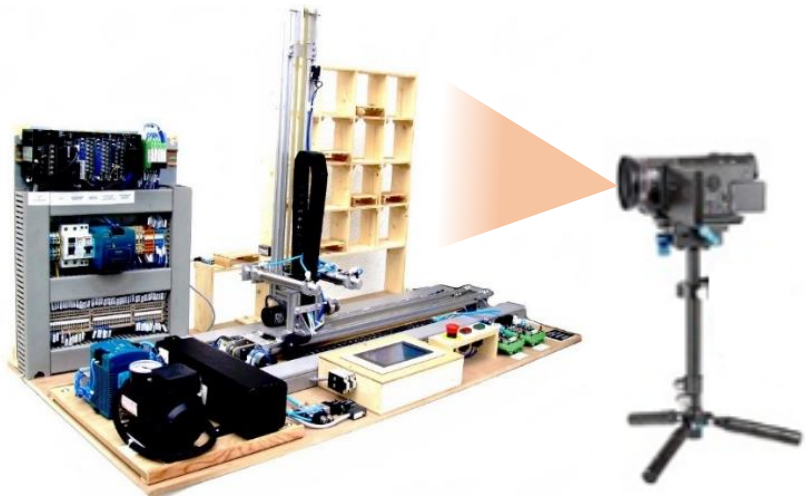
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrotécnica Energia e Automação



PROCESSAMENTO DE IMAGEM NUM SIMULADOR DE ARMAZENAMENTO AUTOMÁTICO

DANIEL JORGE RIBEIRO CARVALHO

(Licenciado em Engenharia Eletrotécnica)

Dissertação para a obtenção do grau de Mestre em
Engenharia Eletrotécnica – Ramo de Automação e Eletrónica Industrial

Orientadores: Professor Doutor Armando José Leitão Cordeiro
Professora Doutora Maria da Graça Vieira de Brito Almeida

Júri:

Presidente: Professor Doutor Luis Manuel dos Santos Redondo
Vogais: Professor Doutor Fernando Manuel Fernandes Melício
Professor Doutor Armando José Leitão Cordeiro

Outubro de 2016

AGRADECIMENTOS

A elaboração deste trabalho só foi possível graças ao apoio de diversas pessoas que, direta ou indiretamente, me acompanharam ao longo deste percurso. Assim, agradeço:

Ao Professor Doutor Armando Cordeiro e à Professora Doutora Maria da Graça Almeida, pela orientação, disponibilidade e espírito crítico, sempre presentes em todas as fases deste trabalho e que muito contribuíram para melhorar o seu conteúdo.

A todos os professores do ISEL pela disponibilidade e ajuda sempre demonstrada.

A todos os que direta ou indiretamente contribuíram para a realização deste trabalho e acompanharam este processo.

À minha esposa, que em todos os momentos a sentia ao meu lado, agradeço com um carinho muito especial a presença, a partilha, a compreensão e o incentivo fundamentais no desenvolvimento deste trabalho.

Por último, tendo consciência que sozinho nada disto teria sido possível, dirijo um agradecimento especial aos meus pais, por serem modelos de coragem, pelo seu apoio incondicional, incentivo, amizade e paciência demonstrada e total ajuda na superação dos obstáculos que ao longo desta caminhada foram surgindo. A eles dedico este trabalho!

RESUMO

Nesta dissertação propõe-se o desenvolvimento de uma solução que permita determinar o estado de ocupação dos alvéolos de um simulador de armazenamento automático com recurso ao processamento de imagem. Trata-se de um simulador já existente no Laboratório de Automação e Robótica do Instituto Superior de Engenharia de Lisboa (ADEEEA) que utilizava apenas as potencialidades do autómato (leitura e escrita de memórias locais) para determinar o estado de ocupação do armazém.

Com a presente dissertação pretende-se adicionar novas funcionalidades ao simulador existente, introduzindo a capacidade de deteção automática em tempo real da presença ou ausência de peças nos respetivos alvéolos.

Esta dissertação teve início com a realização de um estudo preliminar sobre a metodologia a aplicar de modo a determinar o modo mais viável de avaliar o estado de ocupação das várias posições de armazenamento. De seguida procedeu-se à escolha dos meios tecnológicos a empregar para a aplicação da metodologia selecionada, como por exemplo, a câmara digital, o *software* e a unidade de processamento digital. Foi decidido que seriam implementadas duas soluções diferentes. Uma das soluções é baseada num computador pessoal (portátil/secretária) que utiliza, em grande parte, recursos de *software* já consagrados na área da visão computacional, tratando-se de uma solução mais volumosa e com menos portabilidade face ao simulador em causa. A outra solução baseia-se na utilização de um microcomputador *Raspberry Pi,* o qual permite uma redução significativa do tamanho e custo do sistema e acrescenta um elevado grau de portabilidade uma vez que se pretende que a solução adotada seja parte integrante do simulador.

Para além do trabalho de alteração da programação do Controlador Lógico Programável (PLC) existente no simulador por forma a receber as novas funcionalidades e desenvolvimento de *software* para aquisição e tratamento de imagem, foi ainda necessário desenvolver soluções de comunicação digital (*hardware/software*) com o PLC para cada uma das soluções desenvolvidas.

Palavras-chave: Visão Computacional, Processamento e Análise de Imagem, Sistemas de Armazenamento Automático, *Raspberry Pi*, Comunicação RS232.

ABSTRACT

This thesis proposes the development of a solution to determine the occupancy status of the compartments of an automatic storage simulator using image processing. This is an existing simulator at the Automation and Robotics Laboratory of the Instituto Superior de Engenharia de Lisboa (ADEEEA) that used only the automaton capabilities (reading and writing local memories) to determine the warehouse occupancy status.

The present work intends to add new features to the existing simulator, introducing the automatic detection capability in real time of the presence or absence of parts in the respective compartments.

This work began with a preliminary study on the methodology to apply to determine the most viable way to assess the state of occupation of the various storage locations. This was followed by the choice of the technological means to use for applying the selected method, such as the digital camera, the software and the digital processing unit. It was decided that two different solutions would be applied. One of the solutions is based on a personal computer (laptop/desktop) that uses mainly software features already established in the field of computational vision, being a bulkier solution and with less portability when compared to the simulator in question. The other solution is based on the use of a microcomputer *Raspberry Pi*, which allows a significant reduction in size and system cost and adds a high degree of portability since it is intended that the solution adopted is an integral part of the simulator.

In addition to the work of programming modification of the Programmable Logic Controller (PLC) on the simulator in order to receive the new features and software development for image acquisition and treatment, it was also necessary to develop digital communication solutions (hardware/software) with the PLC for each of the developed solutions.

Key Words: Computational Vision, Image Processing and Analysis, Automated Storage Systems, *Raspberry Pi*, RS232 Communication.

“One machine can do the work of a
hundred ordinary man, but no machine
can do the work of one extraordinary man!”
Elbert Hubbard

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS

2D	Bidimensional
3D	Tridimensional
A/D	Analógico-Digital
ARM	*Advanced RISC Machine*
ASCII	*American Standard Code for Information Interchange*
ASIC	*Application Specific Integrated Circuit*
B&W	Preto e Branco (*Black and White*)
CCD	*Charge-Coupled Device*
CMOS	*Complementary metal-oxide-semiconductor*
CR	*Carriage Return*
DC	Corrente Contínua (*Direct Current*)
DLL	Biblioteca de vínculo dinâmico (*Dynamic-link library*)
DSP	*Digital Signal Processor*
EPC	*Eletronic Product Code*
FCS	*Frame Check Sequence*
FIFO	*First-In-First-Out*
FPGA	*Field Programmable Gate Array*
FPGAA	*Field Programmable Gate Analog Array*
FPS	Ecrãs por Segundo (*Frames per Second*)
GPIO	*General Purpose Input/Output*
HDMI	*High-Definition Multimedia Interface*
HMI	Interface Homem-Máquina (*Human-Machine Interface*)
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
LCD	*Liquid Crystal Display*
LIFO	*Last-In-First-Out*
MCU	*Micro-Controller Unit*
MP	*Megapixels*
MPU	*Micro-Processor Unit*
PC	Computador Pessoal (*Personal Computer*)
PCB	Placas de Circuito Impresso (*Printed Circuit Board*)

PCI	*Peripheral Component Interconnect*
PLC	Controladores Lógicos Programáveis (*Programmable Logic Controller*)
RAD	Desenvolvimento Rápido de Aplicações (*Rapid Application Development*)
RFID	*Radio Frequency IDentification*
RGB	Sistema de cores aditivas Vermelho, Verde, Azul (*Red, Green, Blue*)
RIA	*Robotics Industries Association*
ROI	*Region Of Interest*
SCARA	*Selective Compliance Assembly Robot Arm*
SoC	*System on a Chip*
TTL	*Transistor-Transistor Logic*
USB	*Universal Serial Bus*
VGA	*Video Graphics Array*
WLAN	*Wireless Local Area Network*
WMS	*Warehouse Management System*

1. INTRODUÇÃO

No estado atual da economia mundial existe uma elevada concorrência entre as empresas de modo a aumentar a produtividade e a qualidade dos seus produtos ao mais baixo custo possível. Com a constante evolução da tecnologia as empresas têm implementado com sucesso alguns dos fatores atrás referidos, em grande parte devido à aplicação de novas tecnologias ao nível da automatização dos seus processos.

A utilização de sistemas que emulam a visão humana e que, em determinados aspetos a superam, é uma realidade nos dias de hoje. Sistemas dotados de processamento de imagem por computador são cada vez mais aplicados em diversas áreas da indústria, como por exemplo na indústria alimentar, farmacêutica, aeroespacial, aeronáutica, e ainda em muitas áreas da medicina ou do entretenimento.

De facto, a utilização de sistemas com processamento de imagem tem permitido melhorar a vida dos humanos em muitos aspetos e, no caso da indústria, retirar o operador de tarefas repetitivas ou mesmo perigosas em muitos casos. No entanto, convém não esquecer que os sistemas automatizados com funções integradas de processamento de imagem, tal como acontece também com outras tecnologias, carecem muitas vezes de medidas de segurança adicionais uma vez que na ausência das condições ideais de funcionamento, tais como a iluminação inadequada, as vibrações, as poeiras e outros fatores podem condicionar o desempenho da aplicação e trazer riscos acrescidos.

Quando se automatiza uma aplicação com recurso ao processamento de imagem pretende-se que esse sistema simule o mais possível a inteligência e a experiência humana por vezes com uma velocidade de processamento muito mais elevada. Para se alcançar esse objetivo as várias soluções são normalmente baseadas em autómatos programáveis, sensores, câmaras, computadores digitais e sistemas de iluminação, aos quais se juntam técnicas de aquisição e tratamento de imagem por vezes combinadas com algoritmos de inteligência artificial que conseguem atingir um elevado grau de sofisticação. É neste sentido que se pretende desenvolver e aprofundar o tema desta dissertação, ou seja, através da utilização de um sistema automatizado assistido por visão computacional de modo a melhorar o seu desempenho e a trazer-lhe funcionalidades que dificilmente se conseguiriam introduzir com outras soluções por um custo tão reduzido.

1.1. Enquadramento e Motivação

Esta dissertação visa a obtenção do grau de Mestre em Engenharia Eletrotécnica – Ramo Automação Industrial e o tema em questão encontra-se claramente enquadrado na área do curso, especialmente nas vertentes das disciplinas de Automação e da Robótica, onde os conhecimentos aí adquiridos foram essenciais para a elaboração desta dissertação. Por um lado, um sistema de armazenamento automático de peças com um PLC (*Programmable Logic Controller*), sensores, cilindros pneumáticos, ar comprimido, acionadores eletromecânicos, comunicações digitais, entre outros sistemas que fazem parte habitual de um sistema automatizado. Por outro lado, a aquisição, o tratamento e o processamento de imagem com a aquisição e tratamento que fazem parte integrante do conteúdo da disciplina da Robótica. Outras áreas como a eletrónica de potência e os sistemas digitais estão também naturalmente presentes.

A opção por este tema, para a realização da dissertação, deve-se também em grande parte a uma preferência pessoal por estas áreas da engenharia eletrotécnica, aliada ao gosto pelo desenvolvimento de *software* de aplicação e supervisão e finalmente por esta se encontrar relacionada com objetivos pretendidos para a vida profissional.

1.2. Objetivos

O principal objetivo desta dissertação consiste em desenvolver uma solução que introduza uma melhoria significativa no funcionamento de um simulador de armazenamento existente no Laboratório de Automação e Robótica do ISEL (ADEEEA), introduzindo a capacidade de deteção automática e em tempo real da presença ou ausência de peças nos respetivos alvéolos, utilizando técnicas de processamento de imagem.
Trata-se de um simulador de pequenas dimensões que apresenta a desvantagem de estarem acessíveis e poderem ser removidas ou inseridas manualmente peças num conjunto de 16 alvéolos. Dada a limitação do número de interfaces de entradas e de saídas binárias disponíveis no PLC não existe a possibilidade de levar esta informação ao controlador, originando um funcionamento desadequado se houver intervenção humana de forma indevida. A solução existente baseia-se unicamente no preenchimento de uma área de memória do PLC após a inserção ou remoção automática de peças.

Após o desenvolvimento da solução proposta nesta dissertação, com recurso ao processamento de imagem, será possível atualizar constantemente e em tempo real a área de memória do PLC com a localização exata de cada peça após intervenção humana.

Para cumprir o objetivo principal desta dissertação foi necessário explorar e desenvolver alguns objetivos parciais, tais como:

- Modificação do programa do PLC de modo a efetuar as modificações necessárias e integrar o sistema de processamento de imagem;
- Compreensão e programação do microcomputador *Raspberry Pi*;
- Captação de imagem com o auxílio de uma câmara digital;
- Análise e processamento de imagem com recurso a unidades de processamento digital apropriadas para o efeito (foram adotadas duas soluções diferentes);
- Estudo e comparação de resultados de diferentes algoritmos de processamento de imagem;
- Estudo e programação de protocolos de comunicação digital para interação entre as unidades de processamento digital e o PLC.

Note-se que a opção por efetuar a análise e processamento de imagem por duas soluções diferentes resultou de uma estratégia adotada pelos orientadores e pelo aluno de mestrado, uma vez que a última solução é baseada num dispositivo relativamente recente e do qual não existiam certezas quanto ao seu desempenho em aplicações semelhantes. Assim, como solução de recurso existiria sempre uma outra opção que viabilizaria a realização desta dissertação.

1.3. Estrutura da dissertação

1.3.1. Organização geral

Esta dissertação encontra-se organizada em seis capítulos, todos eles divididos em diversos subcapítulos. Os capítulos que constituem esta dissertação são os seguintes: Capítulo 1 – Introdução; Capítulo 2 – Estado da Arte; Capítulo 3 – Conceitos Gerais – Visão Computacional; Capítulo 4 – Soluções Desenvolvidas; Capítulo 5 – Resultados Experimentais – Casos de Estudo; Capítulo 6 – Conclusões.

O capítulo 1 inicia-se com uma introdução ao tema da dissertação, realçando a sua importância e enquadramento na área da engenharia em que se insere. Definem-se de seguida, os principais objetivos a alcançar, a apresentação da organização geral do documento da dissertação e as convenções adotadas.

No capítulo 2 apresenta-se uma síntese do estado da arte relativa ao tipo de sistemas e tecnologias abordadas neste trabalho.

No capítulo 3 apresenta-se uma breve introdução à visão computacional e alguns conceitos gerais relativos ao processamento de imagem.

Em relação ao capítulo 4, inicia-se com uma abordagem relativa ao sistema de armazenamento automático já existente, descrevendo o seu funcionamento, o equipamento que incorpora bem como as suas limitações. De seguida, este contemplará as soluções desenvolvidas para a aquisição, tratamento e processamento de imagem de acordo com os objectivos estabelecidos.

No capítulo 5 apresentam-se os resultados experimentais com ambas as soluções desenvolvidas. Por fim, no capítulo 6 apresentam-se as conclusões gerais de todo o trabalho desenvolvido. Nessas conclusões insere-se alguns aspetos da presente dissertação que são considerados inovadores e deixam-se indicações para futuros desenvolvimentos no domínio das unidades de processamento digital.

1.3.2. Convenções

Com o objetivo de facilitar a leitura e compreensão deste documento foram utilizadas convenções que se descrevem de seguida:

- Sempre que exista correspondência entre as grandezas presentes nas equações utilizam-se unidades do Sistema Internacional (S.I.) de unidades de medida. Nos múltiplos e submúltiplos dessas unidades as respetivas abreviaturas;
- Recorreu-se, sempre que possível, a conceitos presentes na Língua Portuguesa. Em determinados casos os conceitos surgem acompanhados pela designação na Língua Inglesa, colocada entre parêntesis e em itálico, para que a compreensão do seu significado seja mais fácil e menos ambígua.

2. ESTADO DA ARTE

2.1. Introdução

Neste capítulo é apresentado o estado da arte no que diz respeito às principais áreas técnicas e científicas envolvidas neste trabalho.

Inicialmente desenvolve-se a temática da automação e da robótica, sendo abordado o seu surgimento e a sua necessidade na vertente industrial. De seguida apresentam-se os vários tipos de robôs utilizados na vertente industrial, dando ênfase à anatomia e às suas configurações típicas. Posteriormente, faz-se uma abordagem relativamente aos sistemas embebidos que habitualmente se encontram num robô, onde se especificam os controladores, sensores e atuadores usualmente utilizados neste tipo de sistemas.

No âmbito do presente trabalho serão apresentadas algumas soluções atualmente disponíveis no mercado para sistemas de armazenamento automático (métodos de armazenamento e tecnologias de informação e comunicação) e ainda sistemas assistidos por visão computacional (técnicas aplicadas à visão computacional).

2.2. Automação e Robótica Industrial

Após a segunda metade do século XX, o mundo iniciou uma fase de profundas transformações na área da tecnologia, desencadeada principalmente pelos enormes avanços no conhecimento científico em resultado da Segunda Guerra Mundial. Atualmente somos uma sociedade verdadeiramente tecnológica em quase todas as áreas, desde o entretenimento e a cultura, a educação, a medicina, entre outras, de tal forma que já não nos é possível viver sem ela. Estas evoluções tecnológicas permitiram melhorar muitas tarefas e aplicações na indústria em quase todas as etapas do processo produtivo, tendo-se atingido um nível de automatização e produtividade elevados.

A **automação industrial** integra diversas técnicas e tecnologias visando o fabrico sem a intervenção direta do Homem. Integra sistemas mecânicos, elétricos e eletrónicos incluindo arquiteturas computacionais (*hardware* e *software*), aplicados à operação e controlo do processo produtivo [1] [2].

Os primeiros dispositivos automáticos utilizados na produção industrial datam de há mais de 100 anos. Até à década de 70 do século passado, a maior parte dos automatismos industriais eram essencialmente desenvolvidos com recurso a equipamentos mecânicos projetados e construídos especificamente para realizarem determinadas tarefas. A sequência de operações era fixa e determinada pela configuração do equipamento, não sendo, portanto, fácil acomodar alterações das tarefas ou do produto. A maior parte dos dispositivos eletromecânicos eram usados como elementos lógicos numa rede de condutores elétricos (lógica cablada), densamente interligada. Estes tipos de sistemas possuíam inúmeros problemas associados, sendo alguns deles [2]:

- Tamanho físico da sala de controlo e comprimento das cablagens;
- Complexidade para efetuar a manutenção e detetar falhas;
- Falta de flexibilidade.

As restrições apresentadas pelos controladores eletromecânicos conduziram a que, na década de 60, a *General Motors* iniciasse as primeiras investigações no sentido de encontrar soluções alternativas. Uma década depois, ou seja, na década de 70, em *Bedford*, *Massachusetts*, *Richard Morley* liderou um grupo de engenheiros que apresentou o primeiro autómato programável industrial (PLC) como introdução à lógica programada, que veio gradualmente substituir a lógica cablada até então utilizada [3].

A Figura 2.1 ilustra o aspeto físico do primeiro PLC (modelo 084 da *Modicon*) juntamente dos engenheiros que o desenvolveram.

Figura 2.1 – Primeiro PLC modelo 084 da Modicon [3].

Desta forma, o PLC trouxe inúmeras vantagens para os automatismos, destacando-se as seguintes:

- Eliminaram a necessidade de substituir e juntar *hardware* para cada nova configuração lógica;
- Este novo sistema incrementou drasticamente as funcionalidades e reduziu o espaço de colocação do sistema lógico.

De um modo geral, o PLC é um controlador do estado sólido que analisa em permanência o estado dos equipamentos ligados às suas entradas. Baseado no algoritmo que foi desenvolvido no processador e armazenado na memória, este controla o estado dos sistemas ligados às suas saídas [4].

A constante procura em desenvolver a automação fez com que se evoluísse no sentido de reproduzir a destreza manual do Homem. É neste sentido que a **robótica** é inserida num âmbito industrial. Embora tenha origem na ficção científica, só na segunda metade do século passado é que através da concretização de vários projetos, a robótica se estabeleceu como uma tecnologia suportada em manipuladores mecânicos altamente automatizados, a denominada Robótica Industrial [5].

A definição “oficial” de um Robô Industrial é fornecida pela Associação das Indústrias de Robótica (*Robotics Industries Association* – RIA): considera-se que um robô é um dispositivo mecânico articulado e reprogramável que consegue de forma autónoma, recorrendo à sua capacidade de processamento, obter informação do meio envolvente através de sensores e realizar acções com base nessa informação, manipulando objetos com recurso a atuadores [6].
Em 1955 surgem os primeiros protótipos e em 1961 surge o primeiro Robô Industrial desenvolvido pela *Unimate* para uma aplicação num processo de enformação de peças por fundição. Este tipo de robô, frequentemente designado por manipulador, que apenas realizava tarefas rotineiras, rapidamente evoluiu no sentido de adquirir crescentes capacidades de adaptação a novas circunstâncias aproximando-se cada vez mais das tarefas realizadas pelo Homem. Indubitavelmente os robôs são atualmente equipamentos de excelência na automação com elevada flexibilidade, constituindo a robótica um dos domínios onde se concentram grandes esforços de investigação. A sua utilização massiva

na produção industrial constitui um fator de competitividade, quer em termos de redução de custos, quer em termos de qualidade, proporcionando também o bem-estar para o Homem, libertando-o das tarefas rotineiras, pesadas e poluentes, tais como [2]:

- Transporte de peças pesadas;
- Processos de soldadura;
- Inspeção visual do produto.

A Figura 2.2 ilustra o primeiro robô industrial desenvolvido pela *Unimate* em 1961.

Figura 2.2 - Primeiro Robô (desenvolvido pela Unimate em 1961) [7].

Em resumo, a automação e a robótica industrial têm como objetivo tornar os processos [1]:

- mais seguros;
- mais rápidos;
- mais flexíveis;
- mais eficazes quanto ao controlo de qualidade do produto final;
- mais eficazes quanto à gestão da informação (dados) do processo;
- com menores custos de produção e de gestão;
- menos exigentes fisicamente.

2.3. Tecnologia dos Robôs Industriais

Um robô é essencialmente constituído por duas partes, o robô propriamente dito e o controlador, que contém toda a componente eletrónica para o controlo do robô, bem como os acessórios para interface com o utilizador. Nos casos em que o robô utiliza atuadores pneumáticos ou hidráulicos há ainda necessidade de ligações às redes de ar comprimido ou de óleo pressurizado da fábrica, ou, em alternativa, instalar compressores específicos junto do robô [2].

2.3.1. Anatomia do Robô Industrial

Em termos anatómicos o robô define-se pela configuração física do **corpo**, **braço** e **punho**. A maioria dos robôs industriais são montados numa base fixa ao piso. O corpo está ligado à base e o braço ao corpo, na extremidade do braço está o punho, ligado ao punho está uma "mão" ou manipulador (órgão terminal) [5].

Os movimentos relativos entre estes componentes são assegurados por **juntas** que podem ter movimentos rotativos ou deslizantes conforme se observa na Figura 2.3. A unir as diversas juntas estão os **elos**, membros rígidos.

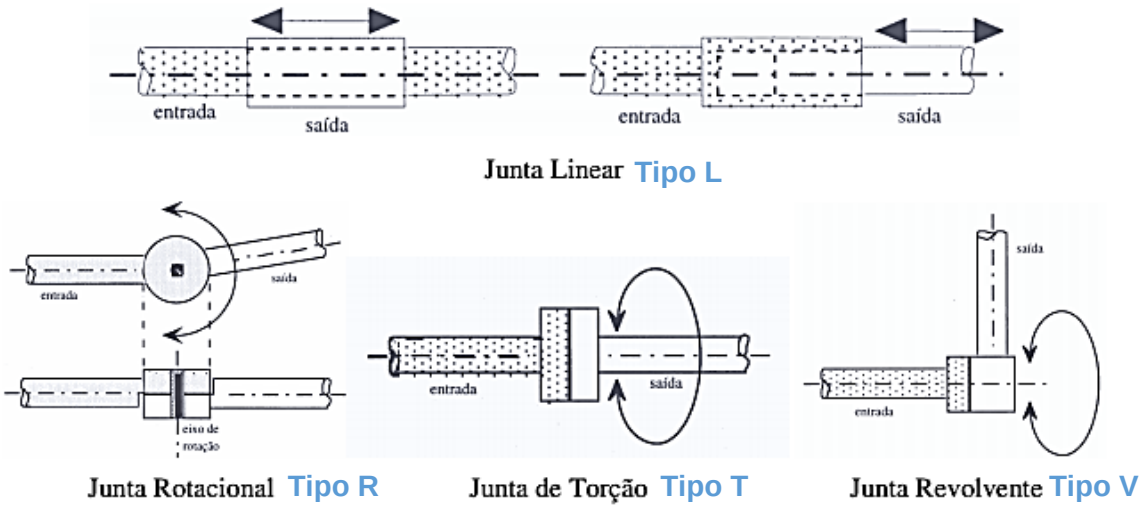


Figura 2.3 - Tipos de juntas usadas em robôs [5].

2.3.2. Configurações típicas do Robô Industrial

Os robôs industriais apresentam-se numa grande variedade de formas, configurações e tamanhos, mas podem na maioria das vezes ser classificados num dos cinco tipos de configurações base: cartesiana, cilíndrica, polar, articulada e SCARA (*Selective Compliance Assembly Robot Arm)* [2].
Associada a cada configuração existe um volume de trabalho, que é o espaço dentro do qual o robô pode movimentar a extremidade do punho. A convenção de usar a extremidade do punho para definir o volume de trabalho procura evitar a confusão que se estabeleceria na sua definição com órgãos terminais de tamanhos diferentes que poderiam ser ligados ao punho. Mais se justifica a sua exclusão por serem elementos opcionais ao robô base [5].

Os cinco tipos de configuração são:

- **Configuração cartesianas:** os seus movimentos assentam em três juntas lineares conforme se ilustra na Figura 2.4. Este tipo de configuração proporciona movimentos lineares segundo três eixos (*x, y, z*) [2].

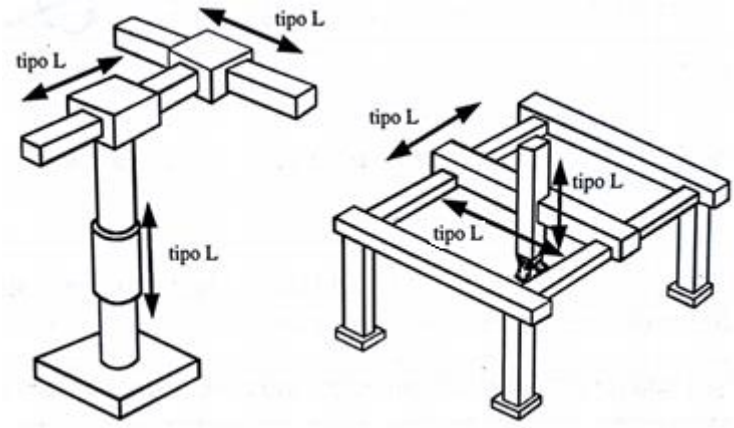


Figura 2.4 - Configuração cartesiana [2].

O volume de trabalho está confinado a um paralelepípedo como se demonstra na Figura 2.5 [2].

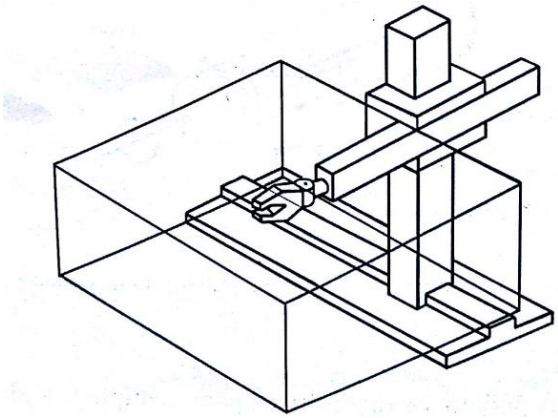


Figura 2.5 - Volume de trabalho de um robô cartesiano [2].

- **Configuração cilíndrica:** combina os movimentos lineares na vertical e na horizontal, como o movimento de rotação no plano horizontal em torno de um eixo vertical. O volume de trabalho é um cilindro parcial com a altura definida pelo curso segundo o eixo vertical, com a espessura definida pelo eixo horizontal e com um arco definido pela excursão angular do movimento de rotação (ver Figura 2.6) [2].

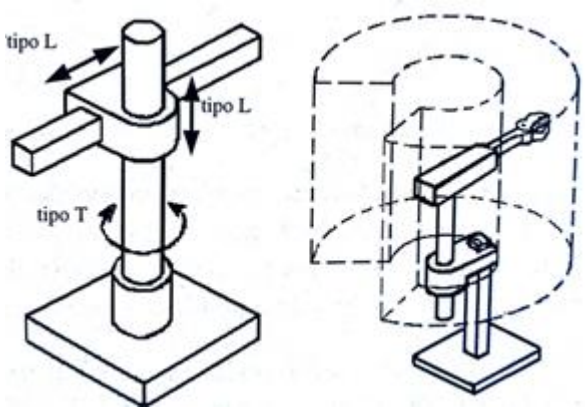


Figura 2.6 - Configuração cilíndrica e respetivo volume de trabalho [2].

- **Configuração polar:** usa um braço telescópico (elo linear) que se pode levantar ou baixar através de uma junta rotativa, montada numa base também rotativa. O volume de trabalho é constituído por parte de uma esfera delimitada pelos cursos das juntas “T” e “R”, e com a espessura definida pelo curso da junta linear (ver Figura 2.7) ([5], [2]).

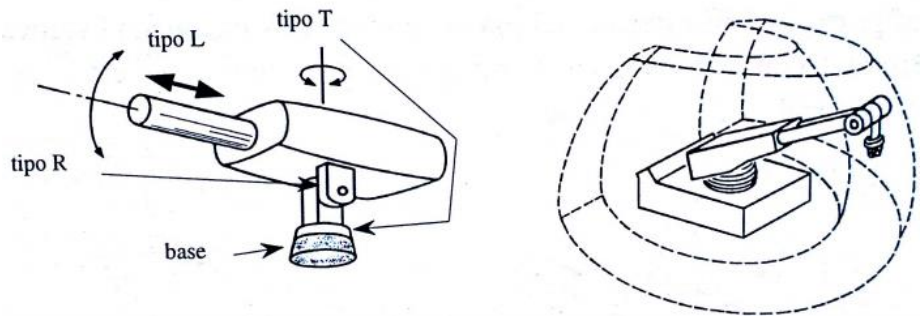


Figura 2.7 - Configuração polar e respetivo volume de trabalho [2].

- **Configuração articulada:** é composto por dois elos retilíneos correspondentes ao antebraço e ao braço humano montados num pedestal vertical. Estes elos estão ligados por juntas rotativas correspondentes ao ombro e ao cotovelo. Um punho está ligado à extremidade do antebraço com as respetivas juntas. O volume de trabalho é constituído por parte de uma esfera com espessura variando de forma não regular (ver Figura 2.8).

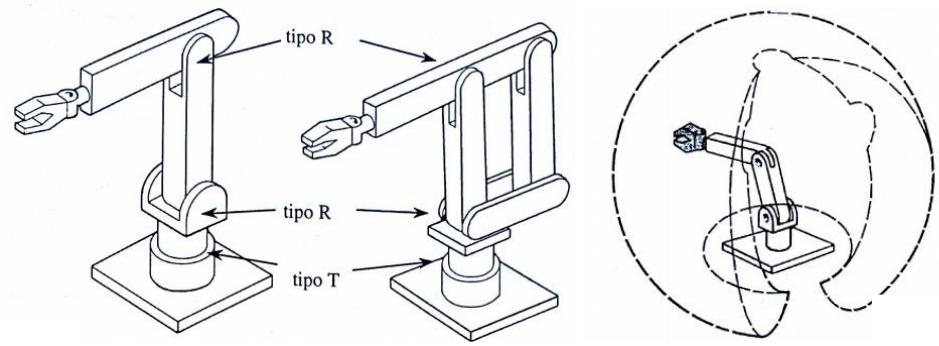


Figura 2.8 - Configuração articulada e respetivo volume de trabalho [2].

- **Configuração SCARA:** é uma combinação das configurações cilíndrica e articulada. A base de eixo vertical suporta uma junta revolvente (tipo V) com o braço de saída na horizontal. Segue-se uma junta rotacional (tipo R) de eixo vertical com o braço de saída suportando uma junta linear (tipo L). As juntas angulares garantem o posicionamento no plano *X*, *Y* enquanto a junta linear permite o posicionamento na vertical. O volume de trabalho é cilíndrico e é limitado pelo curso das diversas juntas (ver Figura 2.9) [2].

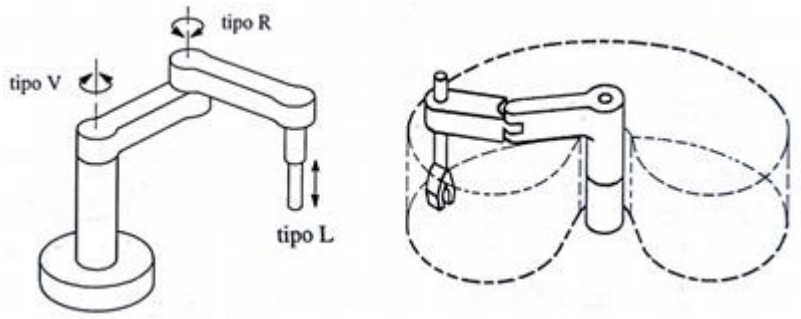


Figura 2.9 - Configuração SCARA e respetivo volume de trabalho [2].

Conforme se pode constatar na descrição da anatomia dos robôs, aos movimentos do braço e do corpo estão associadas três juntas a que correspondem três graus de liberdade. Aos movimentos do punho podem estar associadas duas ou três juntas, em que [5]:

- **Eixo de "*Roll*" (Rotacional):** rotação do punho em torno do eixo do braço do robô;
- **Eixo de "*Pitch*" (Inclinação):** orientação vertical ou inclinação do punho, se o eixo rotacional estiver na posição central corresponde à rotação do punho para cima e para baixo;
- **Eixo de "*Yawl*" (Orientação):** se o eixo rotacional estiver na sua posição central corresponde à rotação do punho para a esquerda e para a direita.

A Figura 2.10 ilustra os graus de liberdade correspondentes a uma configuração típica de um punho com três juntas.

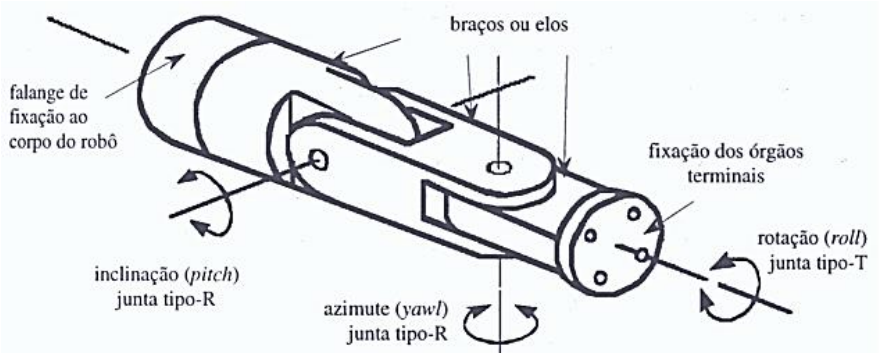


Figura 2.10 - Três graus de liberdade num punho de robô [5].

O robô com as diversas juntas e órgão terminal constitui um sistema de alguma complexidade requerendo uma unidade de coordenação e controlo para realizar os movimentos e as operações requeridas. Estas funções são desempenhadas pelos sistemas embebidos que se encontram acoplados aos robôs [2].

2.4. Sistemas Embebidos

Um sistema embebido é um sistema eletrónico dedicado que é utilizado em diferentes aplicações como telemóveis, robôs, veículos terrestres, sistemas aeroespaciais, entre outros. O sistema embebido pode conter a funcionalidade completa do sistema de diferentes formas - através de *software* executado em processadores ou em maquinaria especializada. Em complemento ao *software* e *hardware*, o desenho de sistemas embebidos pode exigir o desenvolvimento de *software* dependente de *hardware* ou de forma inversa. De forma a ir de encontro a requisitos extremamente exigentes, em termos de fiabilidade e segurança, verificação e validação, desempenho, limitações de consumo energético, dissipação de potência e interligação com outros sistemas, incluindo o "mundo real" (através de sensores e atuadores), estes interfaces *hardware-software* necessitam de ser projetados conjuntamente [8].

A Figura 2.11 ilustra um exemplo da constituição de um sistema embebido. Este é constituído por dois blocos principais, em que no bloco *Computing Platform* (Plataforma Computacional) se encontra fundamentalmente a unidade central de processamento (CPU) responsável por executar o programa e a memória responsável por alocar os dados do programa. No bloco de *I/O Front-End* encontra-se toda a componente que permite a ligação e adaptação de sinais com o exterior (sensores e atuadores). Num sistema embebido é ainda normalmente disponibilizado uma interface com o utilizador (*Human Machine Interface* – HMI).

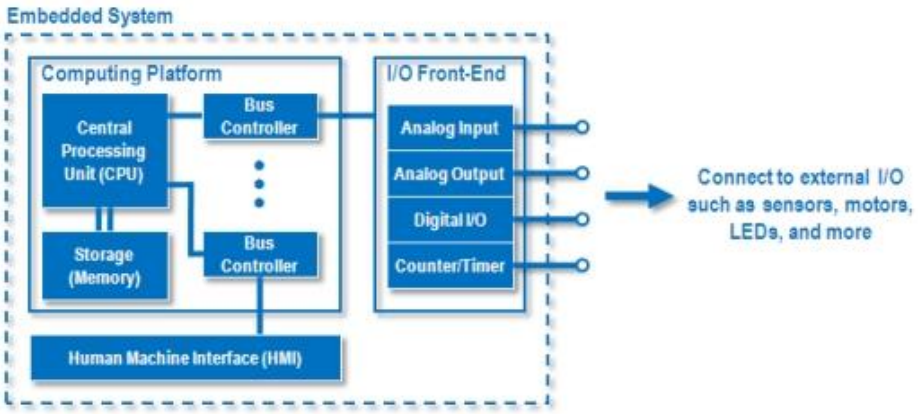


Figura 2.11 – Exemplo da constituição de um sistema embebido [9].

Na implementação destes sistemas muitas vezes discute-se acerca da possibilidade de executar determinadas tarefas por eletrónica dedicada (*hardware*) ou por *software* [10]. Esta é uma decisão tecnológica importante que acarreta consequências a nível dos tempos de resposta que o sistema terá com implicações ao nível do tempo real. Realizar tarefas por *software*, por exemplo, dentro de um microcontrolador (MCU), pode ser uma solução flexível e de baixo custo. A alternativa é a implementação do sistema em *hardware* dedicado que permitirá tempos de processamento muito mais reduzidos. A nível de *hardware* dedicado, as principais soluções tecnológicas são [11]:

- **FPGA – *Field Programmable Gate Array*:** sistema digital baseado em ligações e blocos reconfiguráveis;
- **FPGAA – *Field Programmable Gate Analog Array*:** análogo ao anterior, mas permitindo blocos analógicos;

- **ASIC – *Application Specific Integrated Circuit*:** integrado (não programável) projetado especificamente para a aplicação.

As soluções que utilizam sistemas de processamento via *software* são [11]:

- **MCU – *Micro-Controller Unit*:** processador com periféricos destinados a controlo em tempo real e blocos programáveis;
- **MPU – *Micro-Processor Unit*:** processador destinado a processamento genérico de dados;
- **DSP – *Digital Signal Processor*:** processador que geralmente contem barramento de dados e de endereços de largura diferente, destinado a processamento de dados intensivo contendo alguns periféricos programáveis.

Existe ainda a possibilidade de integrar o *hardware* e *software* [11]:

- **SoC** – ***System on a Chip*:** pode ser constituído por uma FPGA e um MCU/DSP.

Exemplo de um SoC é o *Raspberry Pi*, em que é um dispositivo que contém toda a eletrónica necessária com o intuito de desempenhar as funcionalidades de um computador num único chip. Desta forma, em vez de se possuir um chip individual para o CPU, GPU (*Graphics Processing Unit*), controlador de USB (*Universal Serial Bus*), RAM (*Random Access Memory*), entre outros, está tudo contido num pequeno e único chip (ver Figura 2.12).

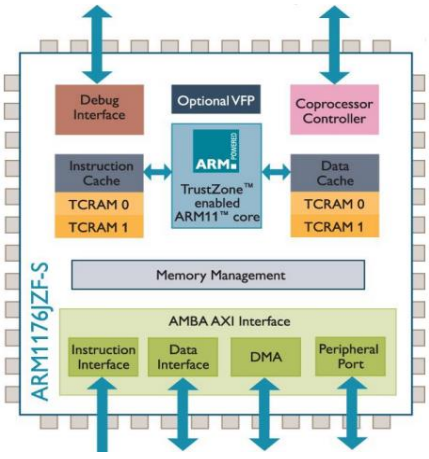


Figura 2.12 – Possível arquitetura de um sistema SoC [12].

2.5. Sensores

Os sistemas autónomos caraterizam-se pela capacidade de realizarem tarefas de forma autónoma num ambiente parcial ou totalmente desconhecido. Para tal, este deve ser dotado com um sistema sensorial que o permita obter informação do meio envolvente, para depois extrair os seus aspetos mais significativos e reagir convenientemente a eventos que por vezes são imprevistos. A qualidade da reação do sistema é proporcional à qualidade e precisão da informação que lhe é fornecida pelos sensores. Assim, raramente um sistema autónomo é equipado com apenas um tipo de sensor. Idealmente, o sistema teria capacidades semelhantes ao ser humano, recebendo várias informações do mundo exterior através de todos os seus sensores, podendo as suas reações se basear num em específico ou na conjugação de vários (técnicas de fusão sensorial) ([2], [13]).
Como exemplo da aplicação de técnicas de fusão sensorial seria o caso do mapeamento em três dimensões (3D) de um objeto a partir de imagens recolhidas por duas câmaras de vídeo. Outro exemplo seria o mesmo mapeamento 3D, conseguido através da combinação de sensores de distância com uma câmara de vídeo [2].

Classificação de sensores

Os sensores podem ser classificados de diversas formas, normalmente em função da sua alimentação, em função da forma como interagem com o meio envolvente, e ainda em relação às grandezas que medem. Note-se que dentro destas classificações podem ainda variar em relação ao seu princípio de funcionamento.

Em relação à sua alimentação, os sensores são normalmente classificados por [13]:

- **Sensores passivos:** são sensores que não possuem alimentação própria e dependem de grandezas exteriores como a temperatura ou a vibração que provocam alterações na sua resistência ou impedância interna as quais são medidas com o auxílio de fontes de alimentação externas ou outros equipamentos;
- **Sensores ativos:** são sensores que não possuem uma fonte de alimentação própria para gerarem os sinais necessários de forma a interagirem com o meio envolvente. Possuem um desempenho muito mais elevado relativamente aos

sensores passivos, contudo, são bastante suscetíveis a interferências. Exemplo destes sensores são os *encoders*, sensores ultrassónicos ou lasers.

Em relação à forma como interagem com o meio envolvente, os sensores são normalmente classificados por [13]:

- **Sensores propriocetivos:** medem o estado interno do sistema, como a velocidade ou a tensão de alimentação (tipicamente grandezas elétricas ou mecânicas);
- **Sensores exterocetivo:** adquirem informações do meio envolvente tais como distâncias, intensidade de luz ou amplitude de som.

Em relação às grandezas que medem, as possibilidades são muitas e apresentam-se alguns exemplos dos tipos de sensores mais comuns [13]:

- **Sensores de posição:** medem o posicionamento de determinados objectos, seja linearmente ou de forma angular (ex.: *Encoder* ou *Resolver*);
- **Sensores de velocidade:** medem a velocidade de deslocamento de determinados objectos, seja linearmente ou de forma angular. Exemplo disso é o caso da taquigeradora ou mesmo do *encoder* que pode também realizar a tarefa de medir a velocidade;
- **Sensores de proximidade:** os sensores de proximidade detetam, sem qualquer contato físico, a presença de um objeto (ex.: fotoeléctricos, indutivos e capacitivos) e baseiam-se normalmente em dois princípios de funcionamento [2]:

    - Modificação de um sinal emitido por perturbação introduzida pelo objeto;
    - Modificação de um sinal emitido por interposição do objeto, impedindo que ele chegue a um recetor, ou por reflexão do sinal para um recetor.

- **Sensores de distância:** os sensores de distância são dispositivos vocacionados para medirem a distância a objetos, enquanto os sensores de proximidade estão mais vocacionados para a deteção de objetos numa vizinhança mais próxima. Frequentemente a informação gerada pelos sensores de distância é utilizada em conjugação com a visão, permitindo acrescentar a terceira dimensão (3D), isto é, o

relevo da superfície, descrita a duas dimensões (2D) pela imagem [2]. Exemplos disso são os sensores de radar, ultrassons ou de raios *Laser*;

- **Sensores de visão:** os sensores de visão ou de imagem são os elementos fundamentais para captação de imagens numa câmara. Estes analogamente agem como a retina do olho humano, em que capta a luminosidade das imagens que são projetadas sobre o sensor. Atualmente existem duas tecnologias para a criação de sensores de visão, a *Charge-Coupled Device* (CCD) e a *Complementary Metal Oxide Semiconductor* (CMOS) [14].

Exemplos de sensores

Apresentam-se seguidamente alguns destes sensores referidos anteriormente.

O ***encoder*** é um dos dispositivos mais utilizado para a determinação do posicionamento angular de diversos objectos, tais como braços de robôs ou veios de máquinas eléctricas, fornecendo muitas vezes informações essenciais ao controlo dos equipamentos. Um *encoder* é basicamente um gerador de impulsos baseado na interação de um feixe de luz com um disco perfurado, que converte um movimento rotativo ou linear numa quantidade específica de impulsos elétricos de onda quadrada [13]. A Figura 2.13 ilustra a forma como é detetado o sentido de rotação de um eixo rotativo utilizando um *encoder* incremental.

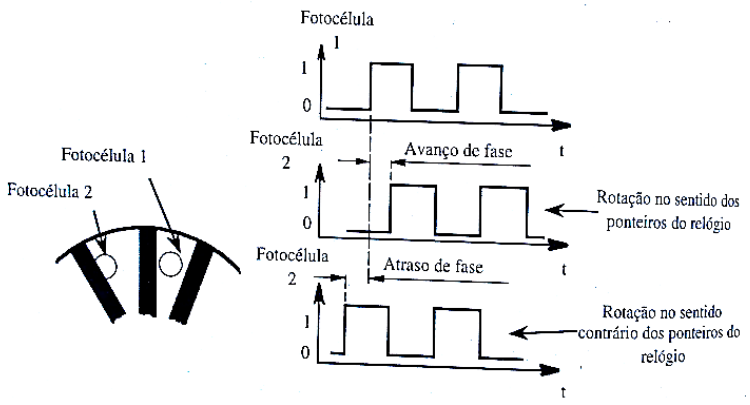


Figura 2.13 - Deteção de sentido de rotação utilizando um encoder [2].

Este tipo de *encoder* gera um trem de impulsos que precisam de ser contados para se obter a posição. Utilizando um segundo par fonte de luz e detetor gerando uma onda desfasada de 90º do par base, é possível obter-se informação sobre o sentido de rotação do disco [13]. O número de segmentos transparentes e escuros no disco determina a resolução do dispositivo, e aumentando o número de linhas do padrão aumenta a resolução por grau de rotação [15]. A Figura 2.14 ilustra o aspeto físico de um *encoder* incremental. Existe também a variante de *encoder* absoluto usando o mesmo princípio de funcionamento ou ainda o *Resolver* com um princípio de funcionamento diferente.

Figura 2.14 - Aspeto físico de um encoder [16].

Os **sensores fotoeléctricos** são constituídos por uma fonte emissora de luz e por um fotodetetor com as respetivas lentes (ver Figura 2.15). O emissor e o recetor estão afastados alguns milímetros e simetricamente orientados para o mesmo ponto focal de modo a que o recetor capte a reflexão do sinal luminoso. É um tipo de sensor muito simples e de baixo custo, contudo, a sujidade das lentes e a influência da luz ambiente são algumas das suas desvantagens [2].

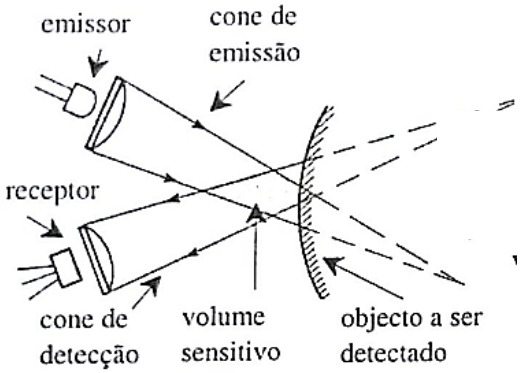


Figura 2.15 - Princípio de funcionamento de um sensor ótico [2].

A Figura 2.16 ilustra o aspeto físico de um sensor ótico de aplicação industrial.

Figura 2.16 - Aspeto físico de um sensor ótico [17].

Os **sensores indutivos** funcionam baseados no princípio de que a indutância de uma bobina se altera quando se aproxima um objeto de material ferromagnético, como se ilustra na Figura 2.17 [2].

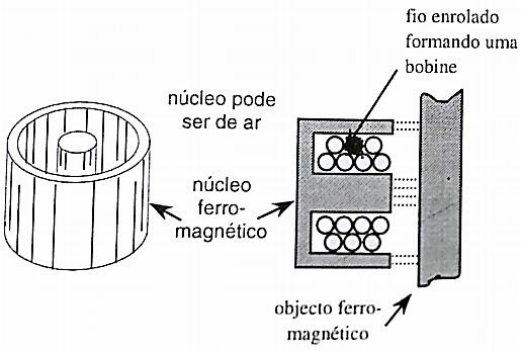


Figura 2.17 - Princípio de funcionamento de um sensor indutivo [2].

A Figura 2.18 ilustra o aspeto físico de um sensor indutivo de aplicação industrial.

.

Figura 2.18 - Aspeto físico de um sensor indutivo de aplicação industrial [18].

Os **sensores capacitivos** baseiam-se no mesmo princípio de funcionamento dos condensadores, constituídos por duas superfícies condutoras, separadas por um dielétrico, cuja capacidade varia diretamente com as áreas frente-a-frente e inversamente com a distância entre elas. Existem normalmente duas variantes, uma para a medição do tipo tudo ou nada (curtas distâncias) e outra para a medição da distância (com precisão) entre a superfície do sensor e a superficie de um objecto (metálico ou não). No primeiro caso é composta por um elétrodo central principal e um elétrodo de compensação (a sua função é garantir que as linhas de campo eletrostático, mesmo nas arestas, mantenham a perpendicularidade para melhor precisão da proporcionalidade entre a distância e a capacidade medida) e a capacidade é medida entre o elétrodo principal e o corpo metálico do sensor. Quando um objecto se aproxima altera a sua capacidade e um circuito condicionador de sinal efectua a detecção do objecto. No segundo caso o sensor é composto apenas por uma das superfícies condutoras sendo a outra uma peça metálica montada a uma certa distância ligada à terra (ver Figura 2.19). Neste caso qualquer objecto que se interponha entre o sensor e a peça metálica modifica a sua capacidade. Neste caso os circuitos condicionadores de sinal são usados não só para detectar como para medir também a distância entre o objecto e o sensor [2].

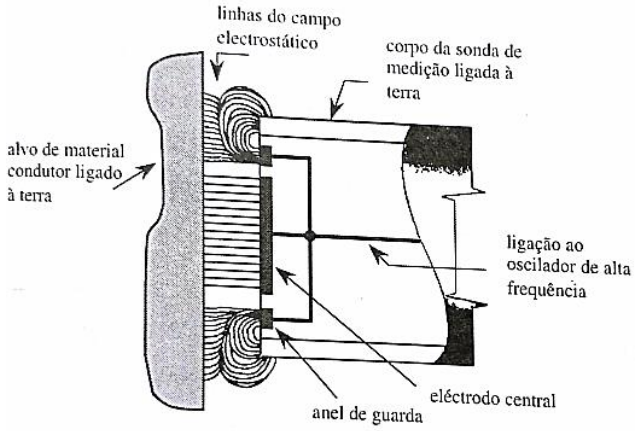


Figura 2.19 - Princípio de funcionamento de um sensor capacitivo [2].

Embora fisicamente muito robusto, este sensor tem algumas restrições relativamente à operação em ambientes poeirentos ou húmidos, ou qualquer outro que altere significativamente a constante dielétrica do circuito [2].

A Figura 2.20 ilustra o aspeto físico de um sensor capacitivo embebido de aplicação industrial.

Figura 2.20 - Aspeto físico de um sensor capacitivo de aplicação industrial [19].

Os **sensores de ultrasons** são dispositivos muito utilizados na medição de distâncias em robótica. A medição da distância baseia-se na medição do tempo de voo de uma onda ultrassónica (ver Figura 2.21), isto é, do tempo que decorre entre o instante em que é emitida uma frequência acima da gama audível até ao instante em que é recebido o eco da reflexão da onda emitida no objeto-alvo [2].

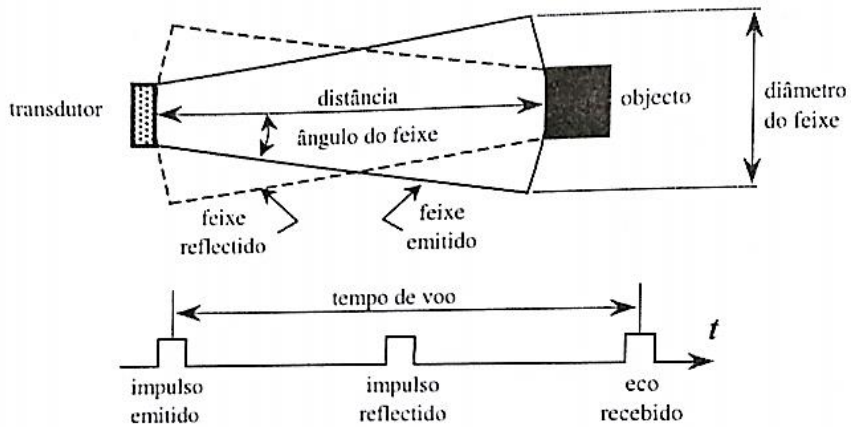


Figura 2.21 - Princípio de funcionamento de um sensor de ultrasons [2].

A Figura 2.22 ilustra o aspeto físico de um sensor de ultrasons de aplicação industrial.

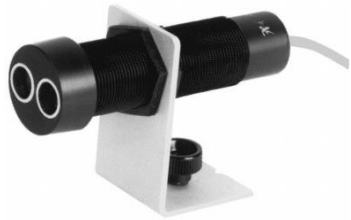


Figura 2.22 - Aspeto físico de um sensor de ultrasons de aplicação industrial [20].

Os **sensores de Raios *Laser*** funcionam como uma alternativa mais precisa, mas bastante mais dispendiosa do que a solução recorrendo aos ultrassons. Normalmente é utilizada a triangulação com emissor e recetor separados, definindo um triângulo com o objeto, em que determina a distância a que este se encontra com base no tempo que o feixe laser demora a ser refletido para o recetor e na velocidade da luz (ver Figura 2.23).

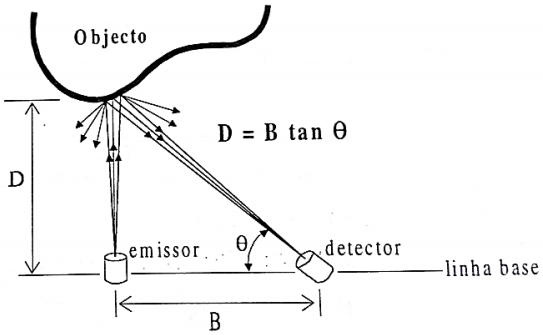


Figura 2.23 - Princípio de funcionamento de um sensor de raios laser [2].

A Figura 2.24 ilustra o aspeto físico de um sensor de raios laser de aplicação industrial.

Figura 2.24 - Aspeto físico de um sensor de raios laser de aplicação industrial [20].

Numa câmara o elemento fundamental para captação de imagens é o **sensor de imagem**. Este analogamente age como a retina do olho humano, em que capta a luminosidade das imagens que são projetadas sobre o sensor.
Atualmente existem duas tecnologias para a criação de sensores de visão, a *Charge-Coupled Device* (CCD) e a *Complementary Metal Oxide Semiconductor* (CMOS) [14].

O **sensor CCD (*Charge-Coupled Device)*** é constituído por uma matriz de elementos sensíveis à luz, que podem ser considerados como condensadores, pois acumulam quantidades de carga variáveis consoante o número de fotões que cada um recebeu (ver Figura 2.25). Para cada imagem, as quantidades de carga são lidas e transferidas para circuitos auxiliares que as convertem para informação compreensível pelo restante equipamento onde será utilizada [13].

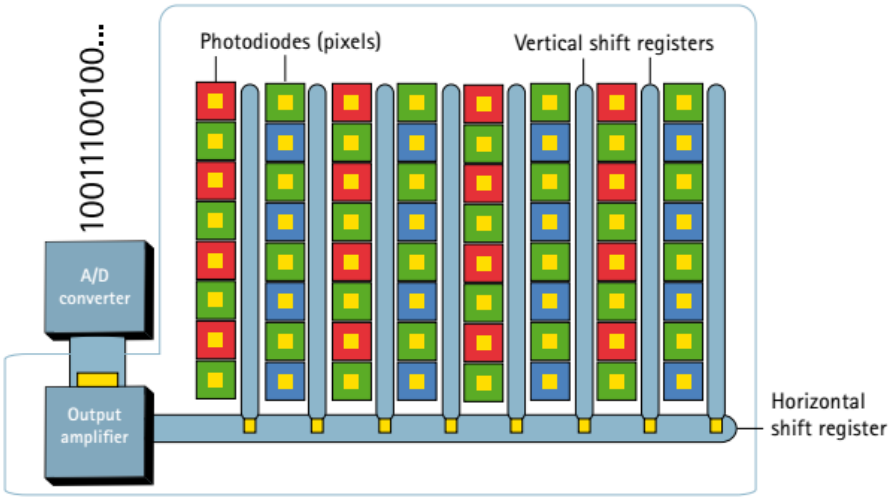


Figura 2.25 - Princípio de funcionamento de um sensor CCD [14].

De referir que o processo de carga é independente da cor, assim, a obtenção de imagens coloridas resulta da existência de conjuntos de pixéis com filtros específicos para as cores fundamentais *Red, Blue and Green* (RGB), sendo apenas sensíveis a estas. Estes pixéis podem ser agrupados no mesmo sensor ou podem ser utilizados simultaneamente três sensores CCD, cada um destinado à receção de uma cor fundamental.

Este tipo de tecnologia tem como desvantagem o consumo de energia e a sua suscetibilidade a alterações das condições de captação de imagem, sobretudo a luminosidade [13].

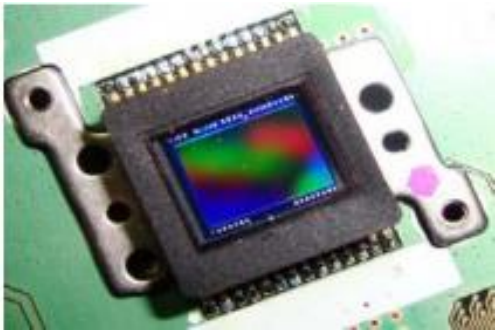


Figura 2.26 - Sensor CCD [13].

Para digitalizar a imagem, é utilizado um conversor analógico/digital (A/D) (ver Figura 2.25) que vai sucessivamente encontrando um valor numérico aproximado correspondente à tensão elétrica de cada um dos pixéis. Cada um destes números é guardado numa posição de memória, construindo-se assim um mapa das intensidades luminosas registadas [2].

Os **sensores CMOS (*Complementary Metal Oxide Semiconductor)*** têm uma estrutura e funcionamento bastante semelhante aos sensores CCD. A sua diferença mais proeminente é a presença, junto a cada pixel, de vários transístores dedicados ao processamento da informação recebida por estes. Graças a esses transístores, a eletrónica necessária para os ciclos de descarga dos pixéis e obtenção de informação é bastante mais simples, o que torna o processo mais rápido, a sua produção mais acessível e o consumo de energia bastante mais reduzido [13]. Ao contrário do sensor CCD, o CMOS já incorpora amplificadores e conversores A/D (analógico/digital) [14]. Tal como seria de esperar, estes sensores também têm desvantagens. Devido ao espaço ocupado pelos circuitos específicos para cada pixel na face do sensor, estes tornam-se menos sensíveis que os CCD [13].

A Figura 2.27 ilustra o principio de funcionamento de um sensor CMOS.

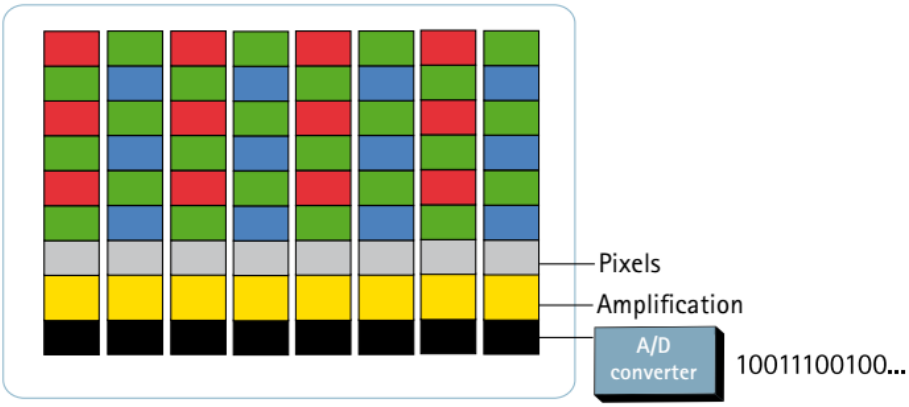


Figura 2.27 - Princípio de funcionamento de um sensor CMOS [14].

2.6. Atuadores

Os atuadores são dispositivos que atuam sobre uma grandeza física do processo, atendendo a comandos que podem ser manuais ou automáticos [15]. Os atuadores elétricos e os atuadores pneumáticos são os mais utilizados em sistemas robóticos.

Atuadores Elétricos

Os atuadores elétricos são muito utilizados principalmente devido à sua facilidade de comando e flexibilidade [15]. O facto da energia elétrica ser a forma de energia mais disseminada nas instalações fabris, é mais uma razão para que os atuadores elétricos sejam os mais desejáveis [2]. O motor passo-a-passo é um dos atuadores elétricos mais comuns em sistemas robóticos, em que é um dispositivo que converte um trem de impulsos de uma tensão contínua (DC) num deslocamento mecânico proporcional ao seu eixo. São utilizados essencialmente para posicionamento de equipamentos, mas podem também ser utilizados para tração, beneficiando assim do seu baixo custo [11].

A Figura 2.28 ilustra o aspeto físico de um motor de passo-a-passo.

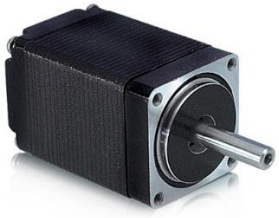


Figura 2.28 - Aspeto físico de um motor de passo-a-passo [21].

Atuadores Pneumáticos

Estes atuadores utilizam o ar comprimido como energia atuante. A sua utilização em ambientes industriais não apresenta restrições, pois na maior parte das instalações fabris existe já uma rede de ar comprimido. Os cilindros acionados por ar comprimido são os atuadores pneumáticos mais utilizados na robótica [2]. Estes podem-se classificar quanto ao seu tipo [22]:

- **Cilindros de efeito simples (ação simples) (ver Figura 2.29 e Figura 2.30):**
  - Realizam trabalho num único sentido;
  - Retorno ao repouso por mola ou força externa;
  - Cursos até 100mm (limitados pela dimensão da mola recuperadora).

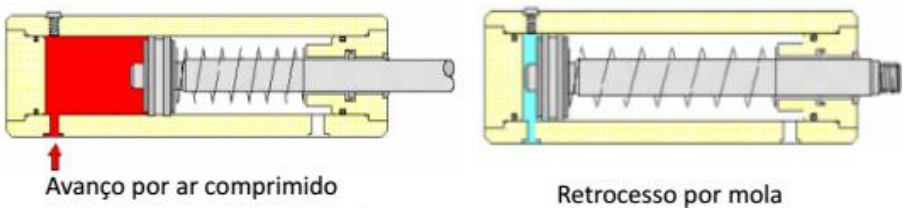


Figura 2.29 - Princípio de funcionamento de um cilindro pneumático de efeito simples [22].

Figura 2.30 - Aspeto físico de um cilindro pneumático de efeito simples [22].

- **Cilindros de duplo efeito (ação dupla) (ver Figura 2.31 e Figura 2.32):**
  - Realizam trabalho nos dois sentidos (avanço e retrocesso);
  - Avanço e retrocesso por efeito do ar comprimido;
  - Cursos até 200mm (curso habitual é entre 100mm e 200mm);
  - Força de avanço maior que a força de retrocesso (devido ao volume ocupado pelo veio do cilindro).

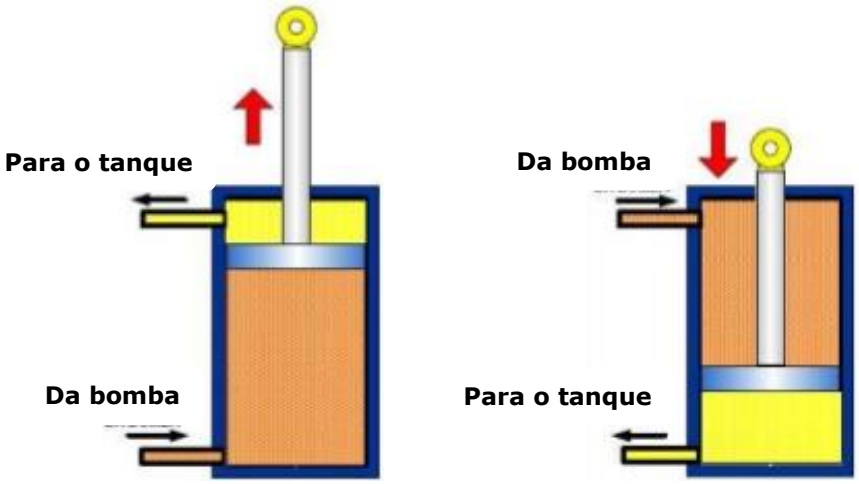


Figura 2.31 - Princípio de funcionamento de um cilindro pneumático de duplo efeito [23].

Figura 2.32 - Aspeto físico de um cilindro pneumático de duplo efeito [24].

Em comparação com os atuadores elétricos, os pneumáticos permitem forças mais elevadas [11]. Contudo, o ar comprimido é em si uma fonte de energia com grandes custos (aproximadamente 10 vezes superior à energia elétrica) em virtude de possuir aproximadamente 50% de perdas (escapes, fugas, baixo rendimento dos compressores, etc.). No entanto, o vantajoso preço dos equipamentos conjuntamente com a grande rentabilidade dos ciclos de trabalho compensa o custo desta energia [22].

2.7. Sistemas de armazenamento

2.7.1. Métodos e exemplos de armazenamento

Com a crescente necessidade das empresas em criar valor para o cliente, no sentido de disponibilizar produtos na quantidade certa, no tempo certo, no local certo e ao custo mínimo, surgiu a necessidade da criação de infraestruturas de armazenamento. Esta necessidade ocorre devido às variações, quer do abastecimento, quer da procura, onde ambas têm comportamentos distintos ao longo do tempo. O armazenamento pode ser classificado essencialmente em duas formas distintas: manual ou automático [25].

Os **armazéns manuais** podem ser representados pelos sistemas que se seguem:

- **Estantes paletização convencional** – é o sistema mais universal para o acesso direto e unitário a cada palete. Por isso, é a solução ótima para armazéns em que é necessário armazenar produtos paletizados com grande variedade de referências. A distribuição e altura das estantes determinam-se em função das características dos empilhadores, dos elementos de armazenamento e das dimensões do local (ver Figura 2.33) [26].

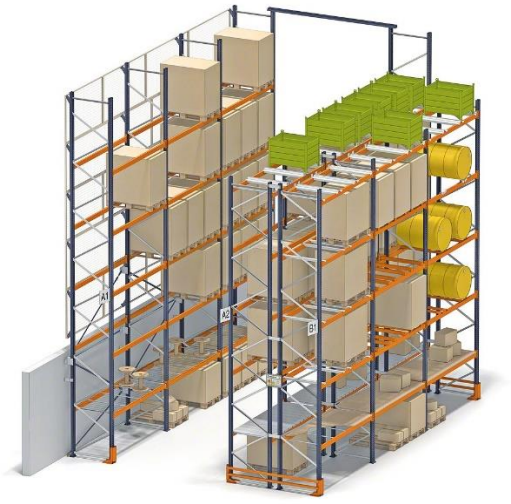


Figura 2.33 - Sistema de estantes paletização convencional [26].

- **Estantes de paletização compacta** – é um sistema de armazenamento por acumulação que facilita a máxima utilização do espaço disponível, tanto em superfície como em altura. Estantes adequadas para produtos homogéneos com baixa rotação e grande quantidade de paletes por referência. Existem dois sistemas para gestão da carga: o sistema *drive-in*, com um único corredor de acesso, e o sistema *drive-through,* com dois acessos à carga, um de cada lado da estante. O sistema compacto é muito utilizado em câmaras frigoríficas, tanto de refrigeração como de congelação, que precisam de aproveitar ao máximo o espaço destinado ao armazenamento dos seus produtos em temperatura controlada (ver Figura 2.34) [27].

Figura 2.34 - Sistema de estantes paletização compacta [27].

- **Estantes paletização *Movirack*** - com a *Movirack* consegue-se compactar as estantes e aumentar a capacidade do armazém sem perder o acesso direto a cada palete. As estantes colocam-se sobre bases móveis guiadas que se deslocam lateralmente. Assim, suprimem-se os corredores e, no momento necessário, abre-se apenas o corredor de trabalho. É o operador que dá a ordem de abertura automática através de um comando à distância ou, de forma manual, acionando um interruptor.

Estas bases dispõem de motores, elementos de movimentação e diferentes sistemas de segurança que devem garantir um funcionamento seguro e eficaz (ver Figura 2.35) [28].

Figura 2.35 - Sistema de estantes paletização Movirack [28].

- **Estantes paletização dinâmica** - este sistema é normalmente aplicável a qualquer sector da indústria ou da distribuição (alimentação, sector automóvel, indústria farmacêutica, química, etc.). Normalmente as palates apresentam uma dimensão fixa. As estantes são constituídas por uma plataforma de rolos, com uma ligeira inclinação que permite o deslizamento das paletes, por gravidade e a uma velocidade controlada, até ao extremo oposto (ver Figura 2.36) [29].

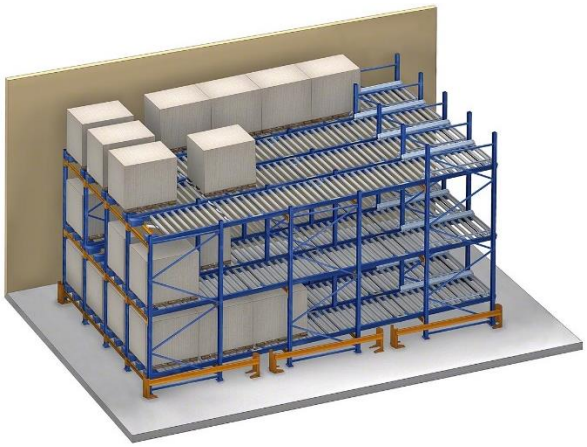


Figura 2.36 - Sistema de estantes paletização dinâmica [29].

- **Estantes paletização *Push-back*** – é um sistema de armazenamento por acumulação que permite armazenar até quatro paletes em profundidade por cada nível. Todas as paletes de um mesmo nível, à exceção da última, assentam sobre um conjunto de carros que se deslocam, por empurrão, sobre os carris de rodagem. Ideal para o armazenamento de produtos de média rotação, com duas ou mais paletes por referência (ver Figura 2.37) [30].

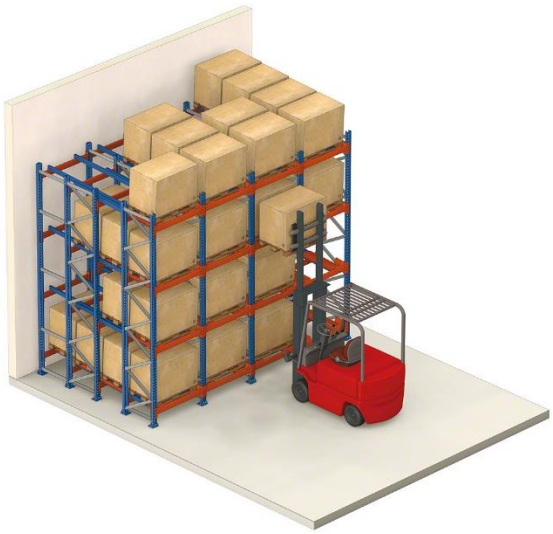


Figura 2.37 - Sistema de estantes paletização Push-back [30].

Os **armazéns automáticos** podem ser representados pelos sistemas que se seguem:

- **Transelevadores (ou transportadores – elevadores) para caixas** - é formado por um corredor central por onde circula um transelevador e duas estantes situadas em ambos os lados para armazenar caixas ou bandejas. Num dos extremos ou numa lateral da estante situa-se a zona de automatização e manipulação, formada por transportadores, onde o transelevador deposita a carga retirada da estante. Os transportadores aproximam a caixa do operador e, uma vez finalizado o seu trabalho, devolvem-na ao transelevador para que a coloque nas estantes.

Todo o sistema é gerido por um *software* de gestão que regista a localização de todos os materiais do armazém e mantém um inventário em tempo real (ver Figura 2.38) [31].

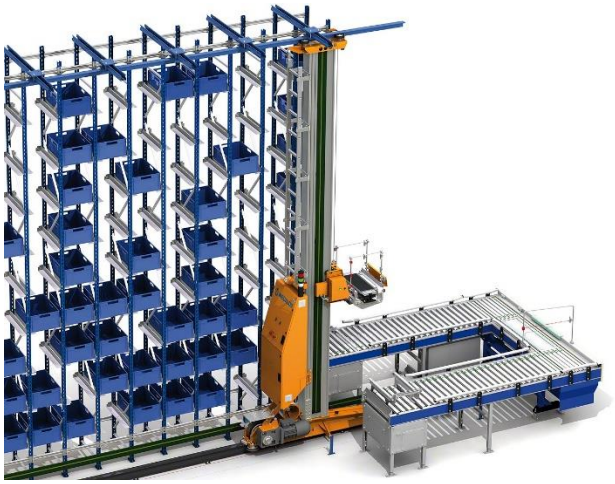


Figura 2.38 - Transelevadores para caixas [31].

- **Carrosséis horizontais e verticais** - são compostos por uma série de prateleiras que rodam (no sentido horizontal ou vertical), entregando os artigos selecionados num ponto de acesso ao operador (ver Figura 2.39) [25].

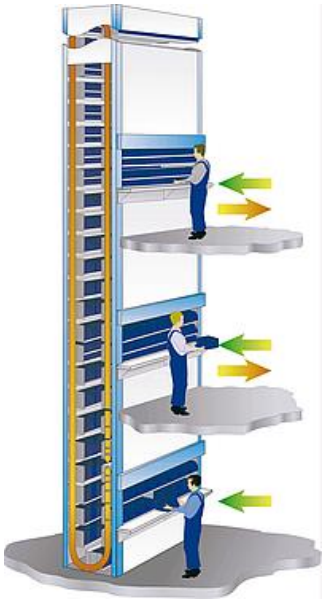


Figura 2.39 - Carrossel vertical [32].

- ***Pallet Shuttle*** – é um sistema de armazenamento semiautomático de alta densidade que facilita a carga e descarga de mercadoria a partir de um carro elétrico, denominado *Pallet Shuttle*. Este faz movimentos internos dentro das estantes de forma autónoma, sem necessidade de que os empilhadores entrem dentro dos corredores do armazém. O operador guia todos os movimentos do *Pallet Shuttle* através de um comando à distância, ao qual transfere as ordens (ver Figura 2.40) [33].

Figura 2.40 - Pallet Shuttle [33].

Todos os métodos de armazenamento demonstrados funcionam no âmbito de um dos dois possíveis critérios para a arrumação dos produtos, o FIFO e o LIFO. O FIFO, que provém do termo em inglês *first-in-first-out*, significa que o primeiro a entrar é o primeiro a sair. Através deste método existe uma melhor forma de controlar as datas de validade dos produtos em *stock*, onde os primeiros produtos armazenados são os primeiros a serem retirados. O LIFO provém do termo em inglês *last-in-first-out*, o que significa último a entrar é o primeiro a sair. Neste método, os produtos colocados em último lugar no armazém são aqueles que são retirados em primeiro lugar [25].

2.7.2. Tecnologias da informação e comunicação

Com o crescente número de empresas e, consequentemente com o aumento da competitividade e da produtividade, o papel das tecnologias da informação e comunicação (TIC) tem vindo a assumir uma importância crescente como resposta a estas

necessidades. A utilização destas tecnologias aplicadas à logística são várias, como computadores, comunicações, *software*, mecanismos de *input*/*output*, entre outras, que têm vindo a desempenhar um papel importante nas empresas, apoiando as tomadas de decisão e a gestão das organizações [25]. Este tipo de soluções tem vindo a ser consolidado num termo designado por WMS ou *Warehouse Management System*.

***Warehouse Management System* (WMS)** - o WMS ou sistema de gestão de armazém consiste num sistema de apoio à gestão dos processos de armazenamento. Este sistema permite uma monitorização de uma forma mais rápida e eficiente dos movimentos nos armazéns, desde as operações de receção, conferência, *picking* (recolha/selecção), expedição, resultando numa redução de erros de *stock,* maior organização e aproveitamento do espaço do armazém e menor dependência das pessoas para estas tarefas.O funcionamento deste sistema inicia-se com a entrada de um artigo em armazém, registado na base de dados e imediatamente disponível no sistema após leitura de um leitor ótico. Todos os processos inerentes ao armazenamento são controlados com a ajuda de terminais portáteis, responsáveis pela leitura dos códigos dos artigos e assim assegurando a rastreabilidade do produto. Durante o tratamento das receções e das expedições de mercadoria, o sistema de gestão informa em tempo-real os operadores onde devem ser colocados ou retirados os artigos. Através deste sistema também é possível o controlo de *stocks*, na medida em que permite emitir avisos de excesso e de ruturas de *stock*, permitindo o reaprovisionamento do armazém atempadamente. Estes processos são realizados com o auxílio de tecnologias de comunicação *wireless*, mais concretamente equipamentos portáteis de leitores óticos permanentemente ligados ao sistema central, permitindo um controlo de tarefas em tempo real reduzindo significativamente os erros [25]. A identificação por código de barras e *Radio Frequency IDentification* (RFID) são duas das tecnologias mais utilizadas nos sistemas WMS. Em sistemas de pequena escala poder-se-á utilizar uma tecnologia mais simples, mas muito limitada, a deteção de objetos por sensores de proximidade.

O **código de barras** é um tipo de linguagem comum a nível internacional, frequentemente utilizado em identificação de mercadorias. Este tipo de tecnologia permite a realização de uma gestão automatizada em todos os tipos de indústria. A leitura desta codificação é realizada com o auxílio de um leitor de código de barras (dispositivos óticos com capacidade para emitir e receber um feixe de luz, visível ou invisível, geralmente

infravermelho), que tem a capacidade de ler e transferir essa informação para uma unidade central (por exemplo, um computador) via *wireless* [34].

O código de barras unidimensional (1D) EAN-13 é um dos mais utilizados em todo o mundo. Trata-se de um código numérico constituído por treze dígitos. A Figura 2.41 ilustra os aspetos principais da estrutura do código EAN-13, com base no exemplo do código 5600338865781. Os dígitos das posições 1, 2 e 3 correspondem a um prefixo atribuído aos países ou organizações nacionais (em Portugal a *Codipor* – Associação Portuguesa de Identificação e codificação de Produtos) e que são designados por *Flag* (560 para Portugal). A 13ª posição é ocupada por um dígito de controlo, que é calculado com um algoritmo específico, e cuja função é detetar erros na leitura e evitar adulterações [25].

Posição dos dígitos

1	Separador lateral	2	3	4	5	6	7	Separador central	8	9	10	11	12	13	Separador lateral

Significado:

País ou Organização (*Flag*)	Código da Empresa Portuguesa (CEP)		Identificação do Produto	Dígito de Controlo	

Figura 2.41 - Estrutura do código EAN-13 [25].

O ***Quick Response Code* (*QR code*)** (ver Figura 2.42) é um código bidimensional (2D) que tem a capacidade de armazenar 4296 caracteres alfanuméricos [35]. Este tipo de código é uma alternativa aos tradicionais códigos de barra 1D.

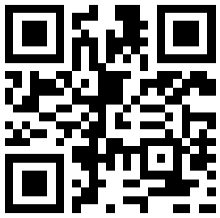


Figura 2.42 - Código QR [36].

A Figura 2.43 ilustra um exemplo do processo de leitura de um código de barras, em que o leitor portátil lê o código e transmite-o para um *software* de gestão.

Figura 2.43 - Processo de leitura de um código de barras [36].

A utilização do tradicional código de barras (1D e 2D) apresenta algumas desvantagens [37]:

- Os motoristas têm de sair dos empilhadores para efetuar a leitura manual dos códigos de barra contidos nas mercadorias;
- Os códigos de barras podem ficar danificados, dificultando a leitura dos respetivos;
- Não é possível editar a informação contida nas etiquetas de códigos de barras;
- Não é possível atualizar a informação em tempo real;
- Possível informação incorreta devido a erros humanos.

Face às desvantagens apresentadas, as empresas sentiram a necessidade de procurar outras soluções mais vantajosas que pudessem aumentar a eficácia e eficiência, redução de recursos e tempos de processamento no seu armazém. A tecnologia RFID tem sido a opção mais solicitada para equipar os sistemas WMS.

Os **sistemas RFID (*Radio Frequency IDentification)*** permitem identificar e acompanhar automaticamente objetos, produtos ou bens, comunicando com estes através de sinais de ondas de rádio. A tecnologia RFID tem-se tornado cada vez mais popular no setor logístico, pois com a aplicação desta na gestão de armazéns apresenta um grande potencial estratégico para o desenvolvimento de modelos integrados, o que se traduz num aumento da eficácia e eficiência, redução de recursos e tempos de processamento. Desta forma, é permitida a rastreabilidade dos produtos e a disponibilização de dados em tempo-real através da interligação entre a tecnologia RFID e a internet (tecnologia *Internet of Things* – IoT), qualquer que seja o local onde se encontrem ao longo de toda a cadeia [25].

Um sistema WMS baseado na tecnologia RFID é composto principalmente por: *hardware* RFID, *software* de gestão, *wireless local area network* (WLAN) e empilhadores. Em específico, o *hardware* RFID é composto por *transponders (tags* RFID*)*, antenas e leitores portáteis RFID [37]. As *tags* RFID ou etiquetas são colocadas em todas as paletes e em cada *rack* (fila) de cada estante (ver Figura 2.44). Uma palete que possua uma *tag* RFID embebida é denominada de palete digital. A *tag* RFID possui toda a informação da mercadoria que a palete contém, tais como: quantidade, nome, data de produção, entre outros. O estado de uma palete digital pode ser "completo", ou "vazio". Como já foi mencionado, a estante possui uma determinada quantidade de *racks*. A *rack* é uma posição de armazenamento para cada palete e é identificada pelo seu número, nome e *rack ID* (RID), ao qual é vinculado à *tag ID* (TID) que está embebida na palete digital. De forma similar à palete digital, o estado da *rack* da estante pode ser "completo" ou "vazio" [37].

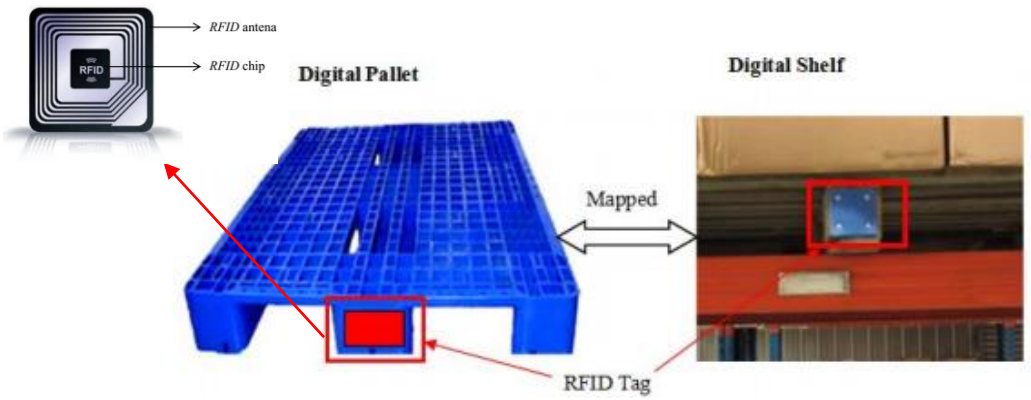


Figura 2.44 - Palete e estante digital (tecnologia RFID) [37].

Os empilhadores utilizados nos sistemas WMS são equipados com sistemas RFID e WLAN (ver Figura 2.45). Cada empilhador é dotado de duas antenas e leitores portáteis como unidades para identificação RFID. Uma das antenas é colocada na frente do empilhador, perpendicular ao chão, com o intuito de detetar as *tags* das paletes colocadas nas estantes. A outra antena é colocada debaixo do empilhador, paralela ao chão, e é responsável por detetar as *tags* que estão em pontos estratégicos do armazém (colocados no chão). O computador instalado em cada um dos empilhadores efetua o pré-processamento dos dados recolhidos pelas duas antenas e envia-os (através da rede WLAN) para o sistema de gestão [37].

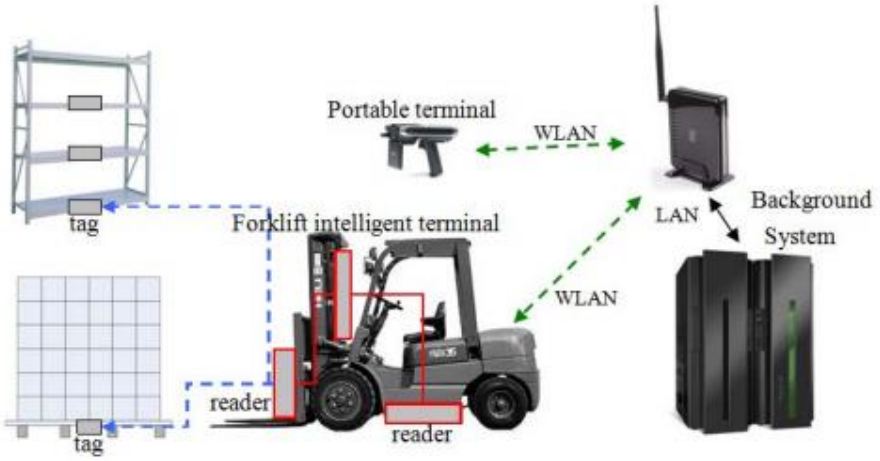


Figura 2.45 - Empilhador dotado de um sistema RFID e WLAN [37].

O processo de operação de um armazém (operação de *picking*) que utiliza a tecnologia RFID pode ser descrito pelas seguintes etapas [38]:

- O sistema de gestão gera uma lista da mercadoria requisitada pelo cliente;
- Através da rede *wireless* WLAN, o sistema procura um empilhador que esteja livre de operação e envia-lhe a lista da mercadoria;
- O motorista do empilhador recebe essa informação e lê as etiquetas e códigos de localização dos respetivos produtos através de leitores móveis, em que envia a informação recolhida novamente para o sistema de gestão (através da rede WLAN);
- O sistema de gestão analisa essa informação e compara com a lista da mercadoria inicial. Caso esteja tudo conforme, o sistema informa o motorista do empilhador para avançar com a recolha dos produtos da zona de armazenamento (através da rede WLAN);

- O sistema RFID atualiza automaticamente a informação contida nas *tags* de localização dos produtos.

Em resumo, um sistema de armazenamento WMS dotado de sistema RFID funciona como um poderoso sistema de aquisição de dados em tempo real, com a vantagem de eliminação de intervenções humanas manuais e visuais, reduzindo assim o tempo de transições e assegurando eficiência e eficácia no processo logístico. Contudo, esta solução acarreta alguns contras principalmente no que refere à elevada quantidade de energia electromagnética que emite, e interferências eletromagnéticas para outros dispositivos colocados no ambiente industrial, e particularmente se é segura ou não para o ser humano em termos de potência do sinal emitido. A aplicação da tecnologia RFID guia-se pela série de normas ISO 18000 [39].

2.8. Sistemas assistidos por visão computacional

2.8.1. Técnicas aplicadas à visão computacional

A deteção e reconhecimento de objetos são dos principais focos de interesse da visão computacional, sobretudo em aplicações na área da robótica, onde os algoritmos de deteção e reconhecimento são a base de grande maioria dos sistemas implementados. O desenvolvimento destes algoritmos é bastante complexo, tendo estes por vezes que prever e ultrapassar dificuldades na deteção devido a variações em ângulos e escalas, variações fotométricas que incluem mudança de brilho e contraste e deformações em perspetiva devido a mudanças de posição por parte do observador [13].

Deteção de objetos: no processo de deteção de objectos são usadas **técnicas de segmentação** de forma a selecionar na imagem possíveis pontos ou objetos de interesse. A segmentação é o processo pelo qual uma imagem digital é dividida em regiões, agrupando os pixéis segundo um determinado critério. A segmentação por cor, intensidade dos pixéis, textura ou algoritmos de deteção de contornos, vértices e descontinuidades são importantes no contexto da análise de imagem [40].

Tal como apresentado nas figuras que se seguem (Figura 2.46 e Figura 2.47) , o processo de segmentação de imagem pode ser efetuado com a aplicação de várias técnicas de deteção de contornos, tais como: *Sobel*, *Prewitt*, *Roberts*, *Canny*, *Laplacian of Guassian* (LoG), algoritmo *Expectation-Maximization* (EM), *Otsu*, entre outros [41].

Operador/Algoritmo	Imagem Segmentada	Imagem Original
Sobel		
Prewitt		
Roberts		
***Laplacian of Gaussian* (LoG)**		

Figura 2.46 – Técnica de segmentação com recurso ao operador Sobel, Prewitt, Roberts e LoG [41].

Operador/Algoritmo	Imagem Segmentada	Imagem Original
Canny		
Otsu		

Figura 2.47 – Ténicas de segmentação com recurso aos operadores Canny e Otsu [41].

O operador *Canny* é o que melhor define os contornos da imagem [41].

A **técnica de subtração do fundo** é uma abordagem amplamente utilizada na inspeção de placas de circuito impresso (PCB), que tem como intuito detetar ausência ou excesso de componentes nas PCBs. Esta técnica é baseada num processo de subtração de pixéis, em que se subtrai o valor de cada pixel da imagem de referência, pelo respetivo pixel da imagem inspecionada. Este processo de subtração tem como resultado uma terceira imagem, e pode ser deduzido pela seguinte expressão [42]:

$$g(i,j) = f_1(i,j) - f_2(i,j) \tag{2.1}$$

g: imagem resultante
f_1: imagem de referência
f_2: imagem inspecionada

A Figura 2.48 ilustra um exemplo de aplicação do método de subtração do fundo em inspeção de PCB.

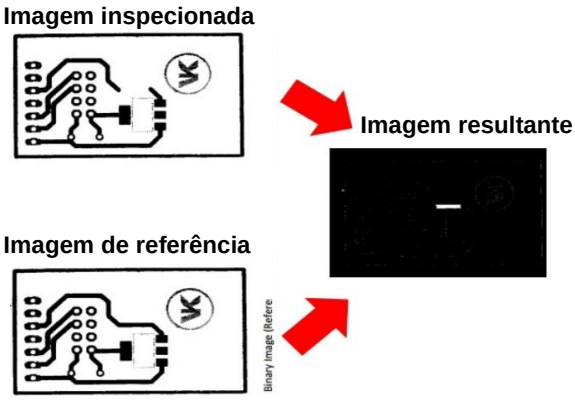


Figura 2.48 - Aplicação do método de subtração do fundo em inspeção de PCB [42].

Reconhecimento de objetos: após a segmentação da imagem, dependendo do objetivo que se pretende para o sistema, por vezes é necessário retirar informações que classifiquem e descrevam os objetos. Caraterísticas como a *área*, o *diâmetro*, o *centro de gravidade* e o *perímetro* são frequentemente utilizadas. Contudo, devido ao facto de serem sensíveis ao fator de escala, podem ser insuficientes para uma classificação correta e completa dos objetos em questão. Casos em que seja necessário distinguir objetos com a mesma forma é necessário usar outros algoritmos. Neste contexto, os histogramas de cor são bastante úteis. Comparando os histogramas de cor de cada objeto, pode-se facilmente obter uma descrição de cada um deles [40].

Com o intuito de tornar o reconhecimento imune às variações referidas anteriormente, existem algoritmos que se baseiam na extração de características específicas, normalmente pertencentes à sua superfície, cantos ou arestas. Dois dos algoritmos mais utilizados para a extração de características e reconhecimento de objetos são o *Scale Invariant Feature Transform* (SIFT) e *Speed Up Robust Features* (SURF). Devido à sua robustez e ao facto de se centrarem em características invariantes à escala e rotação (ver Figura 2.49, Figura 2.50), permite que o reconhecimento seja eficaz em diversos tipos de situações [13].

O **SIFT** é um algoritmo de visão computacional desenvolvido por *David G. Lowe* que é utilizado frequentemente para o reconhecimento de objetos através da deteção e extração de descritores invariantes em escala e rotação. Este é bastante utilizado em aplicações envolvendo a correspondência de diferentes imagens de um objeto ou cena devido a serem razoavelmente invariantes a mudanças de iluminação, ruído na imagem, rotação, escala e pequenas mudanças de ponto de vista ([13], [43]). Este algoritmo é composto por dois elementos distintos, o detetor que é implementado com recurso a funções *Difference of Gaussian* (DoG) e o descritor, que se obtém com base em histogramas de orientações da vizinhança local dos pontos-chave. O DoG é um algoritmo que salienta características particulares de imagens com recurso a subtrações de imagens processadas por filtros gaussianos a diferentes escalas ([13], [43]).

A obtenção dos descritores de uma imagem pode-se resumir nas seguintes etapas ([13], [43]):

- **Deteção de extremos no espaço-escala;**
- **Localização de pontos-chave;**
- **Determinação de orientação;**
- **Descritor de pontos-chave.**

Após a obtenção dos descritores para cada ponto, a tarefa de encontrar a correspondência entre imagens resume-se a encontrar entre os descritores de uma imagem, os melhores candidatos a serem os seus equivalentes na outra imagem [13].

Figura 2.49 - Resultado da aplicação do algoritmo SIFT [44].

O **SURF** é um algoritmo de visão computacional desenvolvido por *Herbert Bay* inspirado no algoritmo SIFT, tendo, portanto, atributos idênticos ([13], [44]). Este foi desenvolvido com o intuito de tornar o processo de identificação de aspetos semelhantes entre imagens mais célere. A principal diferença face ao SIFT reside na fase de deteção dos pontos-chave, que neste é efetuada com recurso a um detetor *Hessian*. Este detetor é sustentado na matriz Hessiana (matriz composta por derivadas parciais de segunda ordem) capaz de identificar zonas de imagem onde se verificam variações acentuadas de intensidade. De referir que neste algoritmo, a determinação e atribuição da orientação aos pontos-chave detetados sustenta-se em transformadas *Haar-wavelets*. Esta transformada é baseada nas funções de *Haar*, em que permite descrever de forma expedita as caraterísticas locais de uma imagem com recurso a um conjunto de coeficientes.

A procura por correspondências segundo este algoritmo pode ser dividida essencialmente em três fases ([13], [45]):

- Encontrar pontos-chave na imagem, definindo-os como aqueles que sejam identificáveis em diferentes condições de imagem;
- Representar por um vetor de características ou descritores a vizinhança dos pontos chave. Devem ser distintivos e robustos ao ruído, às deformações geométricas da imagem e aos erros;
- Comparar os vetores descritores da imagem com o objeto. A comparação é geralmente baseada na distância entre vetores.

Figura 2.50 - Resultado da aplicação do algoritmo SURF [44].

Segundo os dados experimentais obtidos em [44], o algoritmo SURF possui uma maior precisão nos pontos-chave relativamente ao algoritmo SIFT.

2.8.2. Exemplos de sistemas assistidos por visão computacional

A visão computacional encontra-se actualmente em muitas aplicações da indústria apoiando muitas vezes sistemas robóticos na realização de diversas tarefas através da introdução de importantes capacidades sensoriais muitas desejadas para os automatismos desta natureza. Alguns exemplos disso são:

Equipamento de carregamento orientado por sistema de visão (OMRON):

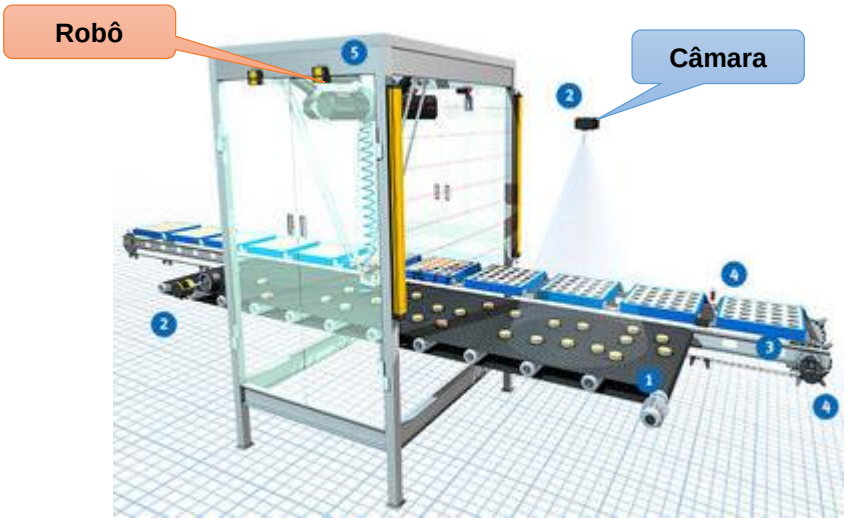


Figura 2.51 - Equipamento de carregamento orientado por sistema de visão [46].

Neste exemplo uma máquina é composta por um robô montado numa estrutura que se sobrepõe a dois tapetes rolantes que funcionam em paralelo (ver Figura 2.51). Um dos tapetes transporta os produtos e o restante as embalagens. O robô escolhe os produtos de acordo com um algoritmo desenvolvido e coloca-os nas embalagens em movimento. A localização imediata dos produtos em movimento é determinada pelo sistema de visão através das imagens adquiridas por uma câmara estacionária. Por outro lado, um sensor de registo é utilizado para acompanhar o posicionamento das embalagens. Este controlo dos dois tapetes permite que o robô recolha e transfira produtos de um tapete para o outro [46].

Este tipo de sistemas, por um lado, utiliza técnicas de deteção de objetos, pois um dos principais desafios neste tipo de automatismos é descobrir a localização dos mesmos. Por outro lado, este automatismo inspeciona a qualidade dos objetos, ou seja, só armazena

produto em conformidade com determinados requisitos de qualidade. Essa conformidade é analisada pela câmara, a qual por exemplo, utiliza técnicas de reconhecimento por "*pattern matching*" (utiliza padrões previamente armazenados, para que seja possível a operação da comparação).

Sistemas multiespectrais aplicados à indústria cerâmica (INFAIMON):

Figura 2.52 - Sistema multiespectral aplicado à indústria cerâmica [47].

A espectroscopia é uma técnica utilizada com regularidade para efectuar um controlo de qualidade específico em laboratórios. As câmaras multiespectrais geram imagens com informação separada em dezenas ou centenas de comprimentos de onda simultaneamente, proporcionando uma grande quantidade de dados sobre a amostra. Neste exemplo (ver Figura 2.52) estas câmaras são a combinação compacta de um espectrógrafo de imagem *ImSpector* e uma câmara matricial monocromática, e geram uma imagem multiespectral que permite resolver, entre outras, aplicações colorimétricas tanto em meios industriais como científicos.

Comparado com os sistemas de imagens baseados em filtros, as câmaras multiespectrais proporcionam alta resolução espacial e espectral, seleção flexível dos diferentes comprimentos de ondas através de *software* e uma ampla cobertura espectral desde ultravioleta até o infravermelho, passando por todo o espectro visível. Atualmente estes sistemas de visão computacional estão a ser utilizados com êxito para o controlo de qualidade na indústria cerâmica, especialmente na análise de argila, uma das matérias primas mais importantes devido ao seu amplo uso na fabricação de porcelana sanitária.

Técnicas espectrográficas, como a refletância difusa no infravermelho e de dispersão de raios-X, permitem identificar e classificar rapidamente o tipo e a qualidade da argila procedente de diferentes zonas. A análise combinada dos parâmetros de cor e de refletância em determinados comprimentos de onda permite diferenciar as amostras segundo a sua origem e determinar a sua pureza [47].

Leitura automática de informação contida nas embalagens (OMRON):

Figura 2.53 - Leitura automática de informação contida nas embalagens [48] .

Este sistema (ver Figura 2.53) possui uma câmara com a capacidade de ler/interpretar caracteres impressos nas embalagens (por exemplo, a validade do produto). Esta câmara também tem a capacidade de informar (através de um protocolo específico de comunicação) a unidade central de processamento se o produto se encontra, ou não, dentro do prazo de validade [48].

A utilização da visão em sistemas automatizados assume cada vez mais uma maior importância, pois possibilita não só um aumento no índice de produtividade como também uma redução de operadores no processo industrial. O aumento do índice de qualidade do produto é também um fator crucial, pois atualmente a concorrência é demasiado elevada para correr riscos em relação à qualidade do produto. Tudo isto resulta normalmente numa redução substancial de custos para a entidade empregadora. O desenvolvimento de novos sistemas e respetivas necessidades requerem uma evolução constante de técnicas associadas à visão computacional.

3. CONCEITOS GERAIS – VISÃO COMPUTACIONAL

3.1. Introdução à visão computacional

A visão computacional pode ser definida como um processo que, através da implementação de algoritmos computacionais, extrai, carateriza e interpreta informações das imagens que são captadas do meio ambiente tridimensional (3D). Todo este processo é conseguido com recurso a sensores óticos, técnicas de processamento digital de imagens, reconhecimento de padrões por métodos analíticos e estatísticos, e aplicações de Inteligência Artificial (IA) [49].

A visão computacional constitui uma das mais importantes capacidades sensoriais desejadas na robótica, à semelhança do que acontece com o Homem, onde aquele sentido é essencial à realização de operações no processo de produção, como a manipulação e a inspeção [2]. A Figura 3.1 demonstra a analogia entre um sistema de visão humana e um sistema de visão computacional.

Figura 3.1 - Analogia entre o sistema de visão humana e computacional [50].

Um dos principais objetivos da visão computacional no contexto da robótica é o reconhecimento de objetos, de entre um conjunto de unidades conhecidas, e a sua posição para desencadear possíveis operações de manipulação. Para os mesmos fins pode também interessar obter o contorno da peça ou o mapa de cores (podendo ser apenas de cinzentos), tendo em vista delimitar um objeto e/ou as suas partes ou relevo.

As mesmas características podem ter interesse para operações de inspeção, visando a identificação, contagem ou seleção de peças [2].

Na implementação da visão computacional, para além do dispositivo de aquisição de imagem propriamente dito (câmara) outros elementos são também essenciais, nomeadamente: o sistema de iluminação, a interface com a unidade de processamento digital e o *software* de processamento de imagem. Dependendo da aplicação em questão, o sistema de aquisição de imagem pode ter especificações exigentes, o mesmo acontece com o processamento computacional, ao qual se pode exigir uma grande capacidade de memória e elevada velocidade de processamento.
Estes requisitos retardaram muito a utilização da visão na robótica há algumas décadas atrás como uma alternativa tecnicamente possível e economicamente viável. Atualmente, com o grande desenvolvimento tecnológico na área dos microcomputadores, em que oferecem capacidades computacionais cada vez mais elevadas a preços cada vez mais baixos, e a banalização da câmara de vídeo, transformada num produto de eletrónica de consumo para entretenimento com qualidade aceitável, tornaram a visão computacional um recurso perfeitamente acessível à robótica industrial e com custos baixos a moderados [2].

Os procedimentos para aplicação de técnicas de visão computacional encontram-se normalmente divididos em seis etapas sequenciais (ver Figura 3.2) [49]:

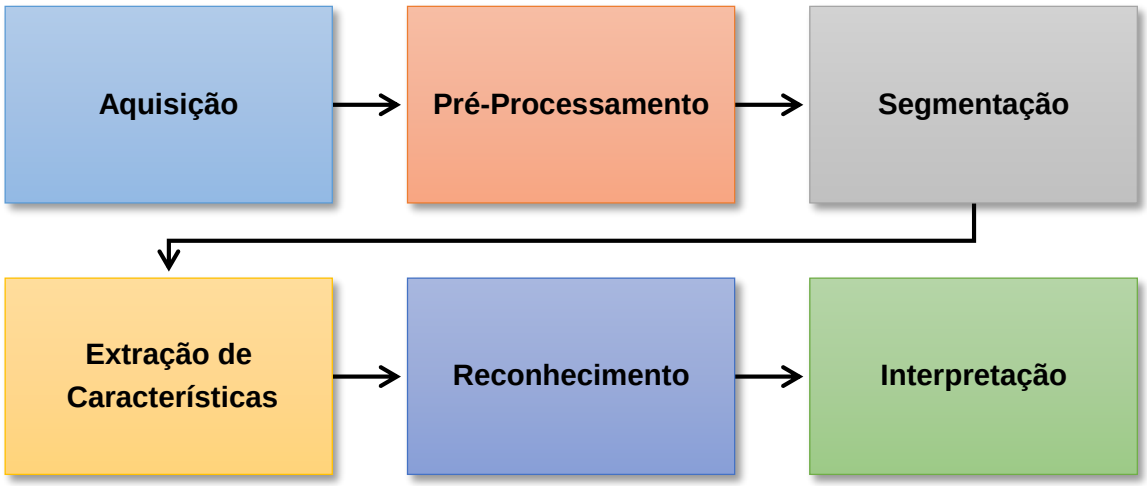


Figura 3.2 - Etapas de um processo de visão computacional.

Analisando de um modo geral cada uma das etapas dos procedimentos de visão computacional, temos ([49], [50]):

- **Aquisição:** aquisição da imagem captada com recurso ao dispositivo de visão (normalmente câmara digital). São utilizadas técnicas especiais de iluminação para obter imagem com contraste suficiente para processamento posterior, pois na imagem adquirida existe uma considerável quantidade de informações que devem ser realçadas, entre as quais: dimensão e representação espacial/geométrica, cor e textura;

- **Pré-Processamento:** compensa deficiências específicas, geradas no momento da aquisição da imagem. Destaca os detalhes da imagem que são de interesse para análise ou que tenham sofrido alguma deterioração;

- **Segmentação:** tem como objetivo isolar regiões de pontos da imagem pertencentes a objetos para posterior extração de atributos e cálculo de parâmetros descritivos;

- **Extração de caraterísticas:** a partir de imagens já segmentadas, esta etapa procura obter dados específicos ou padrões relevantes das regiões ou objetos destacados. Os tipos de dados, padrões ou caraterísticas mais comuns são o número total de objetos, suas dimensões e geometria;

- **Reconhecimento:** o objetivo desta etapa é reconhecer objetos na imagem, agrupando parâmetros de acordo com a sua semelhança para cada região de pixéis encontrada;

- **Interpretação:** O objetivo fulcral de um sistema de visão computacional culmina nesta etapa, em que o sistema consegue interpretar a imagem que captou;

É importante agrupar em categorias estas etapas que constituem um sistema de visão computacional, de acordo com a sua sofisticação e implementação. Considera-se então três níveis de processamento: baixo, médio e alto nível [49].

- Associa-se a processamento de **baixo nível** todos os processos que apenas utilizam as etapas de aquisição e pré-processamento (técnicas de limpeza e aprimoramento de imagem);

- O processamento de **nível médio** diz respeito a todos os processos que extraem, caracterizam e identificam componentes numa imagem resultante do processamento de baixo nível. Este nível de processamento utiliza as etapas de segmentação (técnica de agrupar áreas de uma imagem que têm características semelhantes em entidades distintas representando parte da imagem), extração de caraterísticas (técnica que lida com as características adequadas para distinguir diferentes objetos, como por exemplo, tamanho e forma) e reconhecimento (técnica que identifica objetos, como por exemplo, chave e parafuso);

- Por fim, um processamento de **alto nível** refere-se a um processo relacionado com as tarefas de cognição associadas à visão humana, onde se aplicam técnicas de controlo baseadas em algoritmos de inteligência artificial. Enquanto os algoritmos desenvolvidos em baixo e nível médio abrangem um espetro razoavelmente bem definidos de atividades, o processamento de alto nível é consideravelmente mais vago e especulativo.

A realização desta sequência de etapas é alcançada através da complexidade multidisciplinar que a visão computacional normalmente apresenta (ver Figura 3.3).

Figura 3.3 - Multidisciplinaridade em visão computacional [51].

A visão computacional abrange diversas áreas cientifico-tecnológicas que atravessam a Mecânica, a Eletrotécnica e a Informática.

Desta forma, o sistema consegue tomar decisões a partir da extração de informações do mundo real através de imagem. A decisão pode ser feita a partir de verificações simples a respeito de parâmetros extraídos dos objetos ou de algoritmos mais complexos de inteligência artificial. Contudo, para que seja possível "educar" um sistema a interpretar imagens, é necessário que previamente haja um estudo acerca da imagem em si, desde a sua captação até à sua interpretação.

3.2. Noções de imagem

A palavra "imagem" é um termo que provém do latim "imago", referindo-se à representação visual de um objeto. Uma imagem pode ser adquirida ou gerada pelo ser humano, pois uma imagem consiste em qualquer forma visual da expressão de uma ideia [50].
A aquisição da imagem por parte do ser humano é baseada em 3 sensores, os cones retinais que adquirem cada uma das 3 componentes básicas da cor como se ilustra na Figura 3.4 [52].

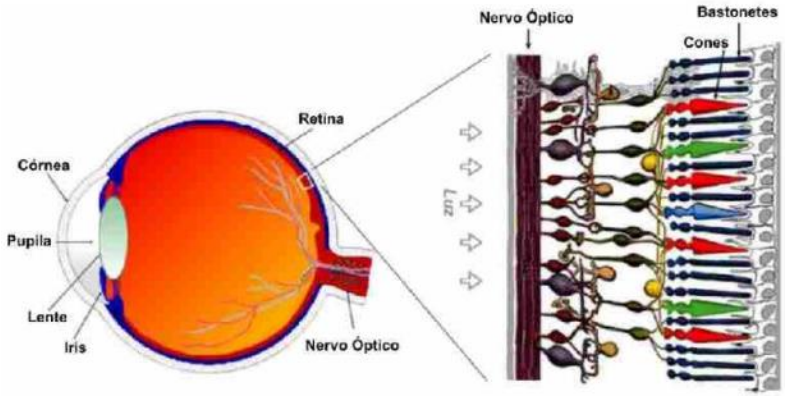


Figura 3.4 - Sensores RGB nos humanos [52].

Este sistema de cor é denominado de RGB (*Red, Green, Blue*), siga anglo-saxónica respetivamente para as cores Vermelha, Verde e Azul.

Cada um destes sensores tem diferente sensibilidade ao longo do espetro visível da luz, conforme ilustrado na Figura 3.5 [52].

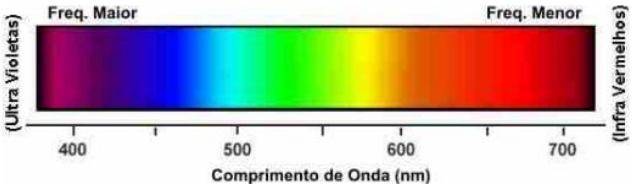


Figura 3.5 - Espetro de cor [52].

É possível então caraterizar a resposta do olho humano a cada frequência de luz, tal como ilustrado na Figura 3.6 [52].

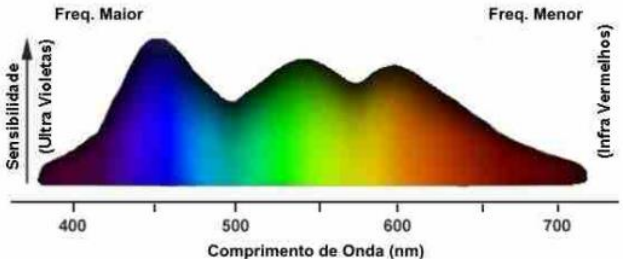


Figura 3.6 - Sensibilidade a cada cor [52].

Cada cor tem uma frequência que estimula de forma diferente os sensores de cor da visão humana. Todas as cores que se podem ver são então obtidas à custa de três cores primárias. O sistema RGB é um sistema aditivo onde outras cores (chamadas de cores secundárias) são conseguidas com recurso às combinações somadas das cores primárias tal como ilustrado na Figura 3.7 [52]

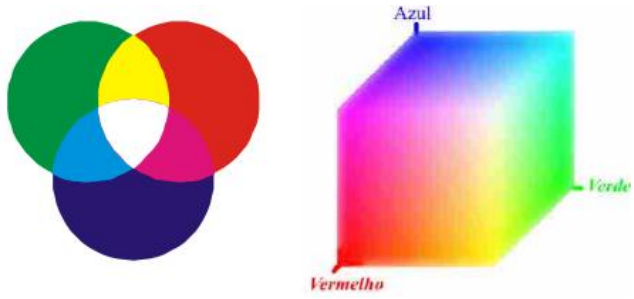


Figura 3.7 - À esquerda, cores primárias e secundárias; à direita o cubo RGB [52].

Tomando então uma definição matemática, qualquer cor pode ser definida pelo conjunto das suas componentes básicas RGB. Uma imagem é então definida pela função:

$$\begin{gathered} rgb\colon I \rightarrow \mathbb{R}^3 \\ p_{uv} \rightarrow (R,G,B) \\ R,G,B \in [0,1], u \in [0,u_{max}-1], v \in [0,v_{max}-1] \end{gathered} \tag{3.1}$$

Esta função associa a cada pixel da imagem de dimensões u_{max} por v_{max} uma cor com componentes (R, G, B) para o vermelho, verde e azul com intensidades que podem ser compreendidas entre 0 e 255 (no caso de uma resolução de 8 bits), ou normalizadas no intervalo unitário [0,1] [52]. Nas imagens binárias (Branco e Preto ou B&W), cada ponto é representado por 0 ou 1 [5].

Uma câmara digital, analogamente à visão humana, adquire uma imagem tridimensional (3D) numa primeira instância. No entanto, para o processamento computacional é necessário projetar a imagem 3D num plano bidimensional (2D). A Figura 3.8 ilustra essa projeção.

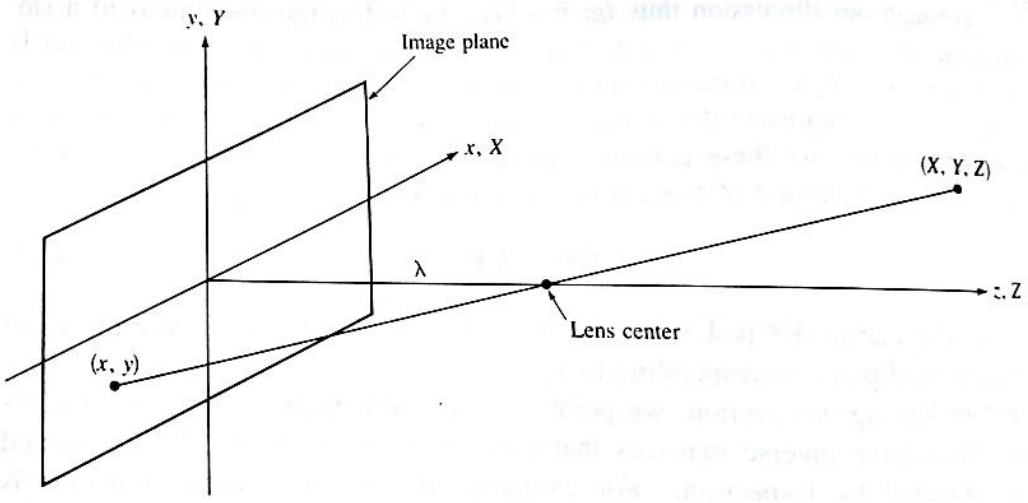


Figura 3.8 - Transformação de perspetiva [53].

Define-se o sistema de coordenadas da câmara (*x, y, z*) como tendo o plano da imagem coincidente com o plano *xy*, e a lente da câmara ao longo do eixo *z*. Assim, o centro do plano da imagem está na origem, e o centro da lente está na coordenada (0, 0, λ). A distância focal da lente é dada pelo parâmetro λ.

À semelhança do sistema em estudo na dissertação, assume-se que o sistema de coordenadas da câmara está alinhado com o sistema de coordenadas do mundo real (*X, Y, Z*). É possível então obter a relação que origina a coordenada (*x, y*) a partir da projeção de um ponto (*X, Y, Z*) num plano de imagem pelas seguintes expressões:

$$\frac{x}{\lambda} = -\frac{X}{Z-\lambda} = \frac{X}{\lambda-Z} \tag{3.2}$$

$$\frac{y}{\lambda} = -\frac{Y}{Z-\lambda} = \frac{Y}{\lambda-Z} \tag{3.3}$$

Nesta etapa, a imagem apresentada num plano bidimensional é composta por uma matriz de dados, representando projeções da cena captada pelo dispositivo de visão. Os elementos da matriz são designados por pixéis. Por definição, um **pixel** é a projeção de uma pequena porção da cena que se reduz a um único valor. Esse valor é a medida da intensidade da luz para esse pixel, como se representa na Figura 3.9 [5].

Se a imagem for RGB, a projeção será realizada para três planos (matrizes), um plano para cada umas das três cores primárias (vermelho, verde e azul). Caso a imagem seja B&W ou numa escala de cinzentos, a projeção é apenas realizada num plano (matriz). Cada matriz tem a dimensão da imagem em pixéis e a cada pixel estará associado um valor entre 0 e 255 (ou entre 0 e 1) que denota a amplitude da respetiva cor nesse pixel (um valor elevado corresponde a uma cor mais clara, um valor muito baixo corresponde a uma cor mais escura).

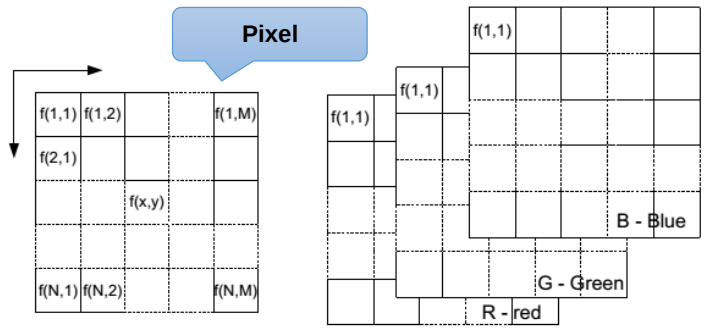


Figura 3.9 - Representação matricial de uma região da imagem [53].

3.3. Aquisição de imagens

A aquisição da imagem é efetuada principalmente por dispositivos responsáveis pela captação e pela digitalização de imagem. Neste trabalho foi utilizada uma câmara digital que já integra por si estes dispositivos. No caso das câmaras analógicas ter-se-ia que colocar circuitos adicionais nos barramentos do computador e seria este o circuito responsável pela digitalização da imagem.

A imagem captada, de um modo geral, consiste em intensidades relativas da luz que correspondem às várias porções da cena. Essas intensidades da luz constituem valores analógicos contínuos que devem ser convertidos na forma digital, como se encontra ilustrado na Figura 3.10 [5].

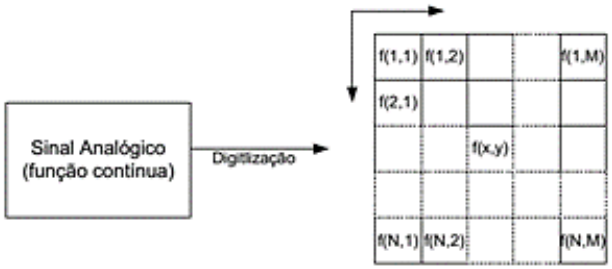


Figura 3.10 - Processo de digitalização de uma imagem [53].

Como já foi referido, ao invés de uma câmara digital, o processo de digitalização não está embutido numa câmara analógica. É neste sentido que, utilizando uma câmara analógica e captada a imagem, ela não é mais do que uma sequência de tensões correspondentes aos pixéis em que a imagem foi discretizada. Esta informação, embora já sob a forma de um sinal elétrico, não é diretamente processável por uma unidade de processamento digital, preparado para processar palavras digitais constituídas por determinado número de *bits*, frequentemente 8 e seus múltiplos.

Captada a imagem, esta precisa de ser digitalizada e guardada em memória antes de se efetuar uma nova aquisição de imagem. A cada pixel corresponde uma posição de memória de armazenamento, no caso de imagem a preto e branco, ou 3 posições de memória, no caso de imagem RGB. Normalmente cada posição de memória ocupa 1 *byte* permitindo descriminar 256 valores distintos [2].

Assim, para uma imagem em escala de cinzentos teremos 256 níveis de cinzento, enquanto que para a imagem RGB teremos 1 *byte* para cada uma das cores base, 256 níveis de cor, perfazendo um total de 256^3 = 16777216, mais de 16 milhões de cores distintas [2].

Normalmente, a interface para a câmara analógica dispõe também de memória para um primeiro armazenamento da imagem digitalizada e possui também um processador específico com uma elevada velocidade de comando do processo de digitalização e armazenamento. Este circuito de interface normalmente está contido numa placa de circuito impresso, vulgarmente designada por *Frame Grabber*, que está diretamente ligado ao barramento do computador (*Peripheral Component Interconnect* - PCI) para uma transferência mais rápida de informação [2]. A Figura 3.11 ilustra uma placa utilizada para o efeito.

Figura 3.11 - PCI-1409 Placa de aquisição de imagem analógica [54].

As câmaras digitais têm dominado o mercado nos sistemas de visão, pois estas possuem um melhor desempenho face às câmaras analógicas: o sinal analógico é muito mais suscetível aos ruídos durante a transmissão que um sinal digital; a câmara digital já inclui o processo de digitalização, capaz, portanto de fornecer diretamente ao computador a imagem digitalizada, sem necessidade da interface (responsável pela conversão do sinal analógico para digital) descrita anteriormente [55].

Consubstanciadas as etapas de captação e digitalização, segue-se para o tratamento dos dados recolhidos da imagem. Utiliza-se para representar imagens bidimensionais (2D), uma função *f(x,y)* onde *x* e *y* são variáveis discretas que representam as coordenadas (linha, coluna) dos pixéis, representando o valor de *f* em qualquer ponto *(x,y)* a intensidade de luz da imagem nesse ponto, como ilustra a Figura 3.12.

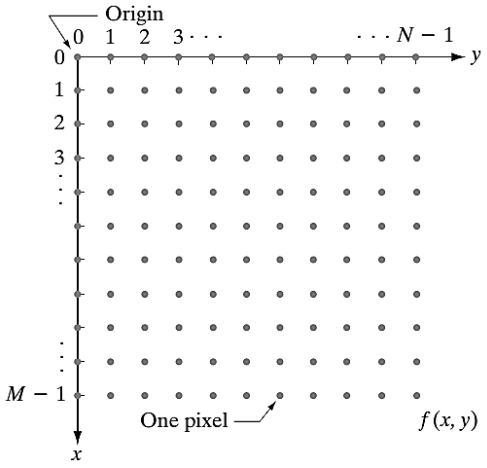


Figura 3.12 - Representação de imagens num sistema bidimensional de coordenadas x-y [49].

Os dois parâmetros de digitalização atrás descritos, o número de pixéis (espacial) e o número de níveis de intensidade de luz da imagem (amplitude) determinam a resolução com que é armazenada a imagem digitalizada. Se as imagens forem amostradas em *N* linhas e *M* colunas, os conjuntos de pixéis podem ser representados na forma matricial como representa a Figura 3.13.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Figura 3.13 - Conjunto de pixéis representados na forma matricial [49].

Grande parte dos algoritmos de processamento de imagem baseiam-se nas relações entre pixéis. É neste sentido que se pretende abordar as relações básicas existentes entre os mesmos.

3.4. Relações básicas entre elementos de imagem

Neste subcapítulo quando se faz referência a um pixel utiliza-se a letra minúscula *p*. Um pixel *p* localizado numa dada coordenada (*x, y*) possui quatro vizinhos, cujas coordenadas são dadas por:

$$(x+1, y)\,, (x-1, y)\,, (x, y+1)\,, (x, y-1) \tag{3.4}$$

Neste conjunto de pixéis, chamados quatro vizinhos de (*p)*, são normalmente designados por N_4 (*p*). As quatro diagonais dos pontos vizinhos relativamente a (*p*) são dadas pela seguinte expressão:

$$(x+1, y+1)\,, (x+1, y-1)\,, (x-1, y+1)\,, (x-1, y-1) \tag{3.5}$$

e são designadas por $N_D(p)$. Estes pontos, juntamente com os quatro pontos vizinhos definidos acima $N_4(p)$, são chamados de oito pontos vizinhos de (*p*) normalmente designados por $N_8(p)$ como ilustra a Figura 3.14 [5].

x-1,y-1	*x-1,y*	*x-1,y+1*
x,y-1	*x,y*	*x,y+1*
x+1,y-1	*x+1,y*	*x+1,y+1*

Figura 3.14 - Vizinhança N8(p) [5].

A medida de distâncias de objetos pertencentes a uma imagem é efetuada, por exemplo, com o auxílio da **distância Euclidiana** (D_e), que nos dá a distância entre pixéis. Dados dois pixéis *p* e *q* de coordenadas (x_1, y_1) e (x_2, y_2), a distância entre os pixéis *p* e *q* é definida pela expressão:

$$D_e(p, q) = [(x_1 - x_2)^2 + (y_1 - y_2)^2]^{1/2} \tag{3.6}$$

3.5. Processamento e análise da imagem

Após ter sido efetuada a aquisição da imagem, a unidade de processamento digital deve ser programada para processar a informação extraída da imagem. Considerando a grande quantidade de dados que uma imagem detém, torna-se conveniente reduzir a quantidade de informação em excesso. Para reduzir a quantidade de informação de um sistema de visão podem ser utilizadas diversas técnicas, tais como:

- Redução de níveis de cor;
- Enquadramento;
- Iluminação.

Redução de níveis de cor

Caso seja possível, reduz-se o número de níveis de cinzento usado pelo sistema de visão computacional. Por exemplo, um registo de 8 bits para cada pixel teria 2^8 = 256 níveis de cinzento. Dependendo da exigência da aplicação, a redução do número de níveis de cinzento implica um número menor de bits para representar a intensidade da luz dos pixéis. O uso de 4 bits iria reduzir o número de níveis de cinzento para 16. Este tipo de conversão permite reduzir de forma significativa o volume de processamento de dados sobre a imagem.

Enquadramento

O enquadramento envolve o uso de apenas uma porção da imagem armazenada na matriz para processamento e análise. Essa porção é denominada de "janela" ou *Region Of Interest* (ROI). Uma "janela" retangular é selecionada para delimitar o componente de interesse e somente os pixéis dentro da janela são analisados.

Iluminação

A boa iluminação da peça cuja imagem se pretende captar é essencial. Assim sendo, deve-se proporcionar uma distinção clara entre o fundo (cenário) e o objeto, ou seja, o fundo deve ser escuro se o objeto é claro ou então o inverso. A iluminação deve também ser uniforme de modo a não criar sombras nem variações de luz na superfície do objeto, permitindo assim que, para além das duas dimensões, se possa ainda obter alguma

informação sobre o relevo. Está comprovado que uma boa iluminação substitui uma grande parte de processamento a aplicar numa imagem. Exemplo disso, são os operadores de contraste, que em caso da utilização de uma boa iluminação, é possível evitar a sua aplicação em algumas situações, otimizando o desempenho do sistema de visão [2].

Outras técnicas, mais específicas e vulgarmente utilizadas, incluem os seguintes procedimentos [5]:

- Conversão para escala de cinzentos;
- Técnica da limiarização;
- Operadores morfológicos;
- Operações estruturais na imagem;
- Reconhecimento do objeto.

3.5.1. Conversão para escala de cinzentos

O reconhecimento intrínseco de imagens implica que as imagens utilizadas sejam previamente tratadas para que as caraterísticas mais significativas das mesmas sejam salientadas em detrimento de outras que sejam irrelevantes à tarefa em causa. Assim, neste trabalho, a maioria das imagens serão trabalhadas em preto e branco, onde previamente foram convertidas para tons de cinzento, ou seja, o valor de cada pixel estará compreendido entre 0 e 255 que denotará o seu nível de cinzento.

A transformação entre o modelo RGB e o modelo *Grayscale* (escala de cinzentos) será feita neste trabalho de acordo com a recomendação BT.601-7 de 2011 da ITU-R (*Radiocommunication Sector of International Telecomunication Union*) [56].
O método utilizado consiste na ponderação da amplitude de cada cor por um fator pré-determinado que valoriza cada componente de cor de forma semelhante à que o olho humano faz. O valor de cada pixel em escala de cinzentos é determinado segundo a equação:

$$Grayscale(x,y) = R(x,y) \times 0.299 + G(x,y) \times 0.587 + B(x,y) \times 0.114 \qquad (3.7)$$

Onde, R(*x, y*), G(*x, y*) e B(*x, y*) correspondem respetivamente às amplitudes do pixel para as cores vermelho, verde e azul.

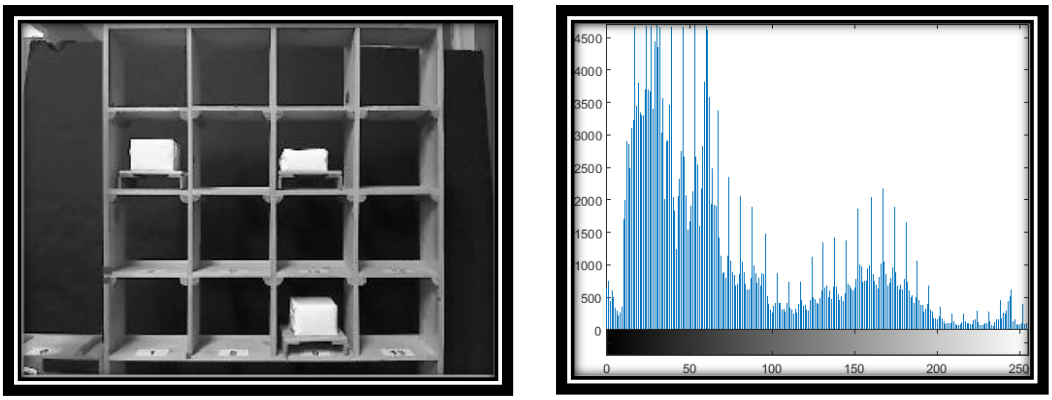
3.5.2. Técnica da limiarização

Após a conversão para escala de cinzentos, a binarização da imagem, ou seja, a sua conversão para uma imagem apenas a preto e branco, é essencial para maximizar as capacidades de deteção necessárias neste algoritmo. De um modo resumido, a técnica da limiarização é uma técnica de conversão binária na qual cada pixel é convertido num valor binário, (0) preto ou (1) branco. O valor fronteira entre o preto e o branco pode ser otimizado através da utilização de um histograma de frequência da imagem. Esta separação, efetuada com base na escolha de um ponto de corte ("*threshold*"), permite evidenciar certos aspetos da imagem em detrimento de outros que não sejam relevantes para a aplicação em causa. O processo de limiarização produz como resultado uma imagem binária, o qual também pode ser denominado como binarização.
A imagem resultante da técnica do limiar é originada a partir da seguinte definição [49]:

$$g(x,y) = \begin{cases} 1, se\ f(x,y) > T \\ 0, se\ f(x,y) \leq T \end{cases} \tag{3.8}$$

Em que $g(x,y)$ é a imagem limiar, $f(x,y)$ é a intensidade do ponto (x,y). Deste modo, examinando a imagem $g(x,y)$ encontram-se pixéis de valor 1, correspondentes aos objetos, enquanto pixéis com valor 0 correspondem ao fundo. A variável *T* é o ponto de corte ou *threshold* que separa os dois grupos de intensidade.

A Figura 3.15 a) expõe uma imagem regular em que cada pixel tem uma tonalidade específica entre 256 possíveis níveis de cinzento. O histograma da Figura 3.15 b) representa em ordenadas a frequência (número de pixéis) correspondentes aos valores com níveis de cinza representados em abcissas.



a) b)

Figura 3.15 - Obtenção de uma imagem binária pela técnica do limiar. a) imagem com todos os níveis de cinzento; b) histograma da imagem.

Após a aplicação desta técnica, todos os píxeis com intensidade menor do que o valor de *threshold* seleccionado são convertidos para a cor preta e os restantes para a cor branca. Um exemplo de aplicação desta técnica pode ser vista na figura abaixo (ver Figura 3.16).

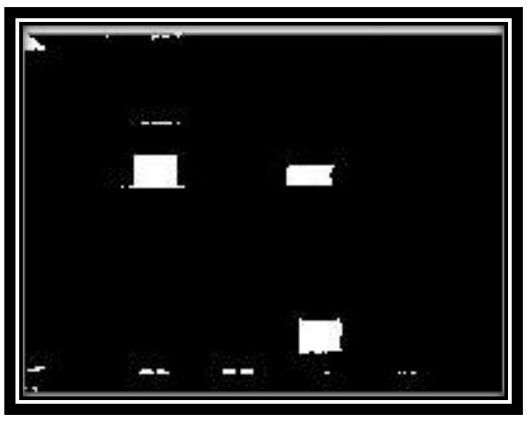


Figura 3.16 - Resultado da técnica do limiar.

Para melhorar a capacidade de diferenciação, são utilizadas técnicas especiais de iluminação para se conseguir um elevado contraste, ou então métodos automáticos de determinação do *threshold* a utilizar, como o método de limiarização bimodal de *Otsu* [57]. No sistema em estudo, o facto de se utilizar peças brancas e um fundo de cor preta facilitou o processo de binarização, pois existe uma grande diferença de intensidade luminosa entre ambos. Quando existem vários objetos com diferentes níveis de intensidade, é aconselhável, por exemplo, dividir a imagem em áreas retangulares menores (ROI) e aplicar a técnica do limiar para cada janela analisada.

3.5.3. Operadores morfológicos

Os operadores morfológicos são aplicados em imagens binárias, e são utilizados com o objetivo de melhorar a imagem após um processo de segmentação, isto é, realça ou compensa imperfeições específicas de uma imagem sujeita a segmentação.
Existem dois operadores morfológicos usualmente utilizados - a “Erosão” e a “Dilatação”. Estes operadores atuam num alcance definido por cada elemento estruturante, que servirá de critério para as modificações ou deteções morfológicas das estruturas ([58] [59]). No sistema em estudo utilizaram-se dois operadores morfológicos, a “Erosão” e o operador “Fecho” que é baseado na “Erosão” e na “Dilatação”. Para outras situações particulares poder-se-á utilizar outros operadores morfológicos, tais como: preenchimento de espaços, abertura, extração de elementos conectados, esqueletos, etc.

- **Dilatação:** permite dilatar a imagem, partindo-se de uma imagem com o objeto preto sobre o fundo branco, por exemplo, conseguida por aplicação do operador binarização. O operador percorre a imagem pixel-a-pixel com um elemento estruturante dando origem a duas situações: se o elemento central do elemento estruturante estiver sobre o pixel branco da imagem, o elemento estruturante da nova imagem torna-se transparente; se o pixel for preto, todo o elemento estruturante assume essa cor na nova imagem.
  A dilatação conseguida depende do formato e dimensão do elemento estruturante, ou seja, se tiver a forma como se apresenta na Figura 3.17, a dilatação é feita em todas as direções. Se for, por exemplo, constituído por uma linha de três pixéis a

dilatação apenas se dá na horizontal; se for uma coluna com três pixéis, a dilatação é apenas na vertical [2].

- **Erosão:** este operador funciona de forma análoga à dilatação, logo genericamente leva a uma diminuição do tamanho dos objetos, em função de um crescimento do fundo da imagem. Se pelo menos um pixel do elemento estruturante coincidir com o fundo da imagem original, então o resultado desta operação local leva a que na imagem de saída este pixel tenha o valor do fundo. Ou seja, no final os pixéis que na imagem original faziam parte do fundo, continuam a fazer parte deste, e alguns pixéis que faziam parte dos objetos passam a fazer parte do fundo. Como os pixéis em causa são normalmente aqueles que fazem parte dos contornos dos objetos, isto leva a que diminua o tamanho destes, podendo mesmo existir divisão do objeto em alguns casos ([58], [59]).

A Figura 3.17 representa o princípio de funcionamento dos operadores morfológicos descritos.

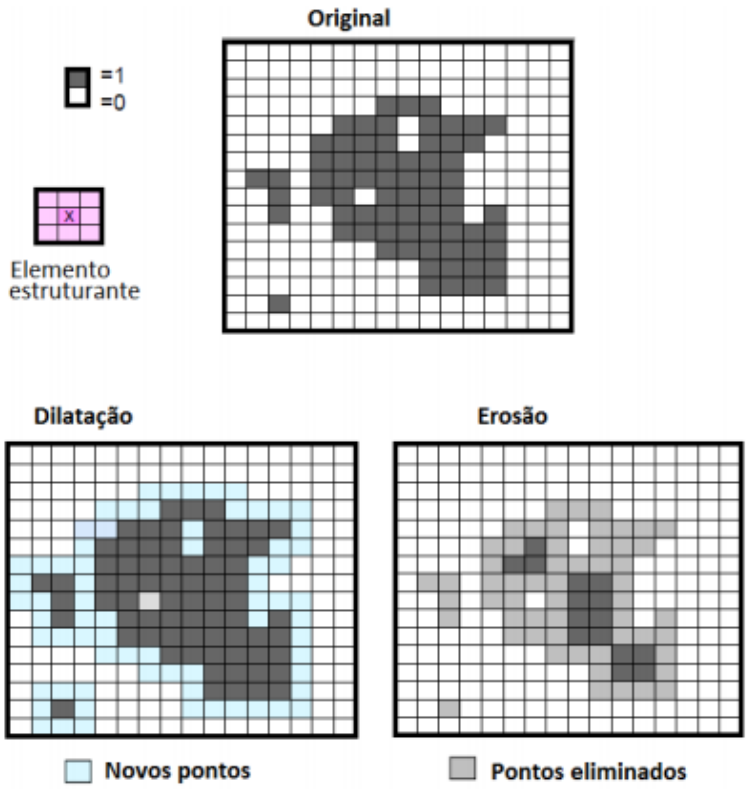


Figura 3.17 - Princípio de funcionamento dos operadores dilatação e erosão [59].

- **Fecho:** este operador efetua uma “dilatação” seguida de uma “erosão” (ver Figura 3.18). Apesar de também suavizar alguns contornos do objeto, a operação de fecho leva a que pequenas distâncias entre objetos, pequenos buracos ou pequenas falhas ao longo do seu contorno sejam preenchidos e passem a fazer parte dos mesmos.

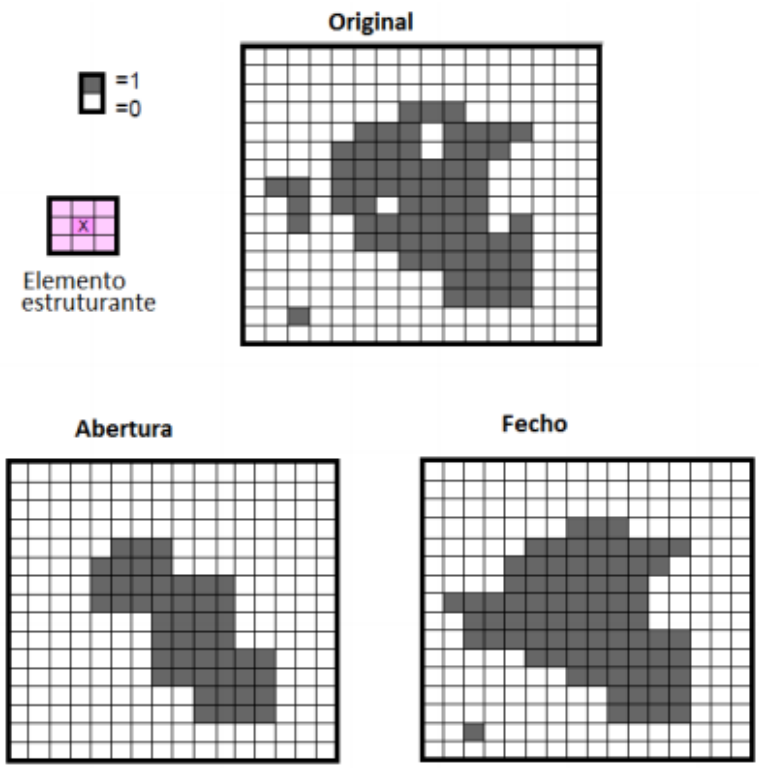
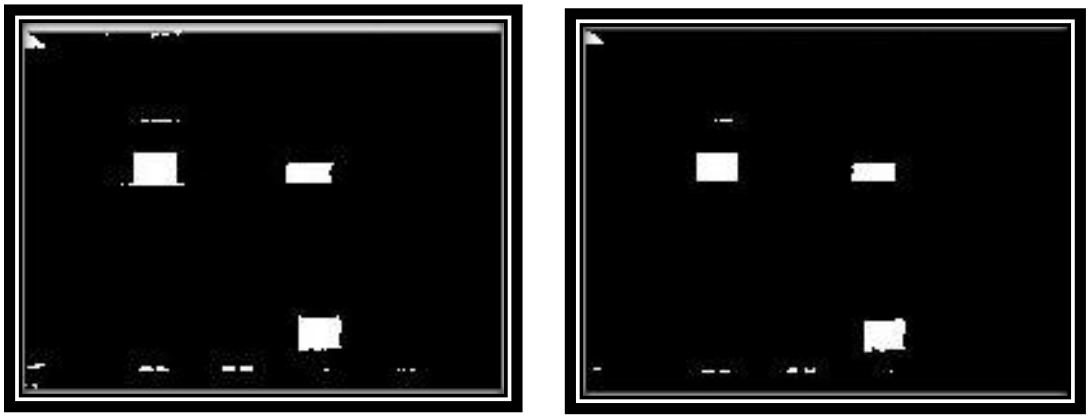


Figura 3.18 - Princípio de funcionamento dos operadores abertura e fecho [59].

Em resumo, com o operador “erosão” conseguiu-se eliminar vários pixéis indesejados em torno dos objetos, e com o operador “fecho” conseguiu-se melhorar o contorno dos objetos (devido à componente de “dilatação”).

A Figura 3.19 ilustra o resultado da aplicação dos operadores morfológicos após a imagem de entrada ter sido sujeita à binarização. É de realçar que são notáveis as diferenças, pois os objetos ficaram melhor definidos quanto à sua forma geométrica.



a) b)

Figura 3.19 - Resultado da aplicação dos operadores morfológicos. a) imagem binária; b) imagem após aplicação de operadores "erosão" e "fecho".

3.5.4. Operadores estruturais em imagens

Em aplicações de visão computacional, é frequentemente necessário distinguir um objeto de outro. É geralmente analisado por meio de características que distinguem de forma específica esse objeto. Uma característica, no contexto dos sistemas de visão, é um parâmetro único que permite uma fácil comparação e identificação. Algumas características de objetos que podem ser usadas na visão computacional incluem a *área*, o *diâmetro*, *centro de gravidade* e o *perímetro* [5].

As técnicas disponíveis para extrair valores de características de objetos bidimensionais podem ser divididas, de um modo geral, em duas categorias: as que tratam de características de contorno e as que tratam de características de área.

As várias características podem ser usadas para identificar o objeto ou peça e determinar a sua localização e/ou orientação [5]:

- **Área:** representa o somatório do número de pixéis brancos em fundo preto que constitui a região e é obtido pela seguinte equação [2]:

$$\text{Área} = \sum_{x=1}^{m} \sum_{y=1}^{n} f(x, y) \tag{3.9}$$

- **Perímetro:** delimita uma área específica, e pode ser determinado anotando a diferença em intensidade de pixéis no contorno e simplesmente contando todos os pixéis na região segmentada que são adjacentes aos pixéis que não estão nessa região, isto é, no outro lado do contorno [5];

- **Diâmetro do círculo:** maior distância entre dois pixéis do objeto [2];

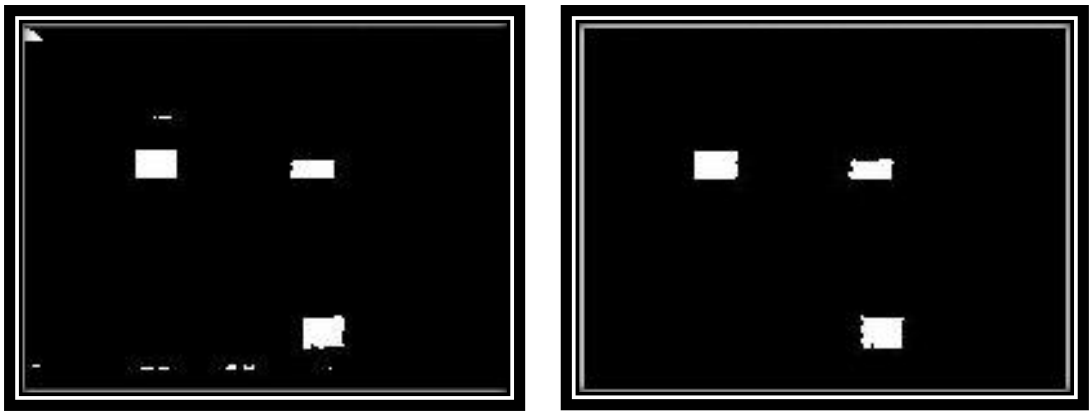
- **Centro de gravidade:** definido pela média das coordenadas dos *n* pixéis que compõem a imagem. É dado pela seguinte equação [2]:

$$CG_x = \frac{1}{n} \sum_x x \quad \text{e} \quad CG_y = \frac{1}{n} \sum_y y \tag{3.10}$$

Um objetivo importante na seleção dessas características é que elas não devem depender da posição ou da orientação do objeto. O sistema de visão não deve depender do facto do objeto ser apresentado numa posição conhecida e fixa em relação à câmara [5].

Extração de objetos indesejáveis: após a análise da Figura 3.19 denota-se que existem vários pixéis indesejados que perturbam a observação dos objetos que realmente interessam para a contabilização da ocupação do sistema. Tendo em consideração tal facto, contabilizam-se as áreas de todos os objetos e apenas se deixam visíveis os objetos que satisafazem uma dada condição.

Deste modo, a Figura 3.20 representa o resultado da aplicação de um processo disponível pela utilização dos operadores estruturais, em que se tira partido da extração da área dos objetos. Nesta figura já se encontram presentes apenas os objetos que interessam para análise do sistema.



a) b)

Figura 3.20 - Resultado da aplicação de um processo dos operadores estruturais: extração de objetos indesejáveis. a) imagem após aplicação de operadores "erosão" e "fecho"; b) imagem tratada.

Cálculo das distâncias entre objetos: conforme foi mencionado anteriormente, "*O sistema de visão não deve depender do objeto ser apresentado numa posição conhecida e fixa em relação à câmara*", é de extrema importância que o objeto a ser estudado possa sofrer ligeiros movimentos em relação à câmara ou vice-versa. No capítulo 4 apresentar-se-á a solução encontrada para esse problema.

Os operadores estruturais são cruciais no sistema em estudo, em que numa perspetiva global, são responsáveis pela extração de objetos indesejáveis e pelo cálculo das distâncias entre objetos, podendo de igual modo eliminar um conjunto de informação de dados que não são relevantes para a solução do sistema.

3.5.5. Reconhecimento do objeto

O passo seguinte no processamento de dados de imagem é identificar o objeto que a imagem representa. Tal como foi mencionado anteriormente, os problemas de identificação são resolvidos usando as informações das características extraídas. O algoritmo de

reconhecimento deve ser suficientemente dotado para identificar especificamente o objeto [5].
As técnicas de reconhecimento de objetos usadas atualmente na indústria podem ser classificadas em duas grandes categorias:

- Técnica *pattern matching*;
- Técnicas estruturais.

A **técnica *pattern matching*** é uma técnica de reconhecimento de padrões que serve para classificar objetos de uma imagem em categorias predeterminadas. O problema nesta técnica é combinar o objeto com uma configuração de características armazenadas, definida como modelo padrão - obtido durante o procedimento de treino no qual o sistema de visão é programado para objetos previamente conhecidos.
Esta técnica é aplicável se não houver necessidade de um grande número de modelos padrões. O procedimento é baseado no uso de um número suficiente de características para minimizar a frequência de erros no processo de classificação. As características do objeto na imagem (por exemplo, a área, o diâmetro e a relação de forma) são comparadas com valores correspondentes armazenados e esses valores constituem o modelo padrão armazenado. Quando se encontra uma combinação, permitindo certas variações estatísticas no processo de comparação, então o objeto foi corretamente classificado [5].

As **técnicas estruturais** de reconhecimento consideram relações entre características ou contorno de um objeto. Por exemplo, se a imagem de um objeto puder ser subdividida em quatro linhas retas ligadas aos seus pontos terminais e as linhas ligadas estão perpendiculares, então o objeto é um retângulo. Esse tipo de técnica, conhecida como reconhecimento sintático de configuração é uma das técnicas estruturais mais usada [5].
O reconhecimento completo de um modelo pode exigir muito tempo de cálculo e é mais apropriado para procurar regiões ou contornos mais simples dentro de uma imagem. Essas regiões mais simples podem então ser usadas para extrair as características exigidas. A maioria dos sistemas de visão em robótica usa essa abordagem para reconhecimento de objetos bidimensionais. Os algoritmos de reconhecimento são usados para identificar cada objeto segmentado numa imagem e atribuir-lhe uma classificação (por exemplo, porca, parafuso, falange, etc.) [5].

Existem muitas outras técnicas de processamento de imagem, algumas abrangentes a vários problemas e outras muito específicas. Procurou-se neste capítulo fazer um apanhado de algumas das técnicas mais comuns e que serão usadas nas soluções aqui apresentadas.

4. SOLUÇÕES DESENVOLVIDAS

4.1. Introdução

Neste capítulo serão apresentadas as soluções desenvolvidas para o controlo do sistema em estudo apresentado. Quer para a solução computacional quer para a solução com microcomputador, serão apresentados os *softwares* e bibliotecas utilizadas, o equipamento utilizado, os vários algoritmos implementados e a descrição funcional. De notar que foi necessário alterar o algoritmo já existente no autómato, pois em determinados aspetos, o seu atual funcionamento era imcompatível com as novas funcionalidades implementadas no sistema.

Atualmente, em aplicações industriais, são usualmente utilizados dois tipos de plataformas computacionais, os computadores convencionais e os chamados *single-board computers*.

Os computadores convencionais são comercializados numa das duas formas: *desktops* ou computadores portáteis. Os *desktops* são compostos por uma torre central que incorpora vários componentes no seu interior e uma série de periféricos acoplados (teclado, rato, colunas e monitor). Este tipo de solução só é apropriado para sistemas que suportem o seu volume estrutural. Já os computadores portáteis, possuindo um volume estrutural muito mais reduzido (já incluem vários periféricos, tais como teclado, rato, colunas e monitor no seu *chassi*), são normalmente utilizados no âmbito industrial, e na maior parte das vezes como ferramenta de programação/teste dos sistemas industrias. Além do seu volume reduzido em comparação com os *desktops*, os computadores portáteis também incluem uma bateria, permitindo o seu uso durante um período considerável sem a conexão permanente a uma fonte de alimentação externa [60].

A outra opção de plataforma computacional são os *single-board computers*. São pequenas placas eletrónicas (do tamanho de um cartão de crédito) que partilham uma arquitetura comum com um computador convencional. Estes incluem um processador, memória RAM, unidade gráfica e unidade de *Inputs*/*Outputs* (entradas/saídas). Este tipo de computadores também possibilitam a conexão de periféricos usuais, tais como o teclado (conexão USB), rato (conexão USB) e monitor (conexão HDMI).

Os *single-board computers* funcionam normalmente com recurso a um processador ARM (similar aos utilizados nos *smartphones*) [60].
Neste trabalho desenvolveram-se duas soluções (em duas unidades de processamento digital diferentes), em que numa delas se utilizou o computador portátil e na outra o *Raspberry Pi*.
Antes de apresentar as soluções desenvolvidas, será feita uma descrição do simulador de armazenamento existente.

4.2. Simulador de armazenamento existente

4.2.1. Descrição do simulador

O simulador de armazenamento automático existente no Laboratório de Automação e Robótica (ADEEEA) do Instituto Superior de Engenharia de Lisboa (ISEL) possibilita a arrumação ou extração de peças de forma automatizada. O processo automatizado é controlado por um autómato programável, cujo funcionamento depende das entradas que lhe estão associadas, permitindo agir sobre os atuadores que estão acoplados às suas saídas. O PLC tem como periférico uma consola HMI, que possibilita a interação do utilizador no funcionamento do sistema. A Figura 4.1 demonstra o aspeto físico do sistema.

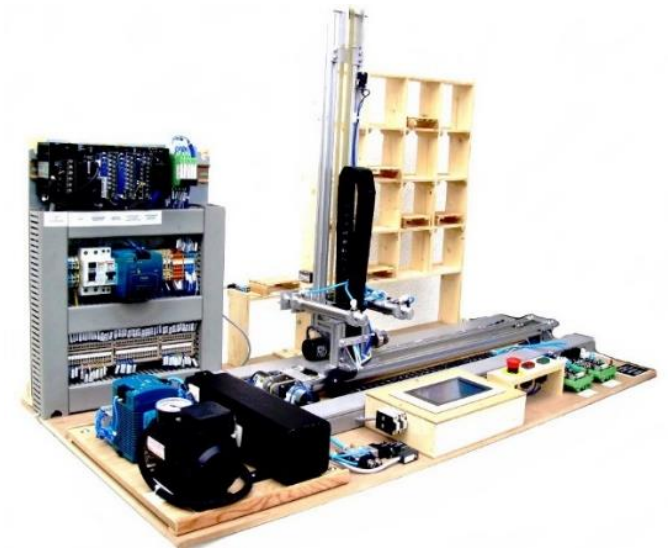


Figura 4.1 - Simulador de armazenamento automático existente no ISEL.

O simulador em questão foi o resultado de um projeto final de curso realizado pelos alunos Filipe Martins, Nº 19568 e Jaidev Bradacim, Nº 19590 no ano de 2006 (ISEL-ADESPA), com orientação do Prof. Doutor João Palma.

O sistema de armazenamento é composto por um quadro elétrico onde se encontra a entrada de energia elétrica acompanhada das suas proteções (fusíveis e disjuntores), proteções essas do autómato programável e das suas interfaces, dos sensores e dos pré-atuadores (relés) e do variador de velocidade, entre outros equipamentos auxiliares (*rack* de ligadores). O fornecimento de ar comprimido aos cilindros pneumáticos do sistema é garantido por uma pequena unidade de ar comprimido localizada junto ao simulador.
Pode-se ainda aferir que toda a componente mecânica móvel é suportada por uma estrutura de alumínio que possibilita o movimento de três eixos (*X, Y e Z*). Nos dois primeiros eixos, *X* e *Y*, a transmissão de movimento é executada através de motores de corrente contínua, conjuntos de correias dentadas e barras de guiamento, e as medições das posições são obtidas através de codificadores incrementais. No terceiro eixo, Z, a transmissão de movimento é efetuada através de cilindros pneumáticos de duplo efeito.
A interação do operador com o sistema efetua-se com recurso a uma consola tátil presente no painel de comando, no qual também se encontram todos os botões necessários para o funcionamento do sistema (botões de pressão e botoneira de emergência). O recurso à consola tátil permite, ao operador, selecionar o local onde pretende armazenar uma determinada peça ou retirar uma determinada peça da estante de armazenamento.

Será apresentada uma breve descrição do equipamento que foi mencionado anteriormente.

Autómato Programável – O PLC instalado neste sistema é um autómato modular do fabricante *OMRON*, constituído pelos seguintes módulos:

- Fonte de alimentação (CJ1W-PA202);
- CPU (CJ1M-CPU12);
- Entradas digitais (CJ1W-ID211);
- Saídas digitais (CJ1W-OC211);
- Contadores (CJ1W-CT021).

Para melhor exemplificar, apresenta-se a Figura 4.2 que ilustra os vários módulos que constituem o autómato programável utilizado no sistema.

Figura 4.2 - Módulos que constituem o PLC modular utilizado no sistema.

Motor DC – São utilizados dois motores de corrente contínua, sendo um deles responsável pelo movimento do eixo *x* (motor 1) e outro pelo movimento do eixo *y* (motor 2). São alimentados a 24V DC através de um circuito eletrónico de controlo de velocidade e sentido de rotação. A Figura 4.3 ilustra o pormenor dos dois motores inseridos no sistema.

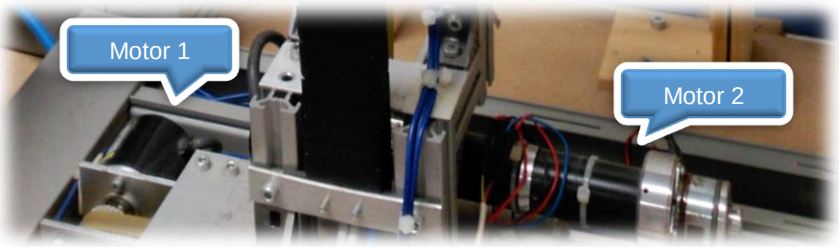


Figura 4.3 - Motores DC utilizados no sistema.

Codificador Incremental (*Encoder*) – Os *encoders* são utilizados para dar informação sobre a posição e a velocidade dos motores de corrente contínua apresentados anteriormente. Estes dispositivos estão acoplados no veio dos motores, gerando impulsos que permitem conhecer a sua localização.

O número de impulsos corresponde à medida da distância, ou seja:

- Contando os impulsos e conhecendo a posição inicial, obtém-se a posição e a velocidade em cada momento;
- O sentido de rotação do *encoder* é determinado por dois sinais desfasados de 90º, em avanço ou atraso em função da direção.

O *encoder* incremental utilizado no sistema tem uma resolução de 500 impulsos por rotação, com uma resposta de 10kHz e a resolução de posição encontra-se nos 0.15mm/impulso. Em baixo mostra-se o pormenor de um *encoder* inserido no sistema, Figura 4.4.

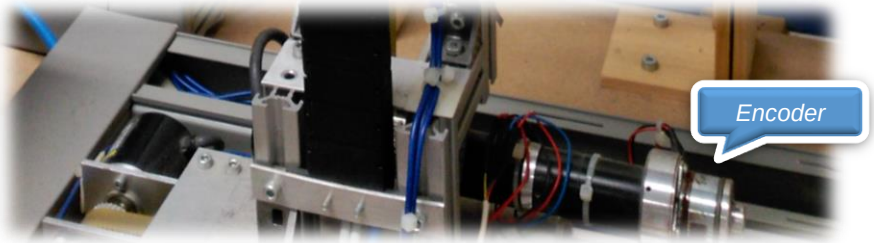


Figura 4.4 - Encoder incremental utilizado no sistema.

Módulo de ar comprimido – Este módulo fornece ar comprimido para os cilindros, para que estes possam atuar de forma a possibilitar o movimento do eixo z (colocação e extração de peças dos alvéolos). O conjunto é constituído por:

- Cilindros de duplo efeito;
- Compressor de ar;
- Válvula 5/2 monoestável;
- Válvula reguladora de pressão;
- Regulador de caudal.

A Figura 4.5 ilustra a válvula 5/2 monoestável utilizada no sistema.

Figura 4.5 - Válvula 5/2 monoestável utilizada no sistema.

A Figura 4.6 ilustra os cilindros de duplo efeito utilizados no sistema.

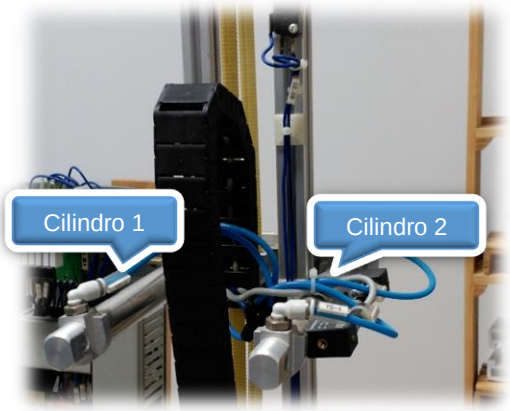


Figura 4.6 - Cilindros de duplo efeito utilizados no sistema.

A Figura 4.7 ilustra o compressor de ar utilizado para alimentar os cilindros pneumáticos de duplo efeito.

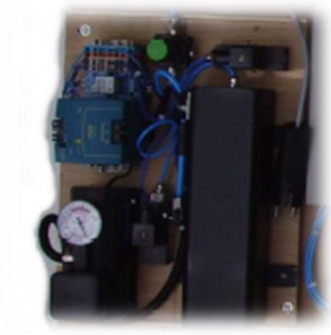


Figura 4.7 - Compressor de ar utilizado no sistema.

Sensores fim de curso – Estes sensores, Figura 4.8, são utilizados para detetar os extremos dos eixos *X, Y* e *Z*.

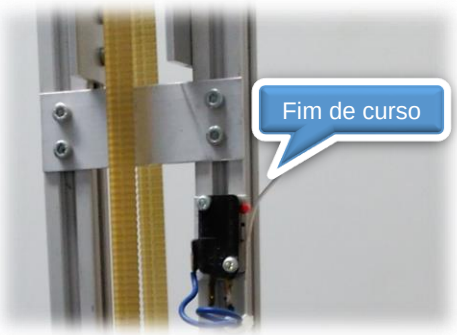


Figura 4.8 - Fins de curso utilizados no sistema.

Consola HMI – Este tipo de equipamento permite a comunicação entre o operador e a máquina, de forma a possibilitar não só definição e alteração de parâmetros, como também supervisionar o sistema. A consola instalada possui ecrã tátil tal como se pode observar na Figura 4.9.

Figura 4.9 - Consola HMI utilizada no sistema.

4.2.2. Instruções de utilização do simulador

De forma a melhor elucidar o funcionamento do simulador do sistema de armazenamento apresenta-se uma breve explicação do guia de utilização, acompanhado de algumas imagens. Ao iniciar o equipamento, a consola tátil apresenta a seguinte tela - Figura 4.10.

Figura 4.10 - Ecrã inicial.

Ao pressionar o botão “início”, surge no ecrã os dados sobre a autoria do sistema - Figura 4.11.

Figura 4.11 - Ecrã que apresenta os dados sobre os autores do sistema.

Ao pressionar o botão “Iniciar” surgem no ecrã cinco botões distintos como ilustra a Figura 4.12.

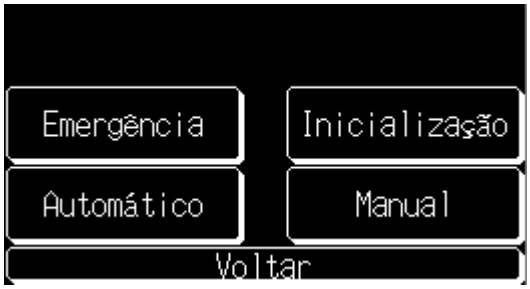


Figura 4.12 - Seleção do modo de funcionamento.

Excluindo o botão “Voltar”, que volta para o ecrã ilustrado na Figura 4.11, os restantes quatros botões assumem funções diferentes:

Botão de “Emergência”: É utilizado para ativar e indicar o estado de emergência, ou seja, acende a luz sempre que este ou a botoneira de emergência no painel de comando esteja pressionada. Esta indicação também está presente no painel de comando através do sinalizador de emergência.

Botão de “Inicialização”: É utilizado para inicializar o sistema, ou seja, antes de começar qualquer operação, é necessário efetuar o reconhecimento e o posicionamento dos eixos nas posições definidas como sendo as iniciais. Este botão acende a luz durante o processo de inicialização, desligando-se ao terminar o processo.

Botão “Automático”: É utilizado para colocar o sistema em funcionamento. Ao entrar neste modo surge um novo ecrã como se apresenta na Figura 4.13.

Figura 4.13 - Modo de funcionamento - Automático.

Ao pressionar nos botões representados por uma seta, os números dos alvéolos vão crescendo ou diminuindo, selecionando então o alvéolo de onde se pretende retirar a peça e onde se pretende inseri-la. Em alternativa existe a opção de selecionar diretamente o alvéolo e verificar o estado de ocupação de cada posição do sistema de armazenamento. Esta alternativa é possível quando se pressiona no botão com aspeto de estante.

A Figura 4.14 representa o ecrã onde se seleciona diretamente os alvéolos para retirar as peças, e a Figura 4.15 representa o ecrã onde é possível selecionar diretamente os alvéolos de destino para as peças.

Figura 4.14 - Seleção direta do alvéolo para retirar peça (alvéolo de origem).

Figura 4.15 - Seleção direta do alvéolo para inserir peça (alvéolo de destino).

Os alvéolos estão numerados de 0 a 16, sendo o número 0 reservado para a zona de receção e entrega de peças.
Se se pretende armazenar alguma peça num alvéolo já ocupado, surgirá uma mensagem de erro no ecrã, tal como está ilustrado na Figura 4.16.

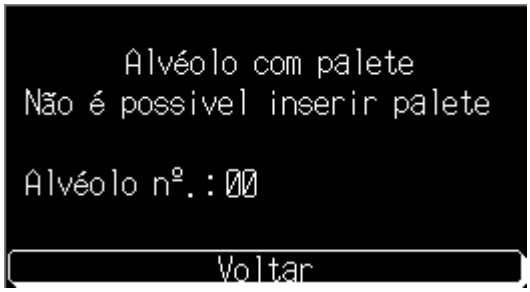


Figura 4.16 - Mensagem de erro – alvéolo ocupado.

Por conseguinte, se se pretende retirar alguma peça de um alvéolo que esteja vazio, surgirá uma mensagem de erro no ecrã, tal como está apresentado na Figura 4.17.

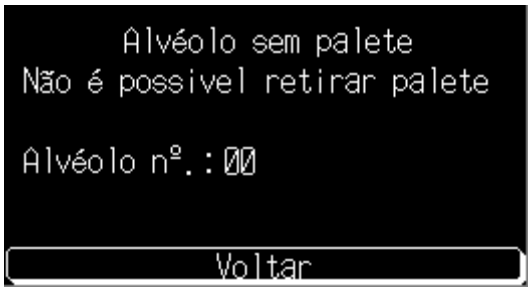


Figura 4.17 - Mensagem de erro – alvéolo vazio.

Toda a componente de controlo do sistema é executado por ação de um autómato programável. Quando o utilizador pretende efetuar uma dada operação, quer seja armazenar ou retirar peças, o autómato atualiza o seu próprio mapa de memória tendo por base a ação efetuada pelo utilizador. Só o alvéolo 0 é que tem instalado um sensor (sensor fotoeléctrico) que identifica a presença de peça.

Para limpar o mapa de memória que contém o registo de ocupação do sistema de armazenamento, é necessário regressar ao menu inicial - Figura 4.10 - e pressionar no botão “Limpar”.

Em funcionamento normal, ao pressionar a botoneira de emergência todas as operações param de imediato. Para retomar novamente o funcionamento é necessário desativar a botoneira de emergência e efetuar o rearme, pressionando o botão verde localizado no painel de comando - procedimento representado na Figura 4.18.

Figura 4.18 - Painel de comando.

Botão "Manual": É utilizado para memorizar as posições dos vários alvéolos do sistema. Ao entrar neste modo surge um novo ecrã - Figura 4.19.

Figura 4.19 - Parametrização da posição de cada alvéolo.

Ao pressionar no botão "Gravar" surgirá um novo ecrã (ver Figura 4.20). Ao entrar neste ecrã é então possível confirmar as alterações efetuadas, em que dá hipótese ao utilizador para confirmar (pressionando no botão "CONFIRMAR") ou cancelar a operação (pressionando no botão "CANCELAR").

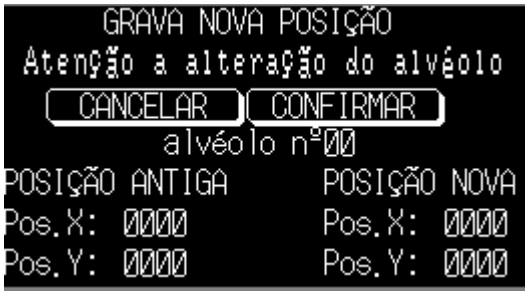


Figura 4.20 - Confirmação da nova parametrização.

4.2.3. Limitações do simulador

O simulador existente no Laboratório de Automação e Robótica utiliza apenas as potencialidades do autómato (memórias locais) para determinar o estado de ocupação do armazém, o qual não se torna suficientemente eficaz quanto ao controlo de *stock* de peças. Tendo em consideração a necessidade de melhorar o desempenho deste simulador de armazenamento automático foi desenvolvido um *upgrade* ao sistema, introduzindo a capacidade de detetar automaticamente e em tempo real a presença ou ausência de peças nos respetivos alvéolos, tal como é exposto na Figura 4.21.

Pretende-se com a introdução destas aplicações que o desempenho do sistema aumente significativamente face ao seu estado atual.

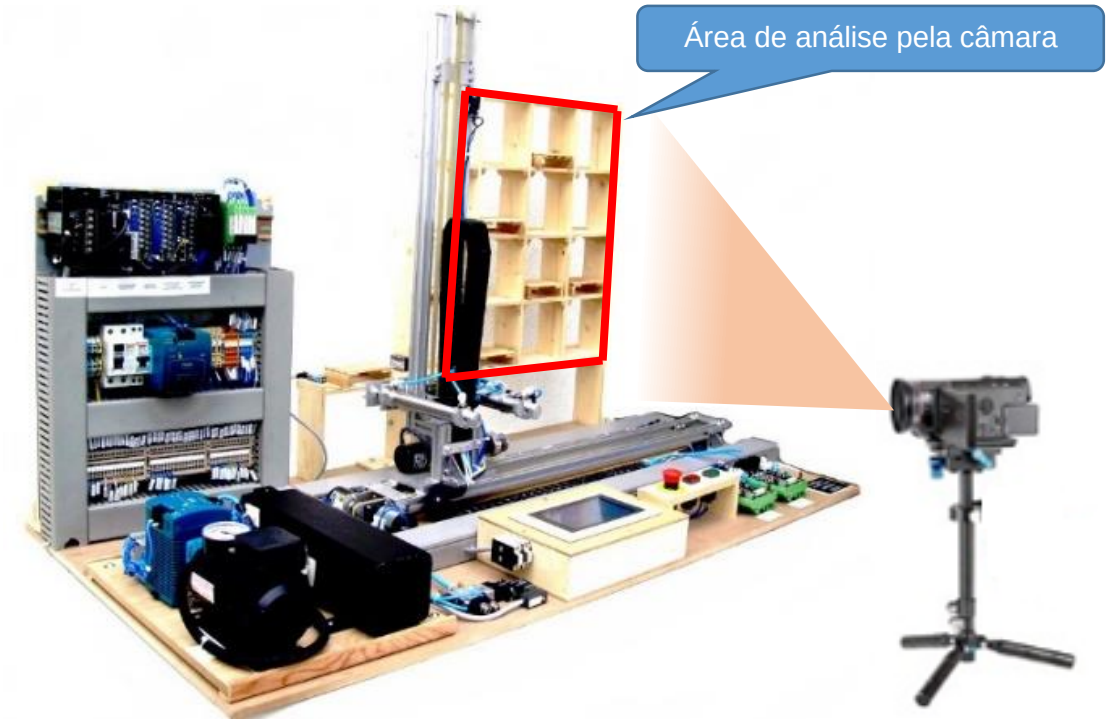


Figura 4.21 - Upgrade do sistema.

Para tal, realizou-se um estudo baseado em métodos e técnicas existentes de **visão computacional**. Desta forma foi possível avaliar quais os algoritmos existentes na visão computacional que melhor se adaptavam às necessidades do sistema. Também foi realizado um estudo de que parte do *software*, gravado no PLC, se alteraria de modo a integrar a nova componente funcional (visão computacional) ao sistema já existente.

4.3. Solução Computacional

A solução computacional (desenvolvida com um computador portátil) foi a primeira a ser desenvolvida, pois a unidade de processamento digital utilizada nesta solução possui uma extensa bibliografia acerca de ferramentas de *software* e respectivas bibliotecas para aquisição, processamento e tratamento de imagem. O facto dos conhecimentos adquiridos na unidade curricular de Sistemas Robóticos (aplicação de técnicas de visão computacional) terem sido aplicados num computador portátil, também facilitou a primeira interação com o sistema. Um computador portátil apropriado para este tipo de aplicações poderá rondar os 500€ no mercado actual.

4.3.1. Software e bibliotecas utilizadas

Para cada etapa foram utilizados os seguintes *softwares*:

Etapa	Software
Processamento e tratamento de imagem	*Matlab* 2015
Aquisição de imagem	*Microsoft Visual Studio* 2015 C# (.net)
Comunicação com autómato	*Microsoft Visual Studio* 2015 C# (.net)
Interface com utilizador	*Microsoft Visual Studio* 2015 C# (.net)

Tabela 4.1- Solução computacional: softwares utilizados.

Para cada *software* foram utilizadas as seguintes bibliotecas:

Software	Biblioteca
***Matlab* 2015**	*Image Processing Toolbox*
***Microsoft Visual Studio* 2015 C# (.net)**	*AForge.dll*
	MWArray.dll
	Project_ImageDotNet.dll

Tabela 4.2 - Solução computacional: bibliotecas utilizadas.

4.3.2. Equipamento utilizado

Quanto ao equipamento utilizado, além do já apresentado no subcapítulo 4.2.1, utilizou-se um computador portátil e uma *webcam*. O computador portátil foi utilizado como unidade de processamento digital, responsável não só pelo processamento e tratamento da imagem, como também pela interação com o autómato programável existente no simulador. A *webcam* foi utilizada para aquisição das imagens.

Computador Portátil

Utilizou-se um computador da marca Asus dotado de um processador da Intel Core i5 2.67GHz e com uma memória RAM de 4GB. Esta unidade digital possui um elevado desempenho, conseguindo atingir velocidades de processamento muito elevadas. Outro modelo menos recente e menos dispendioso poderia também ter sido utilizado.

Figura 4.22 - Unidade de processamento digital: computador portátil.

Webcam

Para aquisição de imagem utilizou-se uma *webcam* comercial da marca Logitech, dotada de um sensor de 2MP, o que possibilita uma resolução de 1600x1200 *pixels*. Possui a captação de imagens em RGB ou B&W. Quanto à conectividade com a unidade de processamento digital, é efetuada através de uma ligação USB.

Figura 4.23 - Captação de imagem: webcam.

4.3.3. Algoritmos desenvolvidos no software *Matlab*

Recorrendo aos conhecimentos adquiridos na unidade curricular de Sistemas Robóticos, onde foi abordada a visão computacional e se utilizou o software *Matlab* para processamento e análise de imagem, desenvolveu-se um algoritmo para receber, processar e analisar as imagens do simulador de armazenamento, devolvendo o resultado (ocupação do armazém) em forma de matriz (4x4). A biblioteca *Image Processing Toolbox* é fundamental para a construção deste algoritmo, pois esta biblioteca dispõe de uma extensa lista de algoritmos e funções direcionadas para o processamento e análise de imagem. O algoritmo desenvolvido é constituído por duas funções principais (ficheiros com extensão ".m"), sendo elas, "*MainProgram*" e "*CoordenatesOfEachPosition*". A função "*MainProgram*" é o tronco principal do algoritmo. A função "*CoordenatesOfEachPosition*" é responsável por procurar as coordenadas de cada uma das 16 posições do simulador de armazenamento. A Figura 4.24 ilustra de um modo macroscópico a construção do algoritmo.

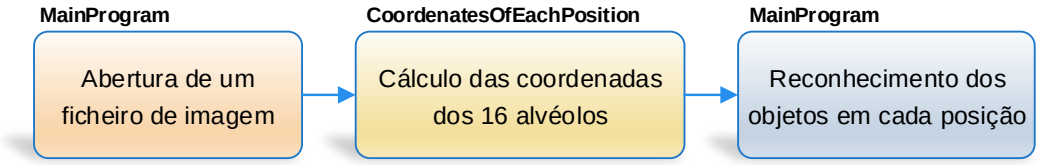


Figura 4.24 - Diagrama de blocos do algoritmo desenvolvido no Matlab (Solução Computacional).

O desenvolvimento deste algoritmo foi essencial para ultrapassar alguns dos vários desafios impostos pelo sistema. Começa-se por demonstrar a metodologia utilizada para determinar automaticamente a localização dos 16 alvéolos do sistema.

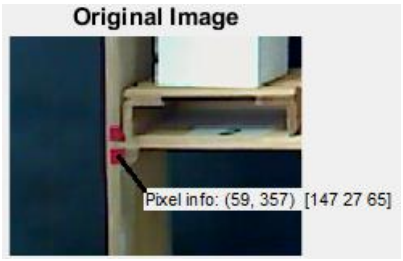
A concretização deste processo inicial foi auxiliada pelo cálculo das distâncias entre umas marcas de referência colocadas no sistema, ou seja, através da utilização do centro de gravidade dessas três marcas de referência $(x1, y1; x2, y2; x3; y3)$ delimita-se o comprimento e largura da primeira posição de armazenamento (onde estão alocadas as marcas de referência). Possuindo este comprimento e largura, consegue-se então descobrir as restantes 15 áreas de análise para verificar o estado de ocupação dos respetivos alvéolos, em termos de coordenadas de imagem.

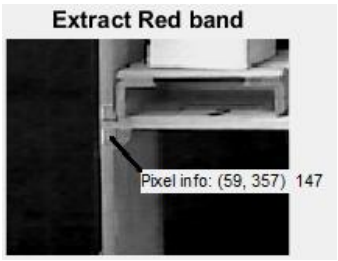
A Figura 4.25 representa a imagem original, onde se observa as marcas de referência (círculos de cor vermelha) no alvéolo da primeira posição.

Figura 4.25 - Imagem original onde se observa as marcas de referência.

Foi utilizada a cor vermelha pelo simples facto de ser uma cor que se realça diante das restantes presentes no sistema, e de ser uma cor facilmente extraída através de um algoritmo de *software*. No fundo poderia ser qualquer uma das cores primárias. Contudo, foi necessário aplicar mais do que um algoritmo para que a imagem apresentasse apenas e exclusivamente as marcas de referência. Para tal, utilizou-se fundamentalmente:

- **extração da cor vermelha da imagem RGB:** este algoritmo efetua a conversão de uma imagem RGB (*Original Image*) para a escala de cinzentos (255 níveis de intensidade luminosa), em que é extraída apenas a parcela correspondente à cor vermelha (*Extract Red band*).



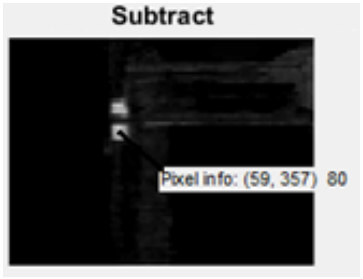


$$Extract\ Red\ band(x,y) = R(x,y) \times 1 + G(x,y) \times 0 + B(x,y) \times 0$$
$$Extract\ Red\ band(59,357) = 147 \times 1 + 27 \times 0 + 65 \times 0$$
$$Extract\ Red\ band(59,357) = 147$$

- **conversão para escala de cinzentos:** este algoritmo efetua a conversão de uma imagem RGB (*Original Image*) para a escala de cinzentos (255 níveis de intensidade luminosa), em que neste caso temos presente o peso de cada uma das três cores (vermelho, verde e azul) (*RGB2GRAY*).

$$RGB2GRAY(x,y) = R(x,y) \times 0.299 + G(x,y) \times 0.587 + B(x,y) \times 0.114$$
$$RGB2GRAY(59,357) = 147 \times 0.299 + 27 \times 0.587 + 65 \times 0.114$$
$$RGB2GRAY(59,357) = 67$$

- **subtração de imagens:** este algoritmo efetua a subtração da intensidade luminosa de cada pixel entre duas imagens. Desta forma foi possível evidenciar toda a cor vermelha existente na imagem de análise com recurso à subtração entre as imagens *Extract Red band* e *RGB2GRAY*, que originou a imagem *Subtract*.



$$Subtract(x,y) = Extract\ Red\ band(x,y) - RGB2GRAY(x,y)$$
$$Subtract(59,357) = Extract\ Red\ band(59,357) - RGB2GRAY(59,357)$$
$$Subtract(59,357) = 147 - 67$$
$$Subtract(59,357) = 80$$

- **técnica da limiarização:** este algoritmo efetua a segmentação da imagem *Subtract*, em que definido o *threshold* (T), cada pixel é convertido num valor binário, ("0" caso *Subtract(x,y)* ≤ T) preto ou ("1" caso *Subtract(x,y)* > T) branco.

O *threshold* (T) foi definido após várias experiências com diferentes níveis de luminosidade a insidir sobre a imagem de análise. O nível escolhido foi de T = 70.

A Figura 4.26 a) apresenta o resultado da aplicação referida anteriormente. De contemplar que a Figura 4.26 b) já possui outros métodos para apresentar a imagem desta forma (métodos que serão abordados no presente capítulo), pois a imagem já se encontra recortada em 16 blocos. O que é de realçar no processo observado na Figura 4.26 é o facto de que a partir das marcas de referência é possível descobrir as coordenadas das 16 posições.

a) b)

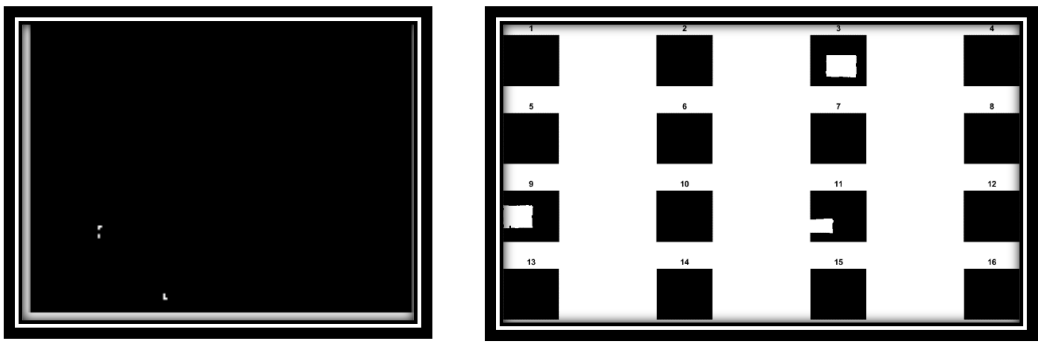


Figura 4.26 - Resultado da aplicação de um processo dos operadores estruturais: cálculo das distâncias entre objetos. a) imagem que contém as marcas de referência; b) imagem recortada em 16 posições.

Como referido anteriormente, com o auxílio de 2 marcas de referência, colocadas na 1ª posição, foi possível determinar o tamanho de cada alvéolo (em altura e comprimento). Uma 3ª marca foi utilizada para determinar a espessura do perfil entre cada alvéolo. O fluxograma ilustrado na Figura 4.27 representa a solução encontrada para determinar automaticamente a localização (coordenadas) dos 16 alvéolos do sistema.

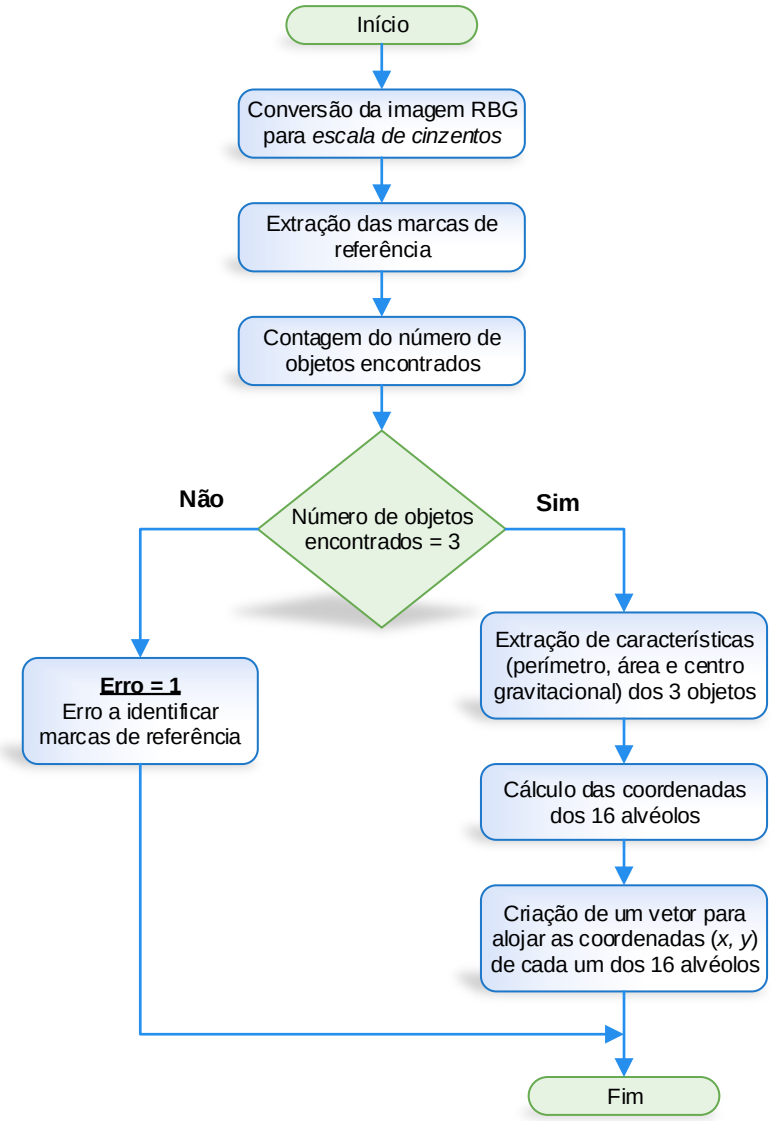


Figura 4.27 - Fluxograma relativo à identificação das coordenadas dos 16 alvéolos (CoordenatesOfEachPosition) (Solução Computacional).

De uma forma mais detalhada, a Figura 4.28 representa a metodologia implementada para a determinação das coordenadas dos 16 alvéolos. De notar que estão apenas representadas algumas deduções essenciais uma vez que as restantes são análogas.

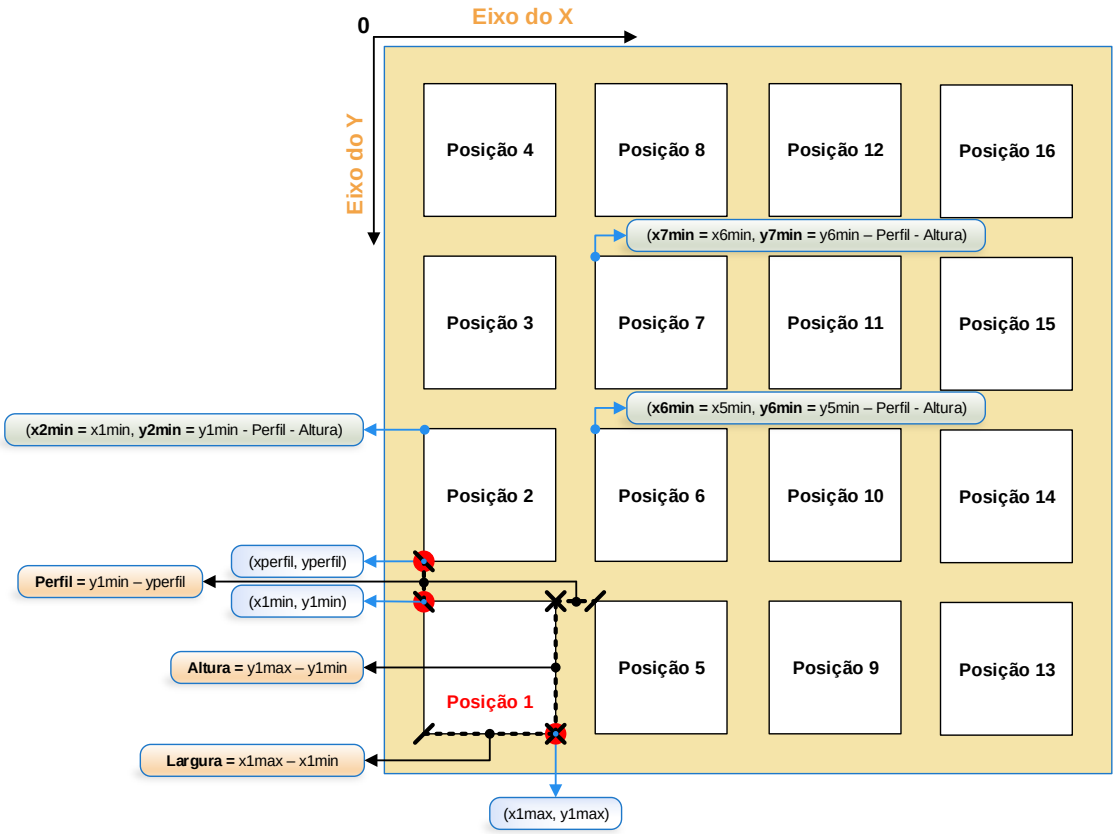


Figura 4.28 - Cálculo das coordenadas dos 16 alvéolos (Solução Computacional).

Com recurso à extração de caraterísticas de objetos (algoritmo disponível na *toolbox* do software *Matlab*), foi possível identificar o centro gravitacional (coordenadas *x* e *y*) das marcas de referência identificadas a vermelho (representadas na Figura 4.28 por um círculo vermelho).

Todo este processo de determinação das coordenadas foi elaborado na função "*CoordenatesOfEachPosition*". Tem como parâmetros de saída uma *flag* de erro para o caso das marcas de referência não terem sido encontradas, e um vetor de 2x17 que contém não só a informação das coordenadas de todos os alvéolos como também a dimensão (altura e largura) do alvéolo da 1ª posição (todos possuem as mesmas dimensões).

A “*MainProgram*” é responsável pela deteção de presença ou ausência de peça em cada alvéolo. Este processo de deteção segue-se pelo fluxograma ilustrado na Figura 4.29.

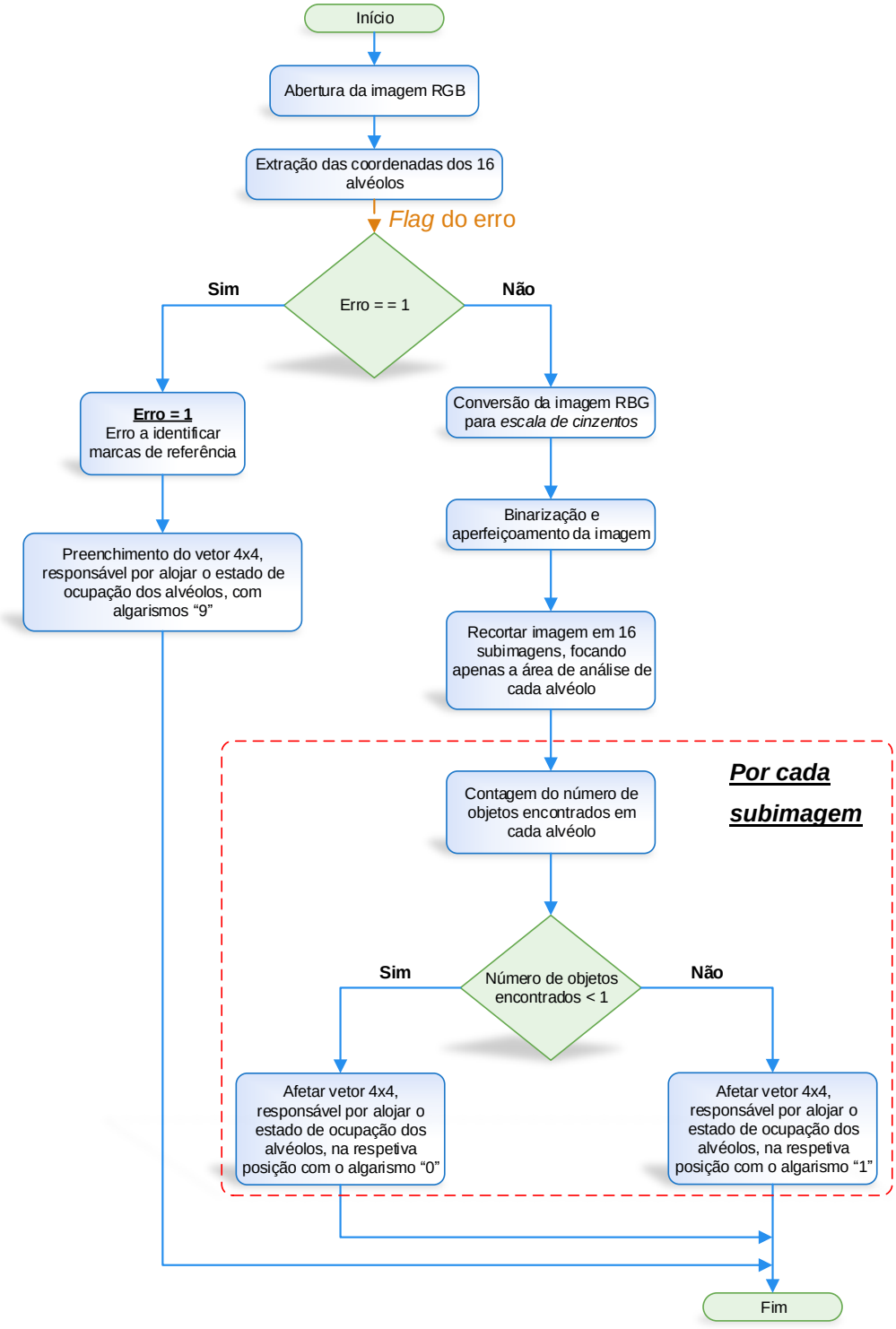


Figura 4.29 - Fluxograma relativo à deteção de peças nos alvéolos (MainProgram) (Solução Computacional).

A função “*MainProgram”* tem como parâmetro de saída um vetor de 4x4 que contém a informação acerca da indicação de presença (algarismo 1) ou ausência (algarismo 0) de peças em cada um dos 16 alvéolos. Caso o vetor seja preenchido com o algarismo 9, significa que as marcas de referência não foram encontradas. Para que este algoritmo pudesse ser integrado no software *Visual Studio C#*, foi necessário exportá-lo para um ficheiro do tipo “biblioteca de vínculo dinâmico” (.dll). O algoritmo poderá ser consultado nos Anexos deste documento.

4.3.4. Algoritmos desenvolvidos no *software Visual Studio C#*

Um ambiente de desenvolvimento ou IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) é um *software* que reúne caraterísticas e ferramentas de apoio ao desenvolvimento de *software* com o objetivo de agilizar processos. Normalmente os IDE facilitam a técnica de RAD (*Rapid Application Development* – Desenvolvimento Rápido de Aplicações), pois baseiam-se em modelos de processo de desenvolvimento de *software* interativo. Analisando alguns IDE, optou-se pelo *Visual Studio C#,* pois além de ser uma ferramenta gratuita e muito produtiva, a sua aplicabilidade está inserida em vários sistemas de automação industrial. Neste *software* desenvolveu-se um algoritmo para:

- aquisição de imagem a partir de uma câmara digital;
- executar o algoritmo desenvolvido no *software Matlab*;
- interação com o autómato por comunicação série.

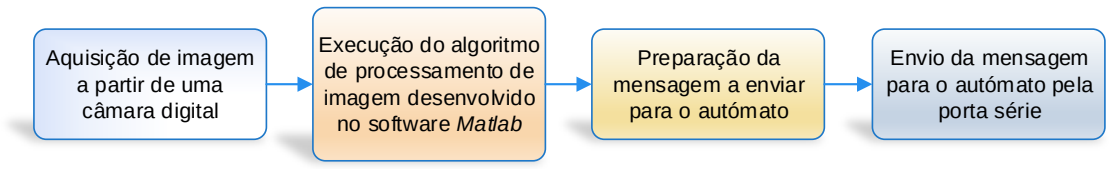


Figura 4.30 - Diagrama de blocos do algoritmo desenvolvido no Visual Studio C# (Solução Computacional).

No âmbito de *software*, a interação entre a câmara digital e o computador é realizada com recurso à biblioteca *Aforge.dll*, em que esta contém funções específicas para activar as funcionalidades da câmara, tais como a captação de fotografias e vídeo.

A captação da imagem só é efetuada quando o braço robótico se encontra na posição de origem (leitura do registo D701 do PLC que indica a presença do braço robótico na posição de origem). Quando é autorizada a captação da imagem, esta é captada e processada pelo algoritmo desenvolvido no *Matlab* (detalhado no subcapítulo 4.3.3). O mesmo retorna um vetor 4x4 (estado de ocupação dos 16 alvéolos e indicação se foram ou não encontradas as marcas de referência). O vetor 4x4 condiciona a interação com PLC, pois se este incluir algarismos 9, são apenas lidos os registos do PLC que contém o estado de ocupação dos alvéolos. Caso contrário, o vetor é interpretado e posteriormente formatado numa mensagem a enviar para o PLC. Este processo segue-se pelo fluxograma ilustrado na Figura 4.31.

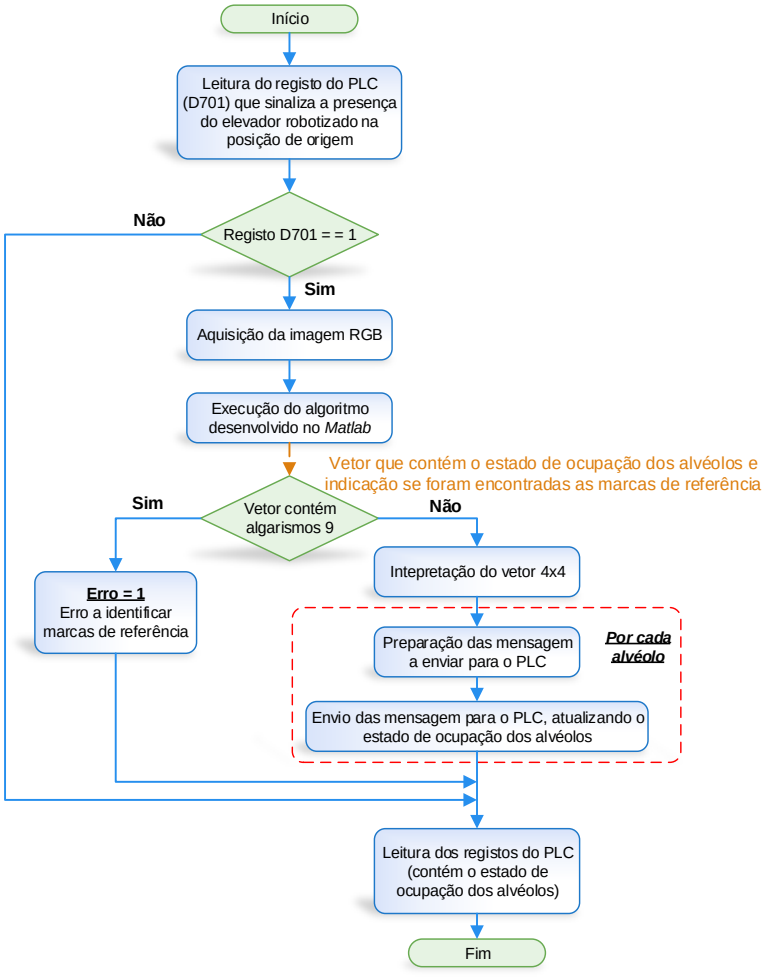


Figura 4.31 - Fluxograma relativo ao algoritmo desenvolvido no Visual Studio C# (Solução Computacional).

Quanto à formatação das mensagens enviadas ou recebidas pelo autómato foi necessário um estudo prévio acerca do protocolo utilizado para a comunicação com o computador. O algoritmo poderá ser consultado nos Anexos deste documento.

Protocolo de comunicação com o autómato da OMRON

Os autómatos da OMRON utilizam um modo de comunicação específico para comunicações série, denominado de "*C-mode (Host Link) commands*", em que proporcionam um controlo de várias operações entre o autómato e o computador. Estas operações incluem leitura e escrita de registos na memória do autómato, mudanças de modo de operação, entre outras [61]. Podem ser estabelecidos até 32 (0 a 31) endereços de autómatos, mas as características da ligação RS-232 apenas suporta uma ligação ponto a ponto. A trama de comunicação pode conter no máximo 131 caracteres de informação. Os caracteres são enviados e recebidos no formato ASCII (*American Standard Code for Information Interchange*). O formato para os comandos *Host Link* enviados pelo computador está ilustrado na Figura 4.32.

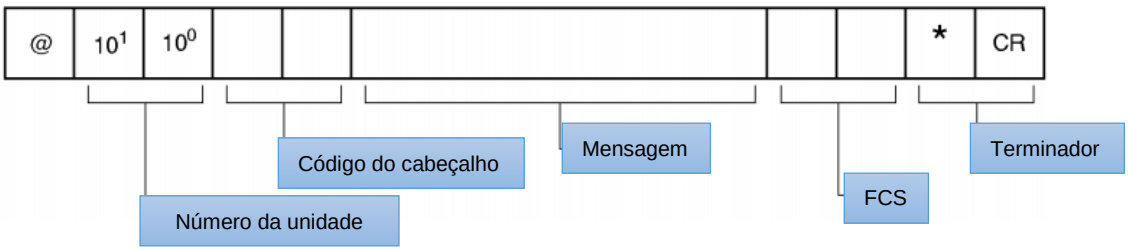


Figura 4.32 - Formato do envio de um comando [61].

Onde:

- @: deve ser colocado no início do comando;
- **Número da unidade**: delimitado de 0 a 31 para cada autómato;
- **Código do cabeçalho**: expresso em dois caracteres;
- **Mensagem**: colocação de parâmetros;
- **FCS**: cálculo do FCS (*Frame Check Sequence*). De seguida será detalhado;
- **Terminador**: colocação de "*" e um caracter terminador ("CR - *Carriage Return*") para indicar o fim do comando a enviar.

O formato para os comandos *Host Link* recebidos pelo computador está ilustrado na Figura 4.33.

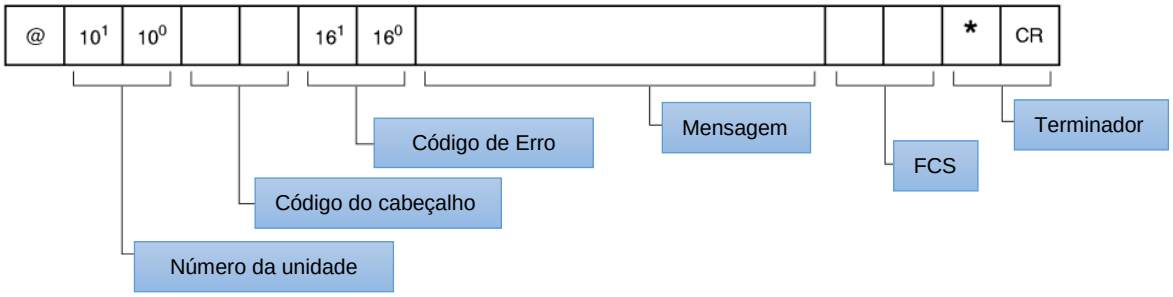


Figura 4.33 - Formato da resposta [61].

Onde:

- @: deve ser colocado no início do comando;
- **Número da unidade**: delimitado de 0 a 31 para cada autómato;
- **Código do cabeçalho**: o código enviado é igual ao recebido;
- **Código de erro**: resultado do comando enviado;
- **Mensagem**: mensagem recebida;
- **FCS**: os 2 caracteres do FCS são recebidos;
- **Terminador**: os 2 caracteres, "*" e um caracter terminador ("*Carriage Return*"), para indicar o fim da resposta.

Os códigos do cabeçalho utilizados neste trabalho foram os seguintes:

- **RD (*DM AREA READ*)**: tem a capacidade de ler um número específico de *words*, designadas por DM *words*;
- **WD (*DM AREA WRITE*)**: tem a capacidade de escrever nas *words*, designadas por DM *words*.

As *words* do PLC da OMRON que contêm o estado de ocupação de cada um dos 16 alvéolos estão numeradas da D601 à D617 (já se encontravam previamente definidas no algoritmo existente do autómato quando o simulador foi criado). A *word* D701 possui a informação se o braço robótico se encontra, ou não, na posição de origem.

Determinação do *Frame Check Sequence* (FCS)

O autómato calcula o valor do FCS (método adoptado pelo fabricante do PLC para a detecção de erros de comunicação) para cada trama que recebe, e valida a existência de erros pela comparação desse valor com o FCS enviado no comando. Sendo assim, o computador deve calcular o valor do FCS para que seja englobado na trama do comando a enviar para o autómato. A Figura 4.34 ilustra a metodologia usada para determinar o valor do FCS.

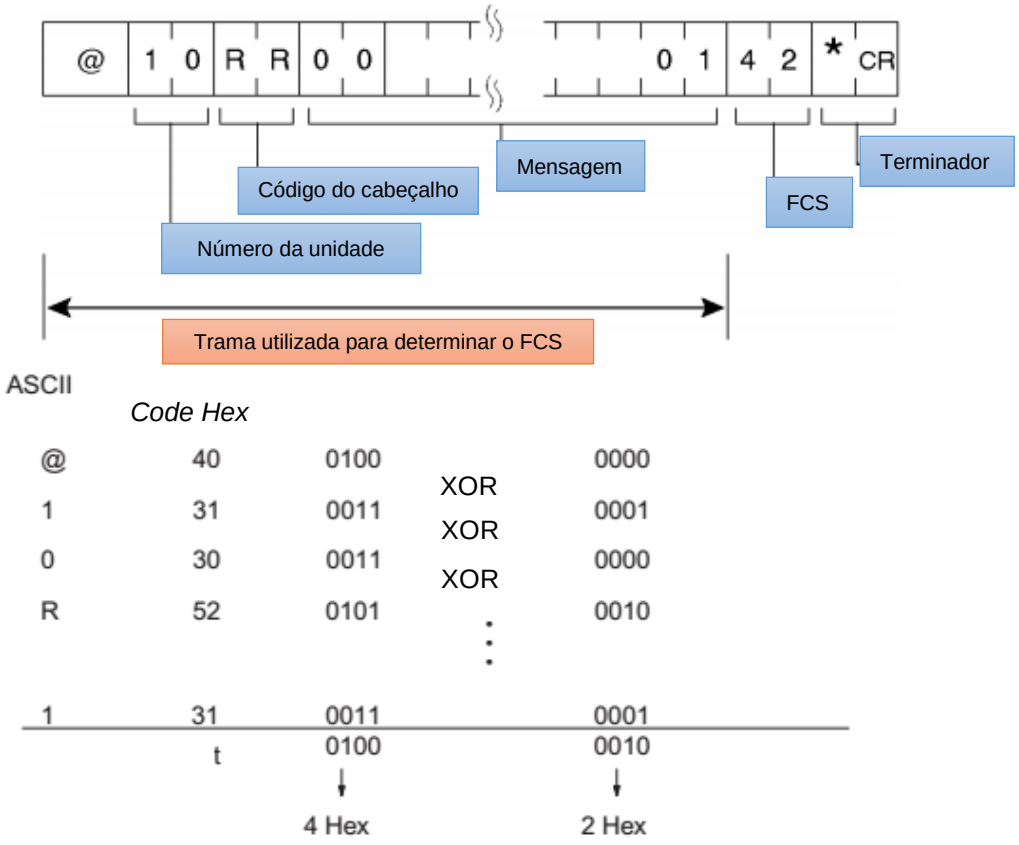


Figura 4.34 - Determinação do FCS [61].

O FCS é um valor de 8 *bits* convertido a partir de dois caracteres ASCII. O valor final de 8 *bits* é o resultado de um OR-exclusivo (XOR) sequencial promovido entre cada caracter, desde o primeiro caracter da trama até ao último caracter pertencente à “mensagem”. A tabela ASCII pode ser consultada nos Anexos deste documento.

4.3.5. Aplicação da solução computacional

A aplicação *Front-End,* desenvolvida como solução computacional (num computador portátil) com recurso ao *Visual Studio C#* permite a monitorização da presença ou ausência de peças no simulador de armazenamento por parte do operador. Esta engloba todos os algoritmos mencionados no subcapítulo 4.3.4. Algumas das configurações disponíveis (manipuláveis pelo operador) neste software não seriam disponibilizadas numa aplicação real de supervisão industrial, ou então teriam acesso restrito (criação de um submenu onde seria requerida uma *password* de autenticação). A Figura 4.35 representa o aspeto do menu inicial da aplicação.

Figura 4.35 - Menu inicial da aplicação Front-End (Solução Computacional).

No menu inicial é possível encontrar a identificação do projeto e dos seus intervenientes, como também a respetiva versão do *software*.

Ao clicar no botão “*Monitoring*” segue-se para a tela principal da aplicação (ver Figura 4.36).

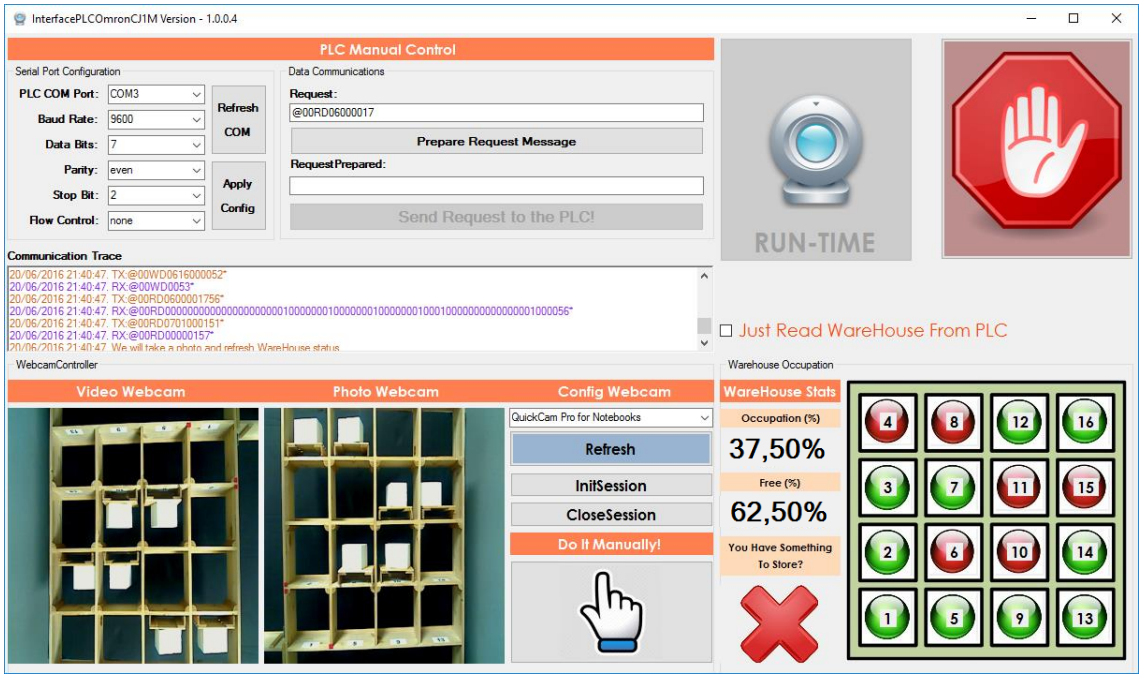


Figura 4.36 - Menu principal da aplicação Front-End (Solução Computacional).

Este *software* permite ao operador:

- Configurar a porta de comunicações para interação com o autómato (***Serial Port Configuration***). Com recurso ao botão “*Refresh COM*” é possível pesquisar novos dispositivos de comunicação USB (neste caso conversores USB-RS232) conectados ao computador portátil. Clicando no botão “*Apply Config*”, a configuração da porta é armazenada num ficheiro auxiliar (evita a configuração por cada vez que se inicia a aplicação);

- Enviar comandos diretos para o autómato (***Data Communication***). Clicando no botão “*Prepare Request Message*”, é efetuada a preparação da mensagem a enviar para o autómato, ou seja, é calculado o FCS e são colocados os caracteres terminadores. Pressionando no botão “*Send Request to the PLC!*”, a mensagem previamente preparada é enviada para o autómato;

- Configurar a webcam (***WebcamController***). Com recurso ao botão "*Refresh*" é possível pesquisar novos dispositivos de aquisição de imagem conectados por USB à unidade digital. Os botões "*InitSession*" e "*CloseSession*" são utilizados para ligar e desligar respetivamente o dispositivo de aquisição de imagem selecionado. Por último, o botão com o símbolo de uma mão é utilizado para realizar todos os algoritmos mencionados no subcapítulo 4.3.4 (apenas 1 ciclo), preenchendo os indicadores representados no "***WareHouse Occupation***" (taxa de ocupação do armazém; estado de ocupação de cada alvéolo; estado de ocupação da posição de carregamento);

- Clicando no botão "*RUN-TIME*", a operação descrita para o botão com o símbolo de uma mão é realizada ciclicamente, até que seja pressionado o botão de "parar".

O estado de ocupação do armazém é dado por indicadores circulares verdes (alvéolo livre), vermelhos (alvéolo ocupado) e azuis (marcas de referência não encontradas).
O campo "*Do You Have Something To Store?*" indica o estado de ocupação da posição de carregamento, em que o símbolo "X" ou o símbolo de "certo" representam respetivamente a ausência ou a presença de peça no alvéolo de carregamento.
Relativamente ao tempo decorrido por cada ciclo, o sistema demora cerca de 2 a 3 segundos (aquisição de imagem, processamento e análise da respetiva e envio das mensagens para o PLC).

4.3.6. Alterações no algoritmo do Autómato

O algoritmo já existente no autómato foi sujeito a alterações, pois em determinados aspetos, o seu atual funcionamento colidia com as novas funcionalidades implementadas no sistema. Cada movimento que o braço robótico exercia (armazenar ou retirar peças dos alvéolos), este ficava imóvel no último alvéolo a que se deslocava (à espera de uma nova ordem pelo operador). Atendendo às novas funcionalidades implementadas, em que é necessária uma visão "limpa" de todo o sistema de armazenamento, o braço robótico foi reprogramado para regressar à posição de origem após conclusão do seu movimento de armazenamento/extração de peças. Sempre que o braço robótico se encontra na posição

de origem é acionada uma *flag* (escolheu-se a *word* D701) para que o software desenvolvido tenha conhecimento do momento mais oportuno para a captação da imagem. O fluxograma da Figura 4.37 representa as alterações efetuadas no algoritmo do autómato já existente anteriormente (assinalado com a cor laranja).

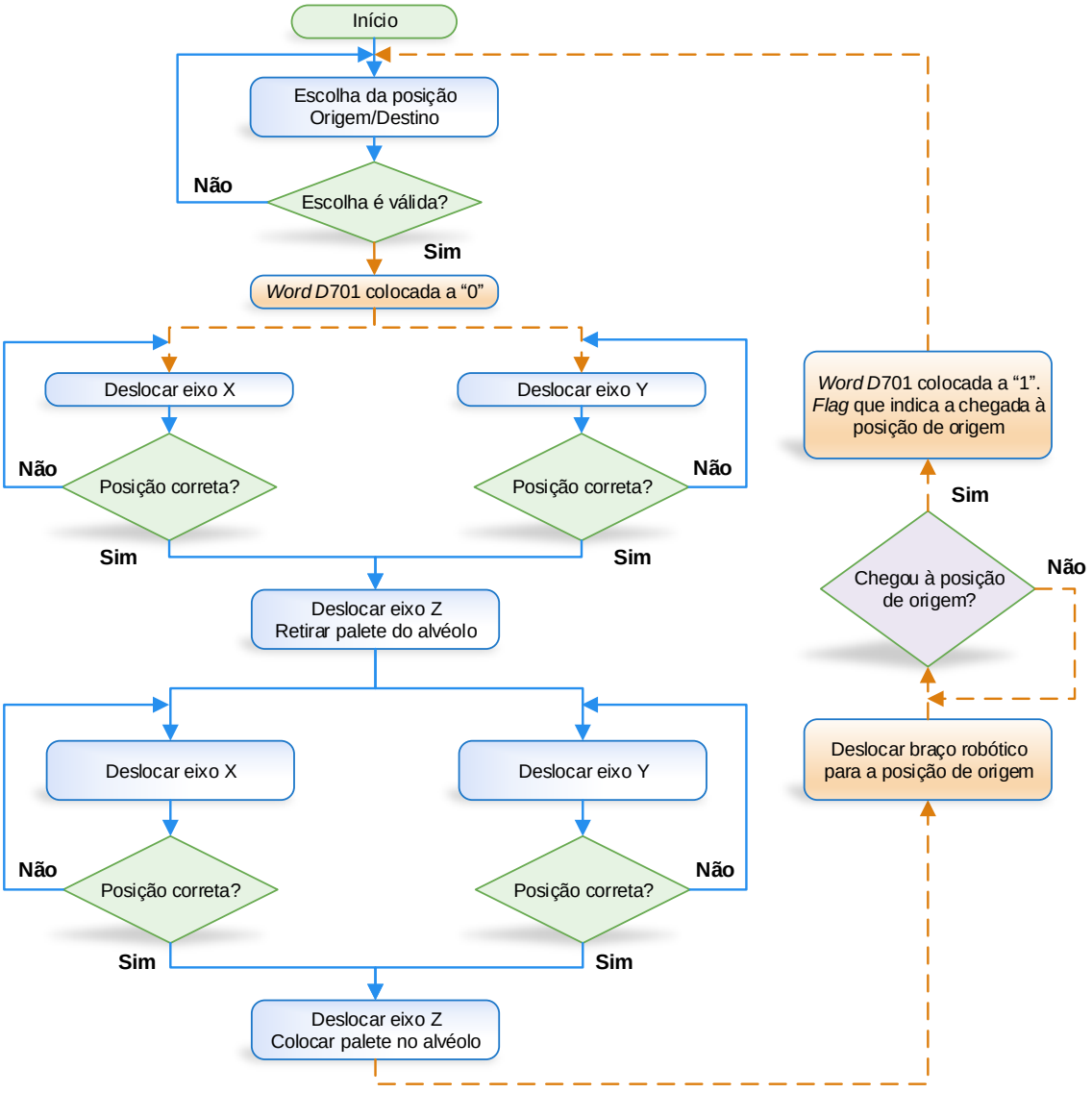


Figura 4.37 - Fluxograma relativo às alterações efetuadas ao algoritmo do autómato [62].

4.4. Solução com Microcomputador

Comparando com a solução descrita no subcapítulo anterior, optou-se por desenvolver uma solução que proporcionasse uma redução no custo de material, um menor volume estrutural e uma maior portabilidade (fisicamente inserido no sistema). Essa solução foi promovida pelo desenvolvimento de uma unidade de processamento digital com recurso ao *Raspberry Pi Model* B+. Este pode ser adquirido atualmente no mercado por 25€.
Todos os modelos do *Raspberry Pi* adotam um fator crucial para a maioria das aplicações, o tamanho reduzido, em que possui a mesma largura e comprimento de um cartão de crédito e uma espessura de cerca de 2 cm (contando com a eletrónica presente, portas USB e *ethernet*). Sendo uma placa eletrónica, é aconselhável a sua proteção contra contactos indevidos. Para tal, existem caixas específicas para alojar o *Raspberry Pi* [60]. Até ao momento, o *Raspberry Pi* é o *single-board computer* mais popular do mercado, com vendas a atingir diversos milhões de unidades. Isto significa que já existe um número considerável de *software developers* que já desenvolvem as suas aplicações com recurso a esta plataforma. Grande parte destes são entusiastas *open-source*, onde importam e criam novas bibliotecas para desenvolver novos *softwares*, escrevem tutoriais e resolvem problemas através de fóruns *online*. Todos estes fatores incrementam as ferramentas disponíveis e tornam o caminho mais facilitado para encontrar soluções que resolvam os problemas que surgem [60].

Nesta solução desenvolveu-se toda a componente de aquisição e tratamento de imagem no computador portátil (no software *Matlab-Simulink* 2015). O *Simulink* possui uma biblioteca específica para a comunicação (através da comunicação *Ethernet*) com o *Raspberry Pi.* Desta forma foi possível desenvolver um modelo *Simulink* e compilá-lo numa aplicação *standalone* para correr no *Raspberry Pi*.
A versão de Linux utilizada no *Raspberry Pi* foi o *Whezzy*, instalada diretamente a partir do *software Matlab* durante a instalação da biblioteca "*Simulink Support Package for Raspberry Pi Hardware"*.

4.4.1. Software e bibliotecas utilizadas

Para cada etapa foram utilizados os seguintes softwares:

Etapa	Software
Processamento e tratamento de imagem	*Matlab-Simulink* 2015
Aquisição de imagem	*Matlab-Simulink* 2015
Comunicação com autómato	*MonoDevelop*

Tabela 4.3 - Solução com microcomputador: softwares utilizados.

Para cada software foram utilizadas as seguintes bibliotecas:

Software	Biblioteca
***Matlab-Simulink* 2015**	*Simulink Support Package for Raspberry Pi Hardware*
	Computer Vision System Toolbox
MonoDevelop	*wiringPi*

Tabela 4.4 - Solução com microcomputador: bibliotecas utilizadas.

É importante salientar que o *software MonoDevelop* e a biblioteca *wiringPi* foram instalados diretamente no *Raspberry Pi*. O propósito da sua utilização será abordado no subcapítulo 4.4.4.

4.4.2. Equipamento utilizado

Quanto ao equipamento utilizado, além do já apresentado no subcapítulo 4.2.1, utilizou-se um *Raspberry Pi Model* B+ e uma *webcam*. O *Raspberry Pi* foi utilizado como unidade de processamento digital, responsável não só pelo processamento e tratamento da imagem, como também pela interação com o autómato programável existente no simulador. A *webcam* foi utilizada para aquisição das imagens (apresentada no subcapítulo 4.3.2).

Raspberry Pi Model B+

A Figura 4.38 ilustra o aspeto físico do *Raspberry Pi Model B+.*

Figura 4.38 - Raspberry Pi Model B+ [63].

O *Raspberry Pi Model B+* é dotado das seguintes caraterísticas [63]:

- *Chip*: *Broadcom* BCM2835 SoC;
- Arquitetura do núcleo: ARM11;
- CPU: 700 MHz baixo consumo de energia ARM1176JZFS;
- GPU: *Dual Core VideoCore IVO Multimedia Co-Processor;*
- Memória RAM: 512MB;
- Sistema Operativo: *Boot* a partir do cartão micro SD, em que funciona sob um sistema operativo Linux;
- Dimensão: 85 x 56 x 17mm;
- Alimentação: Micro USB 5V, 2A.

Quanto aos conectores disponíveis [63]:

- *Ethernet* para conexão a outros dispositivos ou internet;
- HDMI para saída de vídeo;
- *Jack* 3.5mm para saída de áudio;
- 4 conectores USB 2.0;
- 40 pinos GPIO (*General Purpose Input/Output*). Providencia não só 27 GPIO, como também +3.3V, +5V e GND;
- MIPI Câmara (CSI-2);
- *Slot* para cartão de memória micro SD.

4.4.3. Algoritmo desenvolvido no software *Matlab-Simulink*

O algoritmo/modelo desenvolvido no *software Matlab-Simulink* é semelhante ao algoritmo já apresentado no subcapítulo 4.3.3. Existem apenas algumas diferenças no que diz respeito:

- à aquisição de imagem do sistema. No modelo desenvolvido, a imagem é adquirida com recurso a um bloco (Figura 4.39) da biblioteca *Simulink Support Package for Raspberry Pi Hardware*. Este bloco permite adquirir diretamente a imagem a partir de um dispositivo de visão conectado ao *Raspberry Pi*, especificando o ID do dispositivo, a resolução da imagem, o formato de pixéis (RGB ou YCbCr 4:2:2) e o FPS (*Frame Per Seconds*).

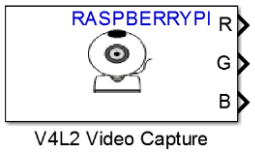


Figura 4.39 - Bloco de aquisição de imagem (Solução com microcomputador).

- à utilização de funções da *toolbox Image Processing Toolbox* (*Matlab*). No modelo desenvolvido, todas as funções dessa *toolbox* (utilizadas na solução computacional), devido à incompatibilidade com o modo de operação pretendido (*standalone*), foram substituídas por blocos da *toolbox Computer Vision System* (*Simulink*). A Figura 4.40 representa o exemplo de uma dessas analogias. Infelizmente para outras situações a analogia não foi tão direta.

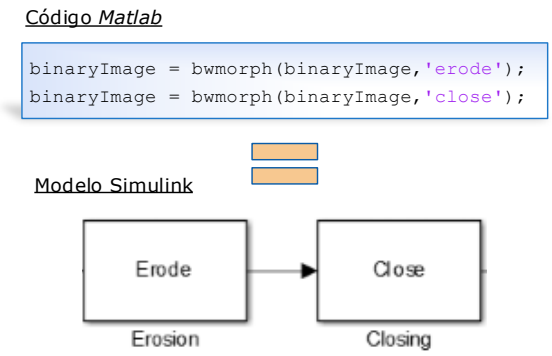


Figura 4.40 - Analogia entre código Matlab e modelo Simulink (Solução com microcomputador).

- à análise do estado de ocupação dos alvéolos. O processo de recorte de imagem (utilizado com o objetivo de focar a área de análise de cada alvéolo) também foi substituído por outro método devido à incompatibilidade já mencionada. Este processo desenvolveu-se numa função do tipo ".m" (bloco "*MATLAB function*" disponível na *toolbox* do *Simulink*). Esta função recebe como parâmetros de entrada:

  - Número de objetos encontrados no sistema;
  - *Centroid* de cada um desses objetos;
  - *Array* de dimensão 1x17 que contém as coordenadas do eixo *X* de cada uma das 16 posições e a largura de cada alvéolo;
  - *Array* de dimensão 1x17 que contém as coordenadas do eixo *Y* de cada uma das 16 posições e a altura de cada alvéolo.

A partir desta informação é possível conhecer o estado de ocupação de cada um dos alvéolos (ver Figura 4.41).

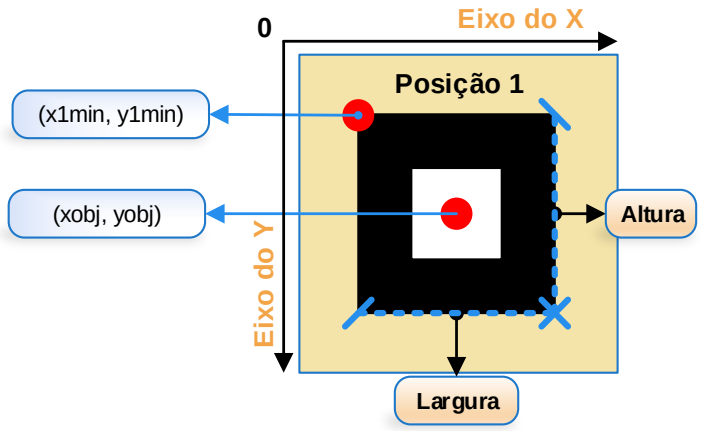


Figura 4.41 - Análise do estado de ocupação do alvéolo (Solução com microcomputador).

A presença/ausência de peça no alvéolo é dada pela seguinte expressão:

$$Ocupação = ((centroid(xobj) > x1\text{min}) \;\&\&\; (centroid(xobj) < x1min + largura) \;\&\&\; (centroid(yobj) > y1\text{min}) \;\&\&\; (centroid(yobj) < y1min + altura)) \quad (4.1)$$

- à exportação do resultado da análise da imagem captada. Quanto a esta matéria, o resultado da análise é escrito em dois ficheiros distintos (armazenados na memória do *Raspberry Pi*, onde se encontra a pasta que contém a aplicação *standalone* compilada pelo *software Simulink*). Um desses ficheiros, denominado por *ErrorLog0.txt*, aloca o resultado de uma *flag* de erro (colocada a "X" significa que as marcas de referência não foram encontradas, caso contrário fica a "Y"), O ficheiro *Array0.txt* contém os 16 caracteres que indicam o estado de ocupação dos 16 alvéolos (indicação de presença com o algarismo "1" e ausência com o algarismo "0").

O algoritmo do modelo desenvolvido no *Simulink* segue-se pelo fluxograma da Figura 4.42.

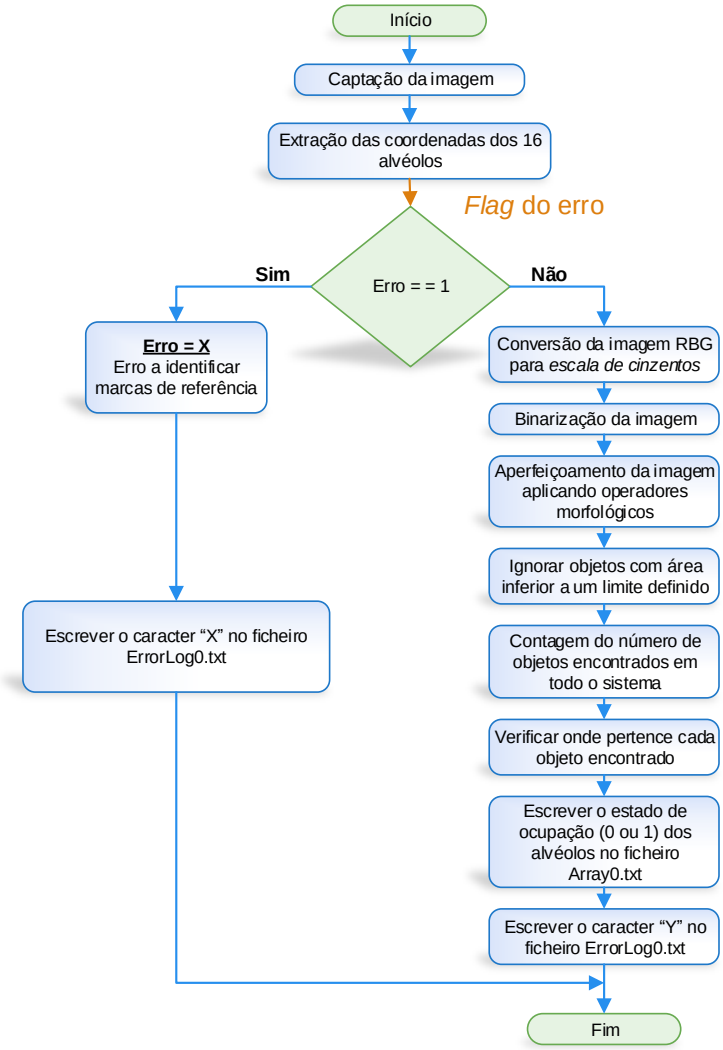


Figura 4.42 - Fluxograma relativo ao algoritmo desenvolvido no Simulink (Solução com microcomputador).

4.4.4. Algoritmo desenvolvido no *Raspberry Pi*

No *Raspberry Pi* foi desenvolvido um algoritmo em linguagem C no *software MonoDevelop*. A biblioteca *wiringPi* foi essencial para o acesso aos GPIO.

Neste *software* desenvolveu-se um algoritmo para:

- executar o modelo desenvolvido no *Simulink*;
- interação com o autómato por comunicação série.

A Figura 4.43 ilustra de um modo macroscópico a construção do algoritmo.

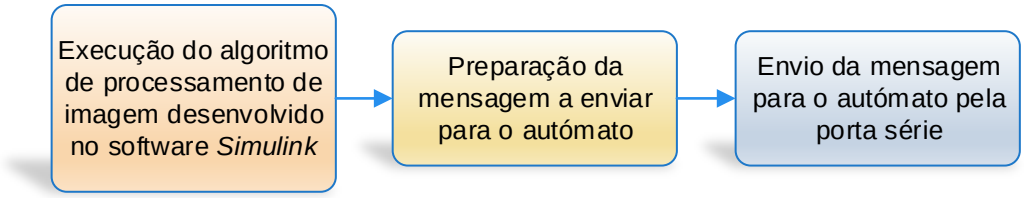


Figura 4.43 - Diagrama de blocos do algoritmo desenvolvido no MonoDevelop (Solução com microcomputador).

A execução do modelo do *Simulink* só é efetuada quando o braço robótico se encontra na posição de origem (leitura do registo D701 do PLC que indica a presença do braço robótico na posição de origem). Assim que o braço robótico atinge a posição de origem, é dada a ordem para executar o modelo do *Simulink*, onde é captada e analisada a imagem. Terminado este processo, é feita a leitura dos ficheiros exportados pelo modelo (*ErrorLog0.txt* e *Array0.txt*), onde serão devidamente lidos e interpretados. Um desses ficheiros, denominado por *ErrorLog0.txt*, reporta o resultado de uma *flag* de erro (colocada a "X" significa que as marcas de referência não foram encontradas, caso contrário fica a "Y"), O ficheiro *Array0.txt* contém os 16 caracteres que indicam o estado de ocupação dos 16 alvéolos (indicação de presença de peça com o algarismo "1" e ausência de peça com o algarismo "0"). Caso o caracter lido no ficheiro *ErrorLog.txt* seja "Y", o vetor que está contido no ficheiro *Array0.txt* é interpretado e posteriormente formatado numa mensagem a enviar para o PLC.

Este processo segue-se pelo fluxograma ilustrado na Figura 4.44.

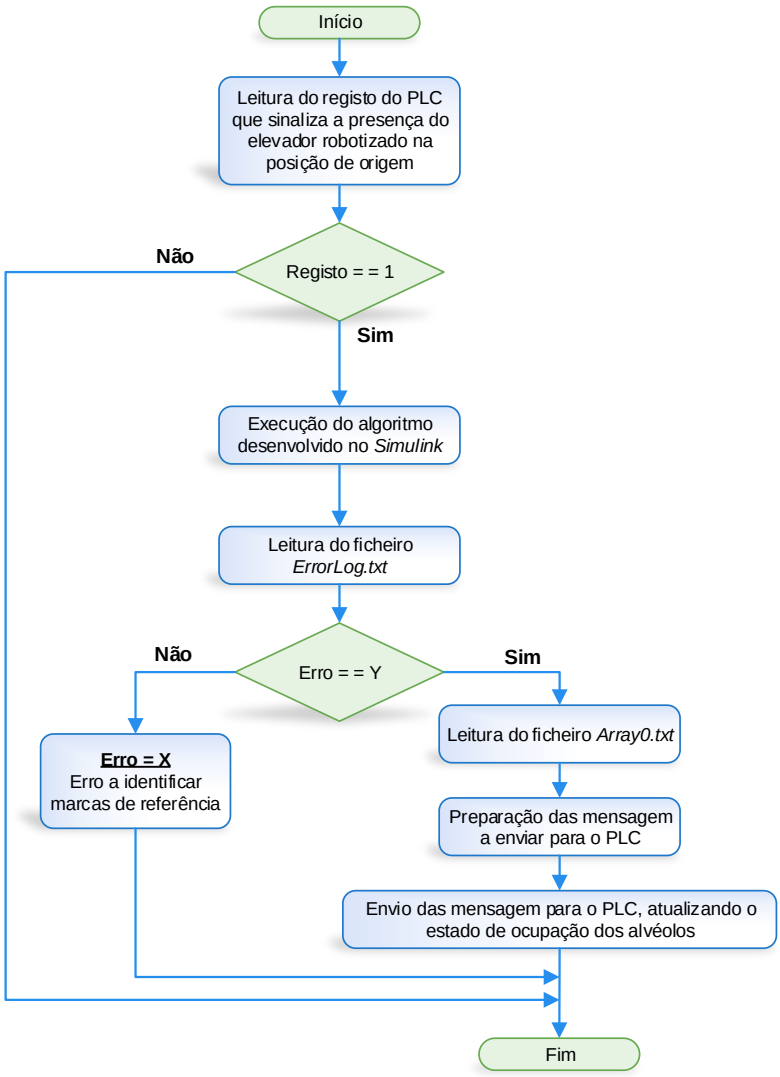


Figura 4.44 - Fluxograma relativo ao algoritmo desenvolvido no MonoDevelop (Solução com microcomputador).

Quanto à formatação das mensagens enviadas ou recebidas pelo autómato, o algoritmo é semelhante ao desenvolvido na solução computacional, apenas adaptado a uma linguagem de programação diferente. Relativamente ao tempo decorrido por cada ciclo, o sistema demora cerca de 2 a 3 segundos (aquisição de imagem, processamento e análise da respetiva e envio das mensagens para o PLC). Conclui-se que o tempo de ciclo obtido nesta solução é muito semelhante ao da solução computacional.

4.4.5. *Hardware* desenvolvido para o *Raspberry Pi*

Com o intuito de interligar o *Raspberry Pi* ao PLC, foi necessário desenvolver uma placa de circuito impresso para adaptar os sinais de comunicação série entre ambos os dispositivos. Isto porque funcionam com níveis de tensão diferentes (ver Tabela 4.5).

Valor Lógico	PLC	*Raspberry Pi*
	Níveis de Tensão	
	Sinais de dados TXD e RXD	Sinais de dados TXD e RXD
0	+13V	0V
1	-13V	3,3V

Tabela 4.5 - Níveis de tensão dos sinais de comunicação.

A Figura 4.45 representa um diagrama temporal ilustrativo da adaptação de sinais que se pretende fazer para a comunicação entre o PLC (RS232) e o *Raspberry Pi* (CMOS).

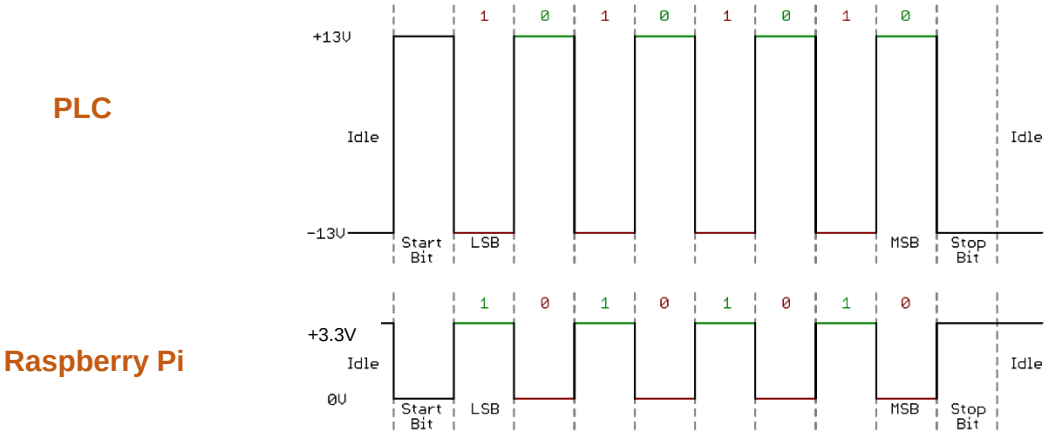
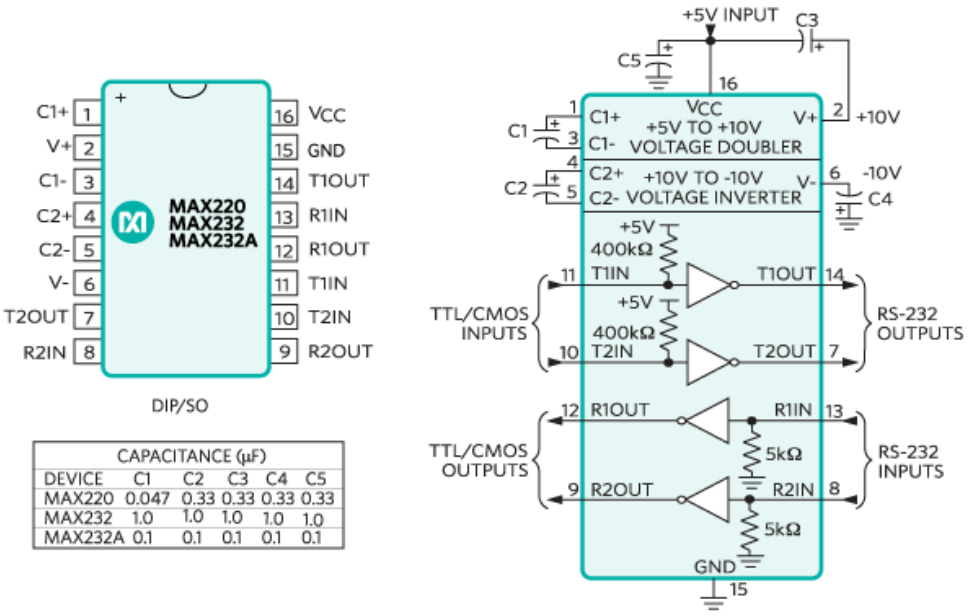


Figura 4.45 - Diagrama temporal dos sinais RS232 e CMOS [64].

A placa desenvolvida para a adaptação descrita tem por base o dispositivo *MAX232*, o qual possui um duplo canal de comunicação, e inclui um gerador de tensão capacitivo para alimentar os níveis de tensão dos sinais RS-232 a partir de uma fonte de 3.3V. Cada um dos canais converte a entrada (sinais RS-232) em sinais lógicos CMOS (0-3.3V), e vice-versa.

A Figura 4.46 representa a configuração dos pinos e o circuito elétrico aconselhado por um fabricante do dispositivo *MAX232*.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	0.047	0.33	0.33	0.33	0.33
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figura 4.46 - Configuração dos pinos (MAX232) e circuito elétrico (conversor RS232 para CMOS) [65].

Além do conversor de RS232 para CMOS, foram também implementados alguns indicadores luminosos (LED) para auxilio da interpretação do estado do sistema, tais como:

- Estado funcional do sistema, ou seja, se o algoritmo desenvolvido no *Raspberry Pi* está a funcionar;
- Estado da comunicação entre o *Raspberry Pi* e o PLC;
- Interpretação das mensagens por parte do PLC, ou seja, se ocorreu algum problema na troca de tramas de dados entre o *Raspberry Pi* e o PLC.

Os restantes GPIO foram disponibilizados com conectores para futuras ligações de *inputs/outputs*. Quanto à conceção da placa de circuito impresso, esta foi desenvolvida no *software Altium Designer*.

É necessário conhecer a configuração dos 40 pinos GPIO disponíveis (ver Figura 4.47) para que seja possível uma ligação direta entre o *Raspberry Pi* e a placa eletrónica adaptadora (por exemplo, com a recurso a um *flat cable* de 40 pinos).

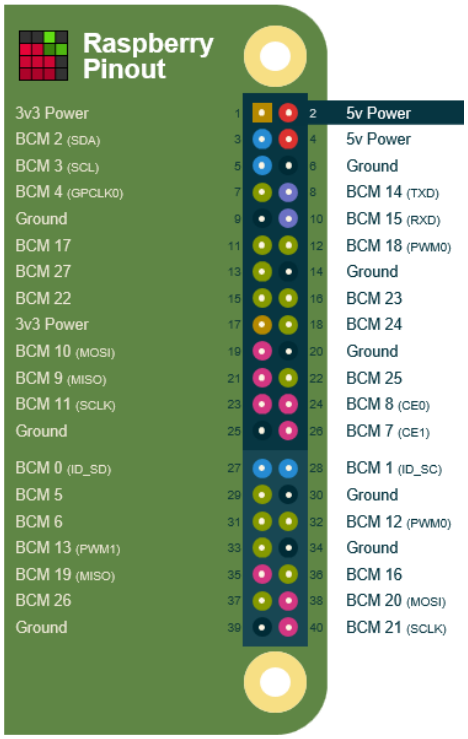


Figura 4.47 - Configuração dos pinos (Raspberry Pi) [66].

A Figura 4.48 representa o aspeto físico da placa desenvolvida para interligar com o *Raspberry Pi*. Toda a informação (circuito elétrico, lista de material e *datasheet*) relativa a esta placa pode ser consultada nos Anexos deste documento.

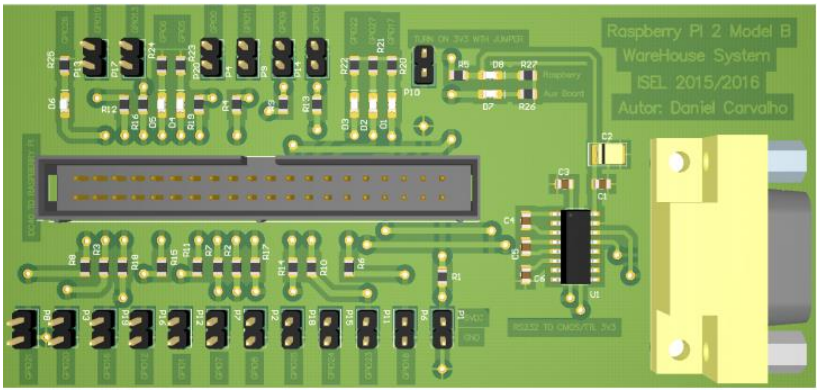


Figura 4.48 - Aspeto físico da placa eletrónica desenvolvida para interligar com o Raspberry Pi.

5. Resultados Experimentais – Casos de Estudo

Este capítulo está dedicado à apresentação de alguns resultados experimentais obtidos nesta dissertação, resultante do desenvolvimento das duas soluções (solução computacional e solução com microcomputador) já descritas no capítulo 4. Pretende-se que os resultados experimentais aqui apresentados permitam demonstrar a viabilidade das soluções propostas nesta dissertação. Os casos de estudo apresentados serão os mesmos para ambas as soluções:

- Situação de marcas de referência não encontradas;
- Situação de braço robótico em movimento;
- Situação de funcionamento normal sem erros.

Solução Computacional

Começa-se por ilustrar (ver Figura 5.1) a integração física da solução computacional no sistema existente, onde se observa os vários equipamentos e respetivas interligações.

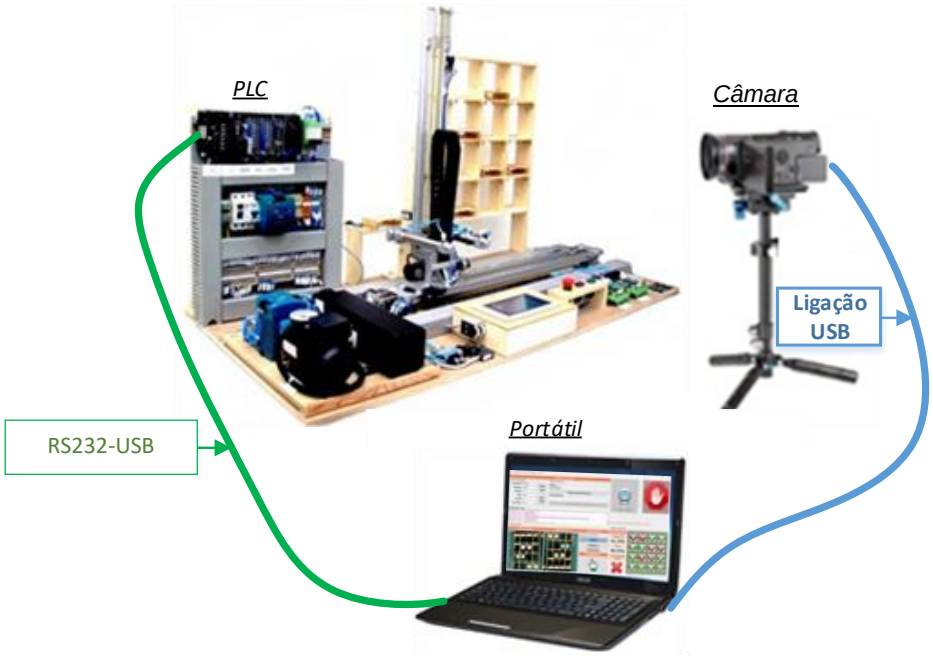


Figura 5.1 - Integração física (Solução Computacional).

Situação de marcas de referência não encontradas (ver Figura 5.2): Nesta situação, pelo facto de não ter sido possível a localização das 3 marcas de referência, o operador é informado pelo sistema (através de mensagens apresentadas na *Communication Trace*) com as seguintes mensagens:

- *"Não foram encontradas as marcas de referência!"*
- *"O estado de ocupação será lido a partir do PLC"*

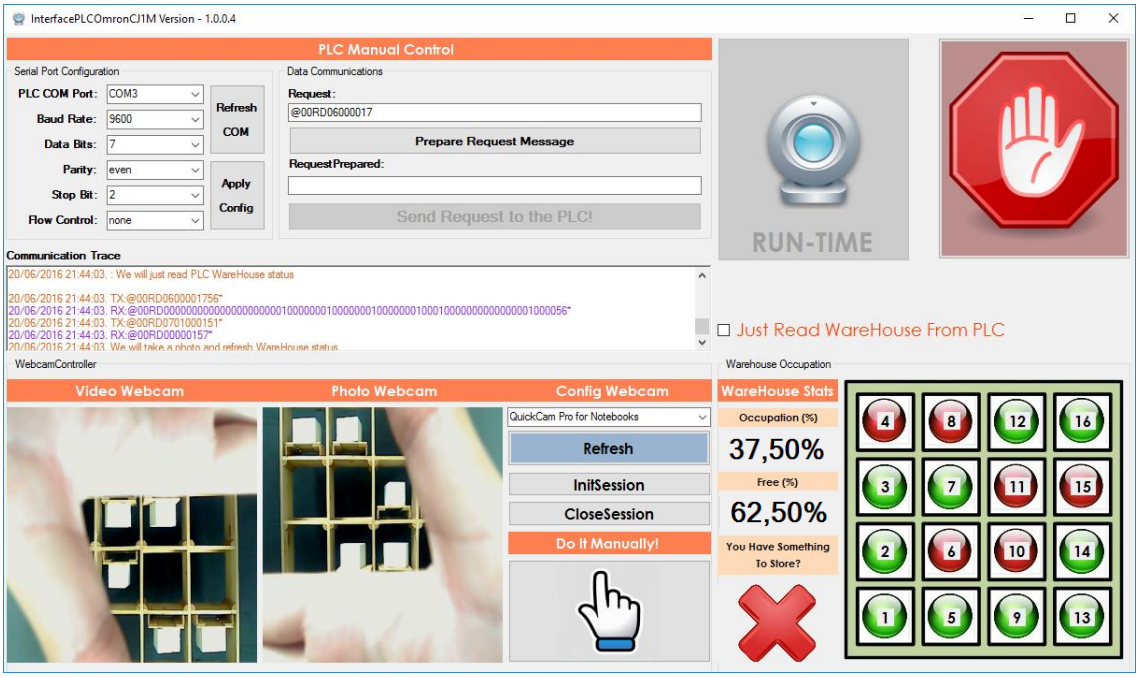


Figura 5.2 - Situação de marcas de referência não encontradas (Solução Computacional).

Neste cenário a fotografia é captada, e após análise desta, como não foram encontradas as 3 marcas de referência, são apenas lidos os registos do autómato que possuem o estado de ocupação dos 16 alvéolos (leitura dos registos D601 ao D6017), e por consequência, os indicadores do sistema de armazenamento (ver Figura 5.2) são atualizados. Verifica-se (através do símbolo "X") também que não existe nenhuma peça na posição de carregamento (leitura do registo D600). O vídeo que está ilustrado no "*Video Webcam*" está a ser captado em tempo real e com uma rotação de 180º face à sua posição de origem. Esta rotação apenas deve-se à forma de como a câmara foi posicionada no suporte físico.

Situação de braço robótico em movimento (ver Figura 5.3): Nesta situação, o operador é informado pelo sistema (através de mensagens apresentadas na *Communication Trace*) com a seguinte mensagem:

- *"Não é possível tirar fotografia!"*

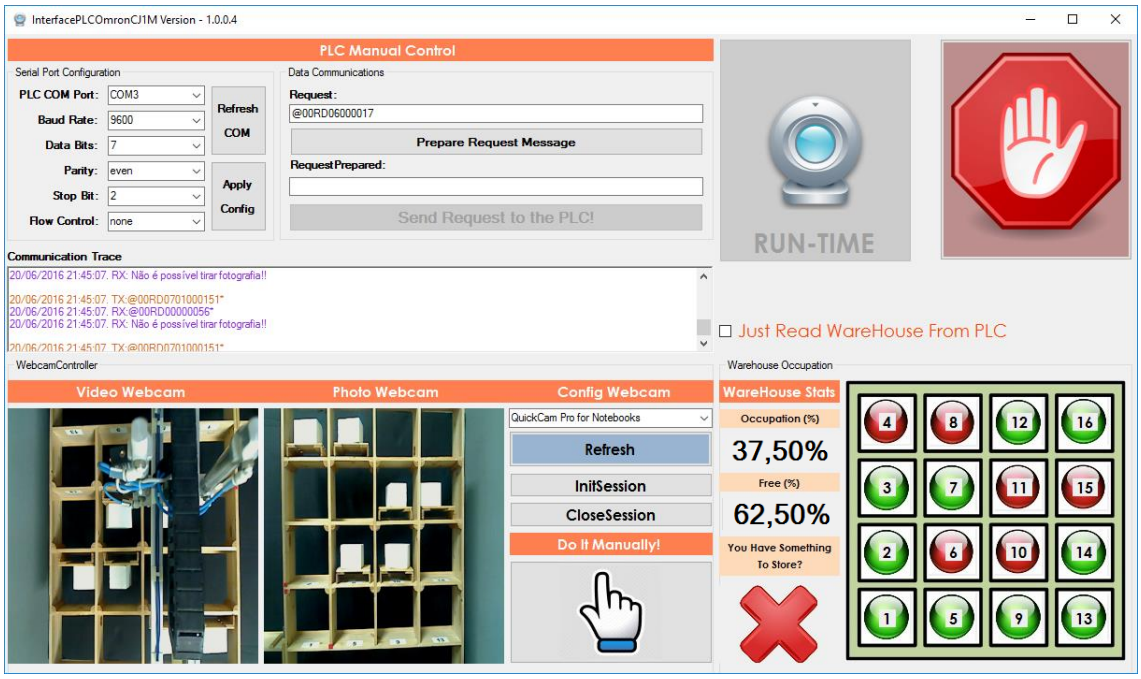


Figura 5.3 - Situação de braço robótico em movimento (Solução Computacional).

Nesta situação, como o braço robótico encontra-se fora da posição de origem, não é possivel tirar fotografia ao sistema, em que serão apenas lidos os registos do autómato que possuem o estado de ocupação dos 16 alvéolos (leitura dos registos D601 ao D6017), e por consequência, os indicadores do sistema de armazenamento (ver Figura 5.3) são atualizados. Verifica-se (através do símbolo "X") também que não existe nenhuma peça na posição de carregamento (leitura do registo D600).

A fotografia que se encontra apresentada na "*Photo Webcam*" (ver Figura 5.3) é a última fotografia válida antes do braço robótico ter iniciado o movimento.

Situação de funcionamento normal sem erros (ver Figura 5.4): Nesta situação, o operador é informado pelo sistema (através de mensagens apresentadas na *Communication Trace*) com a seguinte mensagem:

- *"Será captada a imagem e atualizado o estado de ocupação do armazém"*

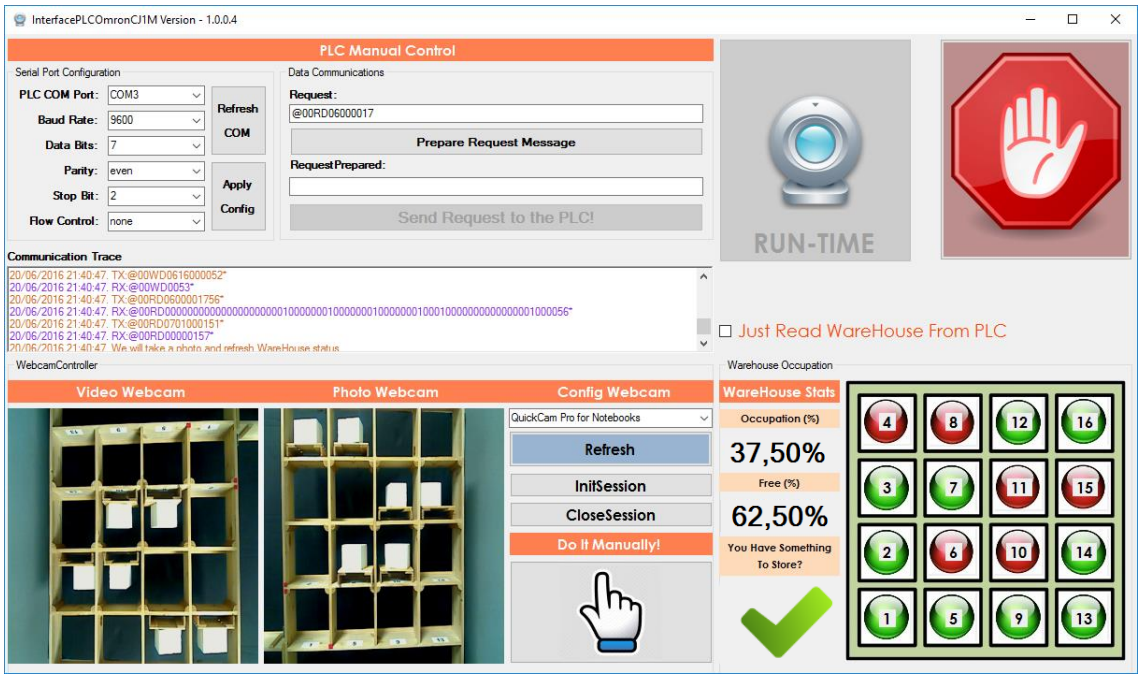


Figura 5.4 - Situação de funcionamento normal sem erros (Solução Computacional).

Nesta situação o sistema tem todas as condições para funcionar na sua normalidade, possibilitando assim a atualização do estado de ocupação dos alvéolos com o auxílio do sistema de visão. Verifica-se (através do símbolo "certo") também que existe uma peça na posição de carregamento (leitura do registo D600).

Solução com Microcomputador

Nesta solução serão apresentados, com o auxílio de um (*Liquid Crystal Display*) LCD conectado por HDMI ao *Raspberry Pi*, os resultados observados na linha de comandos em conjunto com o visualizador de imagens (fotografia que vai sendo adquirida ao sistema).

Começa-se por ilustrar (ver Figura 5.5) a integração física da solução com microcomputador no sistema existente, onde se observa os vários equipamentos e respetivas interligações.

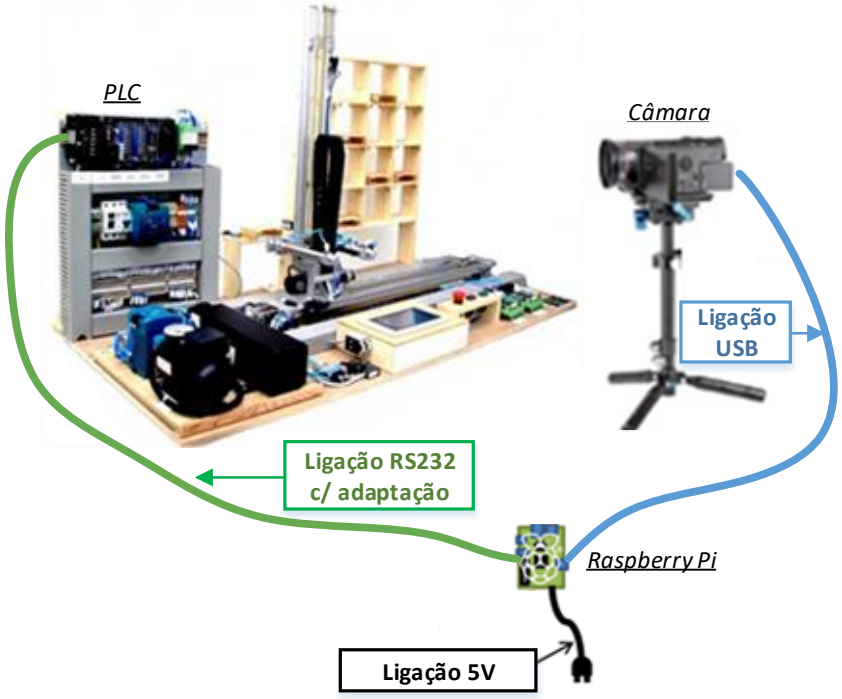


Figura 5.5 - Integração física (Solução com microcomputador).

A interligação designada por “Ligação RS232 c/ adaptação” (ver Figura 5.5) trata-se da placa eletrónica que foi desenvolvida para adaptar os sinais de comunicação entre o *Raspberry Pi* e o PLC. Esta placa eletrónica, como já foi referido, além da adaptação dos sinais de comunicação, possui alguns indicadores acerca do estado do sistema:

- Led Vermelho: O sistema está parado;
- Led Amarelo: Problemas na comunicação entre o *Raspberry Pi* e o PLC;
- Led Verde: O sistema está a funcionar corretamente.

Situação de marcas de referência não encontradas (ver Figura 5.6): Nesta situação, o sistema informa o operador (através de mensagens apresentadas na linha de comandos) com a seguinte mensagem:

- *"As marcas de referência não foram encontradas!"*

Figura 5.6 - Situação de marcas de referência não encontradas (Solução com Microcomputador).

Neste cenário a fotografia é captada, e após análise desta, como não foram encontradas as 3 marcas de referência, o sistema continua a tirar fotografias até que estas sejam encontradas. Pela observação da consola tátil (ver Figura 5.6) é possível comprovar que o estado de ocupação dos alvéolos continua a ser preenchido, mas por acção do PLC.

Situação de braço robótico em movimento (ver Figura 5.7): Nesta situação, o operador é informado pelo sistema (através de mensagens apresentadas na linha de comandos) com a seguinte mensagem:

- *"Mensagem OK - > No entanto agora não se pode tirar fotografia"*

Linha de Comandos

Consola Tátil (HMI)

Figura 5.7 - Situação de braço robótico em movimento (Solução com Microcomputador).

Nesta situação, como o braço robótico encontra-se fora da posição de origem, não é possivel tirar fotografia ao sistema. Pela observação da consola tátil (ver Figura 5.7) é possível comprovar que o estado de ocupação dos alvéolos continua a ser preenchido, mas por acção do PLC.

Situação de funcionamento normal sem erros (ver Figura 5.8): Nesta situação, o operador é informado pelo sistema (através de mensagens apresentadas na linha de comandos) com a seguinte mensagem:

- *"Modelo do Matlab sem erros!"*

Figura 5.8 - Situação de funcionamento normal sem erros (Solução com Microcomputador).

Nesta situação o sistema tem todas as condições para funcionar na sua normalidade, possibilitando assim a atualização do estado de ocupação dos alvéolos com o auxílio do sistema de visão.

6. CONCLUSÕES

Este capítulo é o culminar do trabalho de pesquisa e desenvolvimento que permitiu melhorar o funcionamento do simulador de armazenamento automático existente no Laboratório de Automação e Robótica da ADEEEA, ao qual foi introduzida a capacidade de deteção automática e em tempo real da presença ou ausência de peças nos respetivos alvéolos com recurso ao processamento de imagem.

Aqui são apresentadas as principais conclusões sobre os resultados obtidos, é feito um balanço das limitações da metodologia adotada e são avaliadas possíveis perspetivas de desenvolvimento futuro do trabalho realizado.

6.1. Conclusões gerais

Neste trabalho optou-se por desenvolver duas soluções (solução computacional e solução com microcomputador) capazes de executar esta nova funcionalidade, sendo assim possível optar por uma delas de acordo com a portabilidade do sistema desejado.

Através dos resultados obtidos experimentalmente, foi possível constatar que mesmo sendo o *Raspberry Pi* um dispositivo muito mais pequeno e muito mais barato comparado com o computador portátil, o seu desempenho foi semelhante ou mesmo melhor.

A metodologia proposta para a implementação deste sistema consiste essencialmente em três módulos funcionais, interligados entre si, a aquisição de imagem, o processamento de imagem e a interação entre as unidades de processamento e o PLC.

A aquisição de imagem tem como principal objetivo fornecer à unidade de processamento a imagem digitalizada captada do armazém. Consistindo numa aplicação direta de bibliotecas já existentes que contém o controlo da câmara digital, foi possível efetuar a aquisição de imagem com recurso ao *software* disponível.

O processamento de imagem visa a deteção da presença ou ausência de peças nos respetivos alvéolos. Consistindo numa sequência de processos de conversão de cor, subtração de imagem, binarização e extração de caraterísticas, os resultados a que conduziu são bastante positivos. É de salientar, neste processamento, a determinação automática da localização dos 16 alvéolos, que foi conseguida com o auxílio de três marcas de referência de cor vermelha localizadas no alvéolo da 1ª posição (determina a altura e largura do alvéolo, e espessura da estrutura entre alvéolos). Este aspeto contribuiu para o bom desempenho deste processo, tornando a localização dos 16 alvéolos praticamente imune a ligeiros movimentos da câmara digital em relação ao armazém (área de interesse), não sendo necessário proceder ao moroso processo de calibração da câmara.

A interação entre as unidades de processamento e o PLC permite a atualização em tempo real (visualizada na consola tátil HMI) da presença ou ausência de peças nos respetivos alvéolos. Para tal, foi necessário efetuar um estudo relativo ao protocolo de comunicação utilizado pelo PLC.

Relativamente às linguagens de programação e *softwares* utilizados, *C* (*MonoDevelop*), *.Net* (*Visual Studio C#*), *Matlab* e *Simulink*, foi possível verificar que são bastante versáteis, sobretudo devido à grande quantidade de bibliotecas que lhe podem ser adicionadas. Durante a fase de desenvolvimento do sistema, constatou-se que as bibliotecas *Image Processing Toolbox* e *Computer Vision System Toolbox* disponíveis no *software Matlab e Simulink* respetivamente, oferecem capacidades importantes para a criação de aplicações de visão computacional. Estas bibliotecas possuem potencial para promover pesquisas na área de visão computacional, pois são possuem inúmeras funcionalidades.

Quanto às limitações de ambas as soluções desenvolvidas, a iluminação não controlada e a baixa resolução da câmara utilizada foram dois fatores que por vezes adulteravam o correto funcionamento do sistema. No caso de existir uma luminosidade excessiva a insidir na estante do armazém, a câmara teve dificultade em diferenciar cores de tonalidade clara, e como os níveis de *threshold* do algoritmo são fixos, o sistema apresentou algumas anomalias tendo em conta estas condições do meio envolvente.
Caso de tratasse de um armazém industrial (por exemplo, com uma dimensão de 10x10metros), seria necessário mais do que uma câmara para conseguir cobrir toda a área

de análise. Utilizando técnicas de triangulação de imagens seria possível analisar todo o armazém.

Globalmente, pode-se considerar que os resultados obtidos neste trabalho comprovam que as metodologias adotadas nas duas soluções desenvolvidas permitem desempenhar convenientemente a nova funcionalidade implementada no simulador de armazenamento automático.

6.2. Perspetivas de desenvolvimento futuro

Os resultados obtidos neste trabalho tornam-no um bom ponto de partida para desenvolvimentos futuros neste simulador de armazenamento automático. Durante a sua execução, esteve sempre presente a preocupação de utilizar equipamentos de baixo custo e desenvolver algoritmos simples e organizados, de forma a tornar a sua reutilização viável. Tirando partido de toda a potencialidade do *Raspberry Pi,* (tecnologias *Bluetooth e wi-fi* disponíveis na nova versão *Raspberry Pi 3*) poderá ser possível num futuro próximo introduzir novas funcionalidades a este simulador, tais como a comunicação por *wi-fi* ou *Bluetooth* com outra unidade de processamento (por exemplo um *smartphone*). Quanto a possíveis melhorias em relação à solução desenvolvida, o algoritmo de reconhecimento automático da localização dos alvéolos poderá ser ainda aperfeiçoado, em que, com recurso a mais marcas de referência é possível identificar rotações ou variações de ângulos no que se refere à posição da câmara digital em relação ao armazém. A implementação de algoritmos que atenuem a influência da iluminação ambiente é outro aspecto que deve ser melhorado neste simulador dado que não possui qualquer tipo de iluminação controlada, como por exemplo, a utilização do algoritmo de *Otsu* para detetar automaticamente o melhor *threshold*. Em aplicações futuras, também se poderá dotar o sistema com a capacidade de reconhecer os objetos automaticamente, ou seja, o operador deseja uma peça identificada com a letra “A”, e com recurso a técnicas de reconhecimento de objetos, o sistema possa ser capaz de extrair a peça pretendida. No fundo, seria dado ao operador a possibilidade de escolher entre a extração de um objeto alojado num alvéolo específico (funcionalidade existente) ou por identificação expressa na superfície da peça (funcionalidade para implementação futura).

7. BIBLIOGRAFIA

[1] J. V. P. Ôlas, “Desenvolvimento de um sistema robótico flexível para utilização em farmácias,” Escola Superior de Tecnologia e Gestão de Viseu, Viseu, 2012.

[2] C. Couto, Introdução à Robótica Industrial, Universidade Aberta, 2000.

[3] [Online]. Available: http://eac-ajch.blogspot.pt/2009/09/automato-progrmavel-industrial-plc-1.html..

[4] Schneider Electric, “Centro de Formação: Oque é um PLC (autómato)?,” 2008. [Online]. Available: http://www.schneiderelectric.pt/documents/product-services/training/plc_cfp2008.pdf.

[5] F. O. N. e. G. Almeida, *Sistemas Robóticos,* Lisboa, 2008.

[6] D. D. Ardayfio, Fundamentals of Robotics, Detroit, Michigan: Marcel Dekker, INC., 1987.

[7] Unimate, “Unimate The First Industrial Robot,” [Online]. Available: http://www.robotics.org/joseph-engelberger/unimate.cfm.

[8] INFORUM SIMPÓSIO DE INFORMÁTICA, “Sistemas Embebidos e de Tempo-Real,” 2014. [Online]. Available: http://inforum.org.pt/INForum2014/sessoes/sistemas-embebidos-e-de-tempo-real.html.

[9] National Instruments, “Data Acquisition: I/O for Embedded Systems,” National Instruments, 02 10 2012. [Online]. Available: Data Acquisition: I/O for Embedded Systems.

[10] G. D. Micheli, “"Hardware/Software Co-Design", Proceedings of the IEEE, Vol. 85, No. 3,” Março 1997. [Online]. Available: http://courses.cs.tamu.edu/rabi/cpsc689/lectures/lecture01/hwswcodesign.pdf.

[11] A. J. M. d. Sousa, “Arquitecturas de Sistemas Robóticos e Localização em Tempo Real Através de Visão,” Universidade do Porto, Faculdade de Engenharia, Porto, 2003.

[12] J. H. a. T. Fratangelo, “Raspberry Pi Architecture,” [Online]. Available: http://www.macs.hw.ac.uk/~hwloidl/Courses/F28HS/slides_RPi_arch.pdf.

[13] C. V. R. Coutinho, “Robótica Móvel - Sistema de Condução Autónoma,” ISEL, Lisboa, 2014.

[14] AXIS Communications, “CCD and CMOS sensor technology,” [Online]. Available: http://www.axis.com/files/whitepaper/wp_ccd_cmos_40722_en_1010_lo.pdf.

[15] L. Timóteo, “Automação industrial encoders ópticos rotativos,” 09 02 2013. [Online]. Available: http://pt.slideshare.net/MarioTimotius/automao-industrial-encoders-pticos-rotaivos.

[16] Allen-Bradley, “Encoders ópticos incrementais,” [Online]. Available: http://ab.rockwellautomation.com/pt/motion-control/incremental-optical-encoder.

[17] Festo, [Online]. Available: https://www.festo.com/cat/nl_nl/products_050502.

[18] Festo, [Online]. Available: https://www.festo.com/cat/en_us/products_050501.

[19] Festo, [Online]. Available: http://www.festo-didactic.com/int-en/learning-systems/process-automation/edukit-pa/sensor,capacitive.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xMTE4LjY3NTI&basket=add&vid=8848.

[20] Festo, [Online]. Available: http://www.festo-didactic.com/ov3/media/customers/1100/00904083001075223684.pdf.

[21] Lin Engineering - The Step Motor Specialists, “211 Series - Compact Stepper Motor,” [Online]. Available: http://www.linengineering.com/products/stepper-motors/211-series/.

[22] A. Cordeiro, “Actuadores pneumáticos,” ISEL, Lisboa, 2014.

[23] J. P. Turíbio, “Cilindros,” 24 Fevereiro 2014. [Online]. Available: http://pt.slideshare.net/JoaoPedroTuribio/cilindros-16739036.

[24] Festo, “Atuadores convencionais,” [Online]. Available: https://www.festo.com/cat/pt_pt/products_010204.

[25] S. S. L. Ferreira, “Gestão de armazéns: implementação de um sistema de picking na indústria alimentar,” Escola Superior Agrária, IPSantarém, Santarém, 2012.

[26] Mecalux, “Estantes paletização convencional,” [Online]. Available: http://www.mecalux.pt/estantes-paletizacao/estantes-paletizacao-convencional.

[27] Mecalux, “Estantes paletização compacta,” [Online]. Available: http://www.mecalux.pt/estantes-paletizacao/estantes-paletizacao-compacta.

[28] Mecalux, “Estantes paletização Movirack,” [Online]. Available: http://www.mecalux.pt/estantes-paletizacao/estantes-paletizacao-movirack.

[29] Mecalux, “Estantes paletização dinâmica,” [Online]. Available: http://www.mecalux.pt/estantes-paletizacao/estantes-paletizacao-dinamica.

[30] Mecalux, “Estantes paletização Push-back,” [Online]. Available: http://www.mecalux.pt/estantes-paletizacao/estantes-paletizacao-push-back.

[31] Mecalux, “Transelevadores para caixas,” [Online]. Available: http://www.mecalux.pt/armazens-automaticos-para-caixas/transelevadores-para-caixas.

[32] Kardex Sistemas S.A. , “Megamat - Poupar espaço, aumentar a produtividade,” [Online]. Available: http://www.kardex-remstar.pt/pt/produtos/carrocel-vertical/megamat.html.

[33] Mecalux, “Pallet Shuttle,” [Online]. Available: http://www.mecalux.pt/armazens-automaticos-para-paletes/pallet-shuttle.

[34] S. Hong-ying, “The Application of Barcode Technology in Logistics and Warehouse Management,” Shenyang, China, 2009.

[35] “O que são QR Codes ?,” [Online]. Available: http://www.noseqret.pt/tudo-sobre-qr-codes/.

[36] “E-WMS support of QR or other 2D barcode types,” 17 Junho 2016. [Online]. Available: http://www.exactsoftware.com/docs/docview.aspx?documentid=%7B6d76c2bd-f8be-4709-840a-feba6085149a%7D&NoHeader=1&NoSubject=1.

[37] G. S. C. G. Z. z. Li Minbo, “A RFID-based Intelligent Warehouse Management System Design and Implementation,” Fudan University, Shanghai, China, 2011 Eighth IEEE International Conference on e-Business Engineering.

[38] Y. C. X. M. Bo Yan, “RFID Technology Applied in Warehouse Management System,” School of Economics and commerce, Guangdong, 510006, China, 2008 ISECS International Colloquium on Computing, Communication, Control, and Management.

[39] EC-Council, Wireless Safety, Clifton Park, NY, 2010.

[40] J. Rodrigues, “Algoritmo de seguimento de complexidade variável,” FEUP, Porto, 2012.

[41] T. B. B. Y.Ramadevi, “SEGMENTATION AND OBJECT RECOGNITION USING EDGE DETECTION TECHNIQUES,” *International Journal of Computer Science & Information Technology (IJCSIT),* vol. 2, 2010.

[42] G. K. A. K. Beant Kaur, “Detection and Classification of Printed Circuit Board Defects Using Image Subtraction Method,” IEEE, Patiala, 8 de Março de 2014.

[43] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision,* 5 de Janeiro de 2004.

[44] T. O. M. I. a. L. B. Yuki Sakai, “An Object Tracking System Based on SIFT and SURF Feature Extraction Methods,” International Conference on Network-Based Information Systems, Japan, 2015.

[45] T. T. a. L. V. G. Herbert Bay, “SURF: Speeded Up Robust Features”.

[46] OMRON, “Embaladora/equipamento de carregamento robótico orientado por sistema de visão,” OMRON, [Online]. Available: https://industrial.omron.pt/pt/solutions/packaging/packaging-machine-automation-solutions/robotic-infeed-module.

[47] INFAIMON, “Sistemas Multiespectrais aperfeiçoam o controlo de qualidade da argila na indústria cerâmica,” 13 Maio 2014. [Online]. Available: http://blog.infaimon.com/pt/2014/05/sistemas-multiespectrais-aperfeicoam-o-controlo-de-qualidade-da-argila-na-industria-ceramica/.

[48] OMRON, “FQ2 Smart Camera,” 2015. [Online]. Available: http://www.ia.omron.com/data_pdf/cat/fq2_q193-e1_14_2_csm1006918.pdf?id=3131.

[49] G. e. L. Fu, ROBOTICS - Control, Sensing, Vision and Intelligence, McGraw-Hill Book Co., 1987.

[50] [Online]. Available: http://computacaografica.ic.uff.br/transparenciasvol2cap3.pdf.

[51] H. Zanetti, “Visão Computacional com Python e OpenCV utilizando Kinect,” [Online]. Available: http://pt.slideshare.net/hzanetti/palestra-viso-computacional-tdc-2015.

[52] A. d. S. Carvalho, “Arquiteturas de Sistemas Robóticos e Localização em Tempo Real Através de Visão,” Porto, 2003.

[53] G. e. Woods, Digital Image Processing - Second Edition, 2002.

[54] N. Instruments, “NI PXI/PCI-1409 User Manual,” Fevereiro 2007. [Online]. Available: http://www.ni.com/pdf/manuals/372811c.pdf.

[55] “Image Acquisition,” National Instruments, 19 07 2012. [Online]. Available: http://www.ni.com/white-paper/2808/en/#toc1.

[56] ITU-R, “Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios,” Março 2011. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf.

[57] N. Otsu, “A Threshold Selection Method from Gray-Level Histogram,” em *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 1*, 1979, pp. 62-66.

[58] J. A. Edward R. Dougherty, “An Introduction to Nonlinear Image Processing,” Washington, 1994, pp. 7-16.

[59] J. M. S. Pereira, “Análise de Estruturas em Imagens Médicas: Aplicação ao Sistema Cardiovascular,” FEUP, Porto, 2011.

[60] D. E. R. Gaspar, “Raspberry Pi: a Smart Video Monitoring Platform,” Instituto Superior Técnico, Lisboa, 2014.

[61] OMRON, “Communications Commands - Reference Manual,” Fevereiro 2010. [Online].

[62] F. M. e. J. Badracim, “Projecto ARAU Simulador de Armazém Automático,” ISEL, Lisboa, 2007.

[63] Raspberry Pi, “Raspberry Pi MODEL B+,” [Online]. Available: https://cdn-shop.adafruit.com/datasheets/pi-specs.pdf.

[64] “RS-232 vs. TTL Serial Communication,” Sparkfun, 23 Novembro 2010. [Online]. Available: https://www.sparkfun.com/tutorials/215.

[65] M. Integrated, “+5V-Powered, Multichannel RS-232 Drivers/Receivers,” Maxim Integrated, [Online]. Available: https://www.maximintegrated.com/en/products/interface/transceivers/MAX232.html/tb_tab0.

[66] R. Pi, “Raspberry Pinout,” [Online]. Available: http://pinout.xyz/pinout/pin2_5v_power.

[67] D. Ricardo, Capítulo XXXI dos Princípios da Economia Política e Tributação.

[68] Tompkins, “The Warehouse Management Handbook,” 1998, p. 7.

[69] Saber Eletrónica, “Tecnologia RFID no armazenamento de pallets,” *Saber Eletrónica,* vol. 428, 2008.

[70] S. R. e. P. Norvig, Artificial Intelligence - A Modern Approach, Prentice Hall, 1994.

[71] [Online]. Available: http://paginas.fe.up.pt/~asousa/tsca/Omron/cursos_omr/Teoria1+2+3_V1_0.pdf.

[72] OMRON, “Guia de inspecção e controlo de qualidade,” 2012. [Online].

[73] [Online]. Available: http://eac-ajch.blogspot.pt/2009/09/automato-progrmavel-industrial-plc-1.html.

[74] [Online]. Available: http://compsee.com/industries/warehouse-and-distribution.

[75] Panasonic, “How a CCD Converts Light to Voltage,” [Online]. Available: http://av.jpn.support.panasonic.com/support/global/cs/dsc/knowhow/knowhow27.html.

[76] F. Ponce, Computer Vision - A Modern Aproach, 2002.

[77] [Online]. Available: http://definitions.uslegal.com/r/robotics/.

[78] National Instruments, “Data Acquisition: I/O for Embedded Systems,” [Online]. Available: http://www.ni.com/white-paper/7021/en/.

[79] Allen-Bradley, [Online]. Available: http://ab.rockwellautomation.com/Sensors-Switches/Laser-Sensors/Measurement-Laser-Sensors.

[80] Mecalux, “Transelevadores para paletes,” [Online]. Available: http://www.mecalux.pt/armazens-automaticos-para-paletes/transelevadores-para-paletes.

[81] Xerox, [Online]. Available: http://www.xerox.com/digital-printing/latest/FFSBR-25U.pdf.

[82] M. Piccardi, “Background subtraction techniques: a review,” IEEE, Univ. of Technol., Sydney, NSW, Australia , 2004.

[83] C. P. a. T. P. Anuj Mohan, “Example-Based Object Detection in images by Components,” IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 23, NO. 4, , 2001.

[84] M. B. S. P. B. G. S. A. D. D. P. Sonal D Kalro, “Using Image Processing for Detecting Defects in Printed Circuit Board,” *International Journal of Computer Techniques ,* vol. 2, 2 de Março de 2015.

8. ANEXO A – Algoritmo desenvolvido no *Matlab*

```
%**DISSERTAÇÃO - PROCESSAMENTO DE IMAGEM NUM SIMULADOR DE ARMAZENAMENTO AUTOMÁTICO**
%*****************************************************************************
%**********************MainProgram.m Funtion**********************
%*****************************************************************************
%*****************************DANIEL CARVALHO - A35772*************************
%*****************************************************************************

function [FINALgeneralInfoToPLC, error] = MainProgram(fullFileName)

findGeometric = false; %Flag
error = 0;

%Variables that goes to PLC
AUXgeneralInfoToPLC = zeros(4,4);
% 0: NO object!
% 1: RIGHT object!

%Matrix = 4x4
linhas = 4;
colunas = 4;

% %Directory to the file
% [FileName, PathName] = uigetfile('*.jpg', 'Select an image');
% %Concatenate string (Directory+FileName)
% fullFileName = strcat(PathName,FileName);

%Function that returns the coordenates of each position (position,xmin,ymin)
[arrayCoordenates, error] = CoordenatesOfEachPosition(fullFileName);

if(error == 1)
    FINALgeneralInfoToPLC = [ 9 9 9 9; 9 9 9 9; 9 9 9 9; 9 9 9 9 ];
    error = 1;
    return;
end

%Read image
originalImageIn = imread(fullFileName);
originalImage = imrotate(originalImageIn,180);
%Gives the number of colorbands to decide if we binaryze or not
[rows, columns, numberOfColorBands] = size(originalImage);

%If the image is RGB we convert her to a GrayScale image
if numberOfColorBands <= 1
    grayImage = originalImage;
else
    grayImage = rgb2gray(originalImage);
end

%Binarize the image
binaryImage = grayImage > 210;
%limpar imagem de lixo
binaryImage = bwmorph(binaryImage,'erode');
binaryImage = bwmorph(binaryImage,'close');

%Ignorar objectos <X área!
L=bwlabel(binaryImage,8);
stats=regionprops(binaryImage,'basic');
idx=find([stats.Area]>500);
bw3=ismember(L,idx);
bw2 = medfilt2(bw3, [20 20]);

%**********CUT AN IMAGE AND ANALYZE**********
fprintf('\n**********************ITS TIME TO RECOGNIZE THIS OBJECTS********************\n\n');
%Initialize the matrix position
position = 1;

%"For cicle" the fills all of matrix positions
for objectIndexColunas = 1 : colunas
    for objectIndexLinhas = 1 : linhas
        %Cut the image (16 images)
        img = imcrop(bw2, [arrayCoordenates(1,position) arrayCoordenates(2,position) arrayCoordenates(1,17) arrayCoordenates(2,17)]);

        %Count how many objects was found
        [labeledImage numberOfObjects] = bwlabel(img);

        if(numberOfObjects < 1) %There is NO object!
```

```
    overlayMessage = sprintf('There is NO object in this position');
    fprintf('Position %f x %f dont have object!\n\n', objectIndexLinhas, objectIndexColunas);

else %There is a VALID object!

    %Build the labels to show on screen
    Measurements = regionprops(labeledImage, 'Perimeter', 'Area', 'FilledArea', 'Solidity', 'Centroid');

    %Collect some of the measurements into individual arrays.
    Perimeter = [Measurements.Perimeter];
    Area = [Measurements.Area];
    FilledArea = [Measurements.FilledArea];
    Solidity = [Measurements.Solidity];
    Centroid = [Measurements.Centroid];

    %Calculate circularities
    circularities = Perimeter .^2 ./ (4 * pi * FilledArea);

    %Print the results to command window.
    fprintf('#, Perimeter, Area,    Centroid, Filled Area, Solidity, Circularities, Centroid\n');
    for objectIndex = 1 : numberOfObjects
        fprintf('%8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f\n', ...
            Perimeter(objectIndex), Area(objectIndex), Centroid(objectIndex),...
            FilledArea(objectIndex), Solidity(objectIndex), circularities(objectIndex), Centroid(objectIndex));
    end

   %fprintf('\n**********************ITS TIME TO RECOGNIZE THIS OBJECTS************
    %Gives us which one is a circle, a square or a triangle
    for objectIndex = 1 : numberOfObjects
        shape = 'Undefined object';
        findGeometric = false; %inicializa flag
        message = sprintf('Object %d its undefined\n', objectIndex);
        if circularities(objectIndex) < 1.45 && circularities(objectIndex) > 1.2
            findGeometric = true;
            shape = 'Square';
            message = sprintf('Object %d its a square\n', objectIndex);
        else if circularities(objectIndex) < 1.15
                findGeometric = true;
                shape = 'Circle';
                message = sprintf('Object %d its a circle\n', objectIndex);
            else if circularities(objectIndex) > 1.5
                    findGeometric = true;
                    shape = 'Triangle';
                    message = sprintf('Object %d its a triangle\n', objectIndex);
                end
            end
        end

        if findGeometric
            % Display in overlay above the object
            overlayMessage = sprintf('Object #%d = %s\ncirc = %.2f, s = %.2f', ...
            objectIndex, shape, circularities(objectIndex), Solidity(objectIndex));
        else
            % Display in overlay above the object
            overlayMessage = sprintf('Object #%d = %s\ncirc = %.2f, s = %.2f', ...
            objectIndex, shape, circularities(objectIndex), Solidity(objectIndex));
        end
    end

            %impixelinfo;
            fprintf('Position %f x %f have object!\n\n', objectIndexLinhas, objectIndexColunas);
            AUXgeneralInfoToPLC(objectIndexLinhas,objectIndexColunas) = 1;
        end
        %Increment position of matrix
        position = position+1;
    end
end

%Real Positions for PLC configuration (just for the real view of the matrix)
FINALgeneralInfoToPLC = flipud(AUXgeneralInfoToPLC)
end
```

```
%*******************************************************************************
%**DISSERTAÇÃO - PROCESSAMENTO DE IMAGEM NUM SIMULADOR DE ARMAZENAMENTO AUTOMÁTICO**
%*******************************************************************************
%************************CoordenatesOfEachPosition.m Funtion************************
%*******************************************************************************
%*****************************DANIEL CARVALHO - A35772*****************************
%*******************************************************************************

function [arrayCoordenates, error] = CoordenatesOfEachPosition(fullFileName)
%Function that returns the coordenates of each position (position,xmin,ymin)

%A marca está no perfil, logo é necessário analisar a imagem que está um pouco mais dentro.
margin = 10;
%Variable that gives the error
error = 0;
index = 0;
arrayCoordenates = [];

%Read the orginal image
figure(1)
originalImageIn = imread(fullFileName);
originalImage = imrotate(originalImageIn, 180);
%Matriz dimension (3x2) and this picture is located on row 1 and column 1
subplot(3,2,1);
%Show image
imshow(originalImage, []);
title('Original Image');

%*******************************************************************************
%EXTRACT RED POINTS
im1=originalImage(:,:,1);
%Matriz dimension (3x2) and this picture is located on row 1 and column 2
subplot(3,2,2);
%Show image
imshow(im1, []);
title('Extract Red band');
%*******************************************************************************

%*******************************************************************************
%LOW LEVEL -> CHANGE TO GRAYSCALE (8BITS)
im2=rgb2gray(originalImage);
%Matriz dimension (3x2) and this picture is located on row 2 and column 1
subplot(3,2,3);
%Show image
imshow(im2, []);
title('RGB2GRAY');
%*******************************************************************************

%*******************************************************************************
%SUBTRACT RED POINTS WITH GRAYIMAGE
im3=imsubtract(im1,im2);
%Matriz dimension (3x2) and this picture is located on row 2 and column 2
subplot(3,2,4);
%Show image
imshow(im3, []);
title('Subtract ExtractRedband with RGB2GRAY');
%*******************************************************************************

%*******************************************************************************
%MEDIUM LEVEL -> BINARIZE
%Binarize the image
im4=im3 > 70;
%Matriz dimension (3x2) and this picture is located on row 3 and column 1
subplot(3,2,5);
%Show image
imshow(im4, []);
title('Red Image');
%*******************************************************************************

%*******************************************************************************
%HIGH LEVEL -> RECOGNITION AND INTERPRETATION
%HANG TAGS
%Count how many objects was found
% [labeledImage numberOfObjects] = bwlabel(binaryImage);
[labeledImage numberOfObjects] = bwlabel(im4);
%FEATURES EXTRACTION
% Measurements = regionprops(binaryImage, 'Perimeter', 'Area', 'FilledArea', 'Solidity', 'Centroid');
Measurements = regionprops(im4, 'Perimeter', 'Area', 'FilledArea', 'Solidity', 'Centroid');
%*******************************************************************************
```

```
%Collect some of the measurements into individual arrays.
Perimeter = [Measurements.Perimeter];
Area = [Measurements.Area];
Centroid = [Measurements.Centroid];

    if(numberOfObjects == 3)
        for objectIndex = 1 : numberOfObjects
            %Print the results to command window.
            fprintf('Object: %d\n', objectIndex);
            fprintf('#Perimeter,  Area,    Centroid(x),  Centroid(y)\n');
            fprintf('%8.3f,   %8.3f, %8.3f,    %8.3f\n\n',Perimeter(objectIndex), Area(objectIndex), Centroid(objectIndex+index), Centroid(objectIndex+index+1));
            index = index + 1;
        end

        %Size of each analize area (TOTAL AREA of each position)
        AnalizeAreaWidthMax = Centroid(5) - Centroid(3);
        AnalizeAreaHeighMax = Centroid(6) - Centroid(4);
        perfil = Centroid(4) - Centroid(2);
        fprintf('The MAX size of each analize area is:\nWidth(x)=%8.3f and Heigh(y)=%8.3f\n', AnalizeAreaWidthMax, AnalizeAreaHeighMax);
        fprintf('The perfil Heigh(y)%8.3f\n\n', perfil);

        %*************OUTPUT FUNCTION*************
        %Size of each analize area (REAL AREA to analize on each position)
        x1minRef = Centroid(3) + margin;
        y1minRef = Centroid(4) + margin;

        x1maxRef = Centroid(5) - margin;
        y1maxRef = Centroid(6) - margin;

        AnalizeAreaWidthReal = x1maxRef - x1minRef;
        AnalizeAreaHeighReal = y1maxRef - y1minRef;

        fprintf('The real coordenates of reference position are:\nx1minRef=%8.3f \ny1minRef=%8.3f \nx1maxRef=%8.3f \ny1maxRef=%8.3f\n', x1minRef, y1minRef, x1maxRef, y1maxRef);
        fprintf('The REAL size of each analize area is:\nWidth(x)=%8.3f and Heigh(y)=%8.3f\n', AnalizeAreaWidthReal, AnalizeAreaHeighReal);

        %Calculate the rest of coordenates
        x2minRef = x1minRef;
        y2minRef = y1minRef - margin - perfil - margin - AnalizeAreaHeighReal;

        x3minRef = x2minRef;
        y3minRef = y2minRef - margin - perfil - margin - AnalizeAreaHeighReal;

        x4minRef = x3minRef;
        y4minRef = y3minRef - margin - perfil - margin - AnalizeAreaHeighReal;

        %New column
        x5minRef = x1minRef + AnalizeAreaWidthReal + margin + perfil + margin;
        y5minRef = y1minRef;

        x6minRef = x5minRef;
        y6minRef = y2minRef;

        x7minRef = x5minRef;
        y7minRef = y3minRef;

        x8minRef = x5minRef;
        y8minRef = y4minRef;

        %New column
        %x9minRef = x5minRef + AnalizeAreaWidthReal + margin + perfil + margin;
        x9minRef = x5minRef + AnalizeAreaWidthReal + margin + perfil;
        y9minRef = y5minRef;

        x10minRef = x9minRef;
        y10minRef = y6minRef;

        x11minRef = x9minRef;
        y11minRef = y7minRef;

        x12minRef = x9minRef;
        y12minRef = y8minRef;

        %New column
        %x13minRef = x9minRef + AnalizeAreaWidthReal + margin + perfil + margin;
        x13minRef = x9minRef + AnalizeAreaWidthReal + margin + perfil;
        y13minRef = y9minRef;

        x14minRef = x13minRef;
        y14minRef = y10minRef;

        x15minRef = x13minRef;
        y15minRef = y11minRef;

        x16minRef = x13minRef;
        y16minRef = y12minRef;

        %xmin e ymin of each analize area (REAL AREA to analize on each position)
        arrayCoordenates(1,1) = x1minRef;
        arrayCoordenates(2,1) = y1minRef;

        arrayCoordenates(1,2) = x2minRef;
        arrayCoordenates(2,2) = y2minRef;

        arrayCoordenates(1,3) = x3minRef;
        arrayCoordenates(2,3) = y3minRef;
```

```
        arrayCoordenates(1,4) = x4minRef;
        arrayCoordenates(2,4) = y4minRef;

        arrayCoordenates(1,5) = x5minRef;
        arrayCoordenates(2,5) = y5minRef;

        arrayCoordenates(1,6) = x6minRef;
        arrayCoordenates(2,6) = y6minRef;

        arrayCoordenates(1,7) = x7minRef;
        arrayCoordenates(2,7) = y7minRef;

        arrayCoordenates(1,8) = x8minRef;
        arrayCoordenates(2,8) = y8minRef;

        arrayCoordenates(1,9) = x9minRef;
        arrayCoordenates(2,9) = y9minRef;

        arrayCoordenates(1,10) = x10minRef;
        arrayCoordenates(2,10) = y10minRef;

        arrayCoordenates(1,11) = x11minRef;
        arrayCoordenates(2,11) = y11minRef;

        arrayCoordenates(1,12) = x12minRef;
        arrayCoordenates(2,12) = y12minRef;

        arrayCoordenates(1,13) = x13minRef;
        arrayCoordenates(2,13) = y13minRef;

        arrayCoordenates(1,14) = x14minRef;
        arrayCoordenates(2,14) = y14minRef;

        arrayCoordenates(1,15) = x15minRef;
        arrayCoordenates(2,15) = y15minRef;

        arrayCoordenates(1,16) = x16minRef;
        arrayCoordenates(2,16) = y16minRef;

        %(REAL AREA to analize on each position)
        arrayCoordenates(1,17) = AnalizeAreaWidthReal;
        arrayCoordenates(2,17) = AnalizeAreaHeighReal;
    else
        error = 1; %We didn't found the 3 reference marks
    end
end
```

9. ANEXO B – Algoritmo desenvolvido no *Visual Studio C#*

Class Main.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace InterfacePLCOmron
{
    public partial class Main : Form
    {
        public Main()
        {
            InitializeComponent();
            label_sw_version.Text ="V" + Application.ProductVersion.ToString();
        }

        private void button_ModoManual_Click(object sender, EventArgs e)
        {
            ManualMode manualMode = new ManualMode();
            manualMode.Show();
        }
    }
}
```

Class WorkPort.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.IO.Ports;
using System.Diagnostics;
using System.Threading;
using System.Windows.Forms;

namespace InterfacePLCOmron
{
    class WorkPort
    {
        SerialPort plcCOM = new SerialPort();
        private string namePort;

        public WorkPort(string namePort)
        {
            // TODO: Complete member initialization
            this.namePort = namePort;
            plcCOM.PortName = namePort;
        }

        public void openPort()
        {
            try
            {
                if (!plcCOM.IsOpen)
                    plcCOM.Open();
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

```
public void closePort()
{
    try
    {
        if (plcCOM.IsOpen)
            plcCOM.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

public void sendBuffer(string bufferToSend)
{
    try
    {
        cleanBuffer();
        plcCOM.Write(bufferToSend);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

public string receivedBuffer()
{
    string response = String.Empty;
    Stopwatch timeToStop = new Stopwatch();
    int timeout = 5000;
    bool stop = false;

    plcCOM.ReadTimeout = 5000;

    try
    {
        cleanBuffer();

        timeToStop.Reset();
        timeToStop.Start();

        do
        {
            if (plcCOM.BytesToRead > 0)
            {
                response += plcCOM.ReadExisting();
                if (response != String.Empty && response.Contains("\r"))
                    stop = true;
            }
        }
        while (timeToStop.ElapsedMilliseconds <= timeout && !stop);

        if(!stop && timeToStop.ElapsedMilliseconds >= timeout)
            response = "PLC didn't response!";
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    return response;
}
```

```
public void setConfig(int baudRate = 0, int dataBits = 0, string parity = "", int stopBit = 0, string flowControl
= "")
        {
            try
            {
                if (!plcCOM.IsOpen)
                    plcCOM.Open();

                plcCOM.BaudRate = baudRate;
                plcCOM.DataBits = dataBits;

                switch (parity)
                {
                    case "none":
                        plcCOM.Parity = Parity.None;
                        break;

                    case "even":
                        plcCOM.Parity = Parity.Even;
                        break;

                    case "odd":
                        plcCOM.Parity = Parity.Odd;
                        break;

                    case "space":
                        plcCOM.Parity = Parity.Space;
                        break;

                    case "mark":
                        plcCOM.Parity = Parity.Mark;
                        break;
                }

                switch (stopBit)
                {
                    case 1:
                        plcCOM.StopBits = StopBits.One;
                        break;

                    case 2:
                        plcCOM.StopBits = StopBits.Two;
                        break;
                }

                switch (flowControl)
                {
                    case "none":
                        plcCOM.Handshake = Handshake.None;
                        break;

                    case "Xon/Xoff":
                        plcCOM.Handshake = Handshake.XOnXOff;
                        break;
                }
            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        public void cleanBuffer()
        {
            try
            {
                if (plcCOM.IsOpen)
                {
                    plcCOM.DiscardInBuffer();
                    plcCOM.DiscardOutBuffer();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString());
            }
        }
```

```
        private static string GetStringFromAsciiHex(String input)
        {
            if (input.Length % 2 != 0)
                throw new ArgumentException("input");

            byte[] bytes = new byte[input.Length / 2];

            for (int i = 0; i < input.Length; i += 2)
            {
 // Split the string into two-bytes strings which represent a hexadecimal value, and convert each value to a byte
                String hex = input.Substring(i, 2);
                bytes[i/2] = Convert.ToByte(hex, 16);
            }

            return System.Text.ASCIIEncoding.ASCII.GetString(bytes);
        }
    }
}
```

Class ManualMode.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Diagnostics;
using System.Threading;
using System.Collections;
using AForge.Video;
using AForge.Video.DirectShow;
using MathWorks.MATLAB.NET.Arrays;
using MathWorks.MATLAB.NET.Utility;
using Project_ImageDotNet;

namespace InterfacePLCOmron
{
    public partial class ManualMode : Form
    {
        delegate void pictureBox(PictureBox image);

        private bool DeviceExist = false;
        private FilterInfoCollection videoDevices;
        private VideoCaptureDevice videoSource = null;

        public string namePort = String.Empty;
        public int baudRate = 0;
        public int dataBits = 0;
        public string parity = String.Empty;
        public int stopBit = 0;
        public string flowControl = String.Empty;
        public bool stopped = false;
        bool _runApplication = false;
        public double resultFree = 0.0;
        public double resultOccupation = 0.0;
        public double countZeros = -1;

        SerialPort workPort = new SerialPort();

        Thread _readFromPLC = null;

        public ManualMode()
        {
            InitializeComponent();

            this.Text = this.Text + " Version - " + Application.ProductVersion;

            string[] ports = System.IO.Ports.SerialPort.GetPortNames();

            getCamList();
```

```
            pictureBox1.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox2.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox3.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox4.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox5.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox6.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox7.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox8.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox9.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox10.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox11.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox12.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox13.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox14.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox15.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;
            pictureBox16.Image = InterfacePLCOmron.Properties.Resources.yellow_ball;

            try
            {
                comboBox_PLC_COM.Items.Clear();
                textBox_Request.Clear();
                richTextBox_Trace.Clear();
                label_freeSpace.Text = "0.00%";
                label_occupationSpace.Text = "0.00%";
                textBox_Request.Text = "@00RD06000017"; //@ + 00 (unit node PLC)
                button_SendRequest.Enabled = false;
                button_StopRun.Enabled = false;

                foreach (string s in ports)
                {
                    comboBox_PLC_COM.Items.Add(s);
                }

                comboBox_BaudRate.Text = Config.Default.BaudRate;
                comboBox_DataBits.Text = Config.Default.DataBits;
                comboBox_Parity.Text = Config.Default.Parity;
                comboBox_StopBit.Text = Config.Default.StopBit;
                comboBox_FlowControl.Text = Config.Default.FlowControl;

                if (!String.IsNullOrEmpty(Config.Default.PortName))
                    if (comboBox_PLC_COM.Items.Count > 0)
                        if (comboBox_PLC_COM.Items.IndexOf(Config.Default.PortName) >= 0)
                            comboBox_PLC_COM.SelectedIndex =
comboBox_PLC_COM.Items.IndexOf(Config.Default.PortName);
            }
            catch (Exception exp)
            {
                MessageBox.Show("Get Available Serial Ports Exception:" + exp.Message, Application.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

```
          //get the devices name
        private void getCamList()
        {
            try
            {
                videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);
                comboBox1.Items.Clear();
                if (videoDevices.Count == 0)
                    throw new ApplicationException();

                DeviceExist = true;
                foreach (FilterInfo device in videoDevices)
                {
                    comboBox1.Items.Add(device.Name);
                }
                comboBox1.SelectedIndex = 0; //make dafault to first cam
            }
            catch (ApplicationException exp)
            {
                MessageBox.Show(exp.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                DeviceExist = false;
                comboBox1.Items.Add("No capture device on your system");
            }
        }

        //prepare buffer to send to PLC
        private string RequestPrepared(string bufferToSend)
        {
            string _FCS = String.Empty;
            string commandToSend = String.Empty;
            string charTerminator = "*" + "\r";

            _FCS = FCSCalculator(ASCIItoBinary(bufferToSend));

            return commandToSend = bufferToSend + _FCS + charTerminator;
        }

        #region B U T T O N  A C T I O N S
        private void button_SendRequest_Click(object sender, EventArgs e)
        {
            try
            {
                string bufferToSend = textBox_RequestPrepared.Text;
                string receivedBuffer = String.Empty;

                if (comboBox_PLC_COM.Text != String.Empty && comboBox_BaudRate.Text != String.Empty &&
comboBox_DataBits.Text != String.Empty && comboBox_Parity.Text != String.Empty && comboBox_StopBit.Text !=
String.Empty && comboBox_FlowControl.Text != String.Empty)
                {
                    WorkPort workPort = new WorkPort(comboBox_PLC_COM.Text);
                    workPort.openPort();
                    workPort.setConfig(Convert.ToInt32(comboBox_BaudRate.Text),
Convert.ToInt32(comboBox_DataBits.Text), comboBox_Parity.Text, Convert.ToInt32(comboBox_StopBit.Text),
comboBox_FlowControl.Text);

                    workPort.sendBuffer(bufferToSend);
                    richTextBox_Trace.SelectionColor = Color.Chocolate;
                    richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ". TX:" +
bufferToSend + Environment.NewLine);
                    receivedBuffer = workPort.receivedBuffer();
                    richTextBox_Trace.SelectionColor = Color.BlueViolet;
                    richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ". RX:" +
receivedBuffer + Environment.NewLine);
                    richTextBox_Trace.ScrollToCaret();

                    workPort.closePort();
                }
                else
                    MessageBox.Show("You must select a PortCom and his configs!");
            }
            catch (ApplicationException exp)
            {
                MessageBox.Show(exp.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            button_SendRequest.Enabled = false;
        }
```

```
        private void button_PrepareRequest_Click(object sender, EventArgs e)
        {
            button_SendRequest.Enabled = true;
            textBox_RequestPrepared.Text = RequestPrepared(textBox_Request.Text);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string[] ports = System.IO.Ports.SerialPort.GetPortNames();
            comboBox_PLC_COM.Items.Clear();

            foreach (string s in ports)
            {
                comboBox_PLC_COM.Items.Add(s);
            }

            if (comboBox_PLC_COM.Items.Count > 0)
                comboBox_PLC_COM.SelectedIndex = 0;
        }

        private void button_ApplyConfig_Click(object sender, EventArgs e)
        {
            try
            {
                Config.Default.PortName = comboBox_PLC_COM.Text;
                Config.Default.BaudRate = comboBox_BaudRate.Text;
                Config.Default.DataBits = comboBox_DataBits.Text;
                Config.Default.Parity = comboBox_Parity.Text;
                Config.Default.StopBit = comboBox_StopBit.Text;
                Config.Default.FlowControl = comboBox_FlowControl.Text;

                Config.Default.Save();
            }
            catch (Exception exp)
            {
                MessageBox.Show("Get Available Serial Ports Exception:" + exp.Message, Application.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void button_StopRun_Click(object sender, EventArgs e)
        {
            try
            {
                if (_readFromPLC != null && _readFromPLC.IsAlive)
                    _readFromPLC.Suspend();
            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                button_RunTime_.Enabled = true;
                button_StopRun.Enabled = false;
            }

        }

        private void button_Refresh_Click(object sender, EventArgs e)
        {
            getCamList();
        }

        private void button_closeSession_Click(object sender, EventArgs e)
        {
            if (!(videoSource == null))
                if (videoSource.IsRunning)
                {
                    videoSource.SignalToStop();
                    videoSource = null;
                }
        }
```

```
        private void button_initSession_Click(object sender, EventArgs e)
        {
            if (DeviceExist)
            {
                videoSource = new VideoCaptureDevice(videoDevices[comboBox1.SelectedIndex].MonikerString);
                videoSource.NewFrame += new NewFrameEventHandler(video_NewFrame);
                videoSource.Start();
            }
        }

        private void button_takePicture_Click(object sender, EventArgs e)
        {
            string path_webcam = "C:/Users/d_06_/Desktop/img_webcam.bmp";
            string path_file = "C:/Users/d_06_/Desktop/img1_default.bmp";

            countZeros = 0;

            richTextBox_Trace.Invoke(new EventHandler(delegate
            {
                richTextBox_Trace.SelectionColor = Color.Chocolate;
                richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ". : In this case,
we don't communicate with PLC, just processing the image and gives us the warehouse occupation!" +
Environment.NewLine);
            }));

            try
            {
                if (!(videoSource == null))
                {
                    pictureBox19.Image = (Bitmap)pictureBox18.Image.Clone();
                    pictureBox21.Image = (Bitmap)pictureBox19.Image.Clone();
                    pictureBox19.Image.RotateFlip(RotateFlipType.Rotate180FlipNone);
                    pictureBox21.Image.Save(path_webcam, System.Drawing.Imaging.ImageFormat.Bmp);

                    string[] pos = new string[17];
                    MWNumericArray cellout = null;
                    MWCharArray cellin = null;

                    Project_ImageDotNet.Project_ImageDotNet obj = new Project_ImageDotNet.Project_ImageDotNet();

                    cellin = path_webcam;

                    MWCharArray image_path = (MWCharArray)cellin;

                    cellout = (MWNumericArray)obj.MainProgram(image_path);

                    MWNumericArray pos4 = (MWNumericArray)cellout[1];
                    MWNumericArray pos3 = (MWNumericArray)cellout[2];
                    MWNumericArray pos2 = (MWNumericArray)cellout[3];
                    MWNumericArray pos1 = (MWNumericArray)cellout[4];

                    MWNumericArray pos8 = (MWNumericArray)cellout[5];
                    MWNumericArray pos7 = (MWNumericArray)cellout[6];
                    MWNumericArray pos6 = (MWNumericArray)cellout[7];
                    MWNumericArray pos5 = (MWNumericArray)cellout[8];

                    MWNumericArray pos12 = (MWNumericArray)cellout[9];
                    MWNumericArray pos11 = (MWNumericArray)cellout[10];
                    MWNumericArray pos10 = (MWNumericArray)cellout[11];
                    MWNumericArray pos9 = (MWNumericArray)cellout[12];

                    MWNumericArray pos16 = (MWNumericArray)cellout[13];
                    MWNumericArray pos15 = (MWNumericArray)cellout[14];
                    MWNumericArray pos14 = (MWNumericArray)cellout[15];
                    MWNumericArray pos13 = (MWNumericArray)cellout[16];

                    pos[1] = pos1.ToString();
                    pos[2] = pos2.ToString();
                    pos[3] = pos3.ToString();
                    pos[4] = pos4.ToString();
                    pos[5] = pos5.ToString();
                    pos[6] = pos6.ToString();
                    pos[7] = pos7.ToString();
                    pos[8] = pos8.ToString();
                    pos[9] = pos9.ToString();
                    pos[10] = pos10.ToString();
                    pos[11] = pos11.ToString();
                    pos[12] = pos12.ToString();
```

```
                    pos[13] = pos13.ToString();
                    pos[14] = pos14.ToString();
                    pos[15] = pos15.ToString();
                    pos[16] = pos16.ToString();

                    if(pos[1] == "9")
                    {
                        richTextBox_Trace.Invoke(new EventHandler(delegate
                        {
                            richTextBox_Trace.SelectionColor = Color.Chocolate;
                            richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ". :
We can't find fiducial marks!" + Environment.NewLine);
                        }));
                    }

                    for (int i = 1; i <= 16; i++)
                    {
                        if(pos[i] == "0")
                            ++countZeros;
                    }

                    label_freeSpace.Invoke(new EventHandler(delegate
                    {
                        resultFree = Math.Round(countZeros / 16 * 100, 2);
                        label_freeSpace.Text = resultFree.ToString() + "%";
                    }));

                    label_occupationSpace.Invoke(new EventHandler(delegate
                    {
                        resultOccupation = Math.Round(((16 - countZeros) / 16 * 100), 2);
                        label_occupationSpace.Text = resultOccupation.ToString() + "%";
                    }));

                    for (int i = 1; i <= cellout.NumberOfElements; i++)
                    {
                        RefreshWareHouse(i, pos[i]);
                    }
                }
            }
            catch (Exception exp)
            {
                MessageBox.Show(exp.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
                Console.WriteLine("Error: {0}", exp);
            }
        }

        private void button_RunTime__Click(object sender, EventArgs e)
        {
            try
            {
                button_RunTime_.Enabled = false;
                button_StopRun.Enabled = true;

                if (backgroundWorker1.IsBusy != true)
                {
                    backgroundWorker1.RunWorkerAsync();
                }
                else
                {
                    MessageBox.Show("Thread Busy.", "Process File", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
            }
            catch (Exception exp)
            {
                MessageBox.Show("Start Process:" + exp.Message, Application.ProductName, MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            finally
            {
                if (!backgroundWorker1.IsBusy)
                    button_RunTime_.Enabled = true;
            }
        }
        #endregion
```

```
#region A U X  M E T H O D E S
        public string ASCIItoBinary(string bufferToSend)
        {
            string converted = String.Empty;
            // convert string to byte
            byte[] byteArray = Encoding.ASCII.GetBytes(bufferToSend);

            for (int i = 0; i < byteArray.Length; i++)
            {
                for (int j = 0; j < 8; j++)
                {
                    converted += (byteArray[i] & 0x80) > 0 ? "1" : "0";
                    byteArray[i] <<= 1;
                }
            }
            return converted;
        }

        public static string BinaryToString(string data)
        {
            List<Byte> byteList = new List<Byte>();

            for (int i = 0; i < data.Length; i += 8)
            {
                byteList.Add(Convert.ToByte(data.Substring(i, 8), 2));
            }
            return Encoding.ASCII.GetString(byteList.ToArray());
        }

        public static string BinaryStringToHexString(string binary)
        {
            StringBuilder result = new StringBuilder(binary.Length / 8 + 1);

            // TODO: check all 1's or 0's... Will throw otherwise

            int mod4Len = binary.Length % 8;
            if (mod4Len != 0)
            {
                // pad to length multiple of 8
                binary = binary.PadLeft(((binary.Length / 8) + 1) * 8, '0');
            }

            for (int i = 0; i < binary.Length; i += 8)
            {
                string eightBits = binary.Substring(i, 8);
                result.AppendFormat("{0:X2}", Convert.ToByte(eightBits, 2));
            }

            return result.ToString();
        }

        public string FCSCalculator(string bufferToSend)
        {
            //********************FUNCTION FCS CALCULATOR*******************

            //Arrays (aux1, aux2...) will store which column, for example:
            //@ -> 0100 0000
            //0 -> 0011 0000
            //0 -> 0011 0000
            //R -> 0101 0010
            //D -> 0100 0100

            //aux1 -> 00000
            //aux2 -> 10011
            //aux3 -> 01100
            //aux4 -> 01110
            //aux5 -> 00000
            //aux6 -> 00001
            //aux7 -> 00010
            //aux8 -> 00000
```

```
//Int (xor1, xor2...) will do the XOR operation for each column, for example:
            //xor1 = 0 ^ 0 ^ 0 ^ 0 ^ 0
            //xor2 = 1 ^ 0 ^ 0 ^ 1 ^ 1
            //xor3 = 0 ^ 1 ^ 1 ^ 0 ^ 0
            //xor4 = 0 ^ 1 ^ 1 ^ 1 ^ 0
            //xor5 = 0 ^ 0 ^ 0 ^ 0 ^ 0
            //xor6 = 0 ^ 0 ^ 0 ^ 0 ^ 1
            //xor7 = 0 ^ 0 ^ 0 ^ 1 ^ 0
            //xor8 = 0 ^ 0 ^ 0 ^ 0 ^ 0

            //Result of XOR operation:
            //     Col.1  Col.2
            //@ -> 0100   0000
            //0 -> 0011   0000
            //0 -> 0011   0000
            //R -> 0101   0010
            //D -> 0100   0100

            //Col.1 = CONCATENATE(xor1 + xor2 + xor3 + xor4)
            //Col.2 = CONCATENATE(xor5 + xor6 + xor7 + xor8)

            //FCS = You must convert CONCATENATE(Col.1 + Col.2) to Hexadecimal

            int i = 0;
            string[] aux1 = new string[(bufferToSend.Length) / 8];
            string[] aux2 = new string[(bufferToSend.Length) / 8];
            string[] aux3 = new string[(bufferToSend.Length) / 8];
            string[] aux4 = new string[(bufferToSend.Length) / 8];
            string[] aux5 = new string[(bufferToSend.Length) / 8];
            string[] aux6 = new string[(bufferToSend.Length) / 8];
            string[] aux7 = new string[(bufferToSend.Length) / 8];
            string[] aux8 = new string[(bufferToSend.Length) / 8];

            int xor1 = 0;
            int xor2 = 0;
            int xor3 = 0;
            int xor4 = 0;
            int xor5 = 0;
            int xor6 = 0;
            int xor7 = 0;
            int xor8 = 0;

            string resultXOR_left = String.Empty;
            string resultXOR_right = String.Empty;

            string auxBuffer = bufferToSend;
            string binaryToHex = String.Empty;
            string FCS1 = String.Empty;
            string FCS2 = String.Empty;

            for (i = 0; i < (bufferToSend.Length) / 8; i++)
            {
                aux1[i] = auxBuffer.Substring(i * 8, 1);
                aux2[i] = auxBuffer.Substring(i * 8 + 1, 1);
                aux3[i] = auxBuffer.Substring(i * 8 + 2, 1);
                aux4[i] = auxBuffer.Substring(i * 8 + 3, 1);
                aux5[i] = auxBuffer.Substring(i * 8 + 4, 1);
                aux6[i] = auxBuffer.Substring(i * 8 + 5, 1);
                aux7[i] = auxBuffer.Substring(i * 8 + 6, 1);
                aux8[i] = auxBuffer.Substring(i * 8 + 7, 1);
            }

            for (i = 0; i < (bufferToSend.Length) / 8; i++)
            {
                xor1 ^= Convert.ToInt32(aux1[i]);
                xor2 ^= Convert.ToInt32(aux2[i]);
                xor3 ^= Convert.ToInt32(aux3[i]);
                xor4 ^= Convert.ToInt32(aux4[i]);
                xor5 ^= Convert.ToInt32(aux5[i]);
                xor6 ^= Convert.ToInt32(aux6[i]);
                xor7 ^= Convert.ToInt32(aux7[i]);
                xor8 ^= Convert.ToInt32(aux8[i]);
            }

            resultXOR_left = Convert.ToString(xor1) +
                             Convert.ToString(xor2) +
                             Convert.ToString(xor3) +
                             Convert.ToString(xor4);
```

```
    resultXOR_right = Convert.ToString(xor5) +
                      Convert.ToString(xor6) +
                      Convert.ToString(xor7) +
                      Convert.ToString(xor8);

    FCS1 = resultXOR_left.PadLeft(4, '0');
    FCS2 = resultXOR_right.PadLeft(4, '0');

    binaryToHex = BinaryStringToHexString(FCS1 + FCS2);

    return binaryToHex;
}
#endregion

private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
{
    try
    {
        if (_readFromPLC != null && _readFromPLC.IsAlive)
            _readFromPLC.Resume();
        else
        {
            //Start Read Bool Position From PLC
            _readFromPLC = new Thread(new ParameterizedThreadStart(doIt_readFromPLC));
            _readFromPLC.Start();
        }
    }
    catch (Exception innerExp)
    {
        MessageBox.Show(innerExp.Message, "Error Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
        e.Cancel = true;
        e.Result = innerExp.Message;
    }
    finally
    {
        if (workPort != null && workPort.IsOpen)
            workPort.Close();
    }
}

public void doIt_readFromPLC(object input)
{
    try
    {
        string path_webcam = "C:/Users/d_06_/Desktop/img_webcam.bmp";
        string path_file = "C:/Users/d_06_/Desktop/img1_default.bmp";
        string bufferToSend_ReadPLC = String.Empty;
        string bufferToSend_WritePLC = String.Empty;
        string receivedBuffer = String.Empty;
        string _plcCOM = String.Empty;
        string _baudRate = String.Empty;
        string _dataBits = String.Empty;
        string _parity = String.Empty;
        string _stopBit = String.Empty;
        string _flowControl = String.Empty;
        string[] result = new string[17];
        bool _justRead = false;
        bool _readImageDefault = false;
        int i = 0;

        checkBox_justRead.Invoke(new EventHandler(delegate
        {
            _justRead = checkBox_justRead.Checked;

        }));

        comboBox_PLC_COM.Invoke(new EventHandler(delegate
        {
            _plcCOM = comboBox_PLC_COM.Text;

        }));

        comboBox_BaudRate.Invoke(new EventHandler(delegate
        {
            _baudRate = comboBox_BaudRate.Text;

        }));
```

```
                comboBox_DataBits.Invoke(new EventHandler(delegate
                {
                    _dataBits = comboBox_DataBits.Text;

                }));

                comboBox_Parity.Invoke(new EventHandler(delegate
                {
                    _parity = comboBox_Parity.Text;

                }));

                comboBox_StopBit.Invoke(new EventHandler(delegate
                {
                    _stopBit = comboBox_StopBit.Text;

                }));

                comboBox_FlowControl.Invoke(new EventHandler(delegate
                {
                    _flowControl = comboBox_FlowControl.Text;

                }));

                if (_plcCOM != String.Empty && _baudRate != String.Empty && _dataBits != String.Empty && _parity
!= String.Empty && _stopBit != String.Empty && _flowControl != String.Empty)
                {
                    WorkPort workPort = new WorkPort(_plcCOM);
                    workPort.openPort();
                    workPort.setConfig(Convert.ToInt32(_baudRate), Convert.ToInt32(_dataBits), _parity,
Convert.ToInt32(_stopBit), _flowControl);

                    while (!_runApplication)
                    {
                        bufferToSend_WritePLC = "@00RD07010001";
                        bufferToSend_WritePLC = RequestPrepared(bufferToSend_WritePLC);

                        workPort.sendBuffer(bufferToSend_WritePLC);

                        richTextBox_Trace.Invoke(new EventHandler(delegate
                        {
                            richTextBox_Trace.SelectionColor = Color.Chocolate;
                            richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ".
TX:" + bufferToSend_WritePLC + Environment.NewLine);
                        }));

                        receivedBuffer = workPort.receivedBuffer();

                        richTextBox_Trace.Invoke(new EventHandler(delegate
                        {
                            richTextBox_Trace.SelectionColor = Color.BlueViolet;
                            richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ".
RX:" + receivedBuffer + Environment.NewLine);
                            richTextBox_Trace.ScrollToCaret();
                        }));

                        if (receivedBuffer.ElementAt(10) == '1')
                        {
                            countZeros = -1;

                            checkBox_justRead.Invoke(new EventHandler(delegate
                            {
                                _justRead = checkBox_justRead.Checked;

                            }));

                            if (videoSource != null && !_justRead)
                            {
                                richTextBox_Trace.Invoke(new EventHandler(delegate
                                {
                                    richTextBox_Trace.SelectionColor = Color.Chocolate;
                                    richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime()
+ ". Será captada a imagem e atualizado o estado de ocupação do armazém" + Environment.NewLine);
                                }));

                                pictureBox19.Invoke(new EventHandler(delegate
                                {
```

```
                    pictureBox19.Image = (Bitmap)pictureBox18.Image.Clone();
                    pictureBox21.Image = (Bitmap)pictureBox19.Image.Clone();
                    pictureBox19.Image.RotateFlip(RotateFlipType.Rotate180FlipNone);
                    pictureBox21.Image.Save(path_webcam, System.Drawing.Imaging.ImageFormat.Bmp);
                }));

                string[] pos = new string[17];
                MWNumericArray cellout = null;
                MWCharArray cellin = null;

                Project_ImageDotNet.Project_ImageDotNet obj = new
Project_ImageDotNet.Project_ImageDotNet();

                if (_readImageDefault)
                    cellin = path_file;
                else
                    cellin = path_webcam;

                MWCharArray image_path = (MWCharArray)cellin;

                cellout = (MWNumericArray)obj.MainProgram(image_path);

                MWNumericArray pos4 = (MWNumericArray)cellout[1];
                MWNumericArray pos3 = (MWNumericArray)cellout[2];
                MWNumericArray pos2 = (MWNumericArray)cellout[3];
                MWNumericArray pos1 = (MWNumericArray)cellout[4];

                MWNumericArray pos8 = (MWNumericArray)cellout[5];
                MWNumericArray pos7 = (MWNumericArray)cellout[6];
                MWNumericArray pos6 = (MWNumericArray)cellout[7];
                MWNumericArray pos5 = (MWNumericArray)cellout[8];

                MWNumericArray pos12 = (MWNumericArray)cellout[9];
                MWNumericArray pos11 = (MWNumericArray)cellout[10];
                MWNumericArray pos10 = (MWNumericArray)cellout[11];
                MWNumericArray pos9 = (MWNumericArray)cellout[12];

                MWNumericArray pos16 = (MWNumericArray)cellout[13];
                MWNumericArray pos15 = (MWNumericArray)cellout[14];
                MWNumericArray pos14 = (MWNumericArray)cellout[15];
                MWNumericArray pos13 = (MWNumericArray)cellout[16];

                pos[1] = pos1.ToString();
                pos[2] = pos2.ToString();
                pos[3] = pos3.ToString();
                pos[4] = pos4.ToString();
                pos[5] = pos5.ToString();
                pos[6] = pos6.ToString();
                pos[7] = pos7.ToString();
                pos[8] = pos8.ToString();
                pos[9] = pos9.ToString();
                pos[10] = pos10.ToString();
                pos[11] = pos11.ToString();
                pos[12] = pos12.ToString();
                pos[13] = pos13.ToString();
                pos[14] = pos14.ToString();
                pos[15] = pos15.ToString();
                pos[16] = pos16.ToString();

                Console.ReadLine();

                //WE CAN'T FIND FIDUCIAL MARKS
                if (pos[1] != "9")
                {
                    for (int j = 1; j <= 16; j++)
                    {
                        bufferToSend_WritePLC = "@00WD06" + j.ToString().PadLeft(2, '0') +
pos[j].ToString().PadLeft(4, '0');
                        bufferToSend_WritePLC = RequestPrepared(bufferToSend_WritePLC);

                        workPort.sendBuffer(bufferToSend_WritePLC);

                        richTextBox_Trace.Invoke(new EventHandler(delegate
                        {
                            richTextBox_Trace.SelectionColor = Color.Chocolate;
                            richTextBox_Trace.AppendText(Environment.NewLine +
DateTime.Now.ToLocalTime() + ". TX:" + bufferToSend_WritePLC + Environment.NewLine);
                        }));
```

```
                                    receivedBuffer = workPort.receivedBuffer();

                                    richTextBox_Trace.Invoke(new EventHandler(delegate
                                    {
                                        richTextBox_Trace.SelectionColor = Color.BlueViolet;
                                        richTextBox_Trace.AppendText(Environment.NewLine +
DateTime.Now.ToLocalTime() + ". RX:" + receivedBuffer + Environment.NewLine);
                                        richTextBox_Trace.ScrollToCaret();
                                    }));

                                    Thread.Sleep(50);

                                    if (pos[j] == "0")
                                        ++countZeros;
                                }

                                label_freeSpace.Invoke(new EventHandler(delegate
                                {
                                    resultFree = Math.Round(countZeros / 16 * 100, 2);
                                    label_freeSpace.Text = resultFree.ToString() + "%";
                                }));

                                label_occupationSpace.Invoke(new EventHandler(delegate
                                {
                                    resultOccupation = Math.Round(((16 - countZeros) / 16 * 100), 2);
                                    label_occupationSpace.Text = resultOccupation.ToString() + "%";
                                }));
                            }
                            else
                            {
                                richTextBox_Trace.Invoke(new EventHandler(delegate
                                {
                                    richTextBox_Trace.SelectionColor = Color.Red;
                                    richTextBox_Trace.AppendText(Environment.NewLine +
DateTime.Now.ToLocalTime() + ". : Não foram encontradas as marcas de referência!" + Environment.NewLine);
                                    richTextBox_Trace.SelectionColor = Color.Chocolate;
                                    richTextBox_Trace.AppendText(Environment.NewLine +
DateTime.Now.ToLocalTime() + ". : O estado de ocupação será lido apartir do PLC" + Environment.NewLine);
                                }));
                            }
                        }
                        else
                        {
                            richTextBox_Trace.Invoke(new EventHandler(delegate
                            {
                                richTextBox_Trace.SelectionColor = Color.Chocolate;
                                richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime()
+ ". O estado de ocupação será lido apartir do PLC" + Environment.NewLine);
                            }));
                        }

                        //We will read the WareHouse status by PLC "@ 00 RD 0600 0016 FCS * \r"
                        bufferToSend_ReadPLC = "@00RD06000017";
                        bufferToSend_ReadPLC = RequestPrepared(bufferToSend_ReadPLC);
                        workPort.sendBuffer(bufferToSend_ReadPLC);

                        richTextBox_Trace.Invoke(new EventHandler(delegate
                        {
                            richTextBox_Trace.SelectionColor = Color.Chocolate;
                            richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ".
TX:" + bufferToSend_ReadPLC + Environment.NewLine);
                        }));

                        receivedBuffer = workPort.receivedBuffer();

                        richTextBox_Trace.Invoke(new EventHandler(delegate
                        {
                            richTextBox_Trace.SelectionColor = Color.BlueViolet;
                            richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ".
RX:" + receivedBuffer + Environment.NewLine);
                            richTextBox_Trace.ScrollToCaret();

                        }));

                        if (receivedBuffer != "PLC não respondeu!" && receivedBuffer != String.Empty)
                        {
                            for (i = 0; i < result.Length; i++)
```

```
                                {
                                    result[i] = receivedBuffer.ElementAt(10 + 4 * i).ToString();
                                    RefreshWareHouse(i, result[i]);
                                }
                            }

                            Thread.Sleep(1);
                            //workPort.closePort();
                        }
                        else
                        {
                            richTextBox_Trace.Invoke(new EventHandler(delegate
                            {
                                richTextBox_Trace.SelectionColor = Color.BlueViolet;
                                richTextBox_Trace.AppendText(Environment.NewLine + DateTime.Now.ToLocalTime() + ".
RX: Não é possível tirar fotografia!" + Environment.NewLine);
                                richTextBox_Trace.ScrollToCaret();
                            }));
                            Thread.Sleep(50);
                        }
                    }
                }
                else
                    MessageBox.Show("Tem de configurar a porta de comunicações!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Mensagem de Erro", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

              finally
            {
                button_RunTime_.Invoke(new EventHandler(delegate
                {
                    button_RunTime_.Enabled = true;

                }));
            }
        }

        public void RefreshWareHouse(int picture, string result)
        {
            if (result == "9")
            {
                pictureBox1.Invoke(new EventHandler(delegate { pictureBox1.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox2.Invoke(new EventHandler(delegate { pictureBox2.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox3.Invoke(new EventHandler(delegate { pictureBox3.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox4.Invoke(new EventHandler(delegate { pictureBox4.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox5.Invoke(new EventHandler(delegate { pictureBox5.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox6.Invoke(new EventHandler(delegate { pictureBox6.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox7.Invoke(new EventHandler(delegate { pictureBox7.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox8.Invoke(new EventHandler(delegate { pictureBox8.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox9.Invoke(new EventHandler(delegate { pictureBox9.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox10.Invoke(new EventHandler(delegate { pictureBox10.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox11.Invoke(new EventHandler(delegate { pictureBox11.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox12.Invoke(new EventHandler(delegate { pictureBox12.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox13.Invoke(new EventHandler(delegate { pictureBox13.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox14.Invoke(new EventHandler(delegate { pictureBox14.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox15.Invoke(new EventHandler(delegate { pictureBox15.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
                pictureBox16.Invoke(new EventHandler(delegate { pictureBox16.Image =
InterfacePLCOmron.Properties.Resources.blue_aqua_ball__49x50_; }));
```

```
                pictureBox20.Invoke(new EventHandler(delegate { pictureBox20.BackgroundImage =
          InterfacePLCOmron.Properties.Resources.interrogar;
                pictureBox20.BackgroundImageLayout = ImageLayout.Zoom; }));
            return;
          }

switch (picture)
            {
                case 0:
                    if (result == "1" || result == "2")
                    {
                        pictureBox20.Invoke(new EventHandler(delegate {
                        pictureBox20.BackgroundImage = InterfacePLCOmron.Properties.Resources.checkmark;
                        pictureBox20.BackgroundImageLayout = ImageLayout.Zoom;}));
                    }
                    else
                    {
                        pictureBox20.Invoke(new EventHandler(delegate {
                            pictureBox20.BackgroundImage =
InterfacePLCOmron.Properties.Resources._119498563188281957tasto_8_architetto_franc_01_svg_med;
                            pictureBox20.BackgroundImageLayout = ImageLayout.Zoom;
                        }));
                    }
                    break;

                case 1:
                    if(result == "1" || result == "2")
                        pictureBox1.Invoke(new EventHandler(delegate { pictureBox1.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox1.Invoke(new EventHandler(delegate { pictureBox1.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 2:
                    if (result == "1" || result == "2")
                        pictureBox2.Invoke(new EventHandler(delegate { pictureBox2.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox2.Invoke(new EventHandler(delegate { pictureBox2.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 3:
                    if (result == "1" || result == "2")
                        pictureBox3.Invoke(new EventHandler(delegate { pictureBox3.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox3.Invoke(new EventHandler(delegate { pictureBox3.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 4:
                    if (result == "1" || result == "2")
                        pictureBox4.Invoke(new EventHandler(delegate { pictureBox4.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox4.Invoke(new EventHandler(delegate { pictureBox4.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 5:
                    if (result == "1" || result == "2")
                        pictureBox5.Invoke(new EventHandler(delegate { pictureBox5.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox5.Invoke(new EventHandler(delegate { pictureBox5.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 6:
                    if (result == "1" || result == "2")
                        pictureBox6.Invoke(new EventHandler(delegate { pictureBox6.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox6.Invoke(new EventHandler(delegate { pictureBox6.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;
```

```
                case 7:
                    if (result == "1" || result == "2")
                        pictureBox7.Invoke(new EventHandler(delegate { pictureBox7.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox7.Invoke(new EventHandler(delegate { pictureBox7.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 8:
                    if (result == "1" || result == "2")
                        pictureBox8.Invoke(new EventHandler(delegate { pictureBox8.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox8.Invoke(new EventHandler(delegate { pictureBox8.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 9:
                    if (result == "1" || result == "2")
                        pictureBox9.Invoke(new EventHandler(delegate { pictureBox9.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox9.Invoke(new EventHandler(delegate { pictureBox9.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 10:
                    if (result == "1" || result == "2")
                        pictureBox10.Invoke(new EventHandler(delegate { pictureBox10.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox10.Invoke(new EventHandler(delegate { pictureBox10.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;
```

```
case 11:
                    if (result == "1" || result == "2")
                        pictureBox11.Invoke(new EventHandler(delegate { pictureBox11.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox11.Invoke(new EventHandler(delegate { pictureBox11.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 12:
                    if (result == "1" || result == "2")
                        pictureBox12.Invoke(new EventHandler(delegate { pictureBox12.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox12.Invoke(new EventHandler(delegate { pictureBox12.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 13:
                    if (result == "1" || result == "2")
                        pictureBox13.Invoke(new EventHandler(delegate { pictureBox13.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox13.Invoke(new EventHandler(delegate { pictureBox13.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 14:
                    if (result == "1" || result == "2")
                        pictureBox14.Invoke(new EventHandler(delegate { pictureBox14.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox14.Invoke(new EventHandler(delegate { pictureBox14.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 15:
                    if (result == "1" || result == "2")
                        pictureBox15.Invoke(new EventHandler(delegate { pictureBox15.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox15.Invoke(new EventHandler(delegate { pictureBox15.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;

                case 16:
                    if (result == "1" || result == "2")
                        pictureBox16.Invoke(new EventHandler(delegate { pictureBox16.Image =
InterfacePLCOmron.Properties.Resources.red_ball; }));
                    else
                        pictureBox16.Invoke(new EventHandler(delegate { pictureBox16.Image =
InterfacePLCOmron.Properties.Resources.green_ball; }));
                    break;
            }
        }

        private void ManualMode_FormClosing(object sender, FormClosingEventArgs e)
        {
            try
            {
                _runApplication = true;
                if (_readFromPLC != null)
                {
                    if (_readFromPLC.ThreadState.ToString() == "Running")
                        _readFromPLC.Abort();
                    else
                    {
                        _readFromPLC.Resume();
                        Thread.Sleep(10);
                        _readFromPLC.Abort();
                    }
                }
                this.Hide();
            }
```

```
            catch (Exception exp)
            {
                MessageBox.Show("Get Available Serial Ports Exception:" + exp.Message, Application.ProductName,
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        void video_NewFrame(object sender, NewFrameEventArgs eventArgs)
        {
            pictureBox18.Image = (Bitmap)eventArgs.Frame.Clone();
        }
    }
}
```

10. ANEXO C – Algoritmo desenvolvido no *Raspberry Pi*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
#include <termios.h>

#include <wiringPi.h>
#include <wiringSerial.h>

#define  LEDNOMSG 0
#define  LEDERROMSG         1
#define  LEDERROSIM         2
#define  LEDIN              4

//*************************************************************************************************
//********Configuração dos dados da porta série************** 7 bits dados, paridade even e 2 stopbits
void ConfiguraParametrosPorta(int fd)
{
 struct termios options;
 tcgetattr(fd, &options); //Ler parametros actuais (PARA CONSULTAR TODOS OS PARAMETROS ESCREVER NO TERMINAL: "man
tcgetattr"
 options.c_cflag &= ~CSIZE ;
 options.c_cflag |= CS7; //se pretendermos 5, 6 ou 7 bits de dados (usar CS5, CS6 ou CS7)
 options.c_cflag |= PARENB; //Permitir bit de paridade (por defeito par ou even)
 //options.c_cflag |= PARODD; //tirar o comentário se pretendermos a paridade impar ou ODD
 options.c_cflag |= CSTOPB; //tirar o comentário se pretendermos 2 stop bits
 options.c_cc[VTIME] = 20;    // Ten seconds (100 deciseconds) para o Timeout da resposta
 tcsetattr(fd, TCSANOW, &options); //Escrever os novos valores

} //********fim do procedimento de configuração dos dados da porta série**************

//******************Esta função retorna se existem ou não dados disponíveis antes de 2 segundos na porta série
//*************************************************************************************************************
unsigned char NaoExistemDadosDisponiveis(int fd)
{
    int data;
    unsigned int tmp1, tmp2, d_tmp;
    tmp1 = millis();

    do
    {
        data = serialDataAvail(fd);
        tmp2 = millis();
        d_tmp = tmp2 - tmp1;
    } while ((data <= 0) && (d_tmp < 1000));

    fflush(stdout);
    return (d_tmp >= 1000);
}
//***************************Fim da
função*************************************************************************
//*************************************************************************************************************************
*

int main()
{
    int a, b, c, d, i, z, g, st, msgcompleta = 0;
    int FCS1 = 0, FCS2 = 0;
    char FCSCHR1, FCSCHR2;
    char nib1, nib2;
    FILE* my_stream;
    unsigned int tempo1, tempo2, Delta_tempo;

    if (wiringPiSetup() == -1)
    {
        fprintf(stdout, "Não foi possível iniciar o módulo wiringPi: %s\n", strerror(errno));
        return 1;
    }

    //**********Vou definir quais os pinos são de entrada e quais são de saída*****************************
    pinMode(LEDNOMSG, OUTPUT);
    pinMode(LEDERROMSG, OUTPUT);
```

```
    pinMode(LEDERROSIM, OUTPUT);
    //pinMode (LEDIN, INPUT); comment

    for (a = 0; ; a++) //Vai fazer um ciclo infinito para processar toda a informação de forma ciclica no
programa. Sai do programa com CTRL+C
    {
        int fd;

        if ((fd = serialOpen("/dev/ttyAMA0", 9600)) < 0)
        {
            fprintf(stderr, "Não foi possível abrir a porta série: %s\n", strerror(errno));
            return 1;
        }

        ConfiguraParametrosPorta(fd); //Vai configurar dados da porta e respectivo timeout para a resposta do PLC

        //*************************************************************************************************
        //Primeiro vai enviar uma mensagem para o PLC para saber se pode começar a tirar fotografias
        //vai ao registo D0701 e verifica o que lá está

        char msg_init[] = "@00RD0701000151*"; //Esta mensagem já inclui o FCS (51Hex)
        msg_init[16] = 13; //Adiciona o CR para terminar a mensagem
        serialPrintf(fd, msg_init); //Manda a mensagem inicial para o PLC
                                    //fprintf(stdout," %s\n", msg_init); comment

        //*************************************************************************
        //Vai primeiro verificar se há dados vindos do PLC
        if (NaoExistemDadosDisponiveis(fd))
        {
            printf("Erro msg sem resposta \n");  // Não existem dados.
            digitalWrite(LEDNOMSG, HIGH);     // On//Vai dar mensagem e depois irá acender um LED
            delay(500);
            digitalWrite(LEDNOMSG, LOW);    // On//Vai dar mensagem e depois irá acender um LED
        }
        else
        {
            //Como existem dados vai guardar todos os caracteres recebidos até receber o caracter 13 ou 27 para
abandonar o programa
            //digitalWrite (LED, LOW) ; //Vai apagar o LED caso não estivesse apagado comment

            msgcompleta = 0; //Inicializa a flag que indica o fim da mensagem recebida

            char DataIn[16] = "";

            for (b = 0; ; b++) //Vai fazer um ciclo infinito para esperar pela mensagem completa do PLC ou pelo
timeout
            {
                st = (serialGetchar(fd));

                if ((st == 42)) //Para testar está o caracter 42 que é o '*'
                {
                    msgcompleta = 1; //Quando receber o caracter 13 então a mensagem está completa
                    break;// ascii de CR ou timeout sai do ciclo infinito
                }

                if ((st == -1)) //Para testar está o caracter 42 que é o '*' mas deverá ser o 13 que é o CR
                {
                    msgcompleta = 0; //mensagem incompleta por timeout
                    break;// sai do ciclo infinito por timeout (definido no procedimento ConfiguraParametrosPorta)
                }

                DataIn[b] = toascii(st);  //Vai guardando no ciclo os caracteres ascii

                if (msgcompleta == 0)
                {
                    printf("Mensagem incompleta -> Timeout \n");
                    //digitalWrite (LEDERROMSG, HIGH); // On//Vai dar mensagem e depois irá acender um LED
                    delay(100);
                    //digitalWrite (LEDERROMSG, LOW);  // OFF//Vai dar mensagem e depois irá acender um LED
                } //close: if (msgcompleta==0)
                else //Então é porque é mensagem completa = 1
                {
                    int lng = strlen(DataIn);
                    if (((DataIn[5] != '0') || (DataIn[6] != '0')) || (lng < 13))
                    {
                        if ((DataIn[5] != '0') || (DataIn[6] != '0'))
                            printf("Erro na mensagem recebida - > End Code \n ");
```

```
                        if (lng < 13)
                            printf("Erro na mensagem recebida - > Comprimento inesperado \n");

                        //serialFlush(fd); comment
                        //digitalWrite (LEDERROMSG, HIGH);      // On//Vai dar mensagem e depois irá acender um
LED
                        delay(100);
                        //digitalWrite (LEDERROMSG, LOW);       // OFF//Vai dar mensagem e depois irá acender um
LED
                        fflush(stdout);
                    } //close: if (((DataIn[5]!='0') || (DataIn[6]!='0')) || (lng<13))
                    else //Vai realizar um FCS para identicar a presença de um erro
                    {
                        FCS1 = FCS2 = 0;
                        lng = strlen(DataIn) - 2; //O comprimento da mensagem não pode contar com os dois últimos
caracteres do FCS recebido

                        for (c = 0; c < lng; c++)
                        {
                            nib1 = (DataIn[c] >> 4) & 0x0F;
                            FCS1 = FCS1 ^ nib1;
                            nib2 = (DataIn[c]) & 0x0F;
                            FCS2 = FCS2 ^ nib2;
                        }

                        if (FCS1 < 10)
                            FCSCHR1 = (int)('0') + FCS1;
                        else
                            FCSCHR1 = toascii(55 + FCS1);

                        if (FCS2 < 10)
                            FCSCHR2 = (int)('0') + FCS2;
                        else
                            FCSCHR2 = toascii(55 + FCS2);

                        if ((FCSCHR1 != DataIn[11]) || (FCSCHR2 != DataIn[12]))
                        {
                            printf("Erro na mensagem recebida - > FCS error ");
                            //digitalWrite (LEDERROMSG, HIGH);   // On//Vai dar mensagem e depois irá acender um
LED
                            delay(100);
                            //digitalWrite (LEDERROMSG, LOW);    // OFF//Vai dar mensagem e depois irá acender um
LED
                        } //close: if ((FCSCHR1!=DataIn[11]) || (FCSCHR2!=DataIn[12]))
                        else //se também não houver erros de FCS então vamos verificar se o que está no registo é
0 ou 1
                        {
                            if ((DataIn[10]) == '1') //Se a mensagem estiver a 1 podemos tirar a foto e arrancar
com o programa do Matlab
                            {
                                printf("Mensagem OK - > Resultado: A iniciar processo do Matlab...\n");
                                system("sudo ./raspberrypi_test2"); //localização e programa do Matlab/simulink
                                printf("O programa do simulink terminou com sucesso.\n");

                                //Agora vai ficar aqui a leitura do ficheiro de texto dos erros...
                                my_stream = fopen("ErrorLog0.txt", "r");
                                if (my_stream == NULL)
                                {
                                    printf("Não foi possível abrir o ficheiro ErrorLog0\n");
                                    //LIGAR ALGUM LED DE ERRO
                                }
                                else
                                {
                                    char buff2[1];
                                    printf("Ficheiro ErrorLog0.txt aberto\n");
                                    fread(buff2, 1, 1, my_stream); //Ler e guardar em buff2
                                    fclose(my_stream); //Fechar o ficheiro
                                    printf("Ficheiro ErrorLog0.txt fechado\n");
                                    my_stream = NULL;

                                    if (buff2[0] == 'X')
                                    {
                                        printf("As marcas de referência não foram encontradas!\n");
                                        //LIGAR ALGUM LED DE ERRO
                                    }
                                    else
                                    {
                                        printf("Modelo do Matlab sem erros!\n");
```

```
                                    //Agora vai ficar aqui a leitura do ficheiro de texto do array...
                                    my_stream = fopen("Array0.txt", "r");
                                    if (my_stream == NULL)
                                    {
                                        printf("Não foi possível abrir o ficheiro Array0.txt\n");
                                        //LIGAR ALGUM LED DE ERRO
                                    }
                                    else
                                    {
                                        char buff[16];
                                        printf("Ficheiro Array0.txt aberto\n");
                                        fread(buff, 16, 1, my_stream); //Ler e guardar em buff
                                        fclose(my_stream); //Fechar o ficheiro
                                        printf("Ficheiro Array0.txt fechado\n");
                                        my_stream = NULL;

                                        //Este ciclo FOR vai ciclicamente colocar os caracteres na posição
respectiva dentro da mensagem a enviar
                                        for (z = 0; z < 16; z++)
                                        {
                                            if (buff[z] != 'X') // Se o conteudo da posição buff[z] for
diferente de 'X' então envia mensagem. Caso contrário não envia nada
                                            {
                                                char msg[] = "@00WD060000000000"; //Mensagem Base que vai ser
modificada para acomodar as posições de 0601 a 0616 no PLC
                                                msg[12] = buff[z]; //Na posição 13 vai ficar o estado , 0 ou 1
consoante o valor contido no buff
                                                int n = strlen(msg);
                                                g = z + 1;

                                                if (g < 10)
                                                    msg[8] = '0' + g; // se o número da posição for de 0601 a
609 basta substituir a posição 9
                                                else
                                                { // se o número da posição for de 0610 a 616 será necessário
substituir a posição 8 e 9 para acomodar todos os 2 caracteres
                                                    msg[7] = toascii(49);
                                                    msg[8] = toascii(38 + g);
                                                }
                                                //*************Agora vai calcular o FCS para anexar à
mensagem************************

//*****************************************************************************
                                                FCS1 = FCS2 = 0;
                                                for (i = 0; i < n; i++)
                                                {
                                                    nib1 = (msg[i] >> 4) & 0x0F;
                                                    FCS1 = FCS1 ^ nib1;
                                                    nib2 = (msg[i]) & 0x0F;
                                                    FCS2 = FCS2 ^ nib2;
                                                }
                                                fprintf(stdout, " %d %d\n", FCS1, FCS2);

                                                //******Primeiro caracter do FCS**********
                                                if (FCS1 < 10)
                                                    msg[13] = (int)('0') + FCS1;
                                                else
                                                    msg[13] = toascii(55 + FCS1);

                                                //******Segundo caracter do FCS**********
                                                if (FCS2 < 10)
                                                    msg[14] = (int)('0') + FCS2;
                                                else
                                                    msg[14] = toascii(55 + FCS2);

                                                msg[15] = '*'; //Adiciona o caracter terminador
                                                msg[16] = toascii(13); //Adiciona o CR
                                                                       //fprintf(stdout,"msg antes  =%s ", msg
);
                                                                       //fprintf(stdout,"msg =%s\n", msg );
                                                                       //strncat(msg,FCS,4);

                                                serialPrintf(fd, msg);
                                                delay(90); //Tempo de espera entre cada mensagem de
actualização das posições dos alvéolos
                                                fprintf(stdout, " %s\n", msg);
                                            } //close: if (buff[z] != 'X')
                                        } //close: for (z = 0 ; z < 16 ;  z++)
```

```
                                    } //close: else do if (my_stream == NULL)
                                } //close: else do if (buff[1] != '1')
                            } //close: else do if (my_stream == NULL)
                        } //close: if ((DataIn[10])=='1')
                        else
                        {
                            delay(30);
                            printf("Mensagem OK - > No entanto agora nao se pode tirar fotografia\n");
                        }
                    } //close: else do if ((FCSCHR1!=DataIn[11]) || (FCSCHR2!=DataIn[12]))
                } //close: else do if (((DataIn[5]!='0') || (DataIn[6]!='0')) || (lng<13))
            } //close: else do if (msgcompleta==0)
        } //close: for (b=0 ;;b++)
    } //close: else do for (b=0 ;;b++)
    serialClose(fd);
} //close: for (a=0 ;;a++)
return 0;
} //close: int main()
```

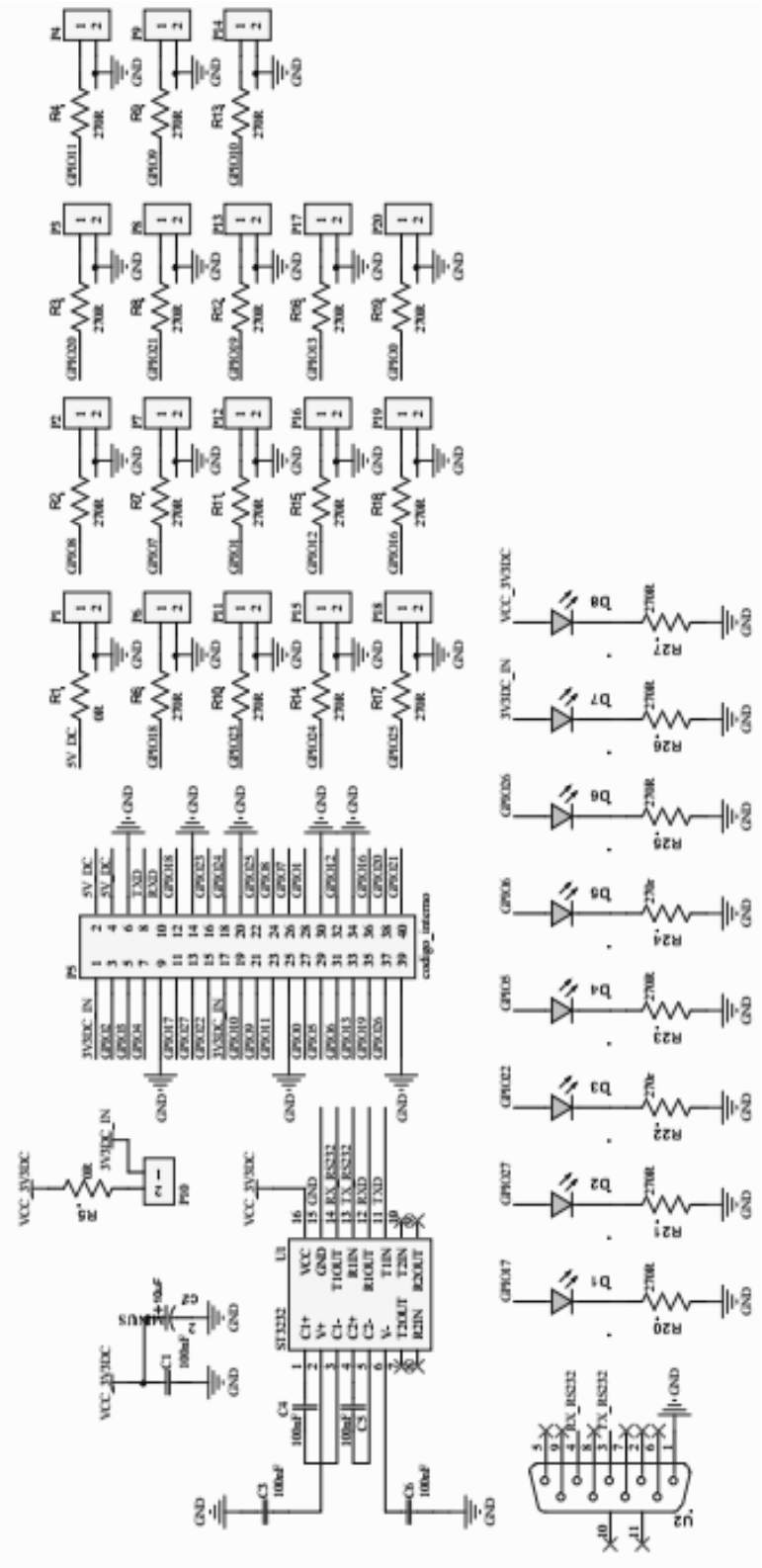
11. ANEXO D – Tabela ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

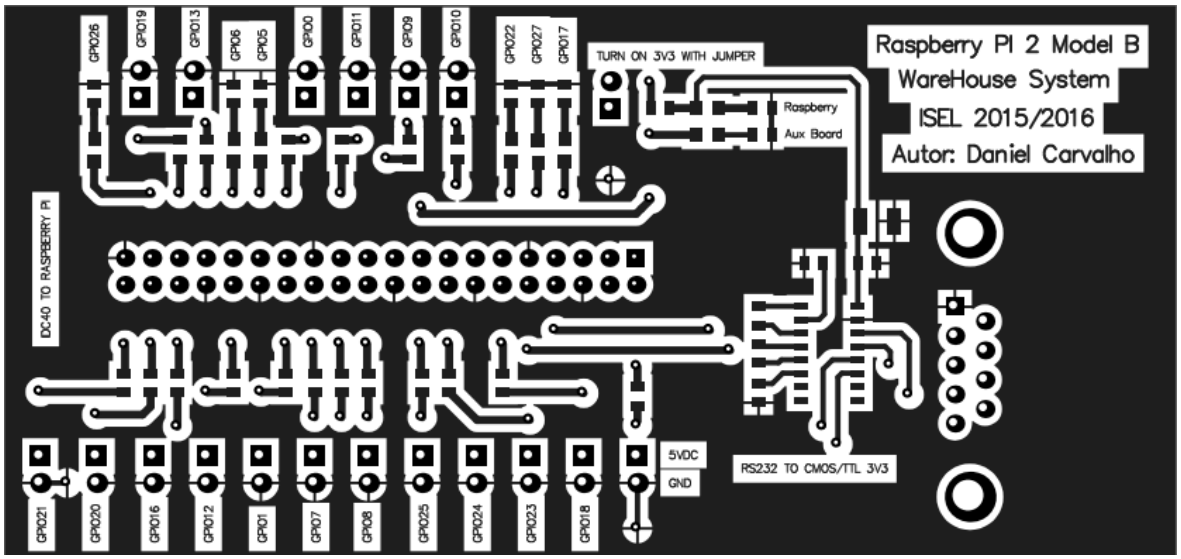
12. ANEXO E – Placa adaptadora para o *Raspberry Pi*

Esquema Elétrico

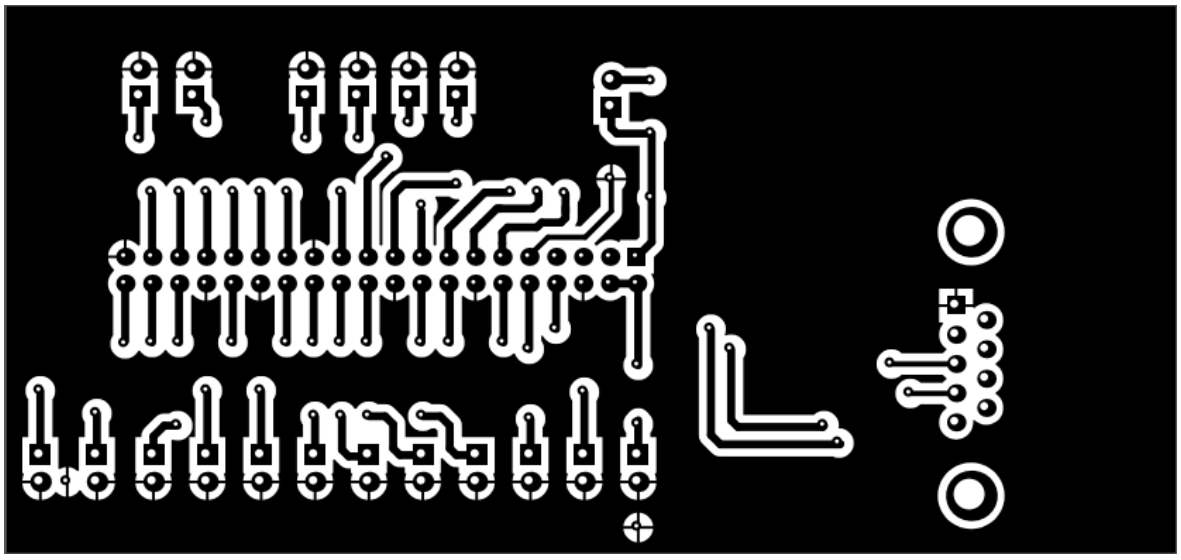


Desenho da PCB

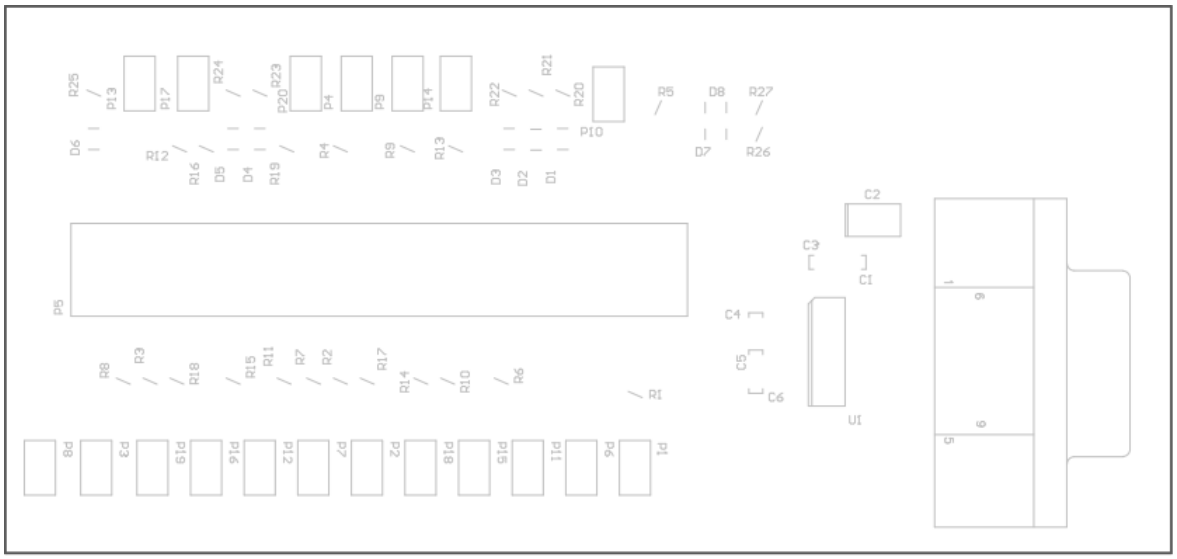
TOP



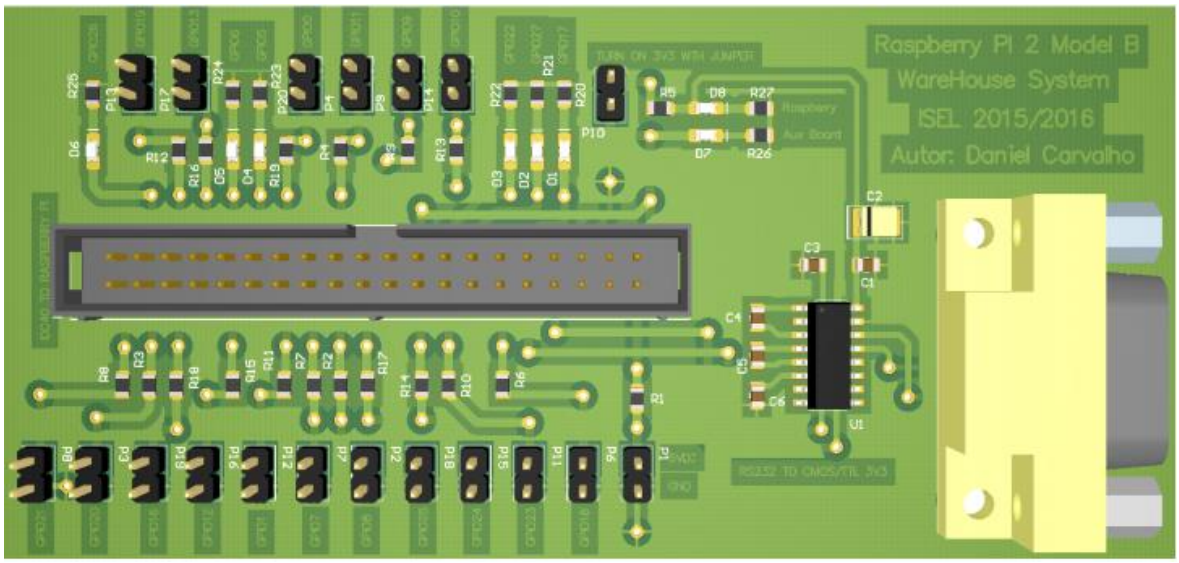
BOT



TOP SILK



PCB 3D



Lista de Material

Quantidade	Designação	Valor	Fornecedor
1	U2	Conector DB9 Female	Farnell
8	D1, D2, D3, D4, D5, D6, D7, D8	Led SMD 0805	Farnell
19	P1, P2, P3, P4, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20	*Header* 2 pinos	Farnell
1	P5	*Header* 40 pinos	Farnell
2	R1, R5	Resistência 0R *SMD 0805*	Farnell
25	R2, R3, R4, R6, R7, R8, R9, R10, R11, R12, R13, R4, R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27	Resistência 270R *SMD 0805*	Farnell
1	C2	Condensador tântalo 10uF	Farnell
5	C1, C3, C4, C5, C6	Condensador cerâmico 100nF	Farnell
1	U1	ST3232	Farnell