David Miguel Ameixa Crispim

Licenciatura em Ciências da
Engenharia Electrotécnica e de Computadores

Desenvolvimento de um Sistema de Registo de Lâminas Histológicas

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador:	José Manuel Matos Ribeiro da Fonseca, Professor Auxiliar com Agregação, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
Co-orientador:	André Teixeira Bento Damas Mora, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri:

Presidente:	Doutor Rodolfo Alexandre Duarte Oliveira, Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
Vogais:	Doutor José Manuel Matos Ribeiro da Fonseca, Professor Auxiliar com Agregação da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
	Doutor João Almeida das Rosas, Professor Auxiliar da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2017

Desenvolvimento de um Sistema de Registo de Lâminas Histológicas

À minha família

Agradecimentos

Antes de mais, gostaria de agradecer aos professores José Manuel Fonseca e André Damas Mora, do Departamento de Engenharia Electrotécnica, pela sua orientação e disponibilidade, partilha de ideias e de experiência, e ajuda em momentos mais complicados.

Gostaria também de agradecer à instituição que disponibilizou as lâminas histológicas, utilizadas para testar o protótipo desenvolvido.

Agradeço aos colegas que me acompanharam desde o início do curso, Inês Martins, Diogo Alves, Frederico Atanásio e Miguel João e os restantes que não vou mencionar, mas que sabem quem são. A vossa amizade e apoio vão ficar, para sempre, guardadas comigo.

Aos meus pais e irmão pelo seu apoio incondicional, carinho e partilha de ideias que me ajudaram a progredir com a construção do protótipo. Um agradecimento em especial ao meu pai, que me ajudou a contruir a estrutura em madeira. Outro agradecimento à minha prima Inês Martins que, para além de prima foi uma colega de curso sempre pronta a ajudar-me quando eu precisava. Por último, à Teresa pelo seu amor, apoio e por estar sempre ao meu lado durante toda esta jornada.

Resumo

O Mundo da Medicina está, constantemente, em busca de alternativas que inovem o mesmo, e que forneçam métodos mais eficientes de execução das tarefas necessárias, quaisquer que estas sejam. Deste modo, esta sensível área da sociedade mantém-se em constante progresso, o que é essencial para o bem-estar e saúde de todos os seus membros.

No seguimento do que foi dito, a solução apresentada nesta dissertação é uma tentativa de inovação no âmbito de registo de informações relativas às amostras de biopsias que são realizadas em institutos de oncologia. O protótipo desenvolvido pretende realizar o dito registo de informação presente nas lâminas histológicas, onde são guardadas as amostras de biopsias, utilizando diferentes tecnologias, entre as quais, motores, sensores e um Raspberry Pi para coordenar todos os mecanismos envolvidos e aplicar o *software* de reconhecimento óptico de caracteres.

Utilizando este protótipo, os profissionais de laboratório poderão usufruir de toda esta informação em formato digital, reduzindo os tempos de acesso e de pesquisa.

Palavras-chave: Lâmina histológica, biopsia, protótipo, reconhecimento óptico de caracteres, Raspberry Pi, Python.

Abstract

The Health Care World is in constant search for changes that may bring innovation and provide more efficient ways of executing tasks. Thus, this sensible area of society shows constant progress, which is essential to the well-being of its members.

Following what has been said, the solution presented in this dissertation is an attempt to innovate the recording of biopsies' information, performed at oncology institutes. The developed prototype aims to perform the histological slides' information recording, where the biopsies are stored, by means of different technologies, among which can be found motors, sensors and a Raspberry Pi responsible for coordinating all mechanisms involved in and applying to the optical character recognition software.

By using this prototype, laboratory professionals will be able to take advantage of all the data in digital format, reducing time wasted on access and search.

Keywords: Histological slide, biopsy, prototype, optical character recognition, Raspberry Pi, Python.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas

IOV - Instituto de Oncologia Virtual

OCR – Optical Character Recognition

ASCII - American Standard Code for Information Interchange

PDF – Portable Document Format

XPS - XML Paper Specification

GPIO – General Purpose Input/Output

LED - Light Emitting Diode

RAM – Random Access Memory

HDMI - High-Definition Multimedia Interface

USB – Universal Serial Bus

IP – Internet Protocol

CPU – Central Processing Unit

PWM – Pulse-Width Modulation

PIL – Python Imaging Library

1

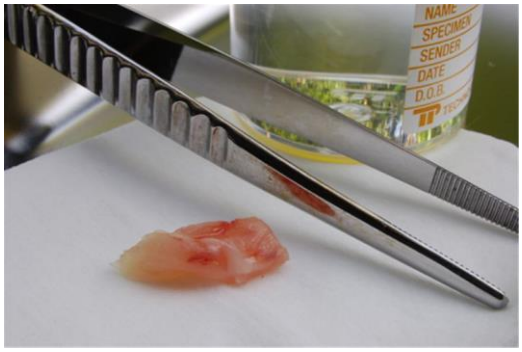
Introdução

1.1 Origem do Problema e Contextualização

O tema desta dissertação foi proposto ao aluno para tentar satisfazer o pedido de criação de um sistema de registo de informações relativas às lâminas histológicas (capaz de colmatar a lacuna criada pela actual inexistência de algo semelhante no Mercado) por uma instituição especializada no combate activo às doenças cancerígenas e/ou oncológicas (nas suas diferentes vertentes, desde a investigação, ao tratamento, à educação e, ainda, à prevenção). Por motivos de ordem ética, o nome desta instituição será omitido no presente documento e substituído por IOV (Instituto de Oncologia Virtual).

As actividades deste tipo de instituições incidem, inicialmente, no estudo e na investigação das causas subjacentes a doenças cancerígenas e/ou oncológicas para que, posteriormente, seja possível reverter os efeitos nocivos das mesmas através de diagnósticos mais céleres e tratamentos mais eficazes. Para além disto, este tipo de instituições procura difundir o conhecimento relacionado com as doenças acima referidas junto da população em geral, de modo a incentivar as pessoas a terem um estilo de vida mais preventivo e, por conseguinte, a diminuir o nível de incidência destas doenças. Para que seja possível proceder ao estudo e à investigação das causas subjacentes a doenças cancerígenas e/ou oncológicas, é necessário (nalguns casos) efectuar uma biópsia e o conjunto de testes que a sucedem (processo ao qual os seguintes parágrafos se irão dedicar).

O artigo "What Happens to Biopsy and Citology Specimens" (The A. C. S. medical and editorial content team, 2015) afirma que, após a remoção do tecido do paciente (Figura 1.1 a)), a amostra é misturada, com uma solução de água e formol (cerca de 10%), de modo a ser preservada para ser mais tarde colocada num recipiente etiquetado com as informações identificativas do paciente e da origem anatómica do tecido (Figura 1.1 b)).



a)

b)

Figura 1.1 - Primeiras duas etapas do procedimento. a) Exemplo de um tecido retirado durante uma biopsia. (Retirado de **http://www.leicabiosystems.com/pathologyleaders/an-intro-duction-to-specimen-preparation**). b) Exemplo de recipientes utilizados na conservação da amostra de tecido. (Retirado de **https://www.therapak.com/catalog/histology**)

O recipiente é então enviado para o laboratório de patologia onde será examinado. A primeira análise é feita a olho nu, ou seja, o patologista ou um assistente treinado examinam a amostra sem microscópio. Deste primeiro exame é possível retirar algumas informações cruciais relativas ao tecido extraído, de entre as quais se destacam as dimensões físicas, a consistência e, no caso de profissionais mais experientes, certos detalhes que possam sugerir, ou não, a existência de uma doença. Após esta análise, o tecido é colocado num recipiente denominado bloco (Figura 1.2), que tem como propósito albergar a amostra enquanto a mesma está a ser processada.

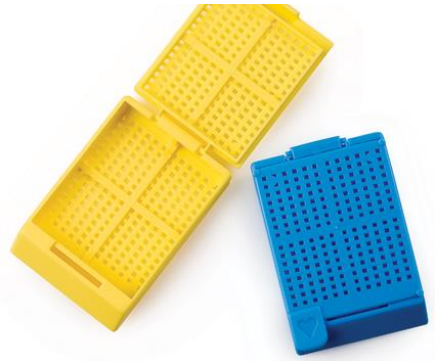


Figura 1.2 - Exemplo de blocos. (Retirado de **http://www.leicabiosystems.com/pt/preparacao-das-amostras/moldes-e-cassetes/estojos-de-biopsia/detalhes/product/biopsy-cassettes**)

Concluído o processo, o tecido é envolvido em parafina quente que, quando arrefecida, cria um invólucro protector. A parafina é uma cera insolúvel em água e de muito baixa reactividade, o que a transforma num bom elemento de protecção. O combinado de parafina e tecido é cortado em camadas bastante finas através da utilização de um instrumento designado por *microtome* (Figura 1.3), e cada camada é subsequentemente colocada numa lâmina histológica. Antes da sua análise, as camadas atravessam ainda um processo de coloração, de modo a facilitar a visualização das células. Feita a análise, as lâminas são arquivadas. O Manual de Boas Práticas Laboratoriais de Anatomia Patológica, publicado em anexo ao Despacho n.º 13 832/2007, de 29.06, dita que as lâminas terão de ser arquivadas durante, pelo menos, dez anos no caso de patologia oncológica e cinco anos em outras situações.

Figura 1.3 - Microtome da Leica Biosystems. (Retirado de http://www.leicabiosystems.com/histology-equipment/microtomes/products/leica-rm2235)

A inexistência de um sistema capaz de registar as diversas informações das lâminas histológicas onde são guardadas as amostras da biopsia[1] foi comunicada pela IOV (ainda que verifique a inexistência de um sistema deste tipo não só na instituição parceira, mas também a larga escala).

1.2 Proposta

Em resposta ao problema comunicado pela IOV, o protótipo desenvolvido visa, como tal, colmatar uma lacuna existente nesta área, para tal criando um método de registo de informações das lâminas histológicas e dos respectivos processos.

Previamente à apresentação da solução, há que referir que foi decidido que o protótipo será criado sob o título de prova de conceito, isto é, um aparelho que representa um modelo prático, limitando-se a executar de forma simples a função para que foi criado. De uma forma prática, tentou-se conjugar o desenvolvimento desta dissertação com materiais acessíveis (oriundos de projectos antigos, por exemplo), de modo a evitar a aquisição do material necessário, devido à ausência de financiamento do projecto.

[1] Remoção de tecido de um organismo vivo com o objectivo de serem realizados exames microscópicos e de diagnóstico (Karkera et al. 2011).

A solução proposta consiste na criação de um sistema que acolherá as lâminas histológicas em forma de torre, as respectivas etiquetas serão submetidas a um *software* de reconhecimento óptico de caracteres (OCR) e, após a etiqueta estar processada, a lâmina é retirada da torre. O protótipo terá, como peça central, um Raspberry Pi responsável por captar a imagem que será alvo de um *software* de OCR e accionar e desactivar os motores envolvidos nos processos de deslocação da torre de lâminas e remoção das mesmas. Desde o momento em que as lâminas são introduzidas até à conclusão do processo, espera-se mínima intervenção humana, restringindo-se ao auxílio da remoção das lâminas após estas serem processadas.

Um sistema deste tipo trará mais-valias aos intervenientes na área, dado que permitirá organizar eficientemente as informações e, por conseguinte, realizar eficazmente as tarefas relacionadas com o registo da informação, visto que esta passará a estar disponível em formato digital, razão pela qual se poderá evitar o acesso físico às lâminas. O número bastante vasto de lâminas que são processadas todos os dias torna a tarefa de registar manualmente as informações em questão pouco exequível, tornando-se, deste modo, num dos motivos que impulsionam esta tecnologia. Cumpre, por fim, salientar, que um sistema deste tipo pretende ser fundamentalmente pragmático, redireccionando o tempo e a preocupação dos trabalhadores para a parte vital desta área – a análise e diagnóstico de doenças cancerígenas e/ou oncológicas.

1.3 Organização

Esta dissertação está estruturada em seis capítulos: Introdução, Estado da Arte, Fundamentos Teóricos, Projecto e Implementação, Resultados e Discussão e, por fim, Conclusões e Trabalho Futuro.

O capítulo Introdução realiza uma contextualização do problema identificado - expondo o procedimento desde a remoção da biopsia à arquivação da mesma. Depois é apresentada a solução que esta dissertação propõe e um pequeno resumo descritivo da mesma.

O capítulo Estado da Arte é responsável por referir o que já foi feito no campo do reconhecimento óptico de caracteres, incluindo produtos no Mercado e aplicações publicadas para este tipo de produtos. Ainda são mencionados produtos relacionados com a área da Saúde e Microscopia, visto ser essa a área de inserção do tema deste projecto.

O capítulo Fundamentos Teóricos tenta fornecer uma explicação teórica por detrás do funcionamento de alguns aparelhos utilizados no projecto, esclarecer o funcionamento do *software* de reconhecimento óptico de caracteres utilizado, abordar aspectos relacionados com o Raspberry Pi e apresentar uma montagem de aumento de tensão (3.3 V $\rightarrow$ 5V).

O capítulo Projecto e Implementação foca-se somente no desenvolvimento do protótipo. Primeiramente, é mostrada uma visão geral de todo o sistema e de todas as montagens presentes no mesmo. Depois, cada elemento é salientado – explicando o porquê de ter sido escolhido e também como funciona. Ainda é mostrado o modo como o protótipo funciona e explicado o código criado para o protótipo final.

O capítulo Resultados e Discussão apresenta os testes realizados aos componentes do projecto, o resultado do *software* de OCR e menciona, por fim, erros que possam ter ocorrido.

Por último, o capítulo Conclusões e Trabalho Futuro realiza algumas considerações finais sobre a dissertação desenvolvida e refere um conjunto de inovações que, posteriormente à realização deste projecto, possam incidir no protótipo criado de forma a o melhorarem.

2

Estado da Arte

O Estado da Arte é um capítulo de grande importância, pois demonstra a pesquisa e reflexão efectuadas pelo autor sobre a temática a ser trabalhada ao longo da dissertação. Esta pesquisa proporciona ao leitor uma visão sobre o estado actual da tecnologia sob análise, conhecimento que é necessário possuir para compreender o carácter inovador da dissertação e para evitar que pesquisas fastidiosas tenham de ser levadas a cabo pelo próprio leitor.

Neste caso em particular, o presente capítulo começa por apresentar algumas limitações, produtos e aplicações referentes aos algoritmos de reconhecimento óptico de caracteres que se verificam actualmente e, por fim, expor alguns produtos relacionados com o que foi desenvolvido que já existam no Mercado.

2.1 Reconhecimento Óptico de Caracteres

De forma a que as informações de carácter identificativo presentes nas lâminas histológicas possam ser registadas, é necessário, numa primeira instância, obter uma imagem que capte toda a informação que se deseja digitalizar. Após este primeiro passo, o protótipo recorrerá a um algoritmo de reconhecimento óptico de caracteres que será utilizado para modificar o formato em que a informação pretendida se encontra tentando, sempre que possível, não perder

qualquer fracção da mesma. Para que esta informação fique registada numa base de dados ou num outro *softwar*e semelhante (parte que a presente dissertação não aborda), é necessário converter o formato de imagem para o formato ASCII, permitindo, deste modo, a sua leitura e manipulação pela máquina. Como esta dissertação apenas tem em vista conceber o sistema físico e obter a informação da imagem, esta será visualizada num documento de texto (.txt), de modo a assegurar a qualidade da informação recolhida.

2.1.1 Limitações

No entanto, a utilização deste tipo de *software* requer sempre alguma atenção visto não existir nenhum produto com uma taxa de sucesso de 100%. A falibilidade que acompanha estes sistemas depende, em larga medida, da má qualidade da imagem original (provocada, por exemplo, por desfocagem ou sujidade). Possíveis limitações de custo podem levar a que a própria tecnologia utilizada possa conter falhas que dificultarão a obtenção de um resultado desejável: o *hardware*, mais especificamente, o *scanner* pode não proporcionar o melhor ambiente para que seja feito o *scan* da imagem através de uma iluminação não uniforme ao longo da sua superfície (White & Rohrer 1983). Tanto (Arica & Yarman-Vural 2002), como (Mangoli & Desai 2016), reconhecem que o reconhecimento óptico de caracteres manuscritos, mais especificamente dos caracteres escritos utilizando uma técnica cursiva[2], apresenta um grande desafio que, presentemente, ainda se verifica. O número quase infinito de variações possíveis, as transições entre caracteres e as sobreposições complicam, de forma acentuada, a tarefa de reconhecimento e descodificação realizada por este tipo de *software*. Dito isto, é normal verificar-se uma redução da taxa de sucesso quando se transita de caracteres dactilografados para manuscritos.

[2] "s.m. (tip.) forma de letra manuscrita, ligeira e mais miúda que o bastardinho; caligrafia num pouco inclinada para diante; adj. diz-se da letra que se faz correndo sobre o papel." Dicionário da Língua Portuguesa. Porto Editora. 7ª edição (pág. 514).

2.1.2 Produtos

Neste capítulo será realizada a recolha e apresentação de alguns produtos de reconhecimento óptico de caracteres, começando pelo Tesseract, passando pelo Readiris 16 e terminando no ABBYY FineReader 14. Serão mencionadas algumas das características que compõem estes produtos, as diferenças entre as suas versões (se existirem) e as suas limitações.

A. Tesseract

Tesseract é um *software* de reconhecimento óptico de caracteres que foi desenvolvido pela HP entre os anos 1984 e 1994, mas apenas em 2005 se tornou *open-source*. Actualmente, é desenvolvido pela Google e pode ser acedido em http://code.google.com/p/tesseract-ocr.

Algumas das características-chave do Tesseract, segundo Ray Smith em "An Overview of the Tesseract OCR Engine" (Smith 2007), estão relacionadas com o modo pouco ortodoxo de como certos aspectos deste *software* foram implementados. O primeiro aspecto mencionado pelo autor é o método utilizado para encontrar as linhas de texto. Este é realizado, numa primeira fase, através de uma análise de componentes ligados[3] em que os conjuntos resultantes são filtrados pela sua altura. Após esta acção, os conjuntos que sobram são organizados tendo em conta a sua coordenada-x, organização esta que possibilita o registo da inclinação da linha de texto, não havendo a necessidade de fazer algum tipo de pré-processamento nesse sentido, salvaguardando a qualidade da imagem.

O segundo aspecto pode ser visto como consequência directa do primeiro e consiste na detecção de texto branco em fundo preto. O Tesseract foi dos primeiros a conseguir realizar esta tarefa sem grande dificuldade.

O terceiro e último aspecto, segundo o autor, pouco convencional em *software* de reconhecimento óptico de caracteres, é o classificador adaptativo que é utilizado pelo Tesseract. Este, ao contrário de classificadores estáticos, apresenta

[3] Técnica de segmentação de imagens que agrupa os pixéis tendo em conta a conectividade entre si.

uma melhor flexibilidade no que toca ao tipo de letra dos caracteres pois o algoritmo é treinado pelas palavras reconhecidas satisfatoriamente, o que leva a um progressivo aumento na qualidade de reconhecimento à medida que o processo avança. São efectuadas duas passagens pela imagem, durante a primeira pode ter sido aprendido algo proveitoso e que poderia ter sido utilizado no reconhecimento de uma palavra precedente.

Apesar destas características impulsionarem este *software*, podem ser indicadas algumas limitações, entre as quais o facto de não ser feita uma análise ao layout da página, não existir uma *interface* incorporada no sistema e o número línguas ser mais reduzido quando em comparação com outros produtos.

B. Readiris 16

Readiris 16 (Figura 2.1) é o mais recente *software* de reconhecimento óptico de caracteres da empresa IRIS[4], criada em território belga no ano de 1987. Em 2013, esta empresa juntou-se ao grupo Canon.

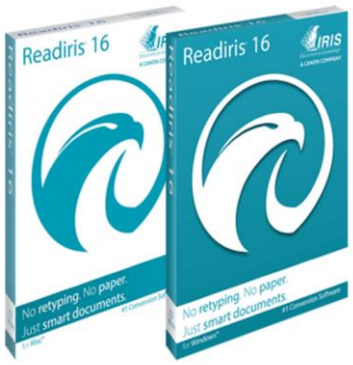


Figura 2.1 - Readiris 16. (Retirado de http://www.irislink.com/PT/c1462/Readiris-16-para-Windows---Software-OCR.aspx)

Com este produto, não há necessidade de qualquer configuração inicial, desde que esteja a ser utilizado juntamente com um *scanner* da IRIS, HP ou Canon. Outro ponto de relevo, é a versatilidade linguística que este *software* apresenta, ao reconhecer documentos em 137 línguas diferentes, incluindo todas as americanas e europeias. Em adição, o Readiris 16 ainda inclui 15 dicionários in-

[4] http://iriscorporate.com

corporados (para as línguas mais comuns). Estes entram em acção quando um caracter não é convertido de maneira satisfatória, corrigindo-o para que o utilizador não perca tempo com revisões. Para além da sua função de reconhecimento de texto, o Readiris 16 utiliza uma ferramenta de compressão para ficheiro PDF e XPS, que usufrui da tecnologia Intelligent High-Quality Compression (iHQC), criada pela I.R.I.S.

O Readiris 16 apresenta-se em duas versões: Pro e Corporate. Ambas são versões pagas, se bem que a primeira é a mais barata das duas. Tanto a versão Pro como a Corporate possuem funcionalidades em comum como o número ilimitado de páginas por documento, a leitura e codificação de códigos de barras e a criação de PDF's indexados. Porém, como seria espectável, a versão Corporate possui mais algumas funcionalidades às quais os utilizadores da Pro não têm acesso. Entre estas, destacam-se o maior número não só de formatos de saída, como também de níveis de compressão de PDF, a assinatura electrónica de ficheiros PDF e a gestão de ficheiros encriptados.

Das poucas limitações que este *software* apresenta, pode-se identificar a ausência de uma aplicação móvel e o facto de se apenas poder utilizar na máquina em que é instalado.

C. **ABBYY FineReader 14**

FineReader 14 (Figura 2.2) é um produto da ABBYY[5], uma empresa fundada em 1989 especializada em criar soluções relacionadas com a captura de conteúdo através de tecnologias relacionados com a linguagem. Estas soluções têm em vista acelerar processos em empresas, acelerar tomadas de decisão e aumentar as receitas.

[5] https://www.abbyy.com/en-eu

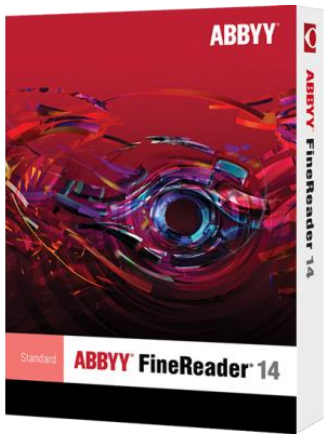


Figura 2.2 - ABBYY FineReader 14 versão Standard. (Retirado de https://macrosolution.com.br/portal/index.php/softwares/finereader-14-standard-edition.html)

De todos os produtos previamente mencionados, este é o que inclui um maior número de línguas, conseguindo reconhecer documentos em 192 línguas diferentes, estando também incluído, no *software*, o apoio de 48 dicionários. Outra característica do produto em questão, é o facto de trazer um editor de texto que permite a comparação entre a imagem original e o texto reconhecido, consentindo a realização de qualquer mudança que o utilizador ache oportuna. Se a conversão automática não for do agrado do utilizador, este pode especificar regiões da imagem ou treinar o programa para reconhecer estilos ou fontes incomuns.

Há semelhança do produto anteriormente apresentado, o ABBYY FineReader 14 está disponível em diferentes versões. Desta feita, podem ser encontradas à venda, as versões Standard, Corporate e Enterprise. A primeira é a mais simples das três, ao englobar apenas duas funcionalidades: a conversão de PDF's e *scans*, e a edição dos primeiros. A segunda, para além das funcionalidades da primeira, possui ainda a comparação entre documentos de diferentes formatos previamente falada e a conversão automática de documentos (agenda da conversão dos documentos para um determinado momento ou quando esse documento é colocado numa certa directoria). Por fim, a última versão, em relação à segunda, tem como única vantagem o processamento de duas vezes mais pacotes em modo automático, no qual a velocidade de processamento é 1.8 vezes mais rápida. Todas as versões são pagas.

2.1.3 Aplicações de OCR

Apesar das suas limitações, este tipo de algoritmos tem enorme potencial e taxas de sucesso bastante atractivas, pelo que é utilizado em numerosas áreas da sociedade em que vivemos. De seguida serão apresentadas algumas aplicações que incluem o reconhecimento óptico de caracteres no seu funcionamento.

A. Verificação automática de assinaturas

Em (Impedoso & Pirlo 2008) são coleccionados uma série de trabalhos que foram realizados neste âmbito. Neste artigo, os autores baseiam a sua exposição no processo de verificação de assinaturas que se encontra esquematizado na Figura 2.3.

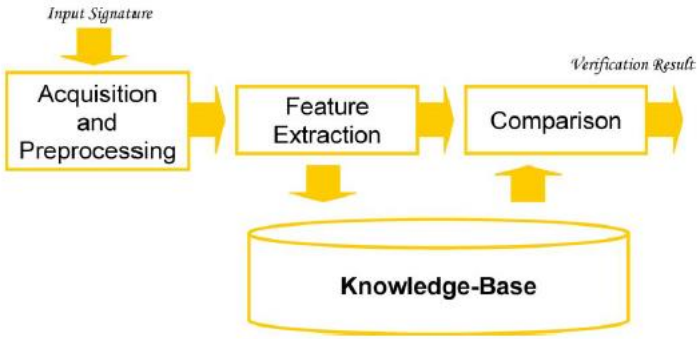


Figura 2.3 - Processos de verificação de assinaturas. (Retirado de (Impedoso & Pirlo 2008))

Analisando este esquema, verifica-se um processo constituído por três fases essenciais: a Aquisição e Pré-processamento, a Extracção de Características e a Comparação.

Em relação à fase de Aquisição e Pré-processamento, os autores consideram dois métodos distintos: o estático (*offline*) e o dinâmico (*online*). No primeiro, utilizado maioritariamente com caracteres impressos, a aquisição é realizada após o processo de escrita estar concluído e a assinatura é constituída pelos tons de cinzentos que compõem a imagem. No segundo, utilizado com o auxílio de uma *interface* (como um *tablet*, por exemplo) e uma caneta electrónica, a aquisição é realizada durante o processo de escrita e a assinatura é constituída por si-

nais electrónicos, gerados com uma certa frequência de amostragem, por dispositivos que compõem o sistema. Neste método ainda são registados outros aspectos interessantes que auxiliam a verificação da assinatura, como por exemplo, a inclinação da caneta, a pressão que é exercida na superfície de escrita e a velocidade com que é realizada assinatura. A grande diferença entre os dois processos acima mencionados consiste no primeiro estar relacionado com o espaço e a intensidade dos pixéis que constituem os caracteres e o segundo tratar a imagem como uma representação espaciotemporal da assinatura. Ainda nesta fase, é realizado o pré-processamento da imagem. O objectivo deste passo é melhorar as condições em que os caracteres serão, numa fase posterior, analisados e realizar a sua segmentação. No caso das assinaturas estáticas são utilizados algoritmos de processamento de imagem como binarização[6], emagrecimento[7], redimensionamento da imagem e remoção de ruído utilizando filtros de mediana[8]. A segmentação é então realizada recorrendo, por exemplo, a algoritmos de componentes ligados ou através de projecções horizontais e verticais do histograma da imagem. Quando se trata de assinaturas dinâmicas, os processos típicos consistem em filtragem e remoção de ruído, utilizando funções gaussianas[9] e transformadas de Fourier[10]. A segmentação, neste caso, é realizada através de sinais indicadores do instante e coordenada em que a caneta entrou e/ou deixou de estar em contacto com a *interface*, análise da velocidade de escrita e considerando a assinatura como uma série de blocos delimitados por interrupções abruptas, entre outros.

---

[6] Conversão de uma imagem em tons de cinzento para preto e branco com base num valor de *threshold* associado.

[7] Operação morfológica realizada em imagens binarizadas para transformar as regiões das imagens em linhas representativas do seu esqueleto.

[8] Filtro que substitui o valor do pixel central pelo valor da mediana dos pixéis vizinhos.

[9] Função do tipo $f(x) = \frac{1}{\sigma \times \sqrt{2 \times \pi}} \times e^{-\frac{1}{2} \times (\frac{x-\mu}{\sigma})^2}$ que representa uma curva em forma de "sino" utilizada, sobretudo, sob um contexto probabilístico.

[10] Ferramenta matemática que decompõe um sinal não-periódico nas suas várias componentes de frequência.

Seguindo para a seguinte fase, a Extracção de Características, o autor separa os atributos de uma assinatura a serem extraídos em dois grupos distintos: funções e parâmetros. No primeiro caso, são utilizadas características como a velocidade, aceleração, pressão e a direcção do movimento da caneta que se distinguem por uma variação temporal, podendo apenas ser utilizadas no caso de ter sido efectuada uma aquisição dinâmica da informação. Os parâmetros, por sua vez, dividem-se em globais e locais. Os primeiros estão relacionados com características referentes a toda a assinatura e abrangem características como o tempo total de assinatura e o número de vezes que o utilizador levantou a caneta. Já os segundos referem-se a fragmentos específicos da assinatura e incluem a altura e/ou largura dos componentes que formam a mesma e a densidade dos pixéis, por exemplo.

O derradeiro passo para, efectivamente, ser possível concluir a veracidade de uma assinatura é a Comparação. Nesta etapa, comparam-se as características extraídas na fase anterior e os dados guardados numa base de conhecimento previamente formada. Existem diversas técnicas que realizam a comparação entre ambas as partes que são escolhidas tendo em conta a melhor alternativa para cada ocasião. São estas o *template matching*, as estatísticas e estruturais. As técnicas de *template matching* incluem classificadores euclidianos e funções de deslocamento. Porém, a técnica mais utilizada é a Dynamic Time Warping[11] (DTW). Já as técnicas de estatísticas englobam as bastante utilizadas redes neuronais[12] e o modelo Hidden Markov[13] (HMM). Em relação à estrutural, são utilizados métodos de comparação recorrendo a gráficos, *strings* e árvores de decisão. Independentemente da técnica utilizada, no final desta etapa é produzida uma resposta booleana que representa a autenticidade da assinatura.

---

[11] Técnica que compara duas sequências temporais e analisa as semelhanças entre as mesmas.

[12] "Um sistema computacional formado por elementos de processamento simples e altamente interconectados, que processam a informação pela sua resposta de estado dinâmica a inputs externos." Dr. Robert Hecht-Nielsen, criador de um dos primeiros neuro computadores.

[13] Ferramenta utilizada para representar distribuições de probabilidade ao longo de várias observações (Ghahramani 2001).

B. Reconhecimento automático de matrículas

O reconhecimento automático de matrículas é vastamente utilizado em diferentes sectores da sociedade. Entre estes encontram-se o controlo das portagens, acesso a parques de estacionamento e monitorização de tráfico (Ibrahim et al., 2012). O artigo previamente mencionado expõe as diferentes etapas e as suas possíveis implementações de modo a realizar a descodificação bem-sucedida de uma matrícula. Estas etapas podem ser observadas no esquema da Figura 2.4.

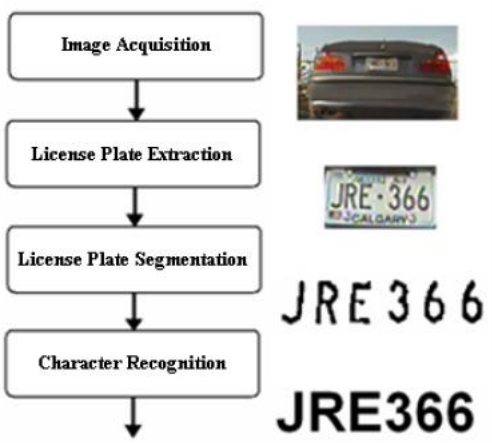


Figura 2.4 - Processos de descodificação de uma matrícula. (Retirado de (Ibrahim et al. 2012))

A Aquisição da Imagem é, meramente, a utilização de uma câmara fotográfica de modo a capturar uma imagem do veículo, na qual está incluída a matrícula do mesmo.

Na seguinte fase é realizada a Extracção da Matrícula. É necessário tirar proveito das características singulares presentes nas matrículas automóveis como o seu formato rectangular ou o contraste entre os seus caracteres (pretos) e o seu fundo (branco) para conseguir distinguir, na imagem capturada, o que é matrícula e o que não é. Ainda nesta fase, os autores enunciam alguns métodos que têm vindo a ser utilizados, nos quais é possível encontrar a Extracção da Matrícula utilizando a informação dos limites da mesma, Extracção da Matrícula utilizando as variações de cor entre os caracteres e o fundo, Extracção da Matrícula utilizando a técnica de componentes ligados para encontrar os limites da matrícula e, por fim, a Extracção da Matrícula através da cor da mesma (apenas

em alguns países, a matrícula apresenta uma cor específica que não o preto e branco).

A terceira fase consiste na Segmentação da Matrícula e é realizada com o intuito de separar os caracteres presentes na mesma. Várias técnicas podem ser utilizadas para realizar a segmentação entre as quais um algoritmo de componentes ligados, projecções verticais e horizontais, um algoritmo de contornos e utilizando informações prévias sobre os caracteres.

Na última fase é realizado, efectivamente, o Reconhecimento dos Caracteres. Um dos métodos mais simples e eficazes é o *template matching*, onde são procurados padrões nos caracteres a serem analisados. Outro método é a extracção e análise - através de redes neuronais ou do modelo Hidden Markov, por exemplo - de características de cada caracter.

C. Escritórios de Advocacia

Escritórios como os de advocacia processam, diariamente, quantidades enormes de documentos de extrema importância, o que requer um sistema que trate a burocracia de forma eficiente. Os sistemas de reconhecimento de caracteres auxiliam estas firmas no que toca à pesquisa e edição destes documentos, onde apenas uma simples palavra-chave é necessária para encontrar a informação desejada. Outra das vantagens que a utilização de um *software* deste tipo proporciona, é a desnecessidade de localizar o documento físico, o que é, com certeza, bem mais demorado do que aceder ao documento em formato electrónico (Mithe et al. 2014), (Singh et al. 2012).

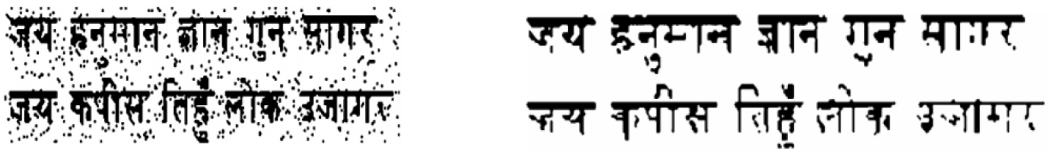
D. Sector bancário

A utilização deste tipo de *software* no sector bancário tem vindo a garantir algumas vantagens, diminuindo não só a quantidade de trabalho manual como, consequentemente, o erro humano. Uma aplicação mais específica pode ser a leitura de cheques, que são introduzidos numa máquina que realiza o *scan* a dados como o montante a ser transferido ou o número da conta, reduzindo o tempo de espera do consumidor pois uma máquina realiza estas operações em

menos tempo, quando comparada a um funcionário (Mithe et al. 2014), (Singh et al. 2012).

Em (Singh et al. 2010) é proposto um esquema para desenvolver sistemas de reconhecimento óptico de caracteres dactilografados, neste caso, do alfabeto Devanagari[14], com o intuito de ser, numa fase posterior, utilizado pela indústria bancária. O sistema proposto considera quatro etapas fundamentais: Pré-processamento, Segmentação, Extracção de Características e Classificação.

A fase de Pré-processamento começa por realizar uma binarização à imagem digitalizada, o que não deve ser problemático pois os caracteres encontram-se em tons de cinzento e, para um fundo claro, um valor de *threshold* baseado no histograma da imagem deverá ser suficiente. Para eliminar o ruído *salt-and-pepper*, ou ruído impulsivo, um filtro de mediana costuma ser bastante eficaz, como se pode confirmar pela Figura 2.5.



a) b)

Figura 2.5 - Aplicação do filtro de mediana. a) imagem com ruído impulsiva antes da utilização de um filtro de mediana. b) após a utilização de um filtro de mediana. (Retirado de (Singh et al. 2010))

Na seguinte fase, a Segmentação é realizada, numa primeira instância, às linhas que delimitam os caracteres. Para tal, são efectuadas projecções horizontais no sentido descendente para encontrar a primeira linha branca directamente superior a um pixel preto e a primeira linha branca directamente inferior a um pixel preto, determinando os limites superior e inferior, respectivamente. Demarcadas as linhas, são efectuadas projecções verticais no interior das mesmas de modo a delimitar as palavras inclusas. Encontradas as palavras, os ca-

[14] Sistema de escrita utilizado em línguas indianas como o Sânscrito ou Híndi.

racteres são finalmente segmentados removendo a *headline* (visível na Figura 2.6), regularmente presente neste alfabeto, através de, mais uma vez, projecções verticais.

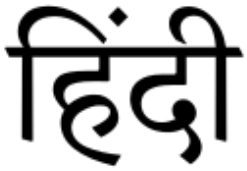


Figura 2.6 - Palavra no alfabeto Devanagari. (Retirado de https://simple.wikipedia.org/wiki/Hindi#/media/File:Hindi.svg)

Neste sistema, a Extracção de Características e a Classificação são baseadas em três atributos: a distância média do pixel central, o histograma baseado na posição do pixel e o histograma baseado no valor do pixel.

Como é possível notar no desenvolvimento que foi efectuado das aplicações acima mencionadas, a estrutura processual não varia muito de uma aplicação para outra, baseando-se, por norma, na aquisição de informação, segmentação, extracção de características e classificação.

Por fim, resta referir que existem bastantes aplicações em diferentes sectores da sociedade para este tipo de *software* para além das mencionadas. Alguns destes sectores incluem a educação, convertendo autênticas bibliotecas para formato electrónico possibilitando a partilha do conhecimento ou a saúde, ajudando pessoas com problemas visuais.

2.2 *Produtos Relacionados*

Na área da Biomédica, estão constantemente a ser concebidos aparelhos que simplifiquem os procedimentos pelos quais os profissionais, neste caso, os técnicos de laboratório, têm que executar. Para além disso, numa área bastante sensível e preocupante como é a das doenças cancerígenas e/ou oncológicas, é habitual o desenvolvimento acentuado de equipamento que acelere estes mesmos procedimentos, seja através de *software* mais intuitivo, seja através de *hardware* mais manuseável.

Actualmente, o Mercado está repleto de equipamentos úteis para quem exerce nesta área, mas como esta dissertação se foca no desenvolvimento de um sistema para visualização e registo de informações identificativas das lâminas histológicas que contenham amostras de tecido, apenas serão mencionados produtos que se assemelhem, em termos de funcionamento e de propósito, ao mesmo. O primeiro tipo de produtos que merece ser mencionado é o *scanner* de lâminas histológicas. O catálogo de produtos do website MedicalExpo[15] apresenta uma série de *scanners* de lâminas histológicas de marcas como a Motic (Figura 2.7 a) ou Leica Biosystems (Figura 2.7 b), todos eles concebidos com o propósito de digitalizar o conteúdo da lâmina (tecido). O normal funcionamento deste tipo de equipamento consiste, muito basicamente, em inserir o conjunto de lâminas permitido pelo dispositivo no compartimento concebido para esse efeito (o número varia de máquina para máquina), o *scanner* embutido na máquina desloca-se de lâmina para lâmina, realizando a digitalização do conteúdo biológico das mesmas. Digitalizações estas que serão, posteriormente, visualizadas através de um *software* próprio.

a) b)

Figura 2.7 - *Scanners*. a) *Scanner* de lâminas histológicas da Motic. (Retirado de http://www.medicalexpo.com/prod/motic-europe/product-69458-717972.html). b) *Scanner* de lâminas histológicas da Leica Biosystems. (Retirado de http://www.medicalexpo.com/prod/leica-biosystems/product-95735-653510.html)

[15] http://www.medicalexpo.com/medical-manufacturer/microscope-slide-scanner-31319.html

O segundo e último tipo de produtos a ser referido nesta revisão é o distribuidor de lâminas de microscópio (Figura 2.8). Este é constituído por uma zona de armazenamento de lâminas (normalmente o número possível de lâminas armazenada varia entre os 50 e 100 lâminas) e uma maçaneta que provoca a saída da lâmina. O seu funcionamento é baseado na acção manual de rodar a maçaneta de modo a que uma lâmina seja ejectada de cada vez.

Figura 2.8 - Exemplo de um distribuidor de lâminas de microscópio. (Retirado de http://www.lifeways.com.my/product/lifeways-microscope-slide-dispenser)

3

Fundamentos Teóricos

O protótipo desenvolvido no âmbito desta dissertação engloba certas tecnologias como motores e controladores. Estas e outras tecnologias essenciais para o correcto funcionamento do protótipo em construção serão aqui apresentadas, procurando fornecer uma explicação sobre os mecanismos que as fazem funcionar, conceitos básicos e aplicações que possam ter.

3.1 Actuador Linear

Um actuador linear, posto de maneira muito simples, é um equipamento que produz movimento em linha recta. Este movimento é criado a partir da conversão de energia (pressão de água, electricidade, força humana, entre outros) que servirá, normalmente, para mover um pistão para trás e para a frente.

As aplicações para este tipo de equipamento são imensas, sendo estes utilizados tanto na indústria automatizada como em periféricos de um computador, tanto em manufactura como em agricultura. São normalmente utilizados com motores, válvulas e interruptores em aplicações que requerem movimento linear.

Neste capítulo serão ainda mencionados alguns tipos de actuadores lineares, exemplificando cada tipo com imagens, atribuindo mais foco ao tipo electromecânico, no qual se vai aprofundar sobre o seu modo de funcionamento, visto ser o tipo de actuador linear utilizado neste projecto.

3.1.1 Tipos de Actuador Linear

Existem vários tipos de actuador linear, cada um com as suas características e funções. O princípio de funcionamento é diferente entre os diversos tipos, porém, mantem-se um objectivo comum: provocar movimento linear.

A divisão em categorias que normalmente se efectua quando se aborda o tema de actuadores lineares é a seguinte: Mecânico, Hidráulico, Pneumático, Piezoeléctrico e Electromecânico.

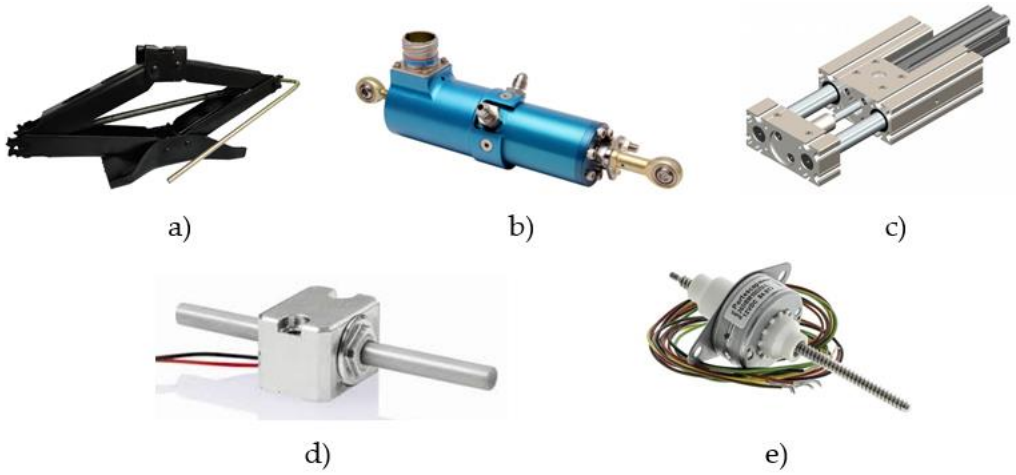


Figura 3.1 – Tipos de actuador linear. a) Mecânico. b) Hidráulico. c) Pneumático. d) Piezoeléctrico. e) Electromecânico.

A destacar, de entre os vários tipos de actuador linear apresentados, o electromecânico.

3.1.2 Actuador Linear Electromecânico

Esta categoria de actuador linear é controlada por um motor eléctrico (DC ou de passo) que possui na sua constituição um segmento com as roscas complementares às um parafuso de avanço (Figura 3.2).

Figura 3.2 - Parafusos de avanço

Com este parafuso imóvel, o movimento circular provocado pelo motor eléctrico faz com que o segmento roscado force o movimento do parafuso de avanço. A direcção com que este parafuso se movimenta depende da direcção de rotação do motor intrínseco ao actuador linear.

Algumas vantagens deste tipo de equipamento residem no facto de ser barato, constituído por um sistema bastante simples e trabalhar com tensões baixas. Por outro lado, algumas das desvantagens relacionam-se com o desgaste das peças e com o compromisso binário-velocidade.

3.2 Motor Servo de Corrente Contínua

O motor servo de corrente contínua (Figura 3.3) é um motor de alta precisão, torque elevado, baixo ruído e de fácil manutenção.

Figura 3.3 - Exemplo de um motor servo

Possui, na sua constituição, um motor DC, um potenciómetro e um circuito e controlo, comunicando a partir de três ligações: alimentação, *ground* e o controlo. O circuito de controlo está embutido na caixa que engloba o motor e

está associado a um eixo. Quando o motor recebe um sinal de controlo – que definirá a sua posição – alimenta o motor DC de modo a girar o seu eixo. À medida que este gira, a resistência do potenciómetro varia possibilitando ao circuito de controlo perceber quanto falta para parar. O sinal de controlo e o modo de utilização de um motor de tipo será explicado em mais detalhe na secção Controlo deste capítulo.

São vários os sectores da sociedade que utilizam motores servo, AC ou CC, devido às suas características. Dentro destes podem ser identificadas as aplicações industriais, robótica, farmacêuticas e linhas de montagem, onde é necessário um trabalho repetitivo, mas preciso. São ainda utilizados em aviões controlados por radiofrequências, no movimento de robôs e para controlar a posição de superfícies como elevadores.

3.2.1 Controlo

Como foi referido, a posição do eixo do motor servo é atribuída através de um sinal de controlo enviado pelo utilizador. Este sinal consiste numa onda PWM (Pulse Width Modulation) cuja duração do pulso define a posição desejada. De modo a ilustrar o funcionamento de um motor deste tipo é apresentada a Figura 3.4.

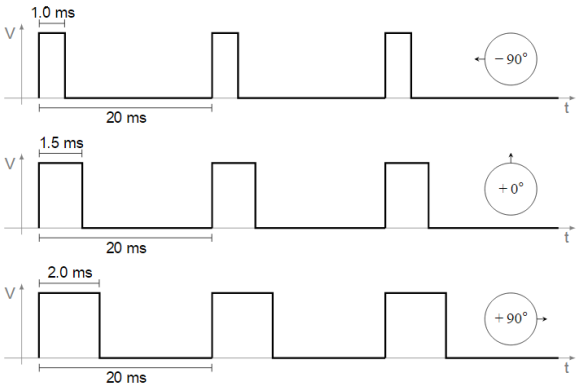


Figura 3.4 - Funcionamento de um motor servo

Esta figura exibe três sinais distintos, onde cada um corresponde a uma posição do eixo diferente. A posição central – aquela que apresenta uma igual possível rotação em ambos os sentidos – corresponde ao pulso do meio (1.5 ms). Se for enviado um pulso com menor duração que 1.5 ms, o motor rodará no

sentido contrário ao dos ponteiros de relógio e, contrariamente, se for enviado um pulso com maior duração, o motor rodará no sentido dos ponteiros de relógio. Para o caso ilustrado na Figura 3.4, o motor exemplificado tem um ângulo máximo de rotação de 180º (90º para cada lado). A posição do motor é actualizada de vinte em vinte milissegundos. Quando um motor servo é ordenado para uma certa posição, ele mantem-na enquanto estiver a receber o sinal para essa mesma posição. Se uma força externa tentar mover o eixo do motor, este resistirá.

3.3 Reconhecimento Óptico de Caracteres (OCR)

O Reconhecimento Óptico de Caracteres, ou OCR, é uma tecnologia que permite realizar a conversão de informação escrita – esteja esta contida em imagens, documentos físicos ou até mesmo electrónicos – para um formato digital manipulável. Diversas áreas da sociedade tiram proveito desta tecnologia, numas é um bem essencial como, por exemplo, no reconhecimento automático de matrículas realizado em certos sistemas de vigilância ou nas portagens, noutras serve como ferramenta para facilitar/acelerar alguns processos como, por exemplo, reconhecimento das informações em cheques bancários ou, simplesmente, para facilitar o acesso aos dados escritos em papel num escritório de trabalho.

3.3.1 Etapas

Este método de converter o estado em que os caracteres escritos se encontram é constituído por uma série de fases sequenciais, cada uma desempenhando o papel de receber a informação proveniente da fase anterior e realizar alguma acção sobre a mesma (própria de cada fase), preparando-a para a próxima etapa. Se uma fase não desempenhar a sua função razoavelmente, poderá colocar em risco a precisão de todo o sistema.

3.3.1.1 Aquisição de Dados

A Aquisição de Dados é a primeira etapa no reconhecimento óptico de caracteres e consiste na captura da imagem ou documento de trabalho – seja atra-

vés da realização de um *scan* ou de uma fotografia – para que este consiga ser analisado pelo *software*. Esta fase, se for mal-executada, pode danificar a imagem e dificultar a tarefa das fases que se seguem, sendo esta uma das principais razões pelas quais estes sistemas podem falhar.

3.3.1.2 Pré-Processamento

A fase de Pré-Processamento tem o objectivo de utilizar a imagem adquirida na fase anterior e optimizá-la – através de técnicas de processamento de imagem – para que o reconhecimento óptico de caracteres propriamente dito seja realizado com a menor das dificuldades.

As técnicas de processamento de imagem utilizadas podem variar dependendo de como se quer deixar a imagem para a próxima fase, mas, de um modo geral, utilizam-se técnicas como:

- Binarização;
- Remoção de ruído: Filtro de mediana, para ruído *salt-and-pepper* ou filtro de média, para ruído gaussiano;
- Operações morfológicas: Emagrecimento, esqueletização, abertura e fecho;
- Correcção da orientação da imagem no caso de esta estar inclinada.

Alguns tipos de *software*, como por exemplo o Tesseract, necessitam que esta etapa seja realizada previamente à utilização do *software* propriamente dito, em outros (normalmente produtos com maiores custos associados), verifica-se a integração do pré-processamento nas funções desempenhadas pelo *software*.

Outro factor que eleva a importância desta fase é o aumento da utilização de máquinas fotográficas digitais para a captura de imagem (como é o caso do presente projecto) pois estas são mais baratas, portáteis e versáteis que a grande maioria dos *scanners*. A grande desvantagem é a qualidade da imagem resultante em comparação com estes últimos. É aí que o pré-processamento faz a diferença, permitindo a correcção de certas falhas provocadas pela utilização de máquinas digitais (Bieniecki et al. 2007).

3.3.1.3 Segmentação

A etapa de Segmentação vem em seguimento da fase de Pré-Processamento e trabalha sobre a imagem resultante da última etapa de modo a separar as sequências de caracteres (frases e palavras) presentes nessa imagem até apenas sobrar um conjunto de imagens, cada uma representada apenas por um único caracter. Em regra geral, a separação de frases em palavras é realizada com bastante facilidade visto haver um espaço significativo a separar cada uma.

Duas técnicas bastante utilizadas nesta fase são as projecções (horizontais, verticais e diagonais) e os componentes ligados. A primeira apresenta as vantagens de ter uma implementação mais simples e de requerer menos processamento, a segunda tem a vantagem de conseguir reconhecer objectos que se encontrem inclusos de outros.

3.3.1.4 Extracção de Características

Esta fase é considerada em (Trier et al. 1996) como a mais importante quando o objectivo é efectuar um reconhecimento de alta qualidade. O objectivo desta fase é encontrar certos aspectos identificativos dos caracteres isolados durante a Segmentação.

Considerando que a imagem a analisar é binária, alguns dos métodos de extracção de características utilizados são:

- As projecções verticais e horizontais dos histogramas;
- Percentagem de pixéis pretos por zona;
- Simetrias horizontais e verticais;
- Número de buracos.

A frisar que o modo como os caracteres serão classificados terá de variar consoante o(s) método(s) utilizado(s) na presente etapa.

3.3.1.5 Classificação

Esta fase é, em alguns casos, a última a ser executada no processo de reconhecimento óptico de caracteres. As características encontradas na etapa anterior são comparadas com um conjunto de características que definem cada clas-

se, podendo, no entanto, ser utilizados diversos métodos para o concretizar. Em (Verma 2012) são mencionados alguns desses métodos, entre os quais se encontram:

- Métodos Estatísticos: k-Nearest Neighbour, Redes Bayesianas, Distância de Mahalanobis;
- Métodos Sintácticos e Estruturais;
- *Template Matching*;
- Redes Neuronais Artificiais.

Como já foi referido, em muitos casos, o reconhecimento óptico de caracteres termina nesta instância, com a classificação de cada elemento. No entanto, há ainda algo mais a fazer, possível de tornar este processo mais fidedigno, que será discutido na próxima e derradeira etapa.

3.3.1.6 Pós-Processamento

A etapa de Pós-Processamento, quando é empregue, é, seguramente, a última de todas as etapas referentes ao reconhecimento óptico de caracteres a ser efectuada. Após a classificação dos caracteres podem existir alguns erros, especialmente entre caracteres que possuam uma morfologia semelhante, tais como o "O" e o "0", o "S" e o "5" e, por fim, o "B" e o "8" (Bassil & Alwani 2012). De forma a combater a permeabilidade ao erro que estes e outros casos apresentam, aplica-se a presente etapa que consiste numa tentativa de corrigir certos erros ortográficos (referentes à língua de conversão) que o resultado final da classificação possa conter.

A correcção é normalmente conseguida com o auxilio de um dicionário que permite encontrar palavras com erros ortográficos. Esta solução obriga à integração de um dicionário com numerosos termos e expressões, o que pode ser considerado como uma desvantagem. Outras soluções podem ser encontradas em (Bassil & Alwani 2012) onde é utilizado um algoritmo que acede às sugestões *online* de soletração da Google e em (Perez-Cortes et al. 2000) que utiliza um algoritmo estocástico[16] de correcção de erros.

[16] Que envolve probabilidades.

3.3.2 Tesseract

O *software* escolhido para a realização do reconhecimento óptico de caracteres foi o Tesseract. Este já foi mencionado no Estado da Arte mas apenas foram referidos alguns aspectos-chave que compõem este *software*. Nesta secção, o Tesseract será apresentado em detalhe para se perceber o que está por detrás do reconhecimento dos caracteres realizado nas etiquetas das lâminas histológicas. Apresentação essa que terá como base o artigo "An Overview of the Tesseract OCR Engine" (Smith 2007).

Antes de começar, o algoritmo assume que a imagem recebida está binarizada e com todo o pré-processamento necessário efectuado. O conjunto de processos que constituem o Tesseract são executados um de cada vez, de forma consecutiva.

A. Determinação das zonas de texto

Com uma página de texto binarizada como *input*, o Tesseract utiliza um algoritmo de componentes ligados de modo a definir as zonas da imagem recebida que contêm texto. Essas zonas seguirão para a próxima fase, onde cada uma delas será analisada individualmente. Esta solução, apesar de pesada em termos de processamento na altura, revelou possuir certas vantagens. O facto de ser possível encontrar linhas inclusas em caracteres tornou possível o reconhecimento de texto invertido, isto é, texto branco em fundo preto, tornando o Tesseract num pioneiro em reconhecer texto invertido.

B. Determinação de linhas

Cada uma das zonas determinadas na fase anterior será alvo de um algoritmo de determinação de linhas (o Tesseract assume que estas zonas são constituídas por texto uniforme). A primeira acção a ser realizada é uma filtragem por alturas. A altura mediana do texto em cada linha servirá para ignorar ruído, acentos e pontuação. Analisando cada zona pela coordenada-x, é possível atribuir um identificador a cada linha enquanto se verifica a inclinação que essa li-

nha possui. Procede-se, então, para a determinação da *baseline* de cada linha através de uma regressão linear utilizando a *least median of squares*[17].

C. Ajustamento das *baselines*

Nesta fase utiliza-se a técnica de interpolação utilizando *splines* quadráticas[18] para ajustar a *baseline*. Dividindo cada linha em grupos com uma certa distância contínua da *baseline* original, realiza-se a interpolação *spline* no grupo mais populoso, que o Tesseract assume ser a nova *baseline*. Na Figura 3.5 pode ser observado um exemplo de texto com a respectiva *baseline* curva.

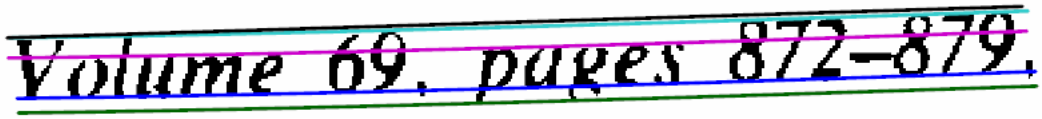


Figura 3.5 - Exemplo de uma *baseline* curva (azul)

D. Determinação do tipo de espaçamento

Um teste que o Tesseract realiza a cada linha consiste em determinar se esta é composta por espaçamentos fixos ou proporcionais (Figura 3.6).

Figura 3.6 - Tipos de Espaçamento. Espaçamento proporcional em cima e espaçamento fixo em baixo

[17] Método utilizado na aproximação de rectas com base num conjunto de pontos utilizando a sua mediana. Uma das mais valias deste método é a resistência a quase 50% de informação contaminada.

[18] Método de interpolação que utiliza diversas funções quadráticas que representam diferente intervalos no domínio em questão. Dentro de cada intervalo, as funções são contínuas, bem como a sua derivada de primeiro grau.

No caso do espaçamento fixo, o Tesseract delimita os caracteres das palavras utilizando o espaçamento determinado.

No caso de o espaçamento ser proporcional, o problema adensa-se bastante. Existem alguns problemas associados a este tipo de espaçamento como os que podem ser observados na Figura 3.7 - Exemplo de texto com alguns obstáculos de reconhecimento. O espaçamento entre os 1's e o espaçamento entre outros caracteres não é sequer parecido. Os f's de "of" e "financial" sobrepõem-se quando analisados pelas coordenada-x. O Tesseract mede a distância entre a *baseline* e a *mean line* (linha de cor rosa na Figura 3.5) e se o valor estiver próximo de um certo *threshold*, o grupo de caracteres é marcado para análise após a fase de reconhecimento das palavras.

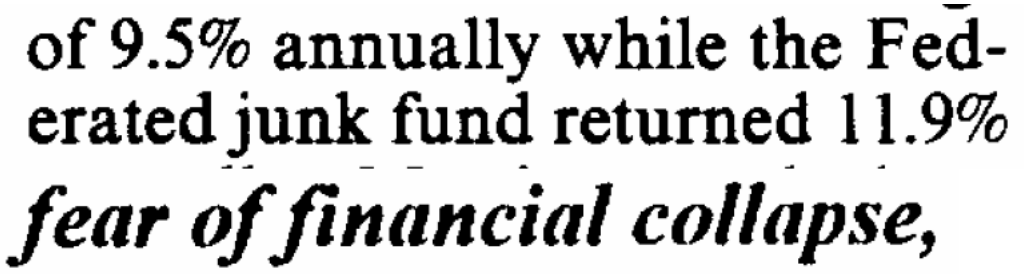


Figura 3.7 - Exemplo de texto com alguns obstáculos de reconhecimento

E. Separação de caracteres unidos

De modo a aumentar a confiança nos resultados obtidos, o Tesseract actua sobre palavras com pouca percentagem de sucesso e tenta separar os seus caracteres. O algoritmo procura por vértices côncavos que possuam vértices opostos ou segmentos de linha. A Figura 3.8 apresenta vértices candidatos e um corte onde o "r" entra em contacto com o "m".

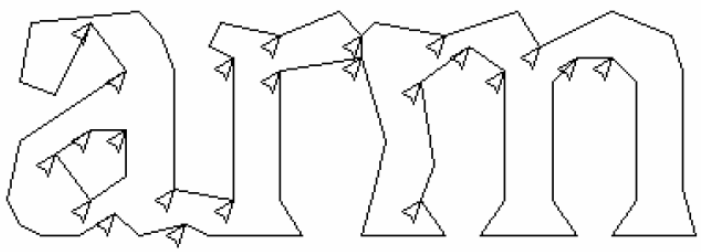


Figura 3.8 - Vértices candidatos a separar os caracteres

Se após o corte a percentagem de sucesso não aumentar, o processo é revertido, voltando tudo ao estado original. Os potenciais recortes são ainda guardados para possível utilização futura.

F. União de caracteres partidos

Após a anterior etapa, se com todos os potenciais cortes esgotados a palavra ainda não apresenta uma percentagem de sucesso aceitável, esta é submetida a um processo de união de caracteres. Este consiste em verificar as possíveis combinações dos vários fragmentos existentes na palavra e avaliá-los através da classificação de conjuntos de fragmentos por classificar.

G. Características

De modo a classificar cada caracter, o Tesseract necessita analisar as suas características. O conjunto de treino utilizado é formado por segmentos de aproximações poligonais do caracter, representado pelas linhas mais finas da Figura 3.9. As linhas mais grossas representam o caracter a ser analisado. Como se pode verificar apenas alguns troços não estão devidamente identificados.

As características são compostas por três dimensões, x, y e o ângulo, podendo existir 50 a 100 num caracter.

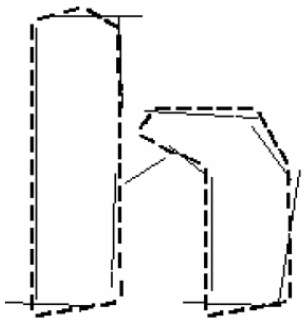


Figura 3.9 - Análise das características de um caracter. As linhas grossas são as analisas enquanto que as linhas finas são as do conjunto de treino

H. Classificação

Esta etapa divide-se em duas fases. Na primeira, é criada uma lista de classes candidatas. Cada característica procura por vectores de *bits*, pertencentes a essas classes, numa tabela. Esses vectores são somados para todas as características, permanecendo para a segunda fase os que apresentarem os maiores resultados.

A segunda fase consiste em testar a semelhança entre as características e os vectores de *bits* que completaram, com sucesso, a primeira fase. É através do cálculo da distância que se comprova qual das classes resultantes é a escolhida.

3.3.2.1 PyTesseract

O *software* Tesseract foi implementado tendo como base a linguagem C/C++. De forma a adaptar este *software* para a linguagem de programação Python, foi criado o PyTesseract, uma ferramenta que possibilita a utilização do Tesseract nesta linguagem. A sua versão actual é a 0.1.7.

3.4 Raspberry Pi

O Raspberry Pi é famosamente conhecido por ser um computador perfeitamente operacional, do tamanho de um cartão de crédito e extremamente barato. Este "minicomputador", desenvolvido pela Raspberry Pi Foundation, tornou-se um fenómeno gigantesco desde o instante em que foi lançado, em Fevereiro de 2012, capturando o interesse de muitos entusiastas de programação e não só. Conforme referiu um dos fundadores da Raspberry Pi Foundation, Eben Upton, "There were 100,000 people on our mailing list wanting a Raspberry Pi and they all put an order in on day one!".

Este computador foi concebido com o intuito de desempenhar um papel de instrumento de aprendizagem, porém, acabou por se tornar num dispositivo com vários propósitos. Pode ser utilizado como um *desktop* normal vulgarmente encontrado em qualquer casa, isto é, ouvir música, ver filmes, jogar, escrever um documento, contendo essa vertente relativa ao uso generalizado de um computador. No entanto, o que o distingue (e a razão pela qual foi utilizado

neste projecto) é o facto de estar bastante direccionado para o controlo e programação de *hardware*. Com o Python como linguagem de programação de referência do Raspberry Pi, o General Purpose Input/Output (GPIO) e a portabilidade que a placa apresenta, o controlo de motores, sensores ou LED's tornou-se trivial, permanecendo apenas a falta de vontade de aprender como único obstáculo.

3.4.1 Origem do Raspberry Pi

A ideia de criar um computador que fosse pequeno e acessível teve início no ano de 2006 na Universidade de Cambridge. Eben Upton, o Director de Estudos em Ciências da Computação na altura, reparou num declínio das capacidades dos alunos que se inscreviam neste curso, em comparação com os dos anos 90. O objectivo inicial do desenvolvimento do Raspberry Pi seria apenas uma tentativa de oferecer um avanço aos alunos, dando-lhes a possibilidade de programar no seu tempo livre e de realizarem os seus próprios projectos. Quando se deu inicio aos trabalhos, percebeu-se que o projecto seria muito mais do que inicialmente previsto. Foi então que, num encontro entre Upton e alguns dos seus colegas, foi criada a Raspberry Pi Foundation.

Passados cerca de cinco anos, foi concluído aquele que seria o primeiro protótipo do Raspberry Pi. Depois de algumas tentativas de financiamento que não correram como esperado, foi de um vídeo feito pela câmara de um telemóvel que tudo mudou. Este foi realizado com o intuito de ser submetido no *blog* de Rory Cellan-Jones, um jornalista de tecnologia, tornando-se viral. Com apenas um protótipo não tão barato quanto isso e um conjunto significativo de entusiastas de computação curiosos, a Raspberry Pi Foundation estava perante o desafio de diminuir o custo de produção da placa, enquanto mantinha as suas características. O lançamento do produto estava marcado para Fevereiro de 2012, destinado a programadores, só no final desse ano iria ser lançada a versão educacional devido a problemas de optimização de *software*. Com uma centena de milhar de pedidos no primeiro dia, a Raspberry Pi Foundation não tinha recursos nem capital para realizar tal tarefa, pelo que recorreu a dois fornecedores de electrónica de renome, a element14 e a RS Components. Devido ao tamanho

do pedido, ambos os *websites* destas empresas ficaram *offline* durante grande parte do dia. Três meses após o lançamento do Raspberry Pi, o número de pedidos ultrapassou o meio milhão de unidades.

3.4.2 A Placa

A versão de Raspberry Pi utilizada neste projecto é o Raspberry Pi 2 Modelo B V1.1 (Figura 3.10), uma placa de segunda geração da Raspberry Pi Foundation.

Figura 3.10 - Raspberry Pi 2 Modelo B V 1.1

Sendo, na sua definição, um computador, não pode existir sem alguns elementos-chave que lhe permitam desempenhar o seu papel sem problemas. Duas das maiores melhorias que foram efectuadas desde o seu predecessor – o Raspberry Pi 1 modelo B+ - foram o processador de 900 MHz quad-core ARM Cortez-A7 da Broadcom e o 1GB de memória RAM, face ao processador de 700 MHz ARMv6 de um único núcleo e aos 512 MB de memória RAM. A maioria dos elementos, no entanto, permaneceu a mesma, começando desde já pela morfologia da placa, na qual apenas algumas inscrições na placa mudaram, as quatro portas USB, o GPIO de 40 p*ins* (Figura 3.11), uma entrada HDMI e Ethernet, um áudio *jack* de 3.5 mm, uma *interface* para a câmara e para um *display*, uma entrada para a alimentação (5V e 2A como valores recomendados) e, por fim, uma ranhura para um cartão SD.

Figura 3.11 - General Purpose Input/Output (GPIO)

3.4.3 Sistemas Operativos

Já foi evidenciado que o Raspberry Pi é uma máquina bastante versátil. O número de diferentes sistemas operativos que este "minicomputador" pode correr vem evidenciar esse facto. Serão apresentados sistemas operativos com diferentes facetas, capazes de preencher uma vasta gama de necessidades que um utilizador de Raspberry Pi poderá requerer.

Dos muitos sistemas operativos existentes para o Raspberry Pi podem-se identificar os seguintes:

- Raspbian: O sistema operativo oficial da Raspberry Pi Foundation e o que será utilizado neste projecto. É baseado no Debian, um sistema operativo *open-source* que, por sua vez, é baseado no Linux. O Raspbian está altamente optimizado para funcionar com o Raspberry Pi, mais especificamente, com os seu CPU de baixa *performance*.
- Open Source Media Center (OSMC): Se o utilizador estiver interessado num OS mais relacionado com multimédia, este é o que se mais adequa às suas necessidades. É *open-source*, baseado no Linux e bastante direccionado para a reprodução de ficheiros multimédia em diversos formatos.
- Ubuntu MATE: Da família do conhecido Ubuntu, distinguindo-se pelo seu ambiente de trabalho. Este sistema operativo integra uma *graphical user interface* (GUI) similar à do Microsoft Windows ou à da Apple Mac e destina-se a utilizadores com necessidades generalizadas.
- Windows 10 IoT Core: Tal como o nome indica, este é um sistema operativo que foi concebido tendo em vista a Internet of Things

(IoT). É utilizado maioritariamente nos chamados dispositivos inteligentes, que reúnem informação – proveniente de sensores – e estão em constante comunicação com a *cloud*.

Existem muitos outros sistemas operativos que o Raspberry Pi consegue correr, entre os quais, o Risc OS, RetroPie, OpenElec, Arch Linux e o Pidora.

3.4.4 Acesso Remoto

Como já foi referido, o Raspberry Pi encontra-se na categoria dos computadores, apesar do seu tamanho. Como tal, para ser utilizado necessita de certos periféricos como teclados, monitores, ratos, entre outros. A não ser que o Raspberry Pi fique fixo num certo lugar, carregar todos estes elementos sempre que se quer utilizar a placa pode ser algo maçador.

Existem duas soluções para este problema, ambas inseridas no âmbito do acesso remoto: activação da Secure Shell (SSH) e a utilização do VNC (Virtual Network Connection). A primeira consiste numa característica do Linux, neste caso do Raspbian, que permite o acesso à linha de comandos do Raspberry Pi a partir de outro computador na mesma rede. O modo como se acede ao Raspberry Pi varia de acordo com o sistema operativo presente no computador *host*. Com Linux ou Mac, o utilizador apenas tem que inserir na linha de comandos "ssh (IP do Raspberry Pi) -l (nome de utilizador)", no caso do Windows, programas como o PuTTY realizam a tarefa com facilidade. O segundo, o VNC, permite o acesso à interface gráfica, caso exista, do Raspberry Pi, ao contrário da activação da SSH. O VNC necessita ser instalado e activado no Raspberry Pi para que o *host* consiga aceder, incluindo definir uma palavra-chave da ligação. Do lado do *host*, o programa VNCViewer, disponível em bastantes plataformas, permite ao utilizador aceder ao Raspberry Pi, após este inserir o IP e a palavra-chave da ligação estabelecida no Raspberry Pi.

De notar que em qualquer um dos casos é necessária uma primeira activação do Raspberry Pi sem acesso remoto para se poder verificar o seu IP.

3.5 Conversor de Nível Lógico

O Mundo em que vivemos é analógico, os sons que ouvimos são analógicos, os sensores distribuídos pelos nossos organismos são analógicos, basicamente, a grande maioria da informação recebida por um ser humano é analógica. No entanto, o Mundo digital apenas conhece dois estados: 1 e 0, HIGH e LOW, ON e OFF.

Os dispositivos electrónicos concebidos para lidar com sinais eléctricos, como foi referido, trabalham com dois estados, mas de uma perspectiva de comportamento, como é que um dispositivo reconhece um sinal como sendo 1 ou 0? A resposta a esta pergunta é o valor de tensão que esse sinal possui. Normalmente, cada dispositivo apresenta no seu *datasheet* o valor de fronteira (ou *threshold*) que separa estes dois estados, bem como os valores (ou intervalos de valores) para os estados HIGH e LOW. A incompatibilidade em relação a estes valores que certos dispositivos apresentam entre si, quando é necessária uma comunicação, cria um problema que é necessário resolver.

Nesta indústria são vulgarmente utilizados valores de referência de 5V, 3.3V ou 2.7V para representar o valor binário HIGH e 0V para o valor LOW. Quando o dispositivo de *output* é de 5V e o de *input* é de 3.3V, por exemplo, o problema resolve-se recorrendo a um divisor de tensão (Figura 3.12).

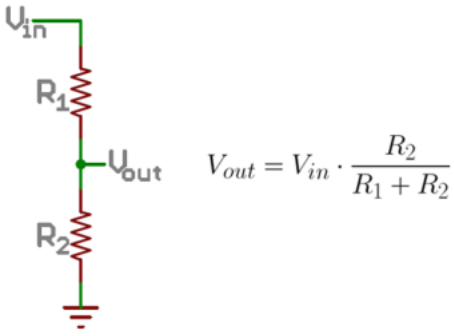


Figura 3.12 - Divisor de tensão

Para este caso específico, fazendo R_2=2*R_1 (10kΩ para R_1 e 20kΩ para R_2, por exemplo) e V_{in} com o valor a converter igual a 5V, V_{out} teria o valor desejado de 3.3V. Para o caso oposto, em que o dispositivo de *output* é de 3.3V e o de *input* é de 5V, a resposta não é muito complicada, mas também não é tão simples

como um divisor de tensão. Uma possível solução é um conversor de nível lógico. Uma possível implementação, e a utilizada neste projecto, é apresentada na Figura 3.13.

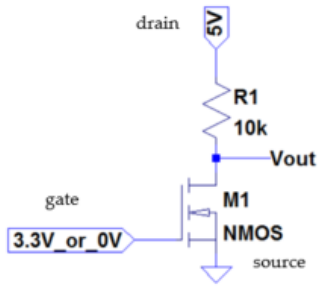


Figura 3.13 - Circuito de um conversor de nível lógico

Esta implementação recorre à tecnologia MOSFET (Metal Oxide Semiconductor Field Effect Transistor). Este é o tipo de transístor mais vulgarmente utilizado em circuitos digitais, podendo estar presentes cerca de centenas de milhar ou mesmo milhões num microprocessador ou memória. Este tipo de transístor apresenta, de um modo simplificado (excluído o *body* por se encontrar normalmente curto-circuitado com a *source*), três terminais: o *drain*, a *source* e a *gate*.

A tensão presente na *gate* (equivalente ao V_{GS} pois a *source* está ligada à massa), oriunda do dispositivo de *output*, controla a corrente que atravessa o transístor. Variações nesta tensão provocam variações na corrente do circuito devido à transcondutância presente nos MOSFETS. Antes de se proceder à análise dos dois estados pretendidos com esta montagem, falta mencionar uma característica muito importante e que é vital para se conseguir efectuar a conversão de tensão de um modo preciso: a tensão de *threshold*, ou V_{th}. Esta tensão depende da tecnologia do transístor e corresponde ao valor de V_{GS} a partir do qual o transístor começa a conduzir corrente. Para o caso de se querer implementar um conversor de nível lógico, o valor desta grandeza necessita sem inferior ao valor máximo de tensão disponibilizado pelo dispositivo de *output*, neste caso, 3.3V.

Estudando a montagem apresentada na Figura 3.13, dois estados são possíveis: $V_{GS} = 0V$ ou $V_{GS} = 3.3V$. Para o primeiro caso, $V_{GS} < V_{th}$, o transístor não

conduz, R1 não é atravessada por corrente, V_{out} = 5V. Para o segundo caso, partindo do princípio que V_{th} < 3.3V e V_{GS} > V_{th}, o transístor já conduz corrente, sendo esta limitada pela resistência nos 500 μA. A queda de tensão provocada pela passagem de corrente na resistência resulta em, aproximadamente, 0V no nó V_{out}. Como foi possível reparar na explicação efectuada, um detalhe a ter em conta é o facto de a tensão de saída desta montagem estar invertida, isto é, para V_{GS} = 0V => V_{out} = 5V e para V_{GS} = 3.3V => V_{out} = 0V. Este pormenor é facilmente resolvido utilizado uma porta lógica NOT.

4

Projecto e Implementação

No capítulo de Projecto e Implementação vão ser apresentados todos os aspectos relacionados com o protótipo em si, começando pela arquitectura de todo o sistema e dos circuitos de mais baixo nível, passando por um olhar mais focado em cada elemento do protótipo, referindo as suas características e funcionalidades e o modo como o utilizador deve proceder para utilizar o protótipo correctamente, terminando numa apresentação e explicação do código escrito com a finalidade de controlar o protótipo.

4.1 Arquitectura do Sistema

Dada a inexistência de sistemas como o que foi desenvolvido, não foi possível basear esta arquitectura em modelos previamente estabelecidos neste âmbito. No entanto, um projecto existente no laboratório mostrou-se ser bastante útil visto ter algumas características semelhantes a este, fornecendo até alguns materiais para a sua construção. De modo a tentar não modificar estes materiais, adaptou-se a arquitectura do protótipo de modo a coexistir com a inalteração desses mesmos materiais.

A estrutura do protótipo foi traçada de forma a proporcionar uma fácil utilização. Uma visão geral do sistema é esquematizada na Figura 4.1, na qual se podem visualizar os diferentes componentes que constituem a máquina e as relações existentes entre cada um desses componentes.

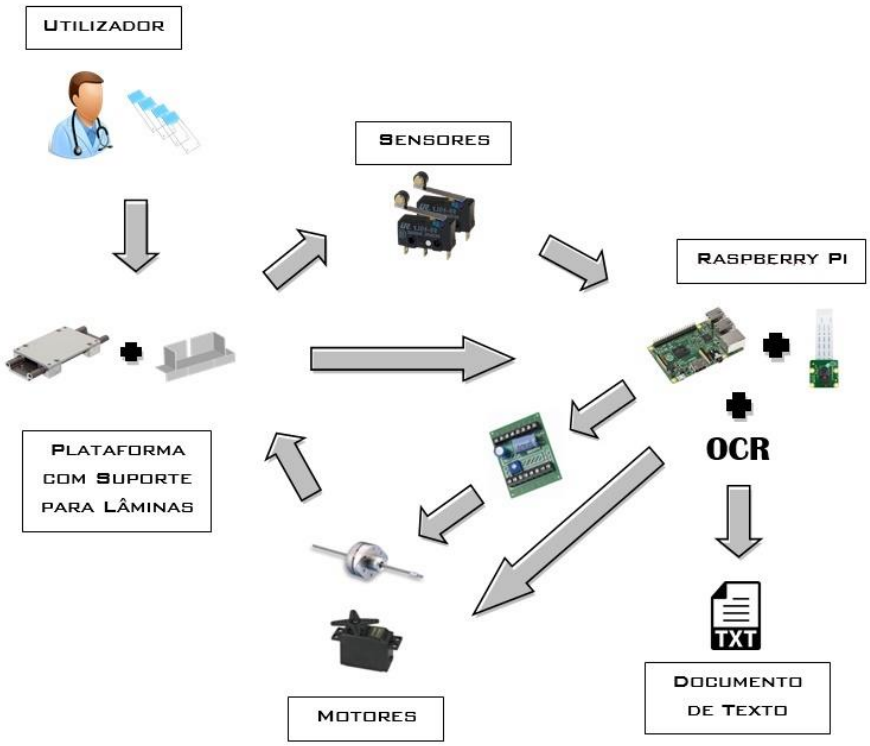


Figura 4.1 - Modelo de arquitectura do sistema

O utilizador interage directamente com o suporte para as lâminas e este por sua vez interage com sensores, motores e com o Raspberry Pi. Este último actua sobre motores, recebe informação dos sensores e cria o *output* do protótipo – o ficheiro de texto com as informações das lâminas. Todo o funcionamento do sistema será aprofundado no capítulo respectivo.

Para além dos intervenientes evidenciados na Figura 4.1, existem alguns componentes - denominadamente electrónicos – que fazem parte e executam um papel também bastante importante. Estes compõem as montagens que possibilitam o correcto funcionamento dos motores, bem como todo o controlo exercido pelo Raspberry Pi. De seguida, serão exibidas essas mesmas montagens que incluem os componentes electrónicos utilizados e as suas ligações.

A. Actuador Linear e *Driver Board*

O actuador linear possui a montagem (Figura 4.2) mais complexa do protótipo pois é utilizado juntamente com uma *driver board* e com outros componentes electrónicos. Em termos de ligações, dos seis fios do actuador linear, dois

são VCC e quatro são as fases que vão servir para o pôr a funcionar. Estes últimos conectam-se à *driver board* enquanto que os primeiros ligam-se à fonte de alimentação, tendo um relé como intermediário. A secção que possui o transístor bipolar e o relé serve para controlar, através do Raspberry Pi, quando é que é feita a alimentação da *driver board*, para que o actuador linear não consuma corrente quando está em repouso. Quando o *pin* do GPIO está activo, a corrente que saí do mesmo atravessa a resistência de 1kΩ e satura o transístor bipolar, passando este a conduzir corrente no seu colector e criando uma diferença de potencial nos terminais do relé. O interruptor troca o terminal de saída, conectando os 12V da fonte de alimentação à *driver board* e, consequentemente, activar o actuador linear.

No segundo lado da *driver board* existem *pins* que controlam o modo de operação do actuador linear e também alguns *outputs*. Os únicos *pins* utilizados desta secção são os 5V de output que se ligam ao ENABLE, o CKI que determina a velocidade do actuador linear e o DIR que define a direcção de rotação.

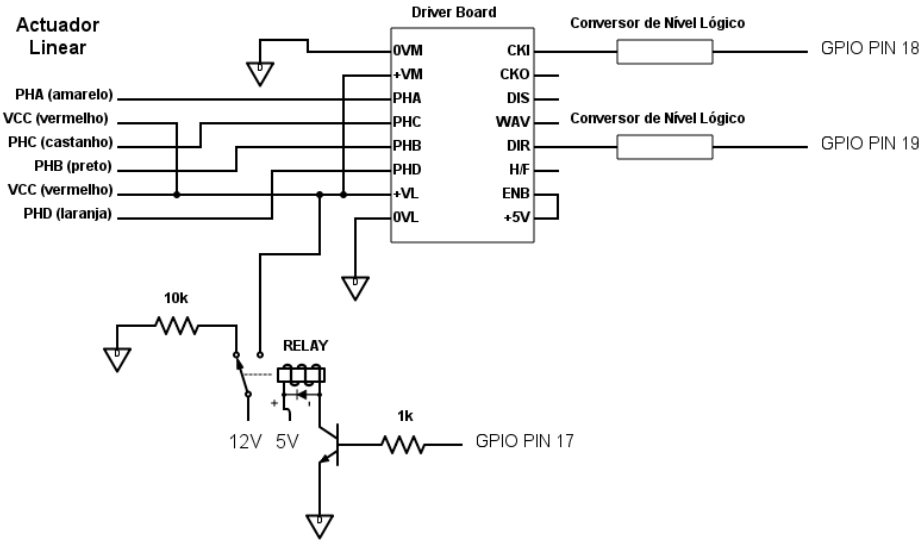


Figura 4.2 - Montagem do actuador linear

B. Motor Servo

Na montagem do motor servo, visível na Figura 4.3, verificam-se as suas três ligações que incluem as de alimentação – VCC e GND – e a de controlo. As

ligações de alimentação encontram-se ligadas directamente à fonte de alimentação, enquanto que o sinal de controlo é conectado ao pino 4 do GPIO do Raspberry Pi. Este último, antes de chegar ao motor, é alvo de um aumento de tensão pelo conversor de nível lógico.

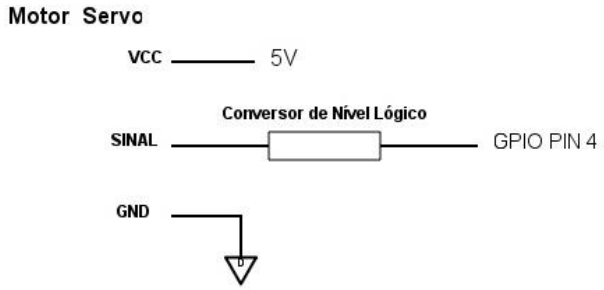


Figura 4.3 - Montagem do motor servo

C. Conversor de Nível Lógico

Nas duas prévias montagens foi utilizada uma "caixa preta" para representar o conversor de nível lógico (apesar deste já ter sido apresentada no capítulo de Fundamentos Teóricos). De seguida, será exibida a montagem que se encontra por detrás dessa "caixa" e que realiza o aumento de 3.3V na saída do GPIO para os 5V que o componente lê.

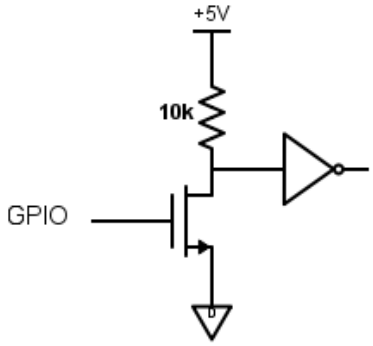


Figura 4.4 - Montagem do conversor de nível lógico

D. *Flash*

Esta montagem (Figura 4.5) consiste, muito simplesmente, num conjunto de leds em série com uma resistência limitadora de corrente de 220 Ω. Esta controla a corrente que atravessa os *leds* e, por sua vez, a intensidade dos mesmos.

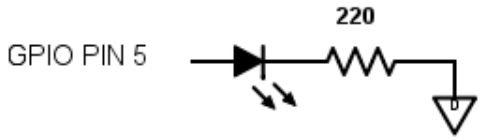


Figura 4.5 - Montagem do flash

E. Sensores

Para terminar a secção das montagens, serão apresentadas as referentes aos dois sensores constituintes do protótipo (Figura 4.6). Ambas são praticamente idênticas, com as poucas alterações a consistirem no *pin* do GPIO (6 e 21) e nas resistências de limitação de corrente (1k e 1.2k, por serem as disponíveis).

Em relação à montagem em si, é bastante simples, contando apenas com o sensor e algumas resistências. Começando pelo estado em repouso, o terminal *normally closed* do sensor está ligado à massa através de uma resistência de 10kΩ, não influenciado os restantes circuitos. No estado activo, o terminal *normally open* recebe os 5V e o divisor de tensão converte-os em 3.3V, que serão lidos pelo Raspberry Pi. A resistência de limitação de corrente é colocada no circuito por se considerar ser boa prática. Se por alguma razão o *pin* do GPIO passa de *input* para *output*, a quantidade de corrente que entra no Raspberry Pi poderia danificar o mesmo, pelo que as resistências de 1k na montagem da esquerda e 1.2k na montagem da direita protegem o Raspberry Pi contra esse cenário.

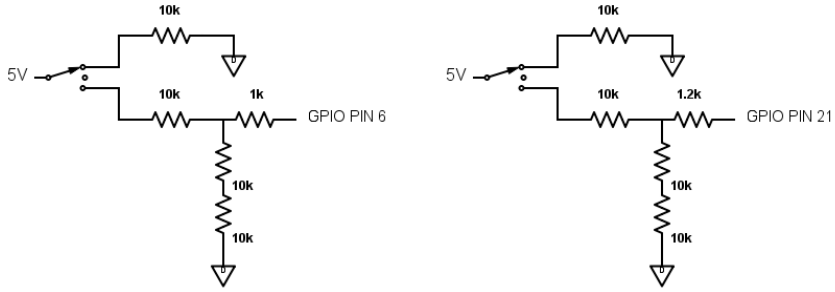


Figura 4.6 - Montagem dos sensores

4.2 Elementos do Protótipo

Como já foi referido, a grande maioria dos materiais utilizados na construção do protótipo foi adquirida de outro projecto, mantendo-se inalterada de modo a promover a longevidade dos materiais e assegurando uma possível utilização em projectos futuros.

Neste capítulo serão apresentados todos os constituintes do protótipo, como os actuadores, sensores, infra-estrutura, controlador e fonte de alimentação. Serão ainda explicadas algumas escolhas que foram tomadas em relação a alguns dos elementos em questão.

4.2.1 Infra-estrutura

A infra-estrutura do protótipo é constituída por um vasto leque de elementos.

Começando por baixo, o suporte do protótipo foi realizado com quatro poleias que seguram a guia metálica, de aproximadamente 1 metro, na vertical. Esta encontra-se perpendicular a uma placa de madeira que consiste no apoio do protótipo. Uma placa ainda foi adicionada com o intuito de se transportar a fonte de alimentação com o resto da estrutura. Estes elementos podem ser visualizados na Figura 4.7.

Figura 4.7 - Secção de suporte do protótipo

Subindo no protótipo encontra-se a estrutura em madeira que alberga a grande maioria dos principais componentes da máquina (Figura 4.8). Esta estrutura suporta as *breadboards*, o Raspberry Pi, a *driver board*, o motor servo e os *leds* que constituem o *flash*.

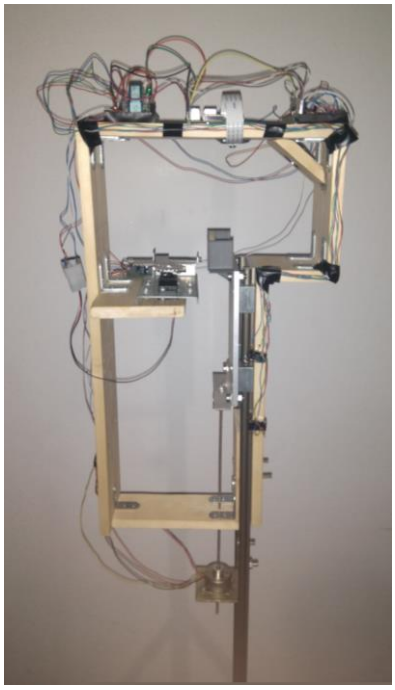


Figura 4.8 – Estrutura em madeira

Como últimos elementos da infra-estrutura temos a plataforma vertical e o suporte para as lâminas.

Estes são ambos elementos fulcrais neste projecto. A primeira pertence ao modelo drylin® W da Igus e é composta por alumínio anodizado enquanto que, o segundo, responsável pelo albergue das lâminas, é um suporte concebido em impressão 3D, pois as suas medidas teriam de ser coincidentes com as medidas das lâminas histológicas. A Figura 4.9 evidencia o desenho do suporte para as lâminas histológicas, realizado no *software* AutoCAD.

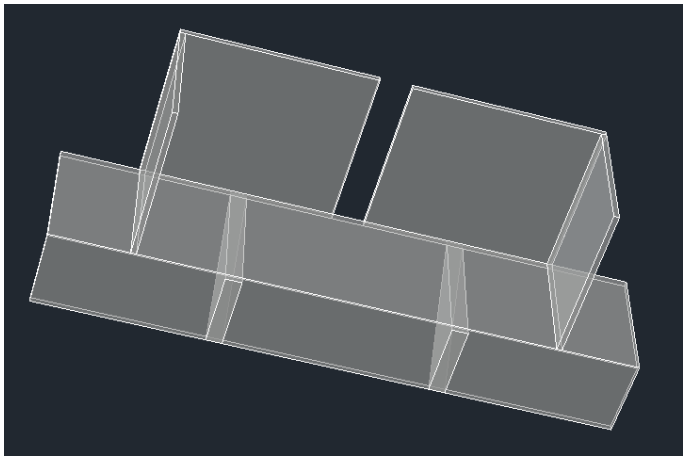


Figura 4.9 - Desenho do suporte para as lâminas histológicas, realizado em AutoCAD

O suporte acopla-se à plataforma vertical no topo da mesma, deixando a parte inferior desta para o acoplamento com o parafuso de avanço utilizado pelo actuador linear. A plataforma vertical desloca-se na calha ou guia previamente falada (Figura 4.10). A guia utilizada tem um comprimento muito superior ao necessário, pelo que se reafirma a tentativa de não modificar material que possa ser utilizado em projectos futuros. A conexão entre estas duas partes é efectuada com a utilização de quatro componentes deslizantes, também evidenciados na Figura 4.10. Estes também são fabricados pela Igus e podem ser identificados pela referência WM-01-10.

Figura 4.10 - Plataforma vertical encaixada na guia

4.2.2 Raspberry Pi 2 Modelo B com Câmara

Pode-se considerar que este é o elemento central do sistema. O Raspberry Pi (Figura 4.11) controla tudo o que há para controlar no protótipo, entre os quais, o actuador linear e o motor servo, recebendo também, informações dos sensores. Todos os sinas de *input* e *output* trocados com os outros elementos são

recebidos e enviados através do GPIO, parte constituinte do Raspberry Pi. Através da utilização de cabos fêmea-macho (também visíveis na Figura 4.11), consegue-se conectar o GPIO a uma *breadboard* com o devido isolamento e assegurar a troca de sinais com os outros elementos de forma segura. Também acoplado ao Raspberry Pi está uma câmara de 8 MP que servirá para fotografar as etiquetas das lâminas.

Figura 4.11 - Raspberry Pi 2 com câmara (à direita)

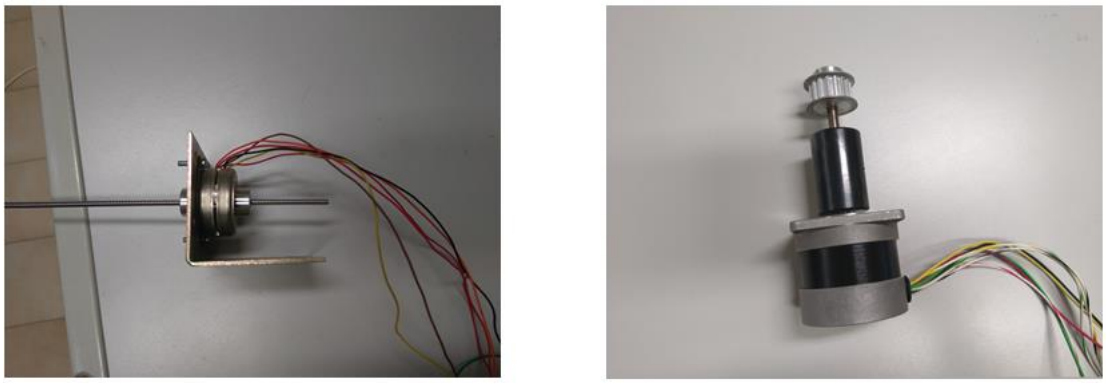
4.2.3 *Flash*

Adicionado como um complemento à fotografia, este conjunto de *leds* (Figura 4.12) funciona como *flash* no momento de captura da imagem. Estes *leds* foram retirados de uma lanterna barata e soldados a fios que os ligam ao Raspberry Pi e ao restante circuito da Figura 4.5. Quando a câmara do Raspberry Pi se prepara para tirar a fotografia à etiqueta da lâmina, estes leds são acessos de maneira a proporcionar uma captura em melhores condições e, por conseguinte, facilitar a tarefa do *software* de reconhecimento óptico de caracteres.

Figura 4.12 - Flash e leds constituintes

4.2.4 Actuador Linear e *Driver Board*

Neste projecto, o movimento da plataforma vertical é proporcionado por um actuador linear, mais especificamente, o 35DBM10B2U-L da Portescap (Figura 4.13 a)). Numa fase inicial do projecto, outra opção estava em aberto, a utilização de um motor de passo bipolar (Figura 4.13 b)), o 440-442 da RS Components. Ambas as soluções apresentavam vantagens e desvantagens, mas acabou-se por escolher o actuador linear, em grande parte devido ao seu formato mais ligeiro e menor peso, pois estes factores facilitavam a sua integração na guia metálica.



a) **b)**

Figura 4.13 - As duas opções. a) Actuador linear. b) Motor de passo

Como qualquer tecnologia deste tipo, o actuador linear consome corrente mesmo quando não está em funcionamento. Este consumo tem a vantagem de oferecer uma certa inércia ao parafuso de avanço porém, quando não é necessária essa inércia, o actuador está a consumir corrente desnecessariamente, po-

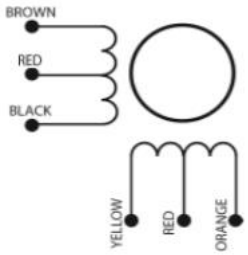
dendo até aquecer demasiado. De modo a resolver este problema, foi adicionado um relé à montagem do actuador linear, com o intuito de cortar esse fluxo de corrente quando este não é necessário.

Algumas características pertinentes deste actuador podem ser examinadas com mais detalhe na Tabela 4.1, retirada do *datasheet*, fornecido pela Portescap, deste componente.

Tabela 4.1 – Dados eléctricos e técnicos do actuador linear

Dados eléctricos	
Tensão Operacional (V)	12
Resistência por fase, ± 10% (Ω)	58.0
Indução por fase (mH)	30.0
Corrente por fase (A)	0.21
Dados técnicos	
Máxima força de retenção (N)	20.9
Mínima força de retenção (não alimentado) (N)	11.1
Máxima Distância (mm)	63.5
Ângulo de passo (°)	7.5 ± 0.5
Passos por revolução	48
Temperatura Operacional (°C)	-20 a 70
Peso (g)	85.2

Como foi previamente referido, o actuador linear é controlado através de seis ligações – duas linhas de alimentação e quatro fases. Em relação às últimas (as ligações de controlo do actuador linear), têm a função de excitar os ímanes permanentes presentes no motor e, consequentemente, provocar o movimento rotativo da secção que está em contacto com o parafuso de avanço. A disposição dos terminais, bem como a ordem com que os mesmos necessitam de ser excitados para provocar o movimento, podem ser observados na Figura 4.14.



UNIPOLAR COIL				
YELLOW	ORANGE	BROWN	BLACK	RED
+	-	+	-	+
+	-	-	+	+
-	+	-	+	+
-	+	+	-	+
+	-	+	-	+

Figura 4.14 - Diagrama do actuador linear e sequência de excitação

Para facilitar o accionamento do actuador utilizou-se uma *driver board* (Figura 4.15).

Figura 4.15 - Driver Board

Esta é uma placa que não só realiza a sequência de excitação dos ímanes automaticamente, como também fornece uma série de *pins* que permitem controlar o actuador de forma mais fácil, sendo possível controlar a velocidade ou direcção que o actuador faz girar o parafuso de avanço. Na Figura 4.16 é possível observar todos os terminais da placa.

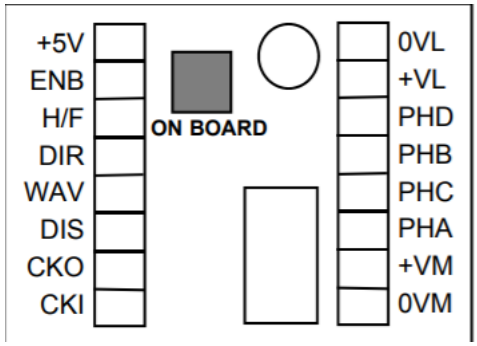


Figura 4.16 - Esquemático da *driver board* com os terminais evidenciados

Começando pela parte esquerda da placa, há terminais tanto de *output* como de *input*:

- 5V: *Output* de 5V;
- ENB: *Input* que quando a HIGH activa a entrada CKI;
- H/F: *Input* que quando a HIGH activa o modo Half *Step*, que causa o actuador linear a dar passos com metade da rotação;
- DIR: *Input* que define a direcção de rotação do parafuso de avanço;
- WAV: *Input* que quando a HIGH activa o modo de onda;
- DIS: *Input* que quando a LOW activa a entrada CKI;
- CKO: *Output* que transmite uma onda do tipo *clock* com uma frequência controlada pelo potenciómetro existente na placa;
- CKI: *Input* na forma de uma onde do tipo *clock* que define a velocidade do actuador linear.

Em relação há parte direita da placa, todos os terminais são mais direccionados para o accionamento do motor. As entradas VL consistem na alimentação das portas lógicas existentes na placa, as entradas VM consistem na alimentação do actuador linear e as restantes coincidem com as fases previamente mencionadas, essenciais para o movimento rotativo.

4.2.5 Motor Servo

O motor servo é responsável por empurrar a lâmina no topo da pilha, após esta ter sido fotografada e analisada. O motor utilizado para este efeito é um motor servo Futaba S3003 (Figura 4.17).

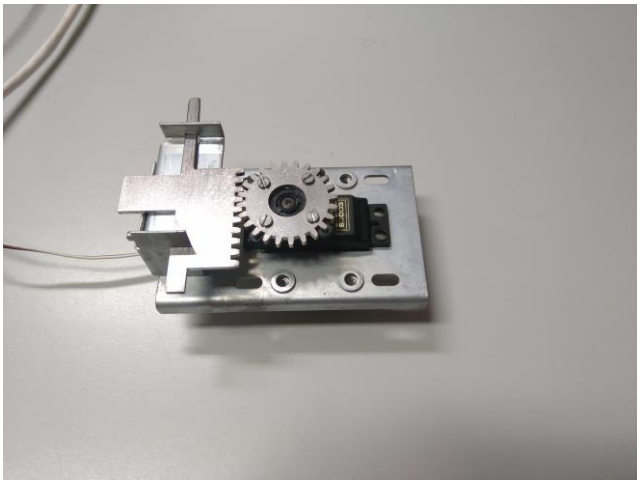


Figura 4.17 - Motor servo Futaba S3003

Como pode ser evidenciado na figura, ao motor foi adicionada uma estrutura metálica de modo a converter o movimento rotativo característico do motor servo num movimento linear, proporcionado pela engrenagem presente na dita estrutura. A extensão da estrutura metálica é a parte deste componente que entrará em contacto com as lâminas.

As características deste motor, retiradas do seu *datasheet*, podem ser observadas na Tabela 4.2.

Tabela 4.2 - Características do motor servo

Modulação	Analógica
Tensão Operacional (V)	4.8 - 6
Torque (kg-cm)	3.17 - 4.10
Velocidade (s/60°)	0.23 - 0.19
Peso (g)	37.0
Alcance Rotacional (°)	60
Ciclo do pulso (ms)	30
Largura do pulso (µs)	500-3000

De notar, em relação à tabela apresentada, que os valores de torque e de velocidade variam de acordo com a tensão que é aplicada ao motor. Assim sendo, com a tensão mínima de 4.8V, o motor apresenta um torque de 3.17 kg-cm e

uma velocidade de 0.23 s/60° e com a tensão máxima de 6V, um torque de 4.10 kg-cm e uma velocidade de 0.19 s/60°.

4.2.6 Sensores de Fim de Curso

No protótipo são utilizados sensores de fim de curso para calibração e interrupção do movimento da plataforma vertical. Estes sensores são, basicamente, interruptores idênticos ao da Figura 4.18. Quando a patilha é pressionada, a tensão presente no terminal comum é transmitida pelo terminal *normally open*. Quando a patilha está em repouso, a tensão é transmitida pelo terminal *normally closed*.

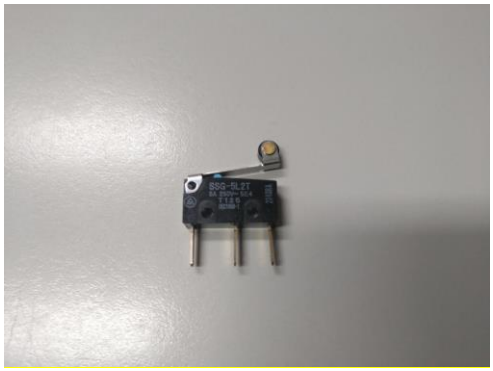


Figura 4.18 – Micro-switch OMRON SSG-5L2T

Foram inseridos, no protótipo, dois destes sensores. O primeiro, colocado numa parte mais superior, promove a interrupção da plataforma vertical para permitir ao utilizador, colocar as lâminas histológicas no respectivo suporte. O segundo encontra-se mais abaixo e tem o propósito de calibrar a máquina, basicamente, consiste num ponto de referência para o protótipo começar a ler as etiquetas.

4.2.7 Fonte de Alimentação

Para todo o sistema trabalhar sem grandes problemas, é necessário estar devidamente alimentado. Para este efeito, utilizou-se uma fonte de alimentação Codegen de 375W (Figura 4.19), já presente no laboratório. Sendo uma fonte de alimentação concebida para computadores, tem a vantagem de apresentar vá-

rios *outputs* de várias tensões, mais especificamente, 3.3V, 5V e 12V. Esta característica torna esta numa fonte bastante flexível em relação ao número de diferentes tipos de equipamento que possam ser introduzidos no protótipo, cobrindo a grande maioria de tensões utilizadas por *hardware* de menores dimensões.

Figura 4.19 - Fonte de alimentação Codegen 375W

4.3 Princípio de Funcionamento

O protótipo foi concebido com a intenção de a sua utilização ser bastante simples e de se realizar o mínimo esforço por parte do utilizador.

Em primeiro lugar, a plataforma movimenta-se num sentido ascendente até activar o sensor superior. O utilizador tem a função de inserir o conjunto de lâminas (no máximo, vinte e nove) no local apropriado e dar a ordem de avanço ao Raspberry Pi, directamente ou através de acesso remoto, de forma a dar início ao processo. A plataforma vertical desce até activar o sensor inferior (de calibração) e dá-se início à leitura das etiquetas das lâminas. O protótipo detém o Raspberry Pi como o "cérebro" das operações, pois é este que não só activa os motores, como também executa o *software* de OCR. As lâminas histológicas, uma a uma, serão alvo de uma fotografia capturada pela câmara conectada ao Raspberry Pi, que se restringirá à secção da etiqueta. Fotografia essa que será pré-processada com a utilização da biblioteca OpenCV de forma a colocar a imagem nas melhores condições para, posteriormente, ser processada pelo Py-Tesseract, de modo a retirar as informações desejadas. O *output* será escrito

num documento de texto de modo a guardar os resultados do OCR. Após uma lâmina ser processada, o Raspberry Pi dá ordem para o motor servo girar e, consequentemente, empurrar a lâmina processada que corresponde à lâmina do topo da torre. Após esta acção, o Raspberry Pi – através do controlador - activa o actuador linear com o intuito de subir a plataforma vertical até que a nova lâmina do topo se encontre na posição da anterior, e assim consecutivamente até não existirem lâminas no suporte.

4.4 Código Implementado

De forma a controlar os motores constituintes do protótipo, bem como realizar o reconhecimento óptico de caracteres e ordenar a câmara do Raspberry Pi a disparar, foi necessário criar determinadas rotinas. Previamente, teve-se que preparar o Raspbian com todas as ferramentas necessárias para a criação destas rotinas. Assim sendo, procedeu-se à instalação do Python, OpenCV e Pytesseract, peças essenciais para o correcto funcionamento do protótipo.

Todos os dispositivos foram testados individualmente, não só para verificar o seu estado como também para haver uma maior facilidade a identificar e corrigir erros. Estes testes serão apresentados no capítulo Resultados e Discussão. Todos os *scripts* realizados no âmbito de teste e o *script* final estão disponíveis na secção de Anexos.

4.4.1 *Script* do Protótipo

Concluindo com sucesso todos os testes individuais, procedeu-se à criação das rotinas finais que o protótipo vai exibir, desde a ordem de início do processo até à última etiqueta ser reconhecida, unindo todos os elementos do protótipo para um único objectivo.

Em relação ao *script* em si, todas as bibliotecas importadas nos testes individuas estão aqui presentes, somente com a adição da biblioteca *sys* (o seu papel será explicado mais à frente). Está ainda dividido em quatro partes distintas: *Setup*, *Functions*, *Operation* e *Cleaning*. De forma a evitar redundâncias, a

numeração dos *pins*, bem como as funções criadas nos testes individuais mantêm-se para esta rotina e não serão de novo mencionados em detalhe.

A. Setup

Nesta secção preparam-se todos os *pins* do GPIO que são utilizados na rotina. Os motores e o *flash* são definidos como *outputs* enquanto que os sensores como *inputs*. Também a câmara é iniciada, estabelecendo a resolução das fotografias para 1920x1080.

B. Functions

Na seguinte etapa estão as cinco funções desenvolvidas que serão chamadas ao longo de todo o *script*. Estas cinco funções são: *pushSlide()*, *takePicture()*, *imageProcessing(image)*, *OCR(image)* e *saveData(file, text)*. Destas, apenas a primeira foi utilizada na secção de testes, enquanto que as outras foram criadas apenas para o *script* do protótipo. O papel destas funções passa por aumentar a legibilidade do código através da segmentação de cada tarefa, visível através das suas implementações, que consistem em instruções utilizadas durante os testes.

C. Operation

A secção *Operation* é, de resto, a mais importante de todo o *script*. Nesta, está traduzido, em código, o que foi explicado no capítulo *Princípio de Funcionamento*. Em primeiro lugar, criou-se o ficheiro em que se vão gravar as informações das lâminas em modo *append*, pois ao longo da rotina serão adicionados pedaços de texto ao ficheiro. Depois, pergunta-se ao utilizador quantas lâminas vão ser processadas, resposta esta que será guardada na variável *n_slides*. Se a resposta dada não estiver situada entre 1 ou 29 lâminas, inclusive, o programa é terminado com a instrução *sys.exit(0)*. A plataforma vertical é então levada até ao sensor superior, momento em que o utilizador tem a função de introduzir as lâminas histológicas no suporte e carregar *ENTER* quando tiver terminado. O próximo passo é calibrar a plataforma, que é levada até ao sensor inferior. Tendo em conta o número de lâminas introduzido pelo utilizador, a plataforma

vertical subirá até a lâmina no topo da torre estar alinhada com o motor servo, de modo a dar-se a sua leitura, gravação no ficheiro e expulsão. Este passo é repetido para todas as lâminas até não restar nenhuma na torre. O actuador linear é parado, o ficheiro fechado e o relé desactivado.

De seguida, será exibido um fluxograma tendo em vista a clarificação do funcionamento do protótipo e da estrutura do código acima descrito.

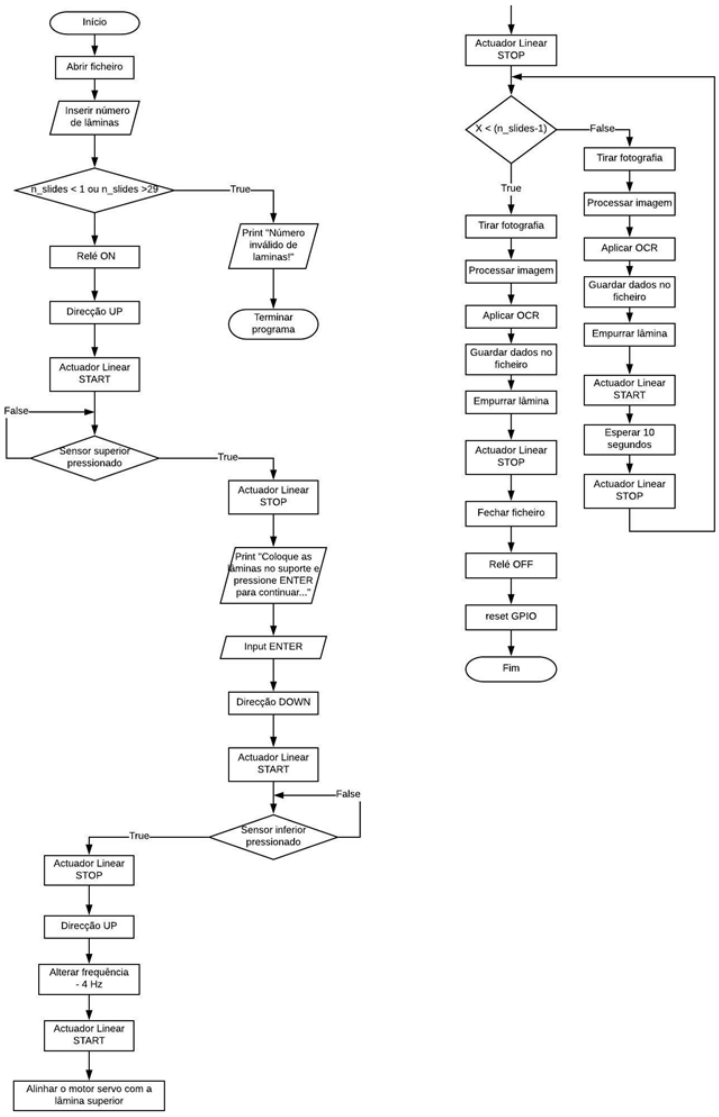


Figura 4.20 - Fluxograma descritivo do código implementado

D. Cleaning

A última etapa da rotina consiste numa boa prática quando se utiliza o GPIO do Raspberry Pi que é executar a instrução *GPIO.cleanup()*. Esta instrução faz *reset* aos *pins* utilizados durante a execução do *script*.

5

Resultados e Discussão

Este capítulo será responsável por apresentar os testes individuais que foram realizados aos componentes do protótipo, o resultado da conversão da informação das etiquetas realizado pelo protótipo, discutir esse mesmo resultado e mencionar certos aspectos negativos do projecto.

5.1 Testes Individuais

Os testes presentes nesta secção serão exibidos na mesma ordem com que foram criados.

5.1.1 Actuador Linear

Os testes realizados ao actuador linear tiveram o objectivo de verificar se as ligações foram efectuadas de forma correcta e foi também um primeiro passo em perceber como o Raspberry Pi e a linguagem Python interagem, tentando manipular a direcção e velocidade de rotação do actuador linear.

Este *script* de teste começa por invocar duas bibliotecas: a *RPi.GPIO* e a *time*. Depois vem uma secção de preparação do GPIO, seleccionando o modo

BCM de disposição dos *pins*[19] e inicializando os *pins* que se vão utilizar neste teste:

- Velocidade – *pin 18*
- Direcção – *pin 19*
- Relé – *pin* 17

Como a *driver board* recebe um sinal de *clock* como definidor de velocidade do actuador linear, utilizou-se o *pin* 18 do GPIO para a velocidade pois este consegue emitir uma onda do tipo PWM.

Os testes a este dispositivo procederam, alterando-se a frequência da onda PWM (mantendo o *duty cycle* a 50%) e verificando o efeito no actuador linear, bem como alterar o estado do *pin* da direcção de LOW para HIGH e vice-versa, verificando, também, o efeito na direcção de rotação do actuador. O relé é activado no início do teste, e desligado no final. Por fim, é realizado o *reset* aos *pins* utilizados durante o programa.

5.1.2 Motor Servo

O segundo dispositivo a ser testado foi o motor servo. Um dos problemas relacionados com este teste foi o facto de que, à semelhança do actuador linear, o motor servo também requer um sinal PWM para ser controlado e apenas o *pin* 18 do GPIO tem essa funcionalidade. Como tal, utilizou-se a biblioteca *wiringPi*, que consegue simular um sinal PWM em qualquer outro *pin*. O próximo passo é a inicialização do único *pin* utilizado neste teste: o 4. Neste *pin* é então emitido o sinal PWM que decide a posição do motor. A última fase deste teste é executar o movimento propriamente dito, fazendo com que a extensão metálica fique completamente estendida, voltando depois à posição inicial, completamente recolhida.

[19] Existem dois modos de disposição dos *pins*: BCM e BOARD. A diferença entre os dois modos é que no primeiro, os *pins* são referidos pelo número presente nos rectângulos verdes na Figura 3.11, enquanto que pelo segundo modo são utilizados os números presentes na placa GPIO, também visível na mesma figura.

5.1.3 PiCamera

Este teste consiste em utilizar a câmara do Raspberry Pi, ou PiCamera, para realizar uma avaliação preliminar, tanto da qualidade da câmara, como da focagem que esta exibe. Para a realização do teste foram precisas importar as bibliotecas *PiRGBArray*, P*iCamera*, *time* e *cv2*. Em primeiro lugar é realizada a inicialização da câmara e do seu *output*. Depois, visualiza-se o que a câmara está a captar e, passados cinco segundos, esta tira a fotografia, a qual é visualizada no ecrã, utilizando métodos do OpenCV.

O maior obstáculo encontrado ao testar a PiCamera foi o facto de esta vir de fábrica com uma focagem fixa ao infinito. Como no protótipo a câmara vai estar relativamente próxima do objecto de fotografia – as lâminas histológicas – a fotografia ficava desfocada. Para solucionar este problema, foi necessário rodar com bastante cuidado a lente que se encontra presa com cola e ir continuamente testando até o foque estar à distância pretendida.

5.1.4 OCR

Este teste foi realizado com objectivo de implementar e verificar a qualidade do *software* PyTesseract.

No *script* de teste foram importadas várias bibliotecas, mais denominadamente, a *print_function*, *pytesseract*, *Image* da Python Imaging Library, *argparse* e *cv2*. A imagem a ser analisada é passada através da linha de comando, bastando inserir *sudo python ocr.py –image (caminho da imagem)*. A imagem é lida e processada por um método da biblioteca OpenCV, mais especificamente, é binarizada de modo a ser facilmente acolhida pelo PyTesseract. Antes de ser alvo do reconhecimento óptico de caracteres, a imagem é exibida no ecrã sendo necessário carregar em qualquer tecla do teclado para continuar o teste. É então aplicado o OCR na imagem e o texto resultante é exibido na linha de comando e guardado num ficheiro de texto.

Estes testes foram realizados em duas imagens (Figura 5.1). Ambos apresentaram resultados bastante satisfatórios. No primeiro, apenas os caracteres que possuíam elementos como acentos e tremas não foram correctamente reco-

nhecidos, enquanto que no segundo, em grande parte devido à ausência dos elementos anteriormente referidos, a taxa de sucesso foi maior, localizando apenas dois erros: 'evangeliza' (evangelizes) e 'resoumes' (resources), este último, muito por culpa da falta de qualidade da palavra original.

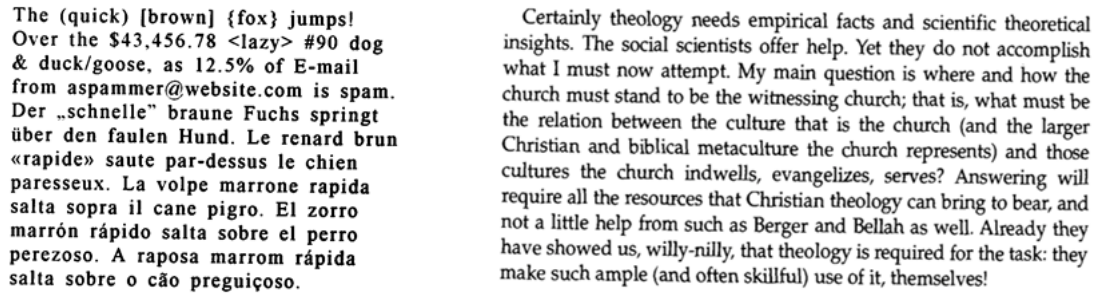


Figura 5.1 - Textos utilizados no teste do *software* de OCR

5.1.5 Sensores

O último teste foi realizado sobre os sensores de fim de curso. O único objectivo deste teste foi confirmar o bom funcionamento dos sensores, isto é, se a tensão presente no terminal comum transita de terminal quando a patilha é pressionada.

Para a realização deste teste foi apenas utilizada a biblioteca *RPi.GPIO*. O *pin* escolhido para receber o sinal do sensor foi o 6. Este é estabelecido como *input* e o programa somente mostra no ecrã '0' ou '1', dependendo da patilha do sensor, se está, ou não, pressionada.

Não houve nenhum problema resultante deste teste, concluindo-se o bom estado do material e da rotina de instruções.

5.2 Teste do Protótipo

Com o protótipo construído e totalmente funcional, procedeu-se ao teste dos motores e dos sensores, bem como da câmara, *flash* e do *software* de reconhecimento óptico de caracteres, só que, desta vez, foi um teste conjunto, com todos os componentes a comunicarem entre si. Nesta secção, decidiu-se dividir estes testes em duas partes, uma mais focada no movimento dos mecanismos, isto é, o movimento que o protótipo tem de realizar durante o funcionamento

normal do mesmo, e a outra, mais relacionada com a câmara e com o *software* de reconhecimento óptico de caracteres.

A. Movimento

Este teste consiste numa simulação do funcionamento normal do protótipo, apenas com a ausência da câmara e do OCR.

O teste começa por colocar a plataforma vertical a subir, a uma velocidade constante, até o sensor superior ser pressionado. Depois de o utilizador colocar as lâminas no suporte, a plataforma desce, com a mesma velocidade, até atingir o sensor inferior. Desta feita, o actuador linear irá alinhar o motor servo com a lâmina que se encontra no topo da torre, subindo a plataforma 10 segundos por cada lâmina a menos em relação ao número máximo possível (29), a uma velocidade de 4 Hz. Com o alinhamento conseguido, o motor servo empurra a lâmina do topo, a qual terá que ser retirado pelo utilizador com a mão. A plataforma é então subida outra vez durante 10 segundos a uma velocidade de 4 Hz de forma a alinhar a segunda lâmina com o motor servo que a vai expulsar e assim consecutivamente, até o suporte estar vazio.

B. Câmara e OCR

Este teste consiste na utilização da câmara e do *software* de reconhecimento óptico de caracteres para validar a correcta conversão dos dados presentes nas etiquetas das lâminas para formato digital. Apenas foram disponibilizadas quatro lâminas, portanto foram realizados quatro testes neste âmbito. A Figura 5.2 mostra as quatro lâminas testadas, bem como o papel do *flash*, a iluminar a zona de interesse das imagens.

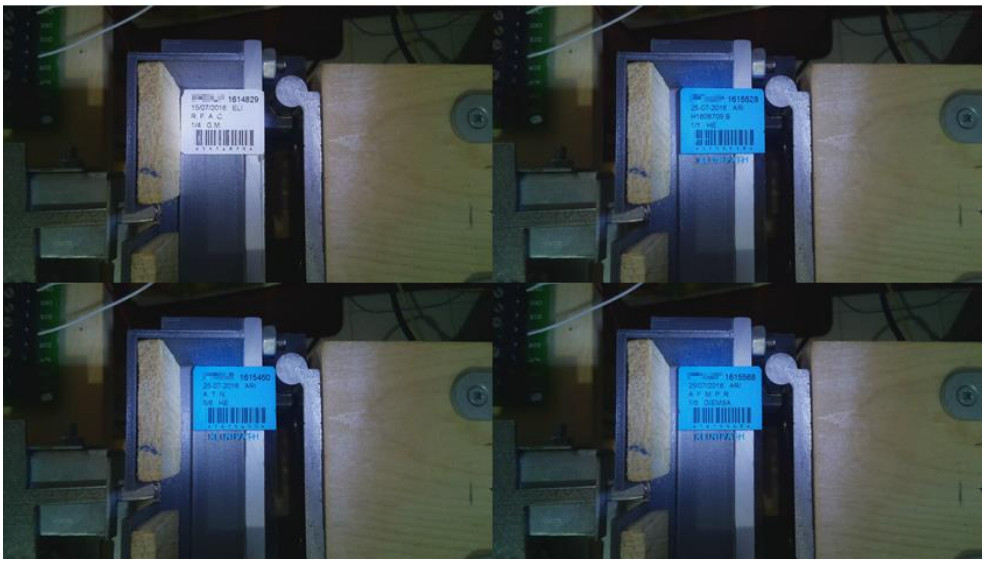


Figura 5.2 - Imagens capturadas pelas câmara nos 4 testes realizados[20]

Estas imagens foram delimitadas pela zona de interesse (etiqueta) e processadas, o que resultou nas imagens da Figura 5.3.

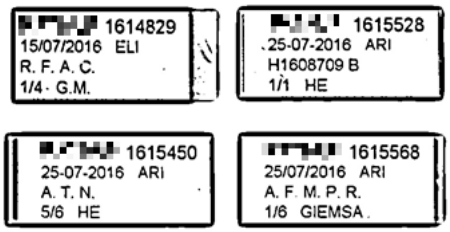


Figura 5.3 - Etiquetas delimitadas e processadas

Estas imagens são então sujeitas ao *software* de reconhecimento óptico de caracteres e a sua informação guardada num ficheiro de texto. Se algum caracter for incorrectamente reconhecido e resultar num caracter que não possa ser codificado em ASCII, o programa não regista a informação dessa lâmina.

As diferenças entre a informação original e a informação após a conversão podem ser visualizadas na Figura 5.4.

[20] O nome da instituição foi e será, ao longo desta dissertação, intencionalmente mantido no anonimato por questões éticas.

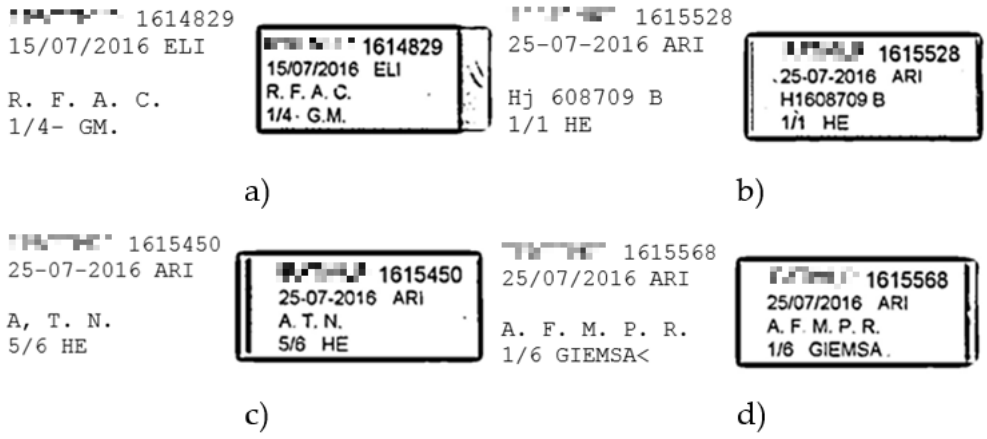


Figura 5.4 - Resultado da conversão e informação original em comparação

Esta imagem mostra o resultado do teste às 4 lâminas disponibilizadas, cuja informação foi registada num ficheiro de texto. Comparando o resultado com a informação original, podem ser apontados alguns erros, por muito pequenos que sejam, em todas as conversões: na conversão a), um ponto a meia altura proveniente de ruído é interpretado com um hífen e um ponto final não é reconhecido; na b), um número "1" é interpretado como um "j"; na c), um ponto final é interpretado como uma vírgula; na d), um ponto proveniente de ruído é interpretado como o símbolo "menor que".

De um ponto de vista estatístico, a percentagem de sucesso de cada conversão foi calculada - dividindo o número de caracteres bem reconhecidos pelo número total de caracteres – assumindo caracteres que tenham surgido no resultado final, provenientes de marcas na imagem de *input*, como um caracter mal reconhecido. A conversão do nome da instituição não se insere nesta estatística, visto estar escondida.

Tabela 5.1 - Percentagem de Sucesso do OCR

Lâminas	Lâmina a)	Lâmina b)	Lâmina c)	Lâmina d)
Percentagem de Sucesso	94,29%	97,06%	96,77%	97,44%

Analisando os resultados obtidos na Tabela 5.1, pode-se confirmar que a informação presente nas lâminas é convertida de um modo bastante aceitável, mostrando percentagens de sucesso a rondar os 96%. De notar que uma fracção

bastante significativa dos erros obtidos tem origem no estado das lâminas testadas, as quais apresentam alguma sujidade e faltas de tinta, o que levou o OCR a reconhecer estas marcas como caracteres e, assim, prejudicando o resultado final da conversão.

Outro resultado de interesse para a análise da *performance* do protótipo são os tempos de execução. Foram cronometrados dois testes distintos: um com 20 lâminas e outro com 10. Desde o momento em que é pressionada a tecla ENTER para começar o processo, o protótipo demorou 8 minutos e 25 segundos a reconhecer 20 lâminas, enquanto que, para 10 lâminas, demorou 6 minutos e 47 segundos. As lâminas histológicas, ao terem 1 mm de espessura cada uma, levaram a uma exigência de precisão bastante elevada, o que resultou no tempo demorado que estes dois testes evidenciaram. Este é reflexo de uma escolha realizada em detrimento da velocidade e em favor do torque e precisão.

6

Conclusões e Trabalho Futuro

Aqui serão feitas as considerações finais acerca desta dissertação, bem como a menção de qualquer inovação que possa ser feita de forma a melhorar o protótipo em qualquer aspecto.

Após a realização de todos os testes, pode-se concluir que o protótipo é capaz de realizar, com pequenos erros, aquilo que foi proposto no início desta dissertação – registar a informação presente nas etiquetas de lâminas histológicas. No entanto, durante todo este processo foram registadas certas ocorrências que merecem ser mencionadas pelo facto de prejudicarem o normal funcionamento do protótipo, com maior ou menor gravidade, podendo ser melhoradas posteriormente.

O primeiro aspecto menos bom, e talvez o que prejudica com mais gravidade o funcionamento do protótipo, é o motor servo. Durante os testes, verificou-se que para além de gerar um ruído significativo – problema causado, provavelmente, pelo facto de o sinal de controlo ser uma simulação de uma onda PWM -, o comprimento da sua extensão não permite a expulsão completa da lâmina da torre, tendo de ser retirada pelo utilizador, depois de estar parcialmente de fora. Outro aspecto prende-se com o facto de as lâminas, por vezes, ficarem pegadas entre si. Este inconveniente dificulta bastante a expulsão das mesmas da torre. Vale também realçar os problemas derivados do estado de algumas etiquetas que, por sujidade ou por falhas de tinta, prejudicaram o *output*

do *software* de reconhecimento óptico de caracteres. Outra consideração prende-se com o mau contacto que, por vezes, existia entre a fonte de alimentação e os fios. Outro ponto menos bom é a velocidade que o actuador linear promove na plataforma, mas, para privilegiar o torque e desprivilegiar a incerteza do actuador linear, é um mal necessário. Por último, durante o período de testes, verificou-se uma certa dificuldade em inserir as lâminas histológicas no suporte impresso a 3 dimensões. Isto deveu-se ao facto de as medidas dimensionadas no AutoCAD serem demasiado exactas, não deixando "margem de manobra" para inserir as lâminas no suporte. Uma folga de, por exemplo, um milímetro resolveria a situação.

Independentemente dos erros ocorridos, existem melhorias ou inovações que se podem efectuar sobre o protótipo, com vista a que este se torne mais completo ou, simplesmente, que execute as tarefas com maior precisão.

Uma delas é a criação de uma base de dados, com vista a guardar toda a informação registada e facilitar o seu acesso. Outra é a criação de uma plataforma de descida, com o objectivo de prescindir, praticamente por completo, de acção por parte do utilizador. Uma última modificação, ou modificações, que se poderão fazer para melhorar o protótipo, de um ponto de vista geral, é utilizar materiais mais apropriados de forma a torná-lo mais conciso, eliminando factores desnecessários como o seu tamanho e o tamanho da fonte de alimentação ou criar um suporte mais estável.

Bibliografia

Arica, N. & Yarman-Vural, F.T., 2002. "Optical Character Recognition for Cursive Handwriting". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6), pp.801–813.

Bassil, Y. & Alwani, M., 2012. "OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion". *Journal of Emerging Trends in Computing and Information Sciences*, 3(1), pp.90–99. Available at: http://arxiv.org/abs/1204.0191.

Bieniecki, W., Grabowski, S. & Rozenberg, W., 2007. "Image preprocessing for improving OCR accuracy". *Proceeding of the 3rd International Conference of Young Scientists "Perspective Technologies and Methods in MEMS Design", MEMSTECH 2007*, pp.75–80.

Ghahramani, Z., 2001. "An Introduction to Hidden Markov Models and Bayesian Networks". *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1), p.2.

Ibrahim, M., Shehata, M. & Badawy, W., 2012. "Automatic License Plate Recognition (ALPR): A State of the Art Review". *IEEE Transactions on Circuits and Systems for Video Technology*, pp.1–14. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6213519.

Impedoso, D. & Pirlo, G., 2008. "Automatic Signature Verification: the State of the Art". *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS*, 38(5), pp.609–635.

Karkera, B. V et al., 2011. "Biopsy: Clinical implications. *Journal of Dentistry and Oral Hygiene*", 3(8), pp.106–108. Available at: http://www.academicjournals.org/JDOH.

Mangoli, M.N. & Desai, S., 2016. "Optical Character Recognition for Cursive

Handwriting". *International Research Journal of Engineering and Technology (IRJET)*, 3(5), pp.792–795.

Mithe, R., Indalkar, S. & Divekar, N., 2014. "Optical character recognition". *International Journal of Emerging Technology and Advanced Enginnering*, 4(5), pp.219–223. Available at: http://www.tutorialspoint.com/dip/Optical_Character_Recognition.htm.

Perez-Cortes, J.C. et al., 2000. Stochastic error-correcting parsing for OCR post-processing. *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, 4, pp.405–408.

Singh, A., Bacchuwar, K. & Bhasin, A., 2012. "A Survey of OCR Applications". *International Journal of Machine Learning and Computing*, 2(3), pp.314–318. Available at: http://www.ijmlc.org/papers/137-L0022.pdf.

Singh, R. et al., 2010. "Optical Character Recognition (OCR) for Printed Devnagari Script Using Artificial Neural Network". *International Journal of Computer Science & Communication*, 1(1), pp.91–95.

Smith, R., 2007. "An Overview of the Tesseract OCR Engine". *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2, pp.629–633.

Team, T.A.C.S. medical and editorial content, 2015. "What happens to biopsy and cytology specimens?" Available at: https://www.cancer.org/treatment/understanding-your-diagnosis/tests/testing-biopsy-and-cytology-specimens-for-cancer/what-happens-to-specimens.html [Accessed January 28, 2017].

Trier, O.D., Jain, a K. & Taxt, T., 1996. "Feature extraction methods for character recognition - A survey". *Pattern Recognition*, 29, pp.641–662.

Verma, R., 2012. "A-Survey of Feature Extraction and Classification Techniques in OCR Systems". *Ijcait*, I(III), pp.1–3. Available at: http://www.ijcait.com/IJCAIT/index.php/www-ijcs/article/view/140.

White, J.M. & Rohrer, G.D., 1983. "Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction". *IBM Journal of Research and Development*, 27(4), pp.400–411. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5390437.

Anexos

Scripts de Teste

Actuador Linear

```
#Library imports
import RPi.GPIO as GPIO
import time

#Set pin setmode
GPIO.setmode(GPIO.BCM)

#Pins declaration
VELOCITY_PIN = 18
DIRECTION_PIN = 19
RELAY_PIN = 17

#Pins setups, output mode and initial state
GPIO.setup(VELOCITY_PIN, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(DIRECTION_PIN, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(RELAY_PIN, GPIO.OUT, initial = GPIO.LOW)

#Associate PWM pulse with velocity pin and insert velocity, in hertz
p=GPIO.PWM(VELOCITY_PIN, 40)

#DIRECTION
#LOW -> DOWN
#HIGH -> UP

#Direction pin set to HIGH
GPIO.output(DIRECTION_PIN, GPIO.HIGH)

#Turn on relay
GPIO.output(RELAY, GPIO.HIGH)

#Start PWM pulse with a 50% duty cycle
```

```
p.start(50)

#Wait 10 seconds
time.sleep(10)

#Stop PWM pulse
p.stop()

#Turn off relay
GPIO.output(RELAY, GPIO.LOW)

#Clean GPIO pins
GPIO.cleanup()
```

Motor Servo

```
#Library import
import wiringpi

#Function that rotates the servo motor in order to push the top slide
#and then retractes until it reaches the inicial position
def pushSlide():
    wiringpi.softPwmWrite(SERVO_PIN, 11)
    wiringpi.delay(2000)
    wiringpi.softPwmWrite(SERVO_PIN, 21)
    wiringpi.delay(2000)

#Set wiringpi pin setmode
wiringpi.wiringPiSetupGpio()

#Pin declaration
SERVO_PIN = 4

#Set pin to output mode
wiringpi.pinMode(SERVO_PIN, 1)

#Create PWM pulse
wiringpi.softPwmCreate(SERVO_PIN, 0, 100)

#Execute functon
pushSlide()
```

PiCamera

```
#Library imports
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2

#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
rawCapture = PiRGBArray(camera)
camera.resolution =(1920, 1088)
camera.vflip = True

#Start camera preview
```

```
camera.start_preview()

#Wait 5 seconds
time.sleep(5)

#Sop camera preview
camera.stop_preview()

#Capture an image from the camera
camera.capture(rawCapture, format="bgr")
image = rawCapture.array

#Display the image on screen and wait for a keypress
cv2.imshow("Image", image)
cv2.waitKey(0)
```

OCR

```
#Library imports
from __future__ import print_function
import pytesseract
from PIL import Image
import argparse
import cv2
import numpy as np

#Grabs the path to the image that will be opened from the command line
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True, help = "Path to the
image")
args=vars(ap.parse_args())

#Read image
image=cv2.imread(args["image"])

#Changes image color scale to gray
image=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

#Applies a binarization filter to the image
image = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 17, 7)

#Display the image on screen and wait for a keypress
cv2.imshow("image", image)
cv2.waitKey(0)

#convert image to PIL format
pil_image  = Image.fromarray(image)

#Applies the ocr software to the image and prints the resultant text
text = pytesseract.image_to_string(pil_image)
print (text)

#Saves text to a file named data.txt
file = open("data.txt", "w+")
file.write(text)
```

Sensores

```
#Library imports
import RPi.GPIO as GPIO

#Set pin setmode
GPIO.setmode(GPIO.BCM)

#Set pin to input mode
GPIO.setup(6, GPIO.IN)

#Print GPIO pin value
print GPIO.input(6)

#Clean GPIO pin
GPIO.cleanup()
```

Script do Protótipo

```
#Library imports
from __future__ import print_function
import RPi.GPIO as GPIO
import wiringpi
import time
import sys
import pytesseract
from PIL import Image
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera

############### SETUP ############################

#Set pin setmode
GPIO.setmode(GPIO.BCM)

#Dismiss warnings
GPIO.setwarnings(False)

##ACTUADOR LINEAR#######

#Pins declaration
VELOCITY_PIN = 18
DIRECTION_PIN = 19
RELAY_PIN = 17

#Pins setups, output mode and initial state
GPIO.setup(VELOCITY_PIN, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(DIRECTION_PIN, GPIO.OUT, initial = GPIO.LOW)
GPIO.setup(RELAY_PIN, GPIO.OUT, initial = GPIO.LOW)

#Associate PWM pulse with velocity pin and insert velocity, in hertz
p=GPIO.PWM(VELOCITY_PIN, 40)

##MOTOR SERVO#########
```

```
#Set wiringpi setmode
wiringpi.wiringPiSetupGpio()

#Pins declaration
SERVO_PIN = 4

#Set pin to output mode
wiringpi.pinMode(SERVO_PIN, 1)

#Create PWM pulse
wiringpi.softPwmCreate(SERVO_PIN, 0, 100)#RANGE-> 0 - 100

##PI CAMERA#############

#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
rawCapture = PiRGBArray(camera)
camera.resolution = (1920, 1080)

##FLASH#####################

#Pins declaration
FLASH_PIN = 5

#Flash pin setup, output mode and initial state
GPIO.setup(FLASH_PIN, GPIO.OUT, initial = GPIO.LOW)

##SENSORS################

#Pins declaration
UP_SENSOR_PIN = 21
DOWN_SENSOR_PIN = 6

#Sensor pins setup, output mode and initial state
GPIO.setup(UP_SENSOR_PIN, GPIO.IN)
GPIO.setup(DOWN_SENSOR_PIN, GPIO.IN)

############# END SETUP ####################
########## FUNCTIONS ########################

#Function that rotates the servo motor in order to push the top slide
#and then retractes until it reaches the inicial position
def pushSlide():
    wiringpi.softPwmWrite(SERVO_PIN, 11)
    wiringpi.delay(2000)
    wiringpi.softPwmWrite(SERVO_PIN, 19)
    wiringpi.delay(2000)

#Function that captures the slide image, turning on the flash leds
previously
#and off afterwards
def takePicture():
    GPIO.output(FLASH_PIN, GPIO.HIGH)
    time.sleep(0.5)
      rawCapture = PiRGBArray(camera)
    camera.capture(rawCapture, format="bgr")
      time.sleep(0.5)
      GPIO.output(FLASH_PIN, GPIO.LOW)
    return rawCapture.array
```

```
#Function that processes the input image by bounding the tag area and
#binarize its contents
def imageProcessing(image):
    mask = np.zeros(image.shape[:2], dtype="uint8")
    (cX, cY) = (image.shape[1] // 2, image.shape[0] // 2)
    cv2.rectangle(mask, (cX-230, cY-200), (cX + 110, cY-55), 255, -1)
    image = cv2.bitwise_and(image, image, mask=mask)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.GaussianBlur(image, (3,3), 0)
    image = cv2.adaptiveThreshold(image, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 17, 7)
    return image

#Function that converts the image to pil format and applies the ocr
software
def OCR(image):
    pil_image  = Image.fromarray(image)
    return pytesseract.image_to_string(pil_image)

#Function that saves the data in the file
def saveData(file, text):
      try:
            file.write(text + "\n\n")
      except UnicodeEncodeError:
            file.write("This slide could not be recognized" + "\n\n")

################### END FUNCTIONS ######################
############ OPERATION ##########################################

#Open file in append mode
file = open("database.txt", "a")

#Input of how many slides will be processed
n_slides=int(input("numero de laminas (max: 29): "))

#Number of slides condition
if n_slides < 1 or n_slides>29:
    print("Numero invalido de laminas!")
    sys.exit(0)

#Turn On Relay
GPIO.output(RELAY_PIN, GPIO.HIGH)

#Set direction pin to HIGH
GPIO.output(DIRECTION_PIN, GPIO.HIGH)

#Retracts the servo motor
wiringpi.softPwmWrite(SERVO_PIN, 19)

#Start PWM pulse with a 50% duty cycle
p.start(50)

#Wait until the upper sensor is activated
while GPIO.input(UP_SENSOR_PIN) == False:
  pass

#Change PWM duty cycle to 0
p.ChangeDutyCycle(0)
```

```
#Put the slides inside the holder and press ENTER
#to continue
raw_input("Coloque as laminas no suporte e pressione ENTER para conti-
nuar...")

#Change direction pin to LOW
GPIO.output(DIRECTION_PIN, GPIO.LOW)

#Wait 1 second
time.sleep(1)

#Change PWM duty cycle to 50
p.ChangeDutyCycle(50)

#Wait until the lower sensor is activated
while GPIO.input(DOWN_SENSOR_PIN) == False:
  pass

#Change PWM duty cycle to 0
p.ChangeDutyCycle(0)

#Wait 1 second
time.sleep(1)

#Change direction pin to HIGH
GPIO.output(DIRECTION_PIN, GPIO.HIGH)

#Change PWM frequency to 4
p.ChangeFrequency(4)

#Wait 1 second
time.sleep(1)

#Change PWM duty cycle to 50
p.ChangeDutyCycle(50)

#Wait 10 seconds for each slide under the maximum number
time.sleep(10 * (29-n_slides))

#Change PWM duty cycle to 50
p.ChangeDutyCycle(0)

#Procedure cycle in which each slide represents
#an iteration
for x in range(0, n_slides-1):
    image = takePicture()
    image = imageProcessing(image)
    text = OCR(image)
    saveData(file, text)
    pushSlide()
    time.sleep(1)
    p.ChangeDutyCycle(50)
    time.sleep(10)
    p.ChangeDutyCycle(0)

image = takePicture()
image = imageProcessing(image)
text = OCR(image)
```

```
saveData(file, text)
pushSlide()

#Stop PWM pulse
p.stop()

#Close file
f.close()

#Turn off relay
GPIO.output(RELAY_PIN, GPIO.LOW)
############# END OPERATION ####################
######### CLEANING ###############

#Clean GPIO pins
GPIO.cleanup()
```