_Sensores.pdf
"Sensores Industriais"
Consultado em Outubro de 2008

[131] http://www.gmsie.usp.br/Downloads/Download.aspx?FileID=118
"Sensores e transdutores"
Consultado em Outubro de 2008

[132] BRAGA, Newton C. "Electrónica Básica para Mecatrónica", Editora Saber. Cap.8-9

[133] http://www.diytrade.com/china/4/products/545332/SNAP_ACTION_MICRO_SWITCH_KW4A_SERIES.html
"Snap action micro switch KW4A series – Dongnan"
Consultado em Novembro de 2008

[134] http://www.dextermag.com/Sensors.aspx

"Reed Switch Sensor, Hall Effect Sensor and Magnetoresistive Sensor - Dexter Magnetics"
Consultado em Novembro de 2008

[135] http://www.robodacta.com.mx/activacioncart-producto.asp?ProductoID=79&CategoriaID=27&SubCategoriaID=45
"RoboDacta - Tu Portal al Mundo de la Robótica Didáctica"
Consultado em Novembro de 2008

[136] http://www.robot-electronics.co.uk/htm/using_the_i2c_bus.htm
"Using the I2C Bus"
Consultado em Outubro de 2008

[137] http://www.ucpros.com/work%20samples/Microcontroller%20Communication%20Interfaces%202.htm
"Microcontroller Interfaces, Part 2"
Consultado em Outubro de 2008

[138] http://ww1.microchip.com/downloads/en/AppNotes/00734b.pdf
Using the PIC® Devices' SSP and MSSP Modules for Slave I^2C^{TM} Communication
Consultado em Outubro de 2008

Bibliografia

[1] http://ezinearticles.com/?expert=Jeff_McRitchie&o=title
MCRITCHIE, Jeff. "EzineArticles.com Expert Author"
Consultado em Maio/Junho de 2008

[2] DEL TORO, Vicent. "Fundamentos de máquinas eléctricas". Prentice Hall do Brasil, 1994

[3] Chapman, Stephen J. "Electric machinery Fundamentals". McGraw-Hill International Edition, 2º edição, 1991

[4] Kostenko, M. P; Piotrovski, L. M. "Máquinas eléctricas I". Editorial MIR, 1975

[5] SOARES, Márcio José. "Placa drive II", Mecatrónica Fácil. Editora Saber, Ano 4, Nº 25, Nov-Dez 2005

[6] OGUIC, P. "Commande de puissance pour moteurs à C.C", Electronique Pratique. Ed. Transoceanic, Nº 314, Março 2007

[7] "Controle de motor DC com escovas", Saber Electrónica. Editora Saber, Ano 43, Nº 420, Janeiro 2008

[8] Machado, J. A. Tenreiro. "Motores passo a passo Controlo e Modos de Funcionamento". Publindústria: Edições técnicas, 1º edição, 1994.

[9] http://ww1.microchip.com/downloads/en/DeviceDoc/39616C.pdf
"Data Sheet PIC18F2331/2431/4331/4431"
Consultado em Outubro de 2007

[10] http://ww1.microchip.com/downloads/en/DeviceDoc/39631D.pdf
"Data Sheet PIC18F2420/2520/4420/4520"
Consultado em Outubro de 2007

[11] http://www.mikroe.com/en/books/picmcubook/ch0/
"mikroElektronika - Free Online Book - PIC Microcontrollers"
Consultado em Setembro de 2008

[12] http://www.mikroe.com/pt/product/books/picbook/00.htm
Microcontroladores PIC
Consultado em Setembro de 2008

[13] SDEC. "LCD Module Description: SF12L64DLGB-E". 2007

[14] BRAGA, Newton C. "Comparação de displays", Saber Electrónica. Editora Saber, Ano 44, Nº 422, Março 2008

[15] BOYER, Etienne. "Os segredos do barramento I^2C", Elektor. Bolina Grupo Editorial, Nº 281, Maio 2008

Anexo I – Código C

```
/* ------------------------------------------------------------------------------------------------------------------
 * Nome do Projecto: Hollowgrail
 * Cliente: ACCO
 *
 * Nome do Ficheiro: lcd.h
 * Descrição do Ficheiro: Definições para utilizar um LCD com controlador
 *                        T6963C usando comunicação i2c
 *
 * Data: 30.10.2007
 * ------------------------------------------------------------------------------------------------------------------ */

#include "p18f2520.h"
#include "delays.h"
#include "string.h"
#include "i2c.h"

/* ********************************* Configuration Bits ********************************* */

#pragma config OSC = HS         // Oscilador do tipo "High-Speed Crystal Resonator"
#pragma config WDT = OFF        // Watchdog Timer
#pragma config LVP = OFF        // Low Voltage ICSP

/* ------------------------------------------- Fim Configuration Bits ------------------------------------------- */

/* ********************************** Pinos de controlo ********************************** */

// Pinos de controlo do display
// Pino C/D (Register Select)
#define LCD_CD          LATAbits.LATA0      // LCD_CD = 0 Entrada de dados
                                            // LCD_CD = 1 Entrada de instruções de comando
// Pino WR (Write)
#define LCD_WR          LATAbits.LATA1      // LCD_WR = 0 Escrita no LCD
// Pino RD (Read)
#define LCD_RD          LATAbits.LATA2      // LCD_RD = 0 Leitura do LCD
// Pino CE (Enable)
#define LCD_CE          LATAbits.LATA3      // LCD_CE = 0 Display fica habilitado
// Pino RST (Reset)
#define LCD_RST         LATAbits.LATA4      // LCD_RST = 0 -> Faz o reset
// Pino FS (Font Select)
#define LCD_FS          LATAbits.LATA5      // LCD_FS = 1 -> 6x8; LCD_FS = 0 -> 8x8;
// Pinos DB0..DB7 - Se forem alterados os pinos é necessário alterar as definições das
// mascaras e as funções vLCD_DataTris, hLCD_DataIn e hLCD_DataOut
#define LCD_DATAB   LATB
#define LCD_DATAC   LATC

// Definição das entradas e saídas do display
#define LCD_CD_TRIS         TRISAbits.TRISA0    // LCD_.._TRIS = 0 Saída
#define LCD_WR_TRIS         TRISAbits.TRISA1    // LCD_.._TRIS = 1 Entrada
#define LCD_RD_TRIS         TRISAbits.TRISA2
#define LCD_CE_TRIS         TRISAbits.TRISA3
```

```
#define LCD_RST_TRIS            TRISAbits.TRISA4
#define LCD_FS_TRIS             TRISAbits.TRISA5
#define LCD_DATAB_TRIS          TRISB
#define LCD_DATAC_TRIS          TRISC

/* ------------------------------------------- Fim Pinos de controlo ------------------------------------------- */

/* ************************************ Definições ************************************ */

// Definições gráficas do LCD
/* ---------------------------------------------------------------------------------------------------------------
        LCD 128x64
        Font 8x8 ou 6x8
        Text home address - 0x0000 to 0x03FF
        Text size - 128/8 * 64/8 = 128 bytes with 8x8 font (usou-se 1k byte,
                                    porque com uma letra diferente podemos precisar de mais)
        Text area - 128/8 with 8x8 font
        Graphic home address - 0x0400 to 0x0?00
        Graphic size - 128/8 * 64 = 1024 bytes
        Graphic area - 128/8 with 8x8 font
--------------------------------------------------------------------------------------------------------------- */
#define lcd_th  0x0000                          // Endereço inicial da memória de texto
#define lcd_gh  0x0400                          // Endereço inicial da memória gráfica
#define lcd_pixel_x  128                        // Largura do display
#define lcd_pixel_y  64                         // Altura do display
#define lcd_caracter_x  8                       // 6x8 - 6; 8x8 - 8
#define lcd_ta  lcd_pixel_x/lcd_caracter_x      // Numero de colunas no texto
#define lcd_ga     lcd_ta                       // Numero de colunas nos gráficos
#define lcd_ts     lcd_ta*(lcd_pixel_y/8)       // Tamanho do texto
#define lcd_gs     lcd_ga*lcd_pixel_y           // Tamanho dos gráficos

// Definições das mascaras para os portos B e C
#define maskB    0b11000000
#define maskC    0b00111111

// Definição do endereço do LCD para a comunicação i2c
#define ADDR_LCD        0b10010000

/* ------------------------------------------------ Fim Definições ------------------------------------------------ */

/* ****************************** Declaração das Variáveis ****************************** */

char i2cData;                       // Variável que guarda os valores recebidos no pino SDA
unsigned char i2cStat;              // Variável que guarda o valor presente no registo de estado
unsigned char flagInterrupt = 0;    // Flag que indica que houve recepção de um dado e permite escrever
                                    // no LCD
/* ---------------------------------------- Fim Declaração das Variáveis ---------------------------------------- */

/* ****************************** Declaração das Funções ****************************** */

// Funções do display

// Inicializa pinos de dados do LCD como entrada ou saída
// Variável de entrada: bIn - Entrada (1) ou saída (0)
void vLCD_DataTris( unsigned char bIn );

// Lê um byte do display
// Valor devolvido: lcdData - Byte lido do display
char hLCD_DataIn( void );
```

```
// Coloca o byte a escrever no display
// Variável de entrada: hInstruction - Instrução a enviar ao LCD
void vLCD_DataOut( unsigned char hInstruction );

// Testar checking flow
void vLCD_BusyCheck( void );

// Envia uma instrução ou um dado para o LCD
// Variáveis de entrada: hInstruction - Instrução a enviar ao LCD
//                       bCD - Comando (1) ou dados (0)
void vLCD_Write( unsigned char hInstruction, unsigned char bCD );

// Envia um endereço para o LCD
// Variáveis de entrada: hAddr -> Endereço
//                       hCmd -> Instrução de comando
void vLCD_WriteAddr( unsigned int hAddr, unsigned char hCmd );

// Inicializa display
void vLCD_Init( void );

// Apaga memória de texto
void vLCD_ClearText( void );

// Apaga memória gráfica
void vLCD_ClearGraphic( void );

// Escreve um caracter no LCD
// Variável de entrada: hCaracter - Caracter a enviar para o LCD
void vLCD_WriteChar( unsigned char hCaracter );

// Escreve uma string numa determinada posição do LCD
// Variáveis de entrada: iX - Coluna
//                       iY - Linha
//                       sPhrase - String a enviar para o LCD
void vLCD_WriteString( unsigned char iX, unsigned char iY, const rom char *sPhrase );

// Desenha um bitmap numa determinada posição do LCD
// Variáveis de entrada: iX - Coluna
//                       iY - Linha
//                       iBitmapX – Total de colunas do bitmap
//                       iBitmapY – Total de linhas do bitmap
//                       vBitmap - Vector com os bytes a enviar para o LCD
void vLCD_WriteBitmap( unsigned char iX, unsigned char iY, unsigned char iBitmapX, unsigned char
iBitmapY, const far rom unsigned char *vBitmap );

// Desenha um pixel no LCD
// Variáveis de entrada: iX - Coluna
//                       iY - Linha
void vLCD_WritePixel( unsigned int iX, unsigned int iY );

// Desenha uma palavra no LCD com um tipo de letra diferente
// Variáveis de entrada: iX - Coluna
//                       iY - Linha
//                       sWord - String a enviar para o LCD
void vLCD_WriteWord( unsigned char iX, unsigned char iY, const rom char *sWord );

// Funções da comunicação por i2c

// Interrupção do i2c
void vI2C_SlaveInterrupt( void );
```

```
// Contém as frases a escrever no LCD
void vLCD_Words( void );

/* ---------------------------------------- Fim Declaração das Funções ---------------------------------------- */



/* ------------------------------------------------------------------------------------------------------------------
 * Nome do Projecto: Hollowgrail
 * Cliente: ACCO
 *
 * Nome do Ficheiro: bitmap.h
 * Descrição do Ficheiro: Declaração dos bitmaps a enviar para o LCD
 *
 * Data: 27.12.2007
 * ------------------------------------------------------------------------------------------------------------------ */

// Variável com o desenho das letras
const far rom unsigned int letra[234] = {
0,2016,48,48,2032,2096,2016,0,0,3072,4064,3120,3120,3120,3120,2016,0,0,0,2016,3072,3072,3072,307
2,2016,0,0,48,2032,3120,3120,3120,3120,2016,0,0,0,2016,3120,3120,4080,3072,2016,0,0,992,1536,4032
,1536,1536,1536,1536,0,0,0,2016,3120,3120,3120,3120,2032,48,2016,3072,4064,3120,3120,3120,3120,3
120,0,0,3072,0,3072,3072,3072,3072,3072,0,0,96,0,96,96,96,96,96,3168,1984,3072,3264,3840,3584,384
0,3456,3264,0,0,3072,3072,3072,3072,3072,3072,1792,0,0,0,2044,3270,3270,3270,3270,3270,0,0,0,1984
,3168,3168,3168,3168,3168,0,0,0,2016,3120,3120,3120,3120,2016,0,0,0,2016,3120,3120,3120,3120,406
4,3072,3072,0,2016,3120,3120,3120,3120,2032,48,48,0,1792,3072,3072,3072,3072,3072,0,0,0,2032,307
2,4080,48,48,4064,0,0,1536,4032,1536,1536,1536,1536,960,0,0,0,3120,3120,3120,3120,3120,2016,0,0,0,
3120,3120,3120,3120,1632,960,0,0,0,3096,3096,3480,3480,3480,2032,0,0,0,3120,1632,960,960,1632,31
20,0,0,0,1056,3120,3120,3120,3120,2032,32,2016,0,4080,240,480,960,1920,4080,0,0
};

// Variável com o desenho da virgula
const far rom unsigned int letra_virgula[9]={0,0,0,0,0,0,1024,1024,2048};

// Variável com o espaço em branco de cada letra
unsigned char vazio_letra[26] = {4, 4, 5, 4, 4, 5, 4, 4, 10, 5, 6, 8, 1, 5, 4, 4, 4, 8, 4, 6, 4, 4, 3, 4, 4, 4};
unsigned char vazio_virgula = 10;

// Variável com a imagem inicial
const far rom unsigned char ResetAmimationA[294] = {
0,0,7,224,0,0,0,0,248,31,0,0,0,7,128,1,192,0,0,28,0,0,48,0,0,48,0,0,24,0,0,96,0,0,6,0,0,192,0,0,3,0,1,0,0,0,
0,128,2,0,0,0,0,64,4,0,0,0,0,96,4,0,0,0,0,48,8,0,0,0,0,16,16,0,0,0,0,24,16,0,0,0,0,12,32,0,0,0,0,4,32,0,0,0,0
,4,64,0,0,0,0,4,64,31,255,248,31,194,64,127,239,252,127,194,64,255,207,254,255,194,65,248,30,30,240,
2,129,195,222,29,224,3,131,195,220,125,192,1,131,135,221,251,128,1,131,129,159,251,128,1,131,195,1
84,127,192,1,131,227,184,61,227,3,67,255,184,61,255,98,65,255,120,124,254,98,64,127,112,248,0,2,64,
0,15,240,0,2,96,0,15,224,0,4,32,0,31,128,0,4,32,0,0,0,0,4,16,0,0,0,0,8,24,0,0,0,0,24,8,0,0,0,0,16,4,0,0,0,0,
32,6,0,0,0,0,96,2,0,0,0,0,64,1,0,0,0,0,128,0,192,0,0,3,0,0,96,0,0,6,0,0,24,0,0,24,0,0,12,0,0,48,0,0,3,128,1,
192,0,0,0,252,62,0,0,0,0,7,224,0,0
};

// Variável com o inverso da imagem inicial
const far rom unsigned char ResetAmimationB[1024] = {
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,2
55,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,255,255,255,255,255,255,248,31,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,7,224,25
5,255,255,255,255,255,255,255,255,255,255,255,255,255,248,127,254,63,255,255,255,255,255,255,255,255,
255,255,255,255,227,255,255,207,255,255,255,255,255,255,255,255,255,255,255,255,207,255,255,231,2
55,255,255,255,255,255,255,255,255,255,255,255,255,159,255,255,249,255,255,255,255,255,255,255,255,25
5,255,255,255,63,255,255,252,255,255,255,255,255,255,255,255,255,255,255,254,255,255,255,255,127,
```

255,255,255,255,255,255,255,255,255,255,253,255,255,255,255,191,255,255,255,255,255,255,255,255,2
55,255,251,255,255,255,255,159,255,255,255,255,255,255,255,255,255,255,251,255,255,255,255,207,25
5,255,255,255,255,255,255,255,255,255,247,255,255,255,255,239,255,255,255,255,255,255,255,255,255
,255,239,255,255,255,255,231,255,255,255,255,255,255,255,255,255,255,239,255,255,255,255,243,255,
255,255,255,255,255,255,255,255,255,255,223,255,255,255,255,251,255,255,255,255,255,255,255,255,255,2
55,223,255,255,255,255,251,255,255,255,255,255,255,255,255,255,255,191,255,255,255,255,251,255,25
5,255,255,255,255,255,255,255,255,255,191,224,0,7,224,61,255,255,255,255,255,255,255,255,255,255,191,1
28,16,3,128,61,255,255,255,255,255,255,255,255,255,255,255,191,0,48,1,0,61,255,255,255,255,255,255,255,
255,255,255,190,7,225,225,15,253,255,255,255,255,255,255,255,255,255,255,126,60,33,226,31,252,255,
255,255,255,255,255,255,255,255,255,255,124,60,35,130,63,254,255,255,255,255,255,255,255,255,255,255,
124,120,34,4,127,254,255,255,255,255,255,255,255,255,255,255,124,126,96,4,127,254,255,255,255,255,
255,255,255,255,255,255,124,60,71,128,63,254,255,255,255,255,255,255,255,255,255,255,124,28,71,19
4,28,252,255,255,255,255,255,255,255,255,255,255,188,0,71,194,0,157,255,255,255,255,255,255,255,255,25
5,255,255,190,0,135,131,1,157,255,255,255,255,255,255,255,255,255,255,191,128,143,7,255,253,255,25
5,255,255,255,255,255,255,255,255,255,191,255,240,15,255,253,255,255,255,255,255,255,255,255,255,255,
159,255,240,31,255,251,255,255,255,255,255,255,255,255,255,255,223,255,224,127,255,251,255,255,25
5,255,255,255,255,255,255,255,255,223,255,255,255,255,251,255,255,255,255,255,255,255,255,255,255,239
,255,255,255,255,247,255,255,255,255,255,255,255,255,255,255,231,255,255,255,255,231,255,255,255,
255,255,255,255,255,255,255,255,247,255,255,255,255,239,255,255,255,255,255,255,255,255,255,255,251,2
55,255,255,255,255,223,255,255,255,255,255,255,255,255,255,255,249,255,255,255,255,159,255,255,255,25
5,255,255,255,255,255,255,255,253,255,255,255,255,191,255,255,255,255,255,255,255,255,255,255,254,255
,255,255,255,127,255,255,255,255,255,255,255,255,255,255,255,255,63,255,255,252,255,255,255,255,255,2
55,255,255,255,255,255,255,255,159,255,255,249,255,255,255,255,255,255,255,255,255,255,255,255,255,231,25
5,255,255,231,255,255,255,255,255,255,255,255,255,255,255,255,255,243,255,255,207,255,255,255,255,255,255
,255,255,255,255,255,255,255,252,127,254,63,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,3,193,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,248,31,255,255,255,255,255,255,255,255,255,25
5,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,243,255,255,255,2
55,255,255,255,255,255,255,255,255,255,255,159,48,51,240,48,56,3,129,255,255,255,255,255,255,255,255,255,1
59,39,147,231,231,147,57,60,255,255,255,255,255,255,255,255,255,147,39,147,231,231,147,57,60,255,255,2
55,255,255,255,255,255,255,147,32,19,231,231,147,57,0,255,255,255,255,255,255,255,255,255,147,39,243,231,2
31,147,57,63,255,255,255,255,255,255,255,255,255,192,112,56,48,48,51,57,129,255,255,255,255,255,255,255,25
5,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
,255,255,255,255,255
};

// Variável com a imagem para inserir o livro
const far rom unsigned char InsertBook[156] = {
96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0
,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,
96,0,0,0,0,6,96,0,0,0,0,6,96,0,0,0,0,6,127,255,255,255,255,254,127,255,255,255,255,254,127,255,255,25
5,255,254,127,255,255,255,255,254,127,255,255,255,255,254,127,255,255,255,255,254
};

// Variável com a imagem para carregar no start
const far rom unsigned char InsertBookPressStart[180] = {
0,63,255,255,255,240,0,96,0,0,0,16,0,160,0,0,0,16,1,32,0,0,0,16,98,32,0,0,0,22,100,32,0,0,0,22,111,224,0
,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,
0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0,0,22,104,0,0,0
,0,22,104,0,0,0,0,22,111,255,255,255,255,246,96,0,0,0,0,6,127,255,255,255,255,254,127,255,255,255,25
5,254,127,255,255,255,255,254,127,255,255,255,255,254,127,255,255,255,255,254,127,255,255,255,255
,254
};

// Variável com a imagem para colocar a argola
const far rom unsigned char PlaceXSpineBelow[448] = {
0,0,0,0,0,0,0,0,0,0,0,14,0,0,0,0,0,0,0,0,0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0,0,15,128,0,0,0,0,0,0,0,0,0,0,
0,15,128,0,0,0,0,0,0,0,0,0,0,0,15,128,0,0,0,0,0,0,0,0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0,0,15,0,7,252,0,0,0,
0,0,0,0,0,0,0,0,63,224,7,252,0,0,0,7,255,255,224,0,0,0,63,224,7,252,0,0,0,7,252,127,224,0,0,0,63,224,3,254

```
,0,0,0,7,252,63,224,0,0,0,63,224,3,254,0,0,0,15,248,31,240,0,0,0,127,192,3,254,0,0,0,15,248,31,240,0,0,0
,127,192,3,254,0,0,0,15,248,31,240,0,0,0,127,128,1,255,0,0,0,31,248,31,248,0,0,0,255,128,1,255,0,0,0,31
,248,15,252,0,0,0,255,128,0,255,128,0,0,63,240,15,252,0,0,1,255,128,0,255,192,0,0,63,240,7,254,0,0,1,2
55,0,0,255,192,0,0,127,224,7,255,0,0,3,255,0,0,127,224,0,0,255,224,3,255,128,0,7,254,0,0,127,240,0,1,2
55,192,3,255,192,0,31,254,0,0,63,252,0,7,255,192,1,255,224,0,127,252,0,0,31,255,0,31,255,128,0,255,25
2,1,255,248,0,0,31,255,255,255,255,0,0,255,255,255,255,240,0,0,15,255,255,255,254,0,0,127,255,255,25
5,224,0,0,7,255,255,255,254,0,0,63,255,255,255,192,0,0,3,255,255,255,248,0,0,31,255,255,255,128,0,0,1
,255,255,255,240,0,0,7,255,255,255,0,0,0,0,127,255,255,192,0,0,3,255,255,254,0,0,0,0,31,255,255,0,0,0,
0,255,255,248,0,0,0,0,7,255,252,0,0,0,0,63,255,224,0,0,0,0,0,63,192,0,0,0,0,1,254,0,0,0
};

// Variável com a imagem para dar início ao processo de furação e encadernação
const far rom unsigned char ReadyToBind[216] = {
63,255,255,255,255,252,32,0,0,0,0,4,32,0,0,0,0,4,32,0,0,0,0,4,32,0,0,0,0,4,32,0,1,128,0,4,32,0,3,192,0,4,3
2,0,6,96,0,4,32,0,12,48,0,4,32,0,25,152,0,4,32,0,49,140,0,4,32,0,97,134,0,4,32,0,193,131,0,4,32,1,129,12
9,128,4,32,3,1,128,192,4,32,2,1,128,64,4,32,2,1,128,64,4,32,3,1,128,192,4,32,1,129,129,128,4,32,0,193,1
31,0,4,32,0,97,134,0,4,32,0,49,140,0,4,32,0,25,152,0,4,32,0,12,48,0,4,32,0,6,96,0,4,32,0,3,192,0,4,32,0,0,
0,0,4,32,0,0,0,0,4,32,0,0,0,0,4,32,0,0,0,0,4,38,102,102,102,102,100,38,102,102,102,102,100,63,255,255,2
55,255,252,6,102,102,102,102,96,6,102,102,102,102,96,6,102,102,102,102,96
};

// Variável com a imagem para colocar uma argola superior (média)
const far rom unsigned char PleaseInsertLargerSpine[480] = {
0,0,0,0,0,0,0,0,0,0,0,0,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15,0,192,0,0,0,0,0,0,0,0,0,0,0,0,15,128,192,0,0,0,0,0,0
,0,0,0,0,0,0,15,128,192,0,0,0,0,0,0,0,0,0,0,0,15,128,192,0,0,0,0,0,0,0,0,0,0,0,15,0,192,0,0,0,0,0,0,0,0
,0,0,0,15,0,192,7,252,0,0,0,0,0,0,0,0,0,0,63,224,124,7,252,0,0,0,7,255,255,224,0,0,0,63,224,0,7,252,0,0,0,
7,252,127,224,0,0,0,63,224,0,3,254,0,0,0,7,252,63,224,0,0,0,63,224,0,3,254,0,0,0,15,248,31,240,0,0,0,12
7,192,0,3,254,0,0,0,15,248,31,240,0,0,0,127,192,0,3,254,0,0,0,15,248,31,240,0,0,0,127,128,0,1,255,0,0,0,
31,248,31,248,0,0,0,255,128,0,1,255,0,0,0,31,248,15,252,0,0,0,255,128,0,0,255,128,0,0,63,240,15,252,0,
0,1,255,128,0,0,255,192,0,0,63,240,7,254,0,0,1,255,0,0,0,255,192,0,0,127,224,7,255,0,0,3,255,0,0,0,127,
224,0,0,255,224,3,255,128,0,7,254,0,0,0,127,240,0,1,255,192,3,255,192,0,31,254,0,0,0,63,252,0,7,255,19
2,1,255,224,0,127,252,0,0,0,31,255,0,31,255,128,0,255,252,1,255,248,0,0,0,31,255,255,255,255,0,0,255,
255,255,255,240,0,0,0,15,255,255,255,254,0,0,127,255,255,255,224,0,0,0,7,255,255,255,254,0,0,63,255,
255,255,192,0,0,0,3,255,255,255,248,0,0,31,255,255,255,128,0,0,0,1,255,255,255,240,0,0,7,255,255,255,
0,0,0,0,0,127,255,255,192,0,0,3,255,255,254,0,0,0,0,0,31,255,255,0,0,0,0,255,255,248,0,0,0,0,0,7,255,25
2,0,0,0,0,63,255,224,0,0,0,0,0,0,63,192,0,0,0,0,1,254,0,0,0,0
};

// Variável com a imagem para colocar uma argola superior (grande)
const far rom unsigned char PleaseInsertLargerSpine2[570] = {
0,0,0,0,0,0,0,0,0,0,0,112,0,0,0,0,0,0,0,0,0,0,0,0,0,0,120,0,0,0,7,252,0,0,0,0,0,0,0,0,120,0,0,0,12,198,0,0,
0,0,0,0,0,0,0,120,0,0,0,12,198,0,0,0,0,0,0,0,0,112,0,0,0,12,198,63,0,0,0,255,255,0,0,0,252,0,0,0,12,198,
63,128,0,0,255,255,0,0,1,252,14,0,0,12,198,31,128,0,1,252,63,128,0,1,252,15,0,192,0,0,31,128,0,1,252,6
3,128,0,1,248,15,128,192,0,0,31,192,0,1,252,63,128,0,1,248,15,128,192,0,0,31,192,0,3,252,63,192,0,3,24
8,15,128,192,0,0,15,192,0,3,248,31,224,0,3,240,15,0,192,0,0,15,224,0,7,248,15,240,0,7,240,15,0,192,7,2
52,7,240,0,8,0,0,16,0,15,224,63,224,124,7,252,7,248,0,23,255,255,236,0,31,224,63,224,0,7,252,3,254,0,1
19,252,127,238,0,127,192,63,224,0,3,254,3,255,255,247,252,63,231,131,255,192,63,224,0,3,254,1,255,2
55,239,248,31,247,255,255,128,127,192,0,3,254,0,255,255,239,248,31,243,255,254,0,127,192,0,3,254,0,
127,255,207,248,31,243,255,254,0,127,128,0,1,255,0,31,255,159,248,31,249,255,248,0,255,128,0,1,255,
0,15,255,159,248,15,253,255,240,0,255,128,0,0,255,128,1,255,63,240,15,252,255,192,1,255,128,0,0,255,
192,0,0,63,240,7,254,0,0,1,255,0,0,0,255,192,0,0,127,224,7,255,0,0,3,255,0,0,0,127,224,0,0,255,224,3,25
5,128,0,7,254,0,0,0,127,240,0,1,255,192,3,255,192,0,31,254,0,0,0,63,252,0,7,255,192,1,255,224,0,127,25
2,0,0,0,31,255,0,31,255,128,0,255,252,1,255,248,0,0,0,31,255,255,255,255,0,0,255,255,255,255,240,0,0,
0,15,255,255,255,254,0,0,127,255,255,255,224,0,0,0,7,255,255,255,254,0,0,63,255,255,255,192,0,0,0,3,2
55,255,255,248,0,0,31,255,255,255,128,0,0,0,1,255,255,255,240,0,0,7,255,255,255,0,0,0,0,0,127,255,255
,192,0,0,3,255,255,254,0,0,0,0,0,31,255,255,0,0,0,0,255,255,248,0,0,0,0,0,7,255,252,0,0,0,0,63,255,224,0
,0,0,0,0,0,63,192,0,0,0,0,1,254,0,0,0,0
};
```

```
// Variável com a imagem do processo de furação
const far rom unsigned char ProgressPunching[48] = {
255,255,255,255,255,255,255,254,255,252,0,0,0,0,0,2,255,248,0,0,0,0,0,2,255,240,0,0,0,0,0,2,255,224,0,
0,0,0,0,2,255,255,255,255,255,255,255,254
};

// Variável com a imagem do processo de encadernação
const far rom unsigned char ProgressBinding[215] = {
0,15,255,255,252,0,31,255,255,252,0,48,0,0,12,0,97,0,0,12,0,195,0,0,12,1,131,0,0,12,3,3,0,0,12,3,3,0,0,1
2,3,63,0,0,12,3,127,0,0,12,3,0,0,0,12,3,0,0,0,12,3,0,0,0,12,3,0,15,192,12,3,0,63,240,12,3,0,112,56,12,3,0,
231,28,12,3,1,199,14,12,3,1,135,6,12,3,3,0,3,12,3,7,7,3,140,3,7,7,3,140,3,7,7,3,140,3,7,7,3,140,3,7,7,3,14
0,3,3,7,3,12,3,3,135,6,12,3,1,199,12,12,3,0,231,28,12,3,0,112,56,12,3,0,63,240,12,3,0,15,192,12,3,0,0,0,1
2,3,0,0,0,12,3,0,0,0,12,3,0,0,0,12,3,0,0,0,12,3,0,0,0,12,3,0,0,0,12,3,0,0,0,12,3,0,0,0,12,3,255,255,255,252,
3,255,255,255,252
};

// Variável com a imagem que indica que a encadernação está concluída
const far rom unsigned char BookComplete[272] = {
0,0,128,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,251,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,4,6,128,0,0,0,0,0,0,0,0,0,0,0,0,0,8,6,64,0,0,0,0,0,0,0,0,0,0,0,0,8,6,64,0,0,0,0,0,0,0,0,
0,0,0,0,0,10,12,76,0,0,1,128,0,0,0,0,0,192,0,96,0,9,140,79,227,241,249,152,31,159,143,248,252,192,252,25
2,252,8,204,76,54,27,13,224,48,48,217,141,134,193,134,97,134,8,108,76,54,27,13,192,48,48,217,141,13
4,193,134,97,134,8,60,76,54,27,13,224,48,48,217,141,134,193,254,97,254,8,60,76,54,27,13,152,48,48,21
7,141,134,193,128,97,128,8,0,71,227,241,249,140,31,159,153,141,252,124,252,60,252,8,0,64,0,0,0,0,0,0,
0,1,128,0,0,0,0,4,0,128,0,0,0,0,0,0,0,1,128,0,0,0,0,3,255,0,0,0,0,0,0,0,0,0,0,0,0,0
};

// Variável com a imagem que indica que a gaveta está cheia
const far rom unsigned char WasteFull[216] = {
3,255,255,255,255,224,4,0,0,0,0,16,8,0,30,240,0,8,16,0,0,16,0,4,16,16,8,49,0,4,16,9,129,56,176,4,16,0,32
,0,156,4,16,0,4,64,28,4,16,31,76,255,24,4,16,14,76,254,88,4,16,78,25,254,81,4,16,96,0,0,3,4,16,96,0,0,7,
4,16,112,0,0,7,4,16,112,0,0,7,4,16,112,0,0,6,4,16,48,0,0,6,4,16,48,0,0,6,4,16,48,0,0,6,4,16,48,0,0,14,4,16,
56,0,0,14,4,16,56,0,0,12,4,16,56,0,0,12,4,16,24,0,0,12,4,16,24,0,0,12,4,16,24,0,0,28,4,16,24,0,0,28,4,16,2
8,0,0,28,4,16,28,0,0,24,4,16,28,0,0,24,4,16,12,0,0,24,4,16,14,0,0,56,4,16,15,255,255,248,4,8,7,255,255,2
40,8,4,0,0,0,0,16,3,255,255,255,255,224
};

// Variável com a imagem que indica que o batente das brocas não se encontra presente
const far rom unsigned char DrillPadMissing[216] = {
3,255,255,255,255,224,4,0,0,0,0,16,8,0,0,0,0,8,16,0,0,0,0,4,16,0,0,0,0,4,16,0,0,0,0,4,16,0,0,0,0,4,16,0,31,
252,0,4,16,0,31,252,0,4,16,0,31,252,0,4,16,0,30,60,0,4,16,0,28,28,0,4,16,0,28,28,0,4,16,0,30,60,0,4,19,25
5,223,125,255,228,16,0,31,124,0,4,16,0,31,124,0,4,19,255,223,125,255,228,16,0,31,252,0,4,16,0,31,124,
0,4,19,255,222,61,255,228,16,0,31,120,0,4,16,0,31,240,0,4,16,0,31,224,0,4,16,0,31,192,0,4,16,0,31,128,0
,4,16,0,31,0,0,4,16,0,30,0,0,4,16,0,28,0,0,4,16,0,24,0,0,4,16,0,16,0,0,4,16,0,0,0,0,4,16,0,0,0,0,4,8,0,0,0,0,
8,4,0,0,0,0,16,3,255,255,255,255,224
};

// Variável com a imagem que indica que a porta de acesso às brocas está aberta
const far rom unsigned char CoverOpen[216] = {
3,255,255,255,255,224,4,0,0,0,0,16,8,0,0,0,0,8,16,0,0,0,0,4,16,31,255,255,192,4,16,16,31,0,64,4,16,16,0,
240,80,4,16,16,0,8,88,4,16,16,0,8,92,4,16,16,0,8,94,4,16,16,0,8,31,4,16,16,0,195,254,4,16,16,0,255,252,4
,16,16,0,255,252,4,16,16,0,255,252,4,16,16,0,255,248,4,16,16,0,255,216,4,16,16,0,254,16,4,16,16,0,224,
80,4,16,16,0,8,64,4,16,16,0,200,64,4,16,16,0,200,64,4,16,16,0,8,64,4,16,16,0,8,64,4,16,16,0,8,64,4,16,16,
0,8,64,4,16,16,0,8,64,4,16,16,0,8,64,4,16,24,0,15,192,4,16,7,192,8,0,4,16,0,62,8,0,4,16,0,1,248,0,4,16,0,0
,0,0,4,8,0,0,0,0,8,4,0,0,0,0,16,3,255,255,255,255,224
};
```

```
/* ------------------------------------------------------------------------------------------------------------------------
 * Nome do Projecto: Hollowgrail
 * Cliente: ACCO
 *
 * Nome do Ficheiro: lcd.c
 * Descrição do Ficheiro: Código para utilizar um LCD com controlador
 *                        T6963C
 *
 * Data: 30.10.2007
 * ------------------------------------------------------------------------------------------------------------------------ */

#include "lcd.h"
#include "bitmap.h"

/* ***************************** Implementação das Funções ***************************** */

// Nos pic18 as interrupções de alta prioridade estão em 0x08h
#pragma code i2c_interrupt = 0x08
void i2c_int (void)
{
        _asm goto vI2C_SlaveInterrupt _endasm
}
#pragma code

// Interrupção do I2C
#pragma interrupt vI2C_SlaveInterrupt
void vI2C_SlaveInterrupt( void )
{
        PIR1bits.SSPIF = 0;             // Apaga a flag de interrupção

        i2cStat = SSPSTAT;              // Guarda o valor presente no registo de estado
        i2cStat &= 0b00101101;          // Mascara para analisar o estado do buffer, modo de escrita
                                        // ou leitura, estado do start bit e bit de endereço ou de dado

        switch(i2cStat)
        {
                // Escrita de um endereço, buffer está cheio
                case 0b00001001: ReadI2C();             // Lê do buffer um endereço
                                 break;
                // Escrita de um dado, buffer está cheio
                case 0b00101001: i2cData = ReadI2C();   // Lê do buffer um dado
                                 flagInterrupt = 1;
                                 break;
        }
}

// Inicializa pinos de dados do LCD como entrada ou saída
void vLCD_DataTris( unsigned char bIn )
{
        if(bIn)         // Pinos de dados do LCD como entradas
        {
                LCD_DATAB_TRIS = LCD_DATAB_TRIS | (255-maskB);
                LCD_DATAC_TRIS = LCD_DATAC_TRIS | (255-maskC);
        }

        else            // Pinos de dados do LCD como saídas
        {
                LCD_DATAB_TRIS = LCD_DATAB_TRIS & maskB;
                LCD_DATAC_TRIS = LCD_DATAC_TRIS & maskC;
        }
```

```
}

// Lê um byte do display
char hLCD_DataIn( void )
{
	unsigned char lcdData;

	// Lê os bit de dados do LCD
	LCD_DATAB = LCD_DATAB & (255-maskB);
	LCD_DATAC = LCD_DATAC & (255-maskC);

	lcdData = LCD_DATAB | LCD_DATAC;

	return lcdData;
}

// Coloca o byte a escrever no display
void vLCD_DataOut( unsigned char hInstruction )
{
	// Escreve os bits de dados no LCD
	LCD_DATAB = LCD_DATAB & maskB;
	LCD_DATAB = LCD_DATAB | (hInstruction & (255-maskB));

	LCD_DATAC = LCD_DATAC & maskC;
	LCD_DATAC = LCD_DATAC | (hInstruction & (255-maskC));
}

// Testa checking flow
void vLCD_BusyCheck( void )
{
	unsigned char data;

	vLCD_DataTris(1);		// Inicializa pinos de dados como entrada

	LCD_CD = 1;			// Instrução de comando
	LCD_RD = 0;			// Define modo de leitura
	LCD_WR = 1;
	LCD_CE = 0;			// Habilita o LCD

	data = hLCD_DataIn();		// Lê um byte do display

	// Testa bits 0 e 1 e não faz nada enquanto estes não forem os dois 1
	while( data & 0x03 == 0 );

	LCD_CE = 1;			// Desabilita o LCD
	LCD_RD = 1;

	vLCD_DataTris(0);		// Inicializa pinos de dados como saída
}

// Envia uma instrução de controlo ou um dado para o LCD
void vLCD_Write( unsigned char hInstruction, unsigned char bCD )
{
	vLCD_BusyCheck();
	vLCD_DataOut(hInstruction);	// Coloca a instrução ou os dados a escrever no display

	if(bCD)
		LCD_CD = 1;		// Define escrita de um comando
	else
		LCD_CD = 0;		// Define escrita de dados
```

```
	LCD_WR = 0;				// Define modo de escrita
	LCD_RD = 1;
	LCD_CE = 0;				// Habilita o LCD
	Nop();
	LCD_CE = 1;				// Desabilita o LCD
	LCD_WR = 1;
}

// Envia um endereço para o LCD
void vLCD_WriteAddr( unsigned int hAddr, unsigned char hCmd )
{
	vLCD_Write(hAddr & 0x00FF, 0);		// Escreve LSB do endereço
	vLCD_Write(hAddr >> 8, 0);		// Escreve MSB do endereço
	vLCD_Write(hCmd, 1);			// Escreve instrução de comando
}

// Inicializa display.
void vLCD_Init( void )
{
	LCD_RST = 0;				// Faz o reset
	Delay100TCYx(100);			// Espera 2ms
	LCD_RST = 1;

	// Mode set
	vLCD_Write(0x80, 1);			// Modo OR
	// Control word set
	vLCD_WriteAddr(lcd_th, 0x40);		// Endereço inicial da memória de texto
	vLCD_WriteAddr(lcd_ta, 0x41);		// Área de texto
	vLCD_WriteAddr(lcd_gh, 0x42);		// Endereço inicial da memória gráfica
	vLCD_WriteAddr(lcd_ga, 0x43);		// Área gráfica
	// Display mode
	vLCD_Write(0x9C, 1);			// Selecciona modo gráfico e de texto sem cursor

	vLCD_ClearText();			// Apaga memória de texto
	vLCD_ClearGraphic();			// Apaga memória gráfica
}

// Apaga memória de texto
void vLCD_ClearText( void )
{
	int i=0;

	// Coloca o apontador para escrever na posição inicial da memória de texto
	vLCD_WriteAddr(lcd_th, 0x24);

	for(i=0;i<=lcd_ts;i++)
		vLCD_WriteChar(' ');	// Limpa toda a memória de texto
}

// Apaga memória gráfica
void vLCD_ClearGraphic( void )
{
	int i=0;

	// Coloca o apontador para escrever na posição inicial da memória gráfica
	vLCD_WriteAddr(lcd_gh, 0x24);

	for(i=0;i<=lcd_gs;i++)
		vLCD_WriteChar(' ');	// Limpa toda a memória gráfica
}
```

```
// Escreve um caracter no LCD
void vLCD_WriteChar( unsigned char hCaracter )
{
	vLCD_Write(hCaracter - 0x20, 0);		// Letra a escrever no display (ascii-20H)
	vLCD_Write(0xC0, 1);				// Escreve dado e incrementa apontador
}

// Escreve uma string numa posição do LCD
void vLCD_WriteString( unsigned char iX, unsigned char iY, const rom char *sPhrase )
{
	unsigned int addr;

	addr = lcd_th + iY*lcd_ta + iX;	// Calcula a posição a escrever no display
	vLCD_WriteAddr(addr, 0x24);	// Coloca o apontador para escrever na posição calculada

	while(*sPhrase)					// Enquanto a string não for '\0'
		vLCD_WriteChar(*sPhrase++);		// Escreve um caracter no LCD
}

// Desenha um bitmap numa determinada posição do LCD
void vLCD_WriteBitmap( unsigned char iX, unsigned char iY, unsigned char iBitmapX, unsigned char
iBitmapY, const far rom unsigned char *vBitmap )
{
	int i=0,j=0;
	unsigned int addr;

	addr = lcd_gh + iY*lcd_ga + iX;	// Calcula a posição a escrever no display
	vLCD_WriteAddr(addr, 0x24);	// Coloca o apontador para escrever na posição calculada

	for( i=0;i<iBitmapY;i++)
	{
		// Desenha no display 8bits do bitmap
		for( j=0;j<iBitmapX;j++)
			vLCD_WriteChar( vBitmap[i*iBitmapX+j] + 0x20);

		iY++;					// Incrementa a linha do display
		addr = lcd_gh + iY*lcd_ga + iX;	// Calcula a nova posição a escrever no display
		vLCD_WriteAddr(addr, 0x24);	// Coloca o apontador para escrever na nova posição
						// calculada
	}
}

// Desenha um pixel no LCD
void vLCD_WritePixel( unsigned int iX, unsigned int iY )
{
	unsigned int addr;
	unsigned char cmd;

	addr = lcd_gh + iY*lcd_ga + iX/lcd_caracter_x;	// Calcula a posição a escrever no display
	vLCD_WriteAddr(addr, 0x24);			// Coloca o apontador para escrever na
							// posição calculada

	// Calcula o comando para o pixel que se pretende escrever
	cmd = 0xF8 | lcd_caracter_x - 1 - (iX % lcd_caracter_x);
	vLCD_Write(cmd, 1);				// Escreve um pixel no LCD
}
```

```
// Desenha uma palavra no LCD com um tipo de letra diferente
void vLCD_WriteWord( unsigned char iX, unsigned char iY, const rom char *sWord )
{
	unsigned int linha, coluna, linha2;
	unsigned int ibit, temp;
	unsigned char indice=0;
	unsigned int sW;

	coluna = iX;
	linha2 = iY;
	while (sWord[indice]!='\0')			// Enquanto a string não for '\0'
	{
		switch(sWord[indice])			// Desenha cada uma das letras
		{
			case ' ':	iX+=4;			// Incrementa quatro colunas do display
					coluna=iX;
					indice++;		// Passa para a letra seguinte
					break;
			case ',':	// Para imprimir as 9 linhas da virgula
					for ( linha=0 ; linha<9 ; linha++ )
					{
						temp=letra_virgula[linha];
						// Para imprimir os 12 pixeis de largura de cada caracter
						for ( ibit=2048 ; ibit>0 ; ibit>>=1 )
						{
						// Analisa a imagem e escreve um pixel no ecrã se necessário
							if (temp&ibit)
								vLCD_WritePixel( iX, iY );
							iX++;
						}
						iX=coluna;	// Para começar a linha seguinte do caracter
						iY++;
					}
					iX+= 12 + 1 - vazio_virgula;	// Para separar mais as letras
					coluna=iX;
					iY = linha2;
					indice++;		// Passar para a letra seguinte
					break;
			case '.':	iY+=6;				// Incrementa seis linhas para
					vLCD_WritePixel( iX, iY );	// escrever um pixel
					iX+=2;
					coluna=iX;
					iY = linha2;
					indice++;		// Passa para a letra seguinte
					break;
			default:	{
					// Para imprimir as 9 linhas de cada letra
					for ( linha=0 ; linha<9 ; linha++ )
					{
						// Para imprimir os 12 pixeis de largura de cada caracter
						temp=letra[(sWord[indice]-'a')*9+linha];
						for ( ibit=2048 ; ibit>0 ; ibit>>=1 )
						{
						// Analisa a imagem e escreve um pixel no ecrã se necessário
							if (temp&ibit)
								vLCD_WritePixel( iX, iY );
							iX++;
						}
						iX=coluna;	// Para começar a linha seguinte do caracter
						iY++;
```

```
                                        }
                                        // Para separar mais as letras
                                        iX+= 12 + 1 - vazio_letra[sWord[indice]-'a'];
                                        coluna=iX;
                                        iY = linha2;
                                        indice++;           // Passa para a letra seguinte
                                        }
                        }
                }
}

void vLCD_Words( void )
{
        unsigned char i,j;

        switch(i2cData)             // Consoante o caracter recebido por i2c
        {
                case 'a':  for(i=0;i<2;i++)
                        {
                                vLCD_WriteBitmap(5,3,6,48,ResetAmimationA);
                                vLCD_WriteWord(33,55,"welcome");
                                for(j=0;j<2;j++)
                                        Delay10KTCYx(250);
                                vLCD_WriteBitmap(0,0,16,64,ResetAmimationB);
                                for(j=0;j<2;j++)
                                        Delay10KTCYx(250);
                                vLCD_ClearGraphic();
                        }
                        break;
                case 'b':  vLCD_WriteBitmap(5,4,6,30,InsertBookPressStart);
                        vLCD_WriteWord(2,42,"ready,insert book");
                        vLCD_WriteWord(25,54,"press start");
                        break;
                case 'c':  break;
                case 'f':  vLCD_WriteBitmap(5,2,6,36,ReadyToBind);
                        vLCD_WriteWord(19,42,"ready to bind");
                        vLCD_WriteWord(24,55,"press start");
                        break;
                case 'h':  vLCD_WriteWord(21,5,"always");
                        vLCD_WriteWord(39,21,"expect");
                        vLCD_WriteWord(46,35,"the best");
                        vLCD_WriteWord(1,53,"punching");
                        vLCD_WriteBitmap(8,54,8,6,ProgressPunching);
                        break;
                case 'o':  vLCD_WriteBitmap(2,5,5,43,ProgressBinding);
                        vLCD_WriteWord(69,24,"gbc.com");
                        vLCD_WriteWord(3,53,"binding");
                        vLCD_WriteBitmap(8,54,8,6,ProgressPunching);
                        break;
                case 's':  vLCD_WriteBitmap(0,3,16,17,BookComplete);
                        vLCD_WriteWord(20,27,"thank you for");
                        vLCD_WriteWord(20,40,"binding with");
                        vLCD_WriteWord(20,52,"gbc");
                        break;
                case 'A': vLCD_WriteBitmap(5,2,6,36,WasteFull);
                        vLCD_WriteWord(28,42,"waste full");
                        vLCD_WriteWord(1,54,"empty waste tray");
                        break;
                case 'B': vLCD_WriteWord(6,42,"drill missing");
                        break;
```

```
                case 'C': vLCD_WriteBitmap(5,2,6,36,DrillPadMissing);
                          vLCD_WriteWord(6,42,"drill pad missing");
                          vLCD_WriteWord(12,54,"check drill pad");
                          break;
                case 'D': vLCD_WriteBitmap(5,9,6,26,InsertBook);
                          vLCD_WriteWord(2,42,"ready,insert book");
                          break;
                case 'E': vLCD_WriteBitmap(5,2,6,36,CoverOpen);
                          vLCD_WriteWord(24,43,"cover open");
                          vLCD_WriteWord(22,54,"close cover");
                          break;
                case 'F': vLCD_WriteString(0, 1,"  Error during  process, please  press Enter to  reset
                                                machine     and remove        sheets");
                          break;
                case '0': vLCD_WriteWord(1,2,"place spine below");
                          vLCD_WriteBitmap(1,16,14,32,PlaceXSpineBelow);
                          vLCD_WriteWord(25,55,"small size");
                          break;
                case '1': vLCD_WriteWord(1,2,"place spine below");
                          vLCD_WriteBitmap(1,16,14,32,PlaceXSpineBelow);
                          vLCD_WriteWord(25,55,"medium size");
                          break;
                case '2': vLCD_WriteWord(1,2,"place spine below");
                          vLCD_WriteBitmap(1,16,14,32,PlaceXSpineBelow);
                          vLCD_WriteWord(25,55,"large size");
                          break;
                case '3': vLCD_WriteWord(17,2,"please place");
                          vLCD_WriteBitmap(1,13,15,38,PleaseInsertLargerSpine2);
                          vLCD_WriteWord(23,53,"larger spine");
                          break;
                case '4': vLCD_WriteWord(17,2,"please place");
                          vLCD_WriteBitmap(1,15,15,32,PleaseInsertLargerSpine);
                          vLCD_WriteWord(23,53,"larger spine");
                          break;
        }
}

/* ---------------------------------------- Fim Implementação das Funções ---------------------------------------- */

void main(void)
{
/* ****************************** Inicialização das Variáveis ****************************** */

        unsigned char i;

        // Inicializa PORTA e PORTB como I/O digitais
        ADCON1 = 0x0F;
        CMCON = 0x07;

        // Inicializa pinos de controlo do LCD como saídas
        LCD_CD_TRIS  = 0;
        LCD_WR_TRIS = 0;
        LCD_RD_TRIS  = 0;
        LCD_CE_TRIS  = 0;
        LCD_RST_TRIS = 0;
        LCD_FS_TRIS = 0;
        // Inicializa pinos de dados do LCD como saídas
        vLCD_DataTris(0);

/* ---------------------------------------- Fim Inicialização das Variáveis ---------------------------------------- */
```

```
	LCD_FS = 0;				// Letra do tamanho 8x8
	vLCD_Init();				// Inicializa LCD

	OpenI2C(SLAVE_7, SLEW_OFF);		// Inicializa i2c
	SSPADD = ADDR_LCD;			// Define endereço de i2c do LCD

	PIE1bits.SSPIE = 1;			// Activa a interrupção do i2c
	RCONbits.IPEN = 1;			// Habilita a prioridade das interrupções
	IPR1bits.SSPIP = 1;			// A interrupção a receber é de alta prioridade
	INTCONbits.GIEH = 1;			// Habilita todas as interrupções de alta prioridade

	while(1)
	{
		if(flagInterrupt)		// Se recebeu um dado por i2c
		{
			vLCD_ClearText();	// Apaga memória de texto
			vLCD_ClearGraphic();	// Apaga memória de gráfica
			vLCD_Words();		// Escreve string no display
			flagInterrupt = 0;
		}
	}
}



/* ------------------------------------------------------------------------------------------------------------------------
 * Nome do Projecto: Hollowgrail
 * Cliente: ACCO
 *
 * Nome do Ficheiro: hollowgrail.h
 * Descrição do Ficheiro: Definições para automatização de uma máquina
 *			de encadernar
 *
 * Data: 06.02.2008
 * ------------------------------------------------------------------------------------------------------------------------ */

#include "p18f4431.h"
#include "delays.h"
#include "sw_i2c.h"

/* ********************************** Configuration Bits ******************************** */

#pragma config OSC = HS		// Oscilador do tipo "High-Speed Crystal/Resonator"
#pragma config WDTEN = OFF	// Watchdog Timer
#pragma config LVP = OFF	// Low Voltage ICSP
#pragma config HPOL = HIGH	// Polaridade dos transistores do lado baixo estão a 1 para PWM par
#pragma config LPOL = HIGH	// Polaridade dos transistores do lado alto estão a 1 para PWM impar
#pragmaconfig PWMPIN = OFF	// Pinos de saída do PWM desactivados durante o reset

/* ---------------------------------------------- Fim Configuration Bits ---------------------------------------------- */

/* ********************************** Pinos de I/O ********************************** */
// Pinos de monitorização dos sensores
// Pino de saída do multiplexer
#define SENSOR_MUX		PORTDbits.RD1

// Pinos dos sensor_drillposition - Detecta posições do ciclo da furação
// SENSOR_DRILLPOSITIONA = 1 - Posição inicial
#define SENSOR_DRILLPOSITIONA	PORTAbits.RA0
// SENSOR_DRILLPOSITIONB = 1 - Posição intermédia
```

```
#define SENSOR_DRILLPOSITIONB       PORTAbits.RA1
// SENSOR_DRILLPOSITIONC = 1 - Posição final
#define SENSOR_DRILLPOSITIONC       PORTAbits.RA2

// Pinos dos sensor_close - Posição do mecanismo de fecho do clickbind
// SENSOR_CLOSE = 1 - Porta aberta
// SENSOR_CLOSE = 2 - Porta fechada
// SENSOR_CLOSE = 7 - Fecha clickbind pequeno
// SENSOR_CLOSE = 6 - Fecha clickbind médio
// SENSOR_CLOSE = 4 - Fecha clickbind grande
#define SENSOR_CLOSEA         PORTAbits.RA3
#define SENSOR_CLOSEB         PORTAbits.RA4
#define SENSOR_CLOSEC         PORTAbits.RA5
#define SENSOR_CLOSE (unsigned char) ((SENSOR_CLOSEA<<2) | (SENSOR_CLOSEB<<1) |
SENSOR_CLOSEC)

// Pino da entrada da interrupção - RC4
// Pinos de controlo dos infravermelhos
#define SENSOR_EMISSOR        LATDbits.LATD0

// Pinos de controlo do multiplexer
#define  CONTROL_MUX          LATC

// Pinos de controlo do motor dc 1 - Prende as folhas
// Pino IN1 - RB0
// Pino IN2 - RB1
// Entrada que indica se o motor está parado junto das folhas
// DC_CURRENT = 1 - Corrente máxima no motor
#define DC1_CURRENT           PORTEbits.RE0

// Pinos de controlo do motor dc 2 - Desloca o bloco de furação
// Pino IN1 - RB2
// Pino IN2 - RB3

// Pinos de controlo do motor dc 3 - Desloca mecanismo de fecho do clickbind
// Pino IN1 - RB5
// Pino IN2 - RB4

// Pinos de controlo do motor dc 4 - Fecha clickbind
// Pino IN1 - RB6
// Pino IN2 - RB7
// Pino dc4_current - I15(mux)      dc4_current = 1 - Corrente máxima no motor

// Pino de controlo do motor dc 5 - Rotação das brocas
#define DC5_CONTROL                 LATDbits.LATD5

// Pinos de controlo do motor de passo - Desloca a mesa
// Pino CE (Enable)
#define STEP_CE         LATDbits.LATD4      // STEP_CE = 1 - Habilita o motor
// Pino RST (Reset)
#define STEP_RST        LATCbits.LATC7      // STEP_RESET = 0 - Faz o reset
// Pino PULSE (Step pulse input)
#define STEP_CLOCK   LATCbits.LATC6         // A mudança de STEP_CLOCK = 0 para
                                            // STEP_CLOCK = 1 faz o motor andar um passo
// Pino CW/CCW (Clockwise/Counterclockwise)
#define STEP_CW         LATCbits.LATC5      // STEP_CW = 1 - Sentido Horário
                                            // STEP_CW = 0 - Sentido Anti-Horário
// Pino que indica o deslocamento máximo do motor de passo para a esquerda
#define  SENSOR_STEPLEFT    PORTEbits.RE1
```

```
// Definição das entradas dos sensores
// SENSOR_.._TRIS = 0 - Saída; SENSOR_.._TRIS = 1 - Entrada
#define SENSOR_MUX_TRIS                 TRISDbits.TRISD1
#define SENSOR_DRILLPOSITIONA_TRIS      TRISAbits.TRISA0
#define SENSOR_DRILLPOSITIONB_TRIS      TRISAbits.TRISA1
#define SENSOR_DRILLPOSITIONC_TRIS      TRISAbits.TRISA2
#define SENSOR_CLOSEA_TRIS              TRISAbits.TRISA3
#define SENSOR_CLOSEB_TRIS              TRISAbits.TRISA4
#define SENSOR_CLOSEC_TRIS              TRISAbits.TRISA5
#define SENSOR_EMISSOR_TRIS             TRISDbits.TRISD0

// Definição das saídas de controlo do multiplexer
#define  CONTROL_MUX_TRIS  TRISC

// Definição das entradas e saídas do motor dc 1
#define DC1_CURRENT_TRIS    TRISEbits.TRISE0            // DC1_.._TRIS = 1 - Entrada

// Definição das entradas e saídas do motor dc5
#define DC5_CONTROL_TRIS    TRISDbits.TRISD5            // DC5_.._TRIS = 0 - Saída

// Definição das entradas e saídas do motor de passo
#define STEP_CE_TRIS                TRISDbits.TRISD4    // STEP_.._TRIS = 0 - Saída
#define STEP_RST_TRIS               TRISCbits.TRISC7
#define STEP_CLOCK_TRIS             TRISCbits.TRISC6
#define STEP_CW_TRIS                TRISCbits.TRISC5
#define  SENSOR_STEPLEFT_TRIS       TRISEbits.TRISE1

/* ---------------------------------------------- Fim Pinos de I/O ---------------------------------------------- */

/* ********************************** Definições ********************************** */

// Definição da mascara para o porto C
#define mask_mux        0b11110000
// Pino do sensor_paper - Detecta presença de papel
#define sensor_paper    0
// Pino do botão Start - Liga a máquina
#define sensor_start    1
// Pino do botão Stop - Pára encadernação
#define sensor_stop     2
// Pino do botão OK - Aceita
#define sensor_ok       3
// Pinos dos botões right e left - Navega na máquina
#define sensor_right    4
#define sensor_left     5
// Pino dos sensor_sizeClick - Detecta tamanho do clickbind
// SENSOR_SIZECLICK = 0 - Clickbind pequeno
// SENSOR_SIZECLICK = 1 - Clickbind médio
// SENSOR_SIZECLICK = 3 - Clickbind grande
#define sensor_sizeClickA       6
#define sensor_sizeClickB       7
// Pino dos sensor_checkClick - Verifica tamanho do clickbind
// SENSOR_CHEKCLICK = 1 - Clickbind pequeno
// SENSOR_CHEKCLICK = 2 - Clickbind médio
// SENSOR_CHEKCLICK = 3 - Clickbind grande
#define sensor_checkClickA      8
#define sensor_checkClickB      9
// Pino do sensor_drawerFull - Detecta se a gaveta está cheia
#define sensor_drawerFull       10
```

```
// Pino do sensor_portOpen - Detecta se a porta de manutenção está aberta
#define sensor_portOpen         11
// Pino do sensor_drillOut - Detecta se as brocas estão mal colocadas
#define sensor_drillOut         12
// Pino do sensor_padOut - Detecta se o batente das brocas está mal colocado
#define sensor_padOut           13
// Pino dc4_current - Indica que o clickbind fechou
#define dc4_current             15
// Pino dc1_current - Indica que as folhas estão presas
#define dc1_current             16
// Pinos dos sensor_drillposition - Detecta posições do ciclo da furação
#define sensor_drillPositionA   17
#define sensor_drillPositionB   18
#define sensor_drillPositionC   19
// Pinos dos sensor_close - Posição do mecanismo de fecho do clickbind
#define sensor_closeA           20
#define sensor_closeB           21
#define sensor_closeC           22
//Pino do sensor_stepleft - Deslocamento máximo do motor de passo
#define sensor_stepleft         23
#define sensor_sizeClick        24
#define sensor_checkClick       25
#define sensor_close            26

// Definição do endereço do LCD para as funções de i2c
#define ADDR_LCD_W  0b10010000          // Escrita no LCD
#define ADDR_LCD_R  0b10010001          // Leitura do LCD

// Definições do tamanho do clickbind
#define click_small     1
#define click_medium    2
#define click_large     3

/* ------------------------------------------------- Fim Definições ------------------------------------------------- */

/* ****************************** Declaração das Variáveis ****************************** */

unsigned char sensor[27];       // Vector que guarda os valores recebidos nos sensores
char cState = ' ';              // Variável que indica em que estado está a encadernação
char cError = ' ';              // Variável que informa de uma situação irregular
char cClick = ' ';              // Variável que informa sobre o tamanho do clickbind
unsigned char flagStop=0;       // Flag que impede a máquina voltar ao funcionamento

/* ---------------------------------------- Fim Declaração das Variáveis ---------------------------------------- */

/* ****************************** Declaração das Funções ****************************** */

// Inicializa pinos de controlo dos multiplexeres como saída
void vSENSOR_ControlTris( void );

// Define o valor dos pinos de controlo dos multiplexeres
// Variável de entrada: iNumber - Valor a enviar para os pinos de controlo dos multiplexeres
void vSENSOR_Control( unsigned char iNumber);

// Lê valor dos sensores
void vSENSOR_Read( void );

// Verifica se a gaveta está cheia
// Valor devolvido: sensorDrawer - Gaveta cheia (1) ou Gaveta vazia (0)
unsigned char iSENSOR_DrawerFull( void );
```

```
// Verifica se a máquina está em condições de começar o ciclo
 void vSENSOR_Error( void );

// Envia um caracter por I2C
// Variáveis de entrada: hAddr -> Endereço do slave
//                       hCaracter -> Caracter a enviar para o slave
void vI2C_SendCaracter( unsigned char hAddr, unsigned char hCaracter );

/* ---------------------------------------- Fim Declaração das Funções ---------------------------------------- */



/* ------------------------------------------------------------------------------------------------------------------
 * Nome do Projecto: Hollowgrail
 * Cliente: ACCO
 *
 * Nome do Ficheiro: motor.h
 * Descrição do Ficheiro: Definições para utilizar um motor de corrente contínua
 *                        utilizando o pic 18F4431
 *
 * Data: 03.01.2008
 * ------------------------------------------------------------------------------------------------------------------ */

#include "p18f4431.h"

/* ********************** Registros de controlo de PWM ********************** */

// Mascara de configuração do PTCON0 a ser usada na função vPWM_Init
// PTOPS3:PTOPS0: PWM Time Base Output Postscale Select bits
#define PTCON0_POST_1_1     0b00001111      // Postscaler 1:1
#define PTCON0_POST_1_2     0b00011111      // Postscaler 1:2
#define PTCON0_POST_1_3     0b00101111      // Postscaler 1:3
#define PTCON0_POST_1_4     0b00111111      // Postscaler 1:4
#define PTCON0_POST_1_5     0b01001111      // Postscaler 1:5
#define PTCON0_POST_1_6     0b01011111      // Postscaler 1:6
#define PTCON0_POST_1_7     0b01101111      // Postscaler 1:7
#define PTCON0_POST_1_8     0b01111111      // Postscaler 1:8
#define PTCON0_POST_1_9     0b10001111      // Postscaler 1:9
#define PTCON0_POST_1_10    0b10011111      // Postscaler 1:10
#define PTCON0_POST_1_11    0b10101111      // Postscaler 1:11
#define PTCON0_POST_1_12    0b10111111      // Postscaler 1:12
#define PTCON0_POST_1_13    0b11001111      // Postscaler 1:13
#define PTCON0_POST_1_14    0b11011111      // Postscaler 1:14
#define PTCON0_POST_1_15    0b11101111      // Postscaler 1:15
#define PTCON0_POST_1_16    0b11111111      // Postscaler 1:16
// PTCKPS1:PTCKPS0: PWM Time Base Input Clock Prescale Select bits
#define PTCON0_PS_1_1       0b11110011      // Prescale 1:1
#define PTCON0_PS_1_4       0b11110111      // Prescale 1:4
#define PTCON0_PS_1_16      0b11111011      // Prescale 1:16
#define PTCON0_PS_1_64      0b11111111      // Prescale 1:64
// PTMOD1:PTMOD0: PWM Time Base Mode Select bits
#define PTCON0_CONT_UD_INT      0b11111111      // Modo Continuous Up/Down Count com
                                                // dupla interrupções
#define PTCON0_CONT_UD          0b11111110      // Modo Continuous Up/Down Count
#define PTCON0_SINGLE_SHOT      0b11111101      // Modo Single-Shot
#define PTCON0_FREE_RUNNING     0b11111100      // Modo Free-Running

// Mascara de configuração do PTCON1 a ser usada na função vPWM_Init
```

```
// PTEN: PWM Time Base Timer Enable bits
#define PTCON1_ON               0b11111111      // Base de tempo do PWM ligada
#define PTCON1_OFF              0b01111111      // Base de tempo do PWM desligada
// PTDIR: PWM Time Base Count Direction Status bits
#define PTCON1_DOWN             0b11111111      // Base de tempo do PWM conta decrescentemente
#define PTCON1_UP               0b10111111      // Base de tempo do PWM conta crescentemente

// Mascara de configuração do PWMCON0 a ser usada na função vPWM_Init
// PWMEN2:PWMEN0: PWM Module Enable bits
#define PWMCON0_PWM_ODD         0b11111111      // Todos os pinos PWM impares habilitados
                                                // para saída de PWM
#define PWMCON0_PWM_1_3         0b11101111      // PWM1, PWM3 habilitados para saída de
                                                // PWM
#define PWMCON0_PWM_ALL         0b11011111      // Todos os pinos de PWM habilitados para
                                                // saída de PWM
#define PWMCON0_PWM_0a5         0b11001111      // PWM0, PWM1, PWM2, PWM3, PWM4
                                                // e PWM5 habilitados para saída de PWM
#define PWMCON0_PWM_0a3         0b10111111      // PWM0, PWM1, PWM2 e PWM3
                                                // habilitados para saída de PWM
#define PWMCON0_PWM_0_1         0b10101111      // PWM0 e PWM1 habilitados para saída de
                                                // PWM
#define PWMCON0_PWM_1           0b10011111      // PWM1 habilitado para saída de PWM
#define PWMCON0_NOT_PWM         0b10001111      // Modulo de PWM desabilitado, todos os
                                                // pinos de PWM sao I/O
// PMOD3:PMOD0: PWM Output Pair Mode bits
// PMOD0
#define PWMCON0_PWM01_IND       0b11111111      // PWM0 e PWM1 estão no modo
                                                // independente
#define PWMCON0_PWM01_COMP      0b11111110      // PWM0 e PWM1 estão no modo
                                                // complementar
// PMOD1
#define PWMCON0_PWM23_IND       0b11111111      // PWM2 e PWM3 estão no modo
                                                // independente
#define PWMCON0_PWM23_COMP      0b11111101      // PWM2 e PWM3 estão no modo
                                                // complementar
// PMOD2
#define PWMCON0_PWM45_IND       0b11111111      // PWM4 e PWM5 estão no modo
                                                // independente
#define PWMCON0_PWM45_COMP      0b11111011      // PWM4 e PWM5 estão no modo
                                                // complementar
// PMOD3
#define PWMCON0_PWM67_IND       0b11111111      // PWM6 e PWM7 estão no modo
                                                // independente
#define PWMCON0_PWM67_COMP      0b11110111      // PWM6 e PWM7 estão no modo
                                                // complementar

// Mascara de configuração do PWMCON1 a ser usada na função vPWM_Init
// SEVOPS3:SEVOPS0: PWM Special Event Trigger Output Postscale Select bits
#define PWMCON1_SEVT_POST_1_1   0b00001111      // Postscaler 1:1
#define PWMCON1_SEVT_POST_1_2   0b00011111      // Postscaler 1:2
#define PWMCON1_SEVT_POST_1_3   0b00101111      // Postscaler 1:3
#define PWMCON1_SEVT_POST_1_4   0b00111111      // Postscaler 1:4
#define PWMCON1_SEVT_POST_1_5   0b01001111      // Postscaler 1:5
#define PWMCON1_SEVT_POST_1_6   0b01011111      // Postscaler 1:6
#define PWMCON1_SEVT_POST_1_7   0b01101111      // Postscaler 1:7
#define PWMCON1_SEVT_POST_1_8   0b01111111      // Postscaler 1:8
#define PWMCON1_SEVT_POST_1_9   0b10001111      // Postscaler 1:9
#define PWMCON1_SEVT_POST_1_10  0b10011111      // Postscaler 1:10
#define PWMCON1_SEVT_POST_1_11  0b10101111      // Postscaler 1:11
#define PWMCON1_SEVT_POST_1_12  0b10111111      // Postscaler 1:12
```

```
#define PWMCON1_SEVT_POST_1_13   0b11001111      // Postscaler 1:13
#define PWMCON1_SEVT_POST_1_14   0b11011111      // Postscaler 1:14
#define PWMCON1_SEVT_POST_1_15   0b11101111      // Postscaler 1:15
#define PWMCON1_SEVT_POST_1_16   0b11111111      // Postscaler 1:16
// SEVTDIR: Special Event Trigger Time Base Direction bit
// Um "Special Event Trigger" irá ocorrer quando a base de tempo do PWM estiver a contar
// de forma decrescente
#define PWMCON1_SEVT_DOWN        0b11111111
// Um "Special Event Trigger" irá ocorrer quando a base de tempo do PWM estiver a contar
// de forma crescente
#define PWMCON1_SEVT_UP          0b11110111
// UDIS: PWM Update Disable bit
#define PWMCON1_UPDC_DIS         0b11111111      // Actualizações do registo do Duty Cycle
                                                 // e do Período estão desabilitadas
#define  PWMCON1_UPDC_EN         0b11111101      // Actualizações do registo do Duty Cycle
                                                 // e do Período estão habilitadas
// OSYNC: PWM Output Override Synchronization bit
// Saídas activadas manualmente pelo registo OVDCON estão sincronizadas com a base de tempo do
// PWM
#define PWMCON1_OVDCON_SYNC      0b11111111
// Saídas activadas manualmente pelo registo OVDCON são assíncronas
#define  PWMCON1_OVDCON_ASYNC    0b11111110

/* ------------------------------------ Fim Registos de controlo de PWM ------------------------------------ */

/* ********************************** Definições ********************************** */

// Definições dos PWM
#define pwm_lfosc                20000000        // Frequência de oscilação
#define pwm_lfreq                20000           // Frequência de PWM do motor dc
#define pwm_iprescaler           1               // Prescaler - Se for alterado é necessário alterar as
                                                 // definições PTCON0_PS_1_x

// Definição para parar os motores dc
#define dc_stop          2
// Definições do motor dc1 e dc2
#define dc_down          1
#define dc_up            0
// Definições do motor dc3
#define dc_back          0
#define dc_front         1
// Definições do motor dc4
#define dc_close         1
#define dc_open          0
// Definições do motor de passo
#define step_left        0
#define step_right       1
// Número de vezes que as brocas se deslocam
#define tableVersion     6       // 4 - versão americana; 6 - versão europeia

/* ------------------------------------------------- Fim Definições ------------------------------------------------- */

/* ****************************** Declaração de Variáveis ****************************** */

unsigned char motorNumber = 1;           // Número do motor, utilizado no reset
int stepCenter = 1350;                   // Número de passos para por a mesa no centro
//int stepCenter = 810;                  // US - 810, Euro - 1350

/* ---------------------------------------- Fim Declaração de Variáveis ---------------------------------------- */
```

```
/* ****************************** Declaração das Funções ***************************** */

// Funções do PWM

// Definições dos registos de controlo do timer e do pwm
// Variáveis de entrada: hPtcon0 - Registo de controlo 0 do timer do PWM
//                       hPtcon1 - Registo de controlo 1 do timer do PWM
//                       hPwmcon0 - Registo de controlo 0 do PWM
//                       hPwmcon1 - Registo de controlo 1 do PWM
void vPWM_Init( unsigned char hPtcon0, unsigned char hPtcon1, unsigned char hPwmcon0, unsigned char hPwmcon1 );

// Calcula o período do pwm
void vPWM_Period( void );

// Calcula o duty cycle do pwm
// Variáveis de entrada: iPerDutyCycle - Duty cycle em percentagem
// Valor devolvido: dutyCycle - Valor do duty cycle a definir em PDCxL e PDCxH
int iPWM_DutyCycle( unsigned char iPerDutyCycle );

// Funções de controlo do motor dc

// Define o sentido de rotação do motor
// Variáveis de entrada: iMotorNumber - Indica o motor dc pretendido: Prende as folhas (1), Desloca
// bloco de furação (2), Desloca mecanismo de fecho do clickbind (3) ou Fecha clickbind (4)
//                       bDirection - Sentido Horário (1) ou Anti-Horário (0)
//                       iPerDutyCycle - Duty cycle em percentagem
void vDC_Mode( unsigned char iMotorNumber, unsigned char bDirection, unsigned char iPerDutyCycle );

// Verifica se o motor está parado junto das folhas
// Variável de entrada: iMotorNumber - Indica o motor dc pretendido: Prende as folhas (1), Desloca
// bloco de furação (2), Desloca mecanismo de feche do clickbind (3) ou Fecha clickbind (4)
// Valor devolvido: motorStop - Motor parado (1) ou motor em andamento (0)
unsigned char bDC_Current( unsigned char iMotorNumber );

// Inicializa motor de passo
void vSTEP_Init( void );

// Faz o motor dar um passo
// Variáveis de entrada: itime - Tempo entre um clock a 1 e outro a 0, define a velocidade do motor
void vSTEP_Clock( unsigned char itime );

// Interrupção externa para parar a máquina
void vSTOP_Interrupt( void );

// Depois de ocorrer uma interrupção, faz parar os motores e envia informação para o LCD
void vINTERRUPT_Stop( void );

// Reiniciar motores
void vINTERRUPT_Restart( void );

/* ---------------------------------------- Fim Declaração das Funções ---------------------------------------- */
```

```
/* ------------------------------------------------------------------------------------------------------------------
 * Nome do Projecto: Hollowgrail
 * Cliente: ACCO
 *
 * Nome do Ficheiro: hollowgrail.c
 * Descrição do Ficheiro: Código para automatização de uma máquina
 *                        de encadernar
 *
 * Data: 06.02.2008
 * ------------------------------------------------------------------------------------------------------------------ */

#include "hollowgrail.h"
#include "motor.h"

/* ************************* Implementação das Funções do motor ************************ */

// Nos pic18 as interrupções de alta prioridade estão em 0x08h
#pragma code stop_interrupt = 0x08
void stop_int (void)
{
	_asm goto vSTOP_Interrupt _endasm
}
#pragma code

// Interrupção que faz parar a máquina
#pragma interrupt vSTOP_Interrupt
void vSTOP_Interrupt( void )
{
	INTCON3bits.INT1IF = 0;		// Apaga a flag de interrupção
	vSENSOR_Read();			// Lê valor dos sensores
	vINTERRUPT_Stop();		// Pára os motores
	cState='u';			// No regresso à main vai para o estado ‘u’
}

// Inicializa pinos de controlo dos multiplexeres como saída
void vSENSOR_ControlTris( void )
{
	CONTROL_MUX_TRIS = CONTROL_MUX_TRIS & mask_mux;
}

// Define o valor dos pinos de controlo dos multiplexeres
void vSENSOR_Control( unsigned char iNumber )
{
	CONTROL_MUX = CONTROL_MUX & mask_mux;
	CONTROL_MUX = CONTROL_MUX | (iNumber & (255-mask_mux));
}

// Lê valor dos sensores
void vSENSOR_Read( void )
{
	unsigned char i;

	for(i=0;i<=15;i++)
	{
		vSENSOR_Control(i);
		// Guarda os valores dos sensores ligados ao multiplexer
		sensor[i] = SENSOR_MUX;
	}
	// Guarda os valores dos sensores ligados directamente no pic
	sensor[dc1_current] = DC1_CURRENT;
```

```
	sensor[sensor_drillPositionA] = SENSOR_DRILLPOSITIONA;
	sensor[sensor_drillPositionB] = SENSOR_DRILLPOSITIONB;
	sensor[sensor_drillPositionC] = SENSOR_DRILLPOSITIONC;
	sensor[sensor_closeA] = SENSOR_CLOSEA;
	sensor[sensor_closeB] = SENSOR_CLOSEB;
	sensor[sensor_closeC] = SENSOR_CLOSEC;
	sensor[sensor_stepleft] = SENSOR_STEPLEFT;
	sensor[sensor_sizeClick] = (sensor[sensor_sizeClickA]<<1) | sensor[sensor_sizeClickB];
	sensor[sensor_checkClick] = (sensor[sensor_checkClickA]<<1) | sensor[sensor_checkClickB];
	sensor[sensor_close] = (sensor[sensor_closeA]<<2) | (sensor[sensor_closeB]<<1) |
				sensor[sensor_closeC];

	if(sensor[sensor_stop] || sensor[sensor_portOpen])
		flagStop = 1;
	else
		flagStop = 0;
}

// Verifica se a gaveta está cheia
unsigned char iSENSOR_DrawerFull( void )
{
	unsigned char sensorDrawer = 0;
	// Variável com o número de vezes seguidas que a gaveta está cheia
	static unsigned char countDrawerFull = 0;

	SENSOR_EMISSOR = 1;
	Delay10KTCYx(5);		// Espera 10ms

	vSENSOR_Read();
	if(sensor[sensor_drawerFull] == 0)
		sensorDrawer = 0;
	else
		sensorDrawer = 1;


	SENSOR_EMISSOR = 0;
	Delay10KTCYx(45);		// Espera 90ms

	return sensorDrawer;
}

// Definições dos registos de controlo do timer e do pwm
void vPWM_Init( unsigned char hPtcon0, unsigned char hPtcon1, unsigned char hPwmcon0, unsigned
char hPwmcon1 )
{
	PTCON0 = (0xFF & hPtcon0);		// Registo de controlo 0 do timer do PWM
	PTCON1 = (0xC0 & hPtcon1);		// Registo de controlo 1 do timer do PWM
	PWMCON0 = (0x7F & hPwmcon0);	// Registo de controlo 0 do PWM
	PWMCON1 = (0xFB & hPwmcon1);	// Registo de controlo 1 do PWM
}

// Calcula o período do pwm
void vPWM_Period( void )
{
	unsigned int period;

	// Cálculo do valor do período - Tpwm = [PTPER + 1]*4*TOSC*PTMRPS
	period = (pwm_lfosc/4) / (pwm_lfreq*pwm_iprescaler) - 1;

	PTPERL = (unsigned char)period;		// LSB do registo do período
```

```
	PTPERH = (unsigned char)(period>>8);	// MSB do registo do período
}

// Calcula o duty cycle do pwm
int iPWM_DutyCycle( unsigned char iPerDutyCycle )
{
	int dutyCycle;
	long freqDutyCycle;

	if( iPerDutyCycle > 0 & iPerDutyCycle <= 100 )
	{
		// Calcula a frequência correspondente a uma percentagem de PWM
		freqDutyCycle = pwm_lfreq;
		freqDutyCycle = freqDutyCycle * 100;
		freqDutyCycle = freqDutyCycle / iPerDutyCycle;

		// Cálculo do valor do duty cycle - Tdc = PDCx*Tosc*PTMRPS
		dutyCycle = pwm_lfosc / (freqDutyCycle*pwm_iprescaler);
	}
	else
		dutyCycle = 0;

	return dutyCycle;
}

// Define o sentido de rotação e duty cycle do motor
void vDC_Mode(unsigned char iMotorNumber, unsigned char bDirection, unsigned char iPerDutyCycle)
{
	int dutyCycle;

	dutyCycle = iPWM_DutyCycle(iPerDutyCycle);	// Calcula o duty cycle

	switch (iMotorNumber)
	{
		case 1:	// Motor dc 1 - Prende as folhas
			switch (bDirection)
			{
				// Se OVDCONDbits.POVDx=0, saída do pino de PWM é controlada
				// pelo valor do OVDCONSbits.POUTx
				// Se OVDCONDbits.POVDx=1, saída do pino de PWM toma o valor
				// do duty cycle
				case 0:	// Sentido Anti-Horário
					OVDCONDbits.POVD0 = 1;
					OVDCONSbits.POUT0 = 0;
					OVDCONDbits.POVD1 = 0;
					OVDCONSbits.POUT1 = 0;
					break;
				case 1:	// Sentido Horário
					OVDCONDbits.POVD0 = 0;
					OVDCONSbits.POUT0 = 0;
					OVDCONDbits.POVD1 = 1;
					OVDCONSbits.POUT1 = 0;
					break;
				case 2:	// Motor parado
					OVDCONDbits.POVD0 = 0;
					OVDCONSbits.POUT0 = 0;
					OVDCONDbits.POVD1 = 0;
					OVDCONSbits.POUT1 = 0;
					break;
			}
```

```
            PDC0L = (unsigned char)dutyCycle;          // LSB do registo do dutycycle
            PDC0H = (unsigned char)(dutyCycle>>8);  // MSB do registo do dutycycle
            break;
case 2:     // Motor dc 2 - Desloca o bloco de furação
            switch (bDirection)
            {
                  case 0:   // Sentido Anti-Horário
                        OVDCONDbits.POVD2 = 1;
                        OVDCONSbits.POUT2 = 0;
                        OVDCONDbits.POVD3 = 0;
                        OVDCONSbits.POUT3 = 0;
                        break;
                  case 1:   // Sentido Horário
                        OVDCONDbits.POVD2 = 0;
                        OVDCONSbits.POUT2 = 0;
                        OVDCONDbits.POVD3 = 1;
                        OVDCONSbits.POUT3 = 0;
                        break;
                  case 2:   // Motor parado
                        OVDCONDbits.POVD2 = 0;
                        OVDCONSbits.POUT2 = 0;
                        OVDCONDbits.POVD3 = 0;
                        OVDCONSbits.POUT3 = 0;
                        break;
            }
            PDC1L = (unsigned char)dutyCycle;          // LSB do registo do dutycycle
            PDC1H = (unsigned char)(dutyCycle>>8);  // MSB do registo do dutycycle
            break;
case 3:     // Motor dc 3 - Desloca mecanismo de fecho do clickbind
            switch (bDirection)
            {
                  case 0:   // Sentido Anti-Horario
                        OVDCONDbits.POVD4 = 1;
                        OVDCONSbits.POUT4 = 0;
                        OVDCONDbits.POVD5 = 0;
                        OVDCONSbits.POUT5 = 0;
                        break;
                  case 1:   // Sentido Horario
                        OVDCONDbits.POVD4 = 0;
                        OVDCONSbits.POUT4 = 0;
                        OVDCONDbits.POVD5 = 1;
                        OVDCONSbits.POUT5 = 0;
                        break;
                  case 2:   // Motor parado
                        OVDCONDbits.POVD4 = 0;
                        OVDCONSbits.POUT4 = 0;
                        OVDCONDbits.POVD5 = 0;
                        OVDCONSbits.POUT5 = 0;
                        break;
            }
            PDC2L = (unsigned char)dutyCycle;          // LSB do registo do dutycycle
            PDC2H = (unsigned char)(dutyCycle>>8);  // MSB do registo do dutycycle
            break;
case 4:     // Motor dc 4 - Fecha clickbind
            switch (bDirection)
            {
                  case 0:   // Sentido Anti-Horario
                        OVDCONDbits.POVD6 = 1;
                        OVDCONSbits.POUT6 = 0;
                        OVDCONDbits.POVD7 = 0;
```

```
                                        OVDCONSbits.POUT7 = 0;
                                        break;
                                case 1: // Sentido Horario
                                        OVDCONDbits.POVD6 = 0;
                                        OVDCONSbits.POUT6 = 0;
                                        OVDCONDbits.POVD7 = 1;
                                        OVDCONSbits.POUT7 = 0;
                                        break;
                                case 2: // Motor parado
                                        OVDCONDbits.POVD6 = 0;
                                        OVDCONSbits.POUT6 = 0;
                                        OVDCONDbits.POVD7 = 0;
                                        OVDCONSbits.POUT7 = 0;
                                        break;
                        }
                        PDC3L = (unsigned char)dutyCycle;          // LSB do registo do dutycycle
                        PDC3H = (unsigned char)(dutyCycle>>8);  // MSB do registo do dutycycle
                        break;
        }
}

// Verifica se o motor esta parado
unsigned char bDC_Current( unsigned char iMotorNumber )
{
        unsigned char motorStop = 0;
        // Variável com o numero de vezes seguidas que a corrente foi máxima no motor1
        static unsigned char countCurrentMax1 = 0;
        // Variável com o numero de vezes seguidas que a corrente foi máxima no motor4
        static unsigned char countCurrentMax4 = 0;
        switch (iMotorNumber)
        {
                case 1: // Motor dc 1 - Prende as folhas
                        // Testa se countCurrentMax ocorreu menos de 3 vezes seguidas
                        if(countCurrentMax1 <= 2)
                        {
                                if(sensor[dc1_current])              // Se a corrente for máxima
                                {
                                        countCurrentMax1++;      // Incrementa countCurrentMax
                                        Delay10KTCYx(50);        // Espera 0.1s
                                }
                                else
                                        countCurrentMax1 = 0;
                        }
                        else
                        {
                                motorStop = 1;                // Motor está parado junto das folhas
                                countCurrentMax1 = 0;
                        }
                        break;

                case 4: // Motor dc 4 -> Fecha clickbind
                        // Testa se countCurrentMax ocorreu menos de 4 vezes seguidas
                        if(countCurrentMax4 <= 2)
                        {
                                if(sensor[dc4_current])              // Se a corrente for máxima
                                {
                                        countCurrentMax4++;      // Incrementa countCurrentMax
                                        Delay10KTCYx(50);        // Espera 0.1s
                                }
                                else
```

```
                                        countCurrentMax4 = 0;
                        }
                        else
                        {
                                        motorStop = 1;    // Motor fechou ou abriu o clickbind
                                        countCurrentMax4 = 0;
                        }
                        break;
        }
        return motorStop;
}

// Inicializa motor de passo
void vSTEP_Init( void )
{
        STEP_CE = 1;                    // Habilita motor de passo
        STEP_RST = 0;                   // Faz o reset ao motor de passo
        STEP_RST = 1;
}

// Faz o motor dar um passo
void vSTEP_Clock( unsigned char itime )
{
        STEP_CLOCK = 0;
        Delay100TCYx(itime);
        STEP_CLOCK = 1;
        Delay100TCYx(itime);
}

// Depois de ocorrer uma interrupção, faz parar os motores e envia informação para o LCD
void vINTERRUPT_Stop( void )
{
        // Pára todos os motores
        vDC_Mode(1,dc_stop,0);
        vDC_Mode(2,dc_stop,0);
        vDC_Mode(3,dc_stop,0);
        vDC_Mode(4,dc_stop,0);
        DC5_CONTROL = 0;
        STEP_CE = 0;

        if(sensor[sensor_portOpen])
        {
                if (cError!='E')
                {
                        cError = 'E';
                        // LDC - "Please close maintenance door"
                        vI2C_SendCaracter(ADDR_LCD_W,cError);
                }
        }
        cError = ' ';
}

// Reiniciar motores
void vINTERRUPT_Restart( void )
{
        int i;

        vSENSOR_Read();
        switch (motorNumber)
        {
```

```
case 1:  // Motor dc 4 - Fecha clickbind
         vDC_Mode(4,dc_open,80);          // Abre mecanismo de fecho
         if(bDC_Current(4))               // Quando estiver aberto, pára
         {
                  vDC_Mode(4,dc_stop,0);
                  motorNumber = 2;        // Passa para o motor DC3
         }
         break;
case 2:  // Motor dc 3 - Desloca mecanismo de feche do clickbind
         switch(sensor[sensor_close])
         {
                  case 1:  // Se a porta está aberta
                           // Desloca-se até à posição de porta fechada
                           while(sensor[sensor_close] != 2)
                           {
                                    vSENSOR_Read();
                                    vDC_Mode(3,dc_front,70);
                           }
                           vDC_Mode(3,dc_stop,0);
                           motorNumber = 3;              // Passa para o motor DC2
                           break;
                  case 2:  // Se a porta está fechada
                           motorNumber = 3;              // Passa para o motor DC2
                           break;
                  case 4:  // Se está na posição de encadernar clickbind grande
                  case 6:  // Se está na posição de encadernar clickbind médio
                  case 7:  // Se está na posição de encadernar clickbind pequeno
                           while(sensor[sensor_close] != 2)
                           {
                                    vSENSOR_Read();
                                    vDC_Mode(3,dc_back,70);
                           }
                           vDC_Mode(3,dc_stop,0);
                           motorNumber = 3;              // Passa para o motor DC2
                           break;
                  default: // Se motor está numa posição desconhecida
                           // Motor vai para a posição de porta aberta
                           while(sensor[sensor_close] != 1)
                           {
                                    vSENSOR_Read();
                                    vDC_Mode(3,dc_back,70);
                           }
                           vDC_Mode(3,dc_stop,0);
                           // Ao atingi-la desloca-se para a posição de porta fechada
                           while(sensor[sensor_close] != 2)
                           {
                                    vSENSOR_Read();
                                    vDC_Mode(3,dc_front,70);
                           }
                           vDC_Mode(3,dc_stop,0);
                           motorNumber = 3;              // Passa para o motor DC2
         }
         break;
case 3:  // Motor dc 2 - Desloca o bloco de furação
         DC5_CONTROL = 1;                 // Liga motor das brocas
         // Motor desloca-se para a posição A
         vDC_Mode(2,dc_down,50);
         if(sensor[sensor_drillPositionA])
         {
                  vDC_Mode(2,dc_stop,0);
```

```
                        DC5_CONTROL = 0;        // Desliga motor das brocas
                        motorNumber = 4;        // Passa para o motor DC1
                }
                break;
        case 4: // Motor dc 1 - Prende as folhas
                vDC_Mode(1,dc_up,50);
                // Motor afasta-se das folhas
                if(bDC_Current(1))              // Quando atinge posição inicial
                {
                        vDC_Mode(1,dc_stop,0);
                        motorNumber = 5;        // Passa para o motor de passo
                }
                break;
        case 5: // Motor de passo - Desloca a mesa
                vSTEP_Init();
                // Deslocamento da mesa para a esquerda até atingir o sensor_stepleft
                STEP_CW = step_left;
                while(sensor[sensor_stepleft] == 0)
                {
                        vSENSOR_Read();
                        vSTEP_Clock(250);
                }
                // Deslocamento da mesa para a direita até atingir a posição central
                STEP_CW = step_right;
                for(i=0;i<stepCenter;i++)
                {
                        vSTEP_Clock(250);
                }
                STEP_CE = 0;                    // Desabilita motor de passo
                motorNumber = 6;
                break;
        }
}

// Verifica se a máquina está em condições de começar o ciclo
unsigned char iSENSOR_Error( void )
{
        unsigned char sensorError = 0;

        if(iSENSOR_DrawerFull())        // Verifica se a gaveta está cheia ou mal colocada
        {
                if (cError!='A')
                {
                        cError = 'A';
                        // LDC - "waste full,empty waste full"
                        vI2C_SendCaracter(ADDR_LCD_W,cError);
                }
        }
        else
        {
                if(sensor[sensor_drillOut] == 0)    // Verifica se brocas estão mal colocadas
                {
                        if (cError!='B')
                        {
                                cError = 'B';
                                // LDC - "drill missing"
                                vI2C_SendCaracter(ADDR_LCD_W,cError);
                        }
                }
                else
```

```
		{
			// Verifica se batente das brocas está mal colocado
			if(sensor[sensor_padOut] == 0)
			{
				if (cError!='C')
				{
					cError = 'C';
					// LDC - "drill pad missing,check drill pad"
					vI2C_SendCaracter(ADDR_LCD_W,cError);
				}
			}
			else
				sensorError = 1;		// Não existem situações irregulares
		}
	}
	return sensorError;
}

// Envia um caracter por I2C
void vI2C_SendCaracter( unsigned char hAddr, unsigned char hCaracter )
{
	SWStartI2C();			// Inicia a comunicação por I2C
	SWWriteI2C(hAddr);		// Envia endereço do slave
	SWWriteI2C(hCaracter);		// Envia caracter para o slave
	SWStopI2C();			// Finaliza a comunicação por I2C
}

/* ---------------------------------- Fim Implementação das Funções do motor ---------------------------------- */

void main(void)
{
/* ***************************** Inicialização das Variáveis ***************************** */

	int i;
	unsigned char iMoveTable;	// Número de vezes que as brocas se deslocam
	int iStepTable = 540;		// Número de passos para andar 8.466mm

	Delay10KTCYx(250);		// Espera 0.5s para não haver problemas durante a
					// programação

	// Inicializa PORTA e PORTE como I/O digitais
	ANSEL0 = 0x00;
	ANSEL1 = 0x00;

	// Inicializa pinos dos sensores como entradas
	SENSOR_MUX_TRIS = 1;
	SENSOR_DRILLPOSITIONA_TRIS = 1;
	SENSOR_DRILLPOSITIONB_TRIS = 1;
	SENSOR_DRILLPOSITIONC_TRIS = 1;
	SENSOR_CLOSEA_TRIS = 1;
	SENSOR_CLOSEB_TRIS = 1;
	SENSOR_CLOSEC_TRIS = 1;
	// Inicializa pinos de controlo dos infravermelhos
	SENSOR_EMISSOR_TRIS = 0;
	// Inicializa pinos de controlo do componente 74HC4067 como saída
	vSENSOR_ControlTris();
	// Inicializa pino de controlo da corrente do motor dc 1 como entrada
	DC1_CURRENT_TRIS = 1;
	// Inicializa pino de controlo do motor dc5 como saída
	DC5_CONTROL_TRIS = 0;
```

```
	// Inicializa pinos de controlo do motor de passo como saídas
	STEP_CE_TRIS = 0;
	STEP_RST_TRIS = 0;
	STEP_CLOCK_TRIS = 0;
	STEP_CW_TRIS = 0;
	SENSOR_STEPLEFT_TRIS = 1;


/* ---------------------------------------- Fim Inicialização das Variáveis ---------------------------------------- */

	vPWM_Init(PTCON0_POST_1_1 &		// Postscaler 1:1
			PTCON0_PS_1_1 &		// Prescale 1:1
			PTCON0_FREE_RUNNING, // Modo Free-Running
			PTCON1_ON &		// Base de tempo do PWM ligada
			PTCON1_UP,		// Base de tempo do PWM conta crescentemente
			PWMCON0_PWM_ALL &	// Todos os pinos de PWM habilitados para saída
			PWMCON0_PWM01_IND &		// PWM0 e PWM1 no modo independente
			PWMCON0_PWM23_IND &		// PWM2 e PWM3 no modo independente
			PWMCON0_PWM45_IND &		// PWM4 e PWM5 no modo independente
			PWMCON0_PWM67_IND,		// PWM6 e PWM7 no modo independente
			PWMCON1_SEVT_POST_1_1 &	// Postscaler 1:1
// Um "Special Event Trigger" irá ocorrer quando a base de tempo do PWM estiver a contar crescentemente
			PWMCON1_SEVT_UP &
// Actualizações do registo do Duty Cycle e do Período estão habilitadas
			PWMCON1_UPDC_EN &
// Saídas activadas manualmente pelo registo OVDCON estão sincronizadas com a base de tempo do PWM
			PWMCON1_OVDCON_SYNC);
	// Valores iniciais das variáveis
	cState = 'a';
	SENSOR_EMISSOR = 0;
	// Inicializa motores
	DC5_CONTROL = 0;
	STEP_CE = 0;
	vPWM_Period();
	vDC_Mode(1,dc_stop,0);
	vDC_Mode(2,dc_stop,0);
	vDC_Mode(3,dc_stop,0);
	vDC_Mode(4,dc_stop,0);

	INTCON3bits.INT1IE = 1;		// Activar a interrupção externa int1
	INTCON2bits.INTEDG1 = 1;	// A interrupção ocorre na transição de 0 para 1
	RCONbits.IPEN = 1;		// Habilita a prioridade das interrupções
	INTCON3bits.INT1IP = 1;		// A interrupção a receber é de alta prioridade
	INTCONbits.GIEH = 1;		// Habilita todas as interrupções de alta prioridade

	// LCD - "GBC welcome"
	vI2C_SendCaracter(ADDR_LCD_W,cState);
	// Espera que termine de aparecer no LCD "GBC welcome"
	for(i=0;i<10;i++)
		Delay10KTCYx(250);

	motorNumber = 1;		// Faz o reset quando a máquina liga
	while (motorNumber<6)
		vINTERRUPT_Restart();

	cState = 'b';
```

```
while(1)
{
	vSENSOR_Read();		// Lê valor dos sensores
	switch(cState)
	{
		case 'b':	if(iSENSOR_Error())			// Se não houverem erros
				{
					if(sensor[sensor_paper])	// Se forem colocadas folhas na mesa
					{
						// LCD - "ready,insert book,press start"
						vI2C_SendCaracter(ADDR_LCD_W,cState);
						cState = 'c';		// Passa para o estado seguinte
					}
					else
					{
						if (cError!='D')
						{
							cError = 'D';
							// LCD - "ready,insert book"
							vI2C_SendCaracter(ADDR_LCD_W,cError);
						}
					}
				}
				break;
		case 'c':	if(sensor[sensor_start])	// Se botão de start for activado
				{
					// Motor dc 1 desloca-se em direcção as folhas
					vDC_Mode(1,dc_down,50);

					// LCD - em branco
					vI2C_SendCaracter(ADDR_LCD_W,cState);
					cState = 'd';		// Passa ao estado seguinte
				}
				else
				{
					if(sensor[sensor_paper] == 0)
						cState = 'b';		// Regressa ao estado anterior
				}
				break;
		case 'd':	if(bDC_Current(1))		// Se as folhas estiverem presas
				{
					vDC_Mode(1,dc_down,10);		// Motor dc 1 pára

					// Motor dc 3 faz abrir a porta para inserir o clickbind
					vDC_Mode(3,dc_back,50);
					cState = 'e';		// Passa ao estado seguinte
				}
				break;
		case 'e':	switch(sensor[sensor_sizeClick])	// Calcula tamanho do Clickbind
				{
					case 0:	// Clickbind pequeno
						cClick = '0';
						// LCD - "place spine below,small size"
						vI2C_SendCaracter(ADDR_LCD_W,cClick);
						cState='f';
						break;
					case 1:	// Clickbind médio
						cClick = '1';
						// LCD - "place spine below,medium size"
						vI2C_SendCaracter(ADDR_LCD_W,cClick);
```

```
                    cState='f';
                    break;
            case 3: // Clickbind grande
                    cClick = '2';
                    // LCD - "place spine below,large size"
                    vI2C_SendCaracter(ADDR_LCD_W,cClick);
                    cState='f';
                    break;
        }
        break;
case 'f': if(sensor[sensor_close] == 1)                 // Se a porta está aberta
        {
                vDC_Mode(3,dc_stop,0);                  // Motor dc 3 pára

                if(sensor[sensor_checkClick] == 0);             // Ausência de clickbind
                else
                {
                        // Consoante o tamanho do clickbind pedido
                        switch (sensor[sensor_sizeClick])
                        {
                        case 0: // Pretende-se argola pequena
                                // LCD - "ready to start binding, press start"
                                vI2C_SendCaracter(ADDR_LCD_W,cState);
                                cState='g';             // Passa ao estado seguinte
                                break;
                        case 1: // Pretende-se argola média

                                // Se a colocada argola média ou grande
                                if(sensor[sensor_checkClick] == click_medium ||
                                   sensor[sensor_checkClick] == click_large)
                                {
                                   // LCD - "ready to start binding, press start"
                                   vI2C_SendCaracter(ADDR_LCD_W,cState);
                                   cState='g';          // Passa ao estado seguinte
                                }
                                else       // Clickbind menor que o adequado
                                {
                                  if(sensor[sensor_checkClick] == click_small)
                                  {
                                    if (cClick!='3')
                                    {
                                       cClick = '3';
                                       // LCD - "please place,larger spine" – media
                                       vI2C_SendCaracter(ADDR_LCD_W,cClick);
                                    }
                                  }
                                }
                                break;
                        case 3: // Pretende-se argola grande

                                // Se a colocada argola grande
                                if(sensor[sensor_checkClick] == click_large)
                                {
                                    // LCD - "ready to start binding, press start"
                                    vI2C_SendCaracter(ADDR_LCD_W,cState);
                                    cState='g';     // Passa ao estado seguinte
                                }
                                else       // Clickbind menor que o adequado
                                {
```

```
                                        if (cClick!='4')
                                        {
                                            cClick = '4';
                                            // LCD - "please place,larger spine" - grande
                                            vI2C_SendCaracter(ADDR_LCD_W,cClick);
                                        }
                                    }
                                    break;
                                }
                            }
                        }
                        break;
case 'g':   if (sensor[sensor_start])    // Se botão de start for novamente activado
                        {
                            // Motor dc 3 faz fechar a porta para dar início à furação
                            vDC_Mode(3,dc_front,50);
                            cState='h';                     // Passa ao estado seguinte
                        }
                        break;
case 'h':   if(sensor[sensor_close] == 2)         // Se a porta está fechada
                        {
                            vDC_Mode(3,dc_stop,0);              // Motor dc 3 pára
                            // LCD - "punching"
                            vI2C_SendCaracter(ADDR_LCD_W,cState);
                            iMoveTable = 0;
                            cState='i';                      // Passa ao estado seguinte
                        }
                        break;
case 'i':   if (iMoveTable > (tableVersion-1))              // Se última furação acabou
                            cState='n';
                        else
                            cState='j';
                        break;
case 'j':   if(iMoveTable == 0)         // Na primeira etapa da furação
                        {
                            vSTEP_Init();     // Inicializa motor de passo
                            // Mesa desloca-se para a posição da primeira furação
                            STEP_CW = step_left;
                            // Enquanto não atingir o sensor
                            while(sensor[sensor_stepleft] == 0)
                            {
                                vSENSOR_Read();
                                vSTEP_Clock(250);
                            }
                            STEP_CE = 0;    // Desabilita motor de passo
                            // Motor dc 2 desloca as brocas desde a posição inicial até ao meio
                            vDC_Mode(2,dc_up,80);
                        }
                        cState = 'k';         // Em todas as etapas da furação passa ao estado seguinte
                        break;
case 'k':   if(sensor[sensor_drillPositionB])    // Em todas as etapas da furação
                        {
                            // Liga motor DC que faz o movimento rotacional das brocas
                            DC5_CONTROL = 1;
                            // Motor DC2 desloca, em rampa de desaceleração, o bloco de furação
                            for(i=0;i<15;i++)
                            {
                                vDC_Mode(2,dc_up,80-i*3);
                                // Dependendo do tempo que demorar a chegar às folhas
                                Delay10KTCYx(100);
```

```
                }
        cState = 'l';
        }
        break;
case 'l':  i=0;
        // Enquanto não atingir a posição C
        while(SENSOR_DRILLPOSITIONC == 0)
        {
                // Motor DC2 desloca, em rampa de aceleração, o bloco de furação
                if(i<8)
                {
                        vDC_Mode(2,dc_up,20+i*3);
                        // Ajustar dependendo do tempo que demorar a chegar a cima
                        Delay10KTCYx(250);
                        i++;
                }
                else
                        vDC_Mode(2,dc_up,40);
        }
        // Motor DC2 desloca bloco de furação para a posição intermédia
        vDC_Mode(2,dc_down,60);
        cState = 'm';
        break;
case 'm': if(sensor[sensor_drillPositionB])
        {
                vDC_Mode(2,dc_stop,0);          // Motor dc 2 pára
                // Desliga motor DC que faz o movimento rotacional das brocas
                DC5_CONTROL = 0;
                iMoveTable++;
                if (iMoveTable == tableVersion)    // Na última etapa da furacao
                {
                        vSTEP_Init();
                        // Motor que desloca a mesa volta a posição inicial
                        STEP_CW = step_left;
                        for(i=0;i<stepCenter;i++)
                                vSTEP_Clock(250);
                        STEP_CE = 0;            // Desabilita motor de passo
                }
                else
                {
                        vSTEP_Init();
                        // Motor que desloca a mesa anda para a direita 8.466mm
                        STEP_CW = step_right;
                        for(i=0;i<iStepTable;i++)
                                vSTEP_Clock(250);
                        STEP_CE = 0;
                }
                cState = 'i';
        }
        break;
case 'n':  // Terminou furação (4 ou 6 vezes)
        vDC_Mode(1,dc_up,50);           // Motor dc 1 afasta-se das folhas
        cState = 'o';
        break;
case 'o':  if(bDC_Current(1))                   // Se tiver chegado a cima
        {
                vDC_Mode(1,dc_stop,0);          // Motor dc 1 pára
                // LCD - "binding"
                vI2C_SendCaracter(ADDR_LCD_W,cState);
```

```
                // Motor dc 3 desloca sistema de encadernação do clickbind
                vDC_Mode(3,dc_front,50);
                cState = 'p';
        }
        break;
case 'p':
        // Consoante o tamanho do clickbind pedido
        switch (sensor[sensor_sizeClick])
        {
                case 0:  // Para o clickbind pequeno

                        // Se o sistema de encadernação estiver na posição de fecho
                        // do clickbind pequeno
                        if(sensor[sensor_close] == 7)
                        {
                                vDC_Mode(3,dc_stop,0);     // Motor dc 3 pára
                                vDC_Mode(4,dc_close,80);  // Motor dc 4 sobe
                                cState='q';
                        }
                        break;
                case 1:  // Para o clickbind pequeno

                        // Se o sistema de encadernação estiver na posição de fecho
                        // do clickbind médio
                        if(sensor[sensor_close] == 6)
                        {
                                vDC_Mode(3,dc_stop,0);      // Motor dc 3 pára
                                vDC_Mode(4,dc_close,80);  // Motor dc 4 sobe
                                cState='q';
                        }
                        break;
                case 3:  // Para o clickbind pequeno

                        // Se o sistema de encadernação estiver na posição de fecho
                        // do clickbind grande
                        if(sensor[sensor_close] == 4)
                        {
                                vDC_Mode(3,dc_stop,0);      // Motor dc 3 pára
                                vDC_Mode(4,dc_close,80);  // Motor dc 4 sobe
                                cState='q';
                        }
                        break;
        }
        break;
case 'q': if(bDC_Current(4))         // Se argola fechou
        {
                vDC_Mode(4,dc_open,80);          // Motor dc 4 desce mecanismo
        cState='r';
        }
        break;
case 'r': if(bDC_Current(4))         // Se argola abriu
        {
                vDC_Mode(4,dc_stop,0);           // Motor dc 4 pára
                // Motor dc 2 desloca as brocas desde a posição do meio até à inicial
                vDC_Mode(2,dc_down,60);
                cState='s';
        }
        break;
case 's': if(sensor[sensor_drillPositionA])
        {
```

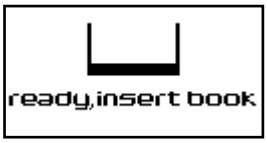
```
                        vDC_Mode(2,dc_stop,0);              // Motor dc 2 pára

                        // Motor dc 3 desloca-se para a posição de porta fechada
                        vDC_Mode(3,dc_back,30);
                        cState='t';
                }
                break;
        case 't':  if(sensor[sensor_close] == 2)        // Se a porta está fechada
                {
                        vDC_Mode(3,dc_stop,0);              // Motor dc 3 pára
                        // LCD - "thank you for,binding with,gbc"
                        vI2C_SendCaracter(ADDR_LCD_W,cState);
                        cState='v';          // Passa para o último estado
                }
                break;
        case 'u':  if(flagStop)         // Enquanto stop activo ou porta das brocas aberta
                        vSENSOR_Read();
                else
                {
                        if(sensor[sensor_start])                // Se botão de start for activado
                        {
                                motorNumber = 1;            // Faz reset a todos os motores
                                while (motorNumber<6)
                                        vINTERRUPT_Restart();
                                cState='v';          // Passa para o último estado
                        }
                        else
                        {
                                if (cError!='F')
                                {
                                        cError = 'F';
                                        // LCD - "Error during process, please press Enter
                                        // to reset machine and remove sheets"
                                        vI2C_SendCaracter(ADDR_LCD_W,cError);
                                }
                        }
                }
                break;
        case 'v':  // Se sensor de presença de papel não o detectar
                if(sensor[sensor_paper] == 0)
                {
                        cError = ' ';
                        cState='b';          // Volta ao estado inicial
                }
                break;
        }
    }
}
```

Anexo II – Lista de mensagens gráficas

Mensagens gráficas de boas vindas

Mensagem gráfica com pedido de inserção do documento a encadernar



Mensagem gráfica com pedido para pressionar botão *start*, após a inserção do documento

Mensagem gráfica com indicação da argola apropriada para a encadernação

Mensagem gráfica com a informação de que a argola utilizada é mais pequena que a apropriada, sendo necessário colocar uma de tamanho médio ou grande

Mensagem gráfica com a informação de que a argola utilizada é mais pequena que a apropriada, sendo necessário colocar uma de tamanho grande

Mensagem gráfica com pedido para pressionar botão *start*, para dar início ao processo de furação

Mensagem gráfica durante o processo de furação

Mensagem gráfica durante o processo de encadernação

Mensagem gráfica de finalização da encadernação

Mensagem gráfica com a informação de que a gaveta se encontra cheia ou mal colocada

Mensagem gráfica com a informação de que o batente das brocas não se encontra colocado

Mensagem gráfica com a informação de que a porta que dá acesso às brocas se encontra aberta