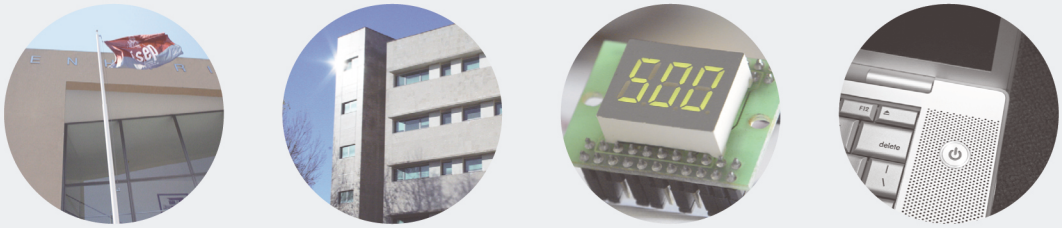
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Aplicação para programação automática de robôs de paletização

FREDERICO EMANUEL MATOS DA SILVA MOURA
outubro de 2017

POLITÉCNICO
DO PORTO

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Aplicação para programação automática de robôs de paletização

FREDERICO EMANUEL MATOS DA SILVA MOURA
Outubro de 2017

POLITÉCNICO
DO PORTO

APLICAÇÃO PARA PROGRAMAÇÃO AUTOMÁTICA DE ROBÔS DE PALETIZAÇÃO

Frederico Emanuel Matos da Silva Moura

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

2017

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha da Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Frederico Emanuel Matos da Silva Moura, nº 1070950, 1070950@isep.ipp.pt
Orientação científica: Manuel Fernando dos Santos Silva, mss@isep.ipp.pt

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

22 de outubro de 2017

À memória de meu tio Noémio.

Agradecimentos

Gostaria de agradecer ao meu orientador, Professor Manuel Fernando dos Santos Silva, pela orientação, apoio e pela enorme disponibilidade mesmo a alguns (largos) quilómetros de distância.

À minha família por toda a ajuda e por todas as oportunidades que me deram, permitindo-me crescer por mim mesmo. Não seria quem sou sem vós.

À minha mulher, Maria João, sem a qual este trabalho nem teria existido. Obrigado por todo o apoio, carinho e confiança incondicional ao longo de tanto tempo.

A todos o meu mais sincero obrigado.

Resumo

O desenvolvimento tecnológico, sentido ao longo dos tempos, tem permitido e até incentivado a que cada vez mais atividades sejam desempenhadas de forma mecânica. Esta tendência tem-se intensificado nos últimos tempos devido às novas exigências impostas pela globalização que não permite momentos de pausa e em que tudo está comunicável vinte e quatro horas por dia. Para responder ao paradigma atual – onde a competitividade de cada empresa e de cada produto constitui um requisito incontestável para a sua sobrevivência – a indústria tem vindo a optar por automatizar – com níveis de exigência cada vez maiores – cada vez mais processos (ou atividades) que, até então, eram realizados por mão-de-obra humana. É comummente aceite que à automatização de processos seja associado um vasto número de benefícios à produção, tais como a redução de custos, a rapidez de produção, produção programada e contínua, e a própria padronização do produto.

Entre as várias atividades que têm sido desenvolvidas de forma automatizada encontram-se as operações de manipulação. Estas são descritas como a movimentação de objetos de um lugar para o outro, sendo a paletização disso exemplo. Acompanhado as exigências atuais, é esperado que os equipamentos que executam o processo de paletização – de objetos com cada vez mais diferentes dimensões e diferentes materiais – sejam mais flexíveis, mais rápidos e mais precisos.

Neste cenário, a robótica assume-se como uma solução cada vez mais procurada, por ser frequentemente associada a conceitos como flexibilidade e precisão. Por sua vez, o seu *software* de programação deve garantir rapidez e eficiência e admitir várias opções.

Tendo em conta estas variáveis, neste trabalho, procurou-se contextualizar e desenvolver uma aplicação para robôs, no caso, do fabricante ABB. Esta aplicação, em conjunto com o *software* de programação offline da marca, permite a programação automática das funções de paletização de um robô. Desenvolvida na linguagem nativa do robô (RAPID), a aplicação detém uma interface de utilizador básica escrita em XML. E por ser em código aberto, permitirá a interação com outro *software* de modo a admitir outras funções.

Para a prossecução dos objetivos foram elaboradas diversas simulações do controlador IRC5 dispondo este do programa desenvolvido para o efeito. As simulações, efetuadas em ambiente RobotStudio, tiveram como protagonista principal um manipulador e correspondente ferramenta, típicos de processos de paletização. A etapa final consistiu em ensaios, utilizando o programa desenvolvido, num robô físico da ABB (modelo IRB140), disponibilizado em laboratório.

Palavras-Chave

Paletização, robótica, programação, ABB, RAPID, RobotStudio

Abstract

Felt throughout time, technological development has allowed and even encouraged increasingly more activities to be performed mechanically. This tendency has lately been amplified due to new requirements imposed by globalisation that does not allow moments of pause and in which everything is available twenty-four hours a day. To reply to the current paradigm – where each company's and product's competitiveness constitute an indisputable prerequisite to its survival – the industry has chosen to automate – with ever greater demands – increasing processes (or activities), which hitherto were carried out by human labour. It is commonly accepted process automation to be associated with a vast number of production advantages, such as cost reduction, fast production, schedule and continuous production, and product standardisation itself.

Amongst the several activities that have been developed in an automated manner are handling operations. These are described as the movement of objects from one place to another, of which palletisation is an example. Keeping up with current demands, it is expected that the equipment that executes palletising processes – of objects with increasingly different dimensions and materials – to be more flexible, faster and more precise.

In this scenario, robotics assumes itself as an increasingly sought-after solution, since it is often associated with concepts such as flexibility and precision. In turn, its programming software should ensure speed and efficiency, and allow several options.

Considering these variables, in this work it was sought to contextualise and develop an application for robots, in this case, having ABB as manufacturer. This application, together with the brand's off-line programming software, allows for automatic programming of a robot's palletising functions. Developed in the robot's native language (RAPID), the application has a basic user interface written in XML. And being open-source, it will allow for the interaction with another software to enable other functions.

In the pursuit of the objectives, several IRC5 controller simulations were made which had the developed program for this purpose. The simulations, carried out in a RobotStudio environment, had as main protagonist a manipulator and corresponding tool, both typical of palletising processes. The final step consisted of tests using the developed program, in a physical ABB robot (model IRB140), made available in the laboratory.

Keywords

Palletising, robotics, programming, ABB, RAPID, RobotStudio

Zusammenfassung

Die technologische Entwicklung, die im Laufe der Zeiten gefühlt wurde, hat erlaubt und sogar ermutigt, dass immer mehr Aktivitäten auf mechanischer Art durchgeführt seien. Diese Tendenz hat sich in den letzten Zeiten verstärkt aufgrund der neuen Anforderungen der Globalisierung, die keine Momente der Pause erlauben und in denen alles vierundzwanzig Stunden am Tag verfügbar werden. Um auf das derzeitige Paradigma zu antworten – wo die Wettbewerbsfähigkeit jedes Unternehmens und jedes Produktes eine unbestreitbare Voraussetzung ihres Überlebens ist – hat sich die Industrie für Automatisierung entschieden – mit steigenden Anforderungen – immer mehr Verfahren (oder Aktivitäten), die bis dahin von menschlicher Arbeit verwirklicht wurden. Es ist allgemein anerkannt, dass Prozessautomatisierung mit einer Vielzahl den Produktionsvorteilen assoziieren ist, wie Kostenreduzierung, Produktionsgeschwindigkeit, programmierte und kontinuierliche Produktion und selbst mit Produktstandardisierung.

Unter den verschiedenen Aktivitäten, die in einer automatisierten Weise entwickelt worden, befinden sich die Handhabungsvorgänge. Diese werden als die Bewegung von Objekten aus einer Stelle zu einer anderen beschrieben, wobei das Palettieren von denen ein Beispiel ist. Um den aktuellen Anforderungen gerecht zu werden, es wird erwartet, dass die Ausstattungen, die den Palettiervorgang ausführen – von Objekten mit zunehmend unterschiedlichen Dimensionen und Materialien – flexibler, schneller und präziser seien.

In diesem Szenario, nimmt die Robotik als eine zunehmend begehrte Lösung an, da sie oft mit Konzepten wie Flexibilität und Präzision verbunden ist. Im Gegenzug, sollte ihre Programmiersoftware die Schnelligkeit und Effizienz sicherstellen und mehrere Optionen zulassen.

Angesichts dieser Variablen, wurde in dieser Arbeit versucht, eine Anwendung für Roboter entwickeln und in einen Kontext einordnen, in diesem Fall mit ABB als Hersteller. Diese Anwendung, zusammen mit der Offline-Programmiersoftware der Marke, ermöglicht die automatische Programmierung den Palettiervorgang eines Roboters. Entwickelt auf der eigenen Sprache des Roboters (RAPID), die Anwendung hat eine grundlegende Benutzeroberfläche, die auf XML geschrieben wurde.

Und weil es Open-Source ist, wird es die Interaktion mit anderer Software ermöglichen, um andere Funktionen zuzulassen.

Bei der Verfolgung der Ziele wurden mehrere Simulationen des IRC5-Controllers durchgeführt, den zu diesem Zweck das entwickelte Programm hatte. Die Simulationen, die in einer RobotStudio-Umgebung durchgeführt wurden, hatten als Hauptprotagonist einen Manipulator und ein Werkzeug, die typisch für Palettiervorgänge sind. Der letzte Schritt bestand aus Tests, die das entwickelte Programm benutzten, in einem physischen ABB-Roboter (Modell IRB140), der im Labor zur Verfügung gestellt wurde.

Schlüsselwörter

Palettisieren, Robotik, Programmierung, ABB, RAPID, RobotStudio

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

ABB	–	ASEA Brown Boveri
API	–	Application Programming Interface
ARLA	–	ASEA programming Robot LAnguage
ARP	–	Automatic Robot Palletizer
ASEA	–	Allmänna Svenska Elektriska Aktiebolaget
BBC	–	Brown, Boveri & Cie.
CAD	–	Computer Aided Design
CPS	–	Cyber-Physical System
EPAL	–	European Pallet Association
IRC	–	Industrial Robot Controller
PLC	–	Programmable Logic Controller
ROS	–	Robot Operating System
TCP	–	Tool Centre Point
SCARA	–	Selective Compliance Assembly Robot Arm
UCS	–	User Coordinate System
XML	–	eXtensible Markup Language

1. INTRODUÇÃO

ro.bó.ti.ca | ʀɔˈbɔtikɐ, ʀuˈbɔtikɐ
nome feminino
"Conjunto de técnicas respeitantes ao funcionamento e utilização de autómatos (robôs) na execução de múltiplas tarefas em substituição do homem." [1]

Inerente à própria condição humana, a busca por métodos e mecanismos que facilitem tarefas penosas ou repetitivas, leva-nos, inevitavelmente, a automatizar. Indissociável deste conceito é o ramo da automação que temos, porventura por razões antropomórficas, como mais próximo: a robótica. Historicamente, a noção de um mecanismo programável capaz de imitar um humano existe desde a Grécia Antiga e, desde então, evoluiu chegando aos tempos modernos depois de, pelo caminho, capturar a atenção e imaginação de cientistas e de inventores. Atualmente, apesar das inúmeras aplicações possíveis para o uso diário de um robô na vida quotidiana, é na indústria que o seu emprego assume um maior impacto, chegando mesmo ao ponto da utilização de robôs em detrimento de trabalhadores humanos gerar controvérsia em diversos patamares. Segundo alguns artigos e estudos, é expectável que os robôs venham a ser responsáveis por cerca de 30% dos postos de trabalho, nos próximos dez anos [2].

De facto, as características intrínsecas de repetibilidade, flexibilidade e fiabilidade do robô, perante a crescente procura por parte da indústria por redução de custos e melhorias de produtividade, apresentam-se como uma gigantesca vantagem perante a mão-de-obra humana ou, até certo ponto, perante maquinaria dedicada para a automatização de diversos processos. Dada esta situação, não se revela surpreendente que o número de robôs em ambiente industrial tenha crescido de forma acentuada ao longo das últimas décadas e assim se preveja continuar. A International Federation of Robotics (IFR) estima que o nível de *stock* de robôs a nível mundial necessário para atender a possíveis requisitos operacionais atinja, aproximadamente, os dois milhões e meio de unidades em 2019 [3], conforme é possível constatar na Figura 1. Este valor, ainda que estimado, torna-se inteligível quando comparado com o seu equivalente no início do século que era de setecentas e cinquenta mil unidades.

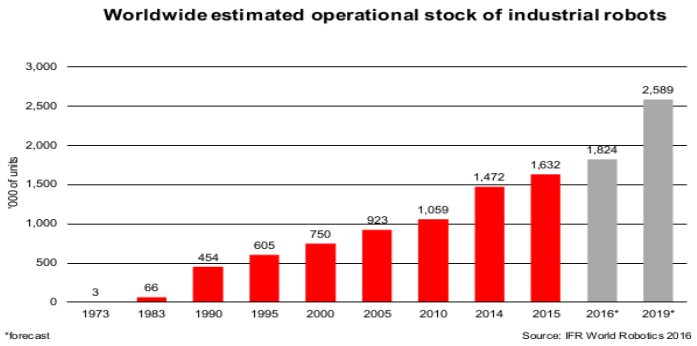


Figura 1 Estimativa de *stock* operacional mundial de robôs industriais [3]

Porém, de acordo com a tendência corrente da aproximação de uma quarta revolução industrial (Indústria 4.0) [4], a interação humana com os sistemas ciber-físicos (*cyber-physical system* - CPS) – nos quais se incluem os robôs – não pode ser descartada, sendo a cooperação inclusive encorajada, dada a impossibilidade da substituição do ser humano pela máquina em diversas tarefas complexas ou que requeiram tomada de decisões, como as em situações não contempladas. Ou seja, assume-se a direção da mudança dos perfis de competências dos trabalhadores, livrando-os de funções fastidiosas e alocando-os a funções com valor acrescentado e criativas, nas quais se enquadra inevitavelmente o *software*. Neste quadro geral, afigura-se um significativo aumento de ferramentas de *software* colocadas à disposição do utilizador ou programador. Acompanhando este aumento generalizado, prevê-se também que o desenvolvimento de aplicações para robôs assuma um papel cada vez mais importante, dado o crescente e elevado número de estas unidades no mercado e o seu variado leque de utilizações.

1.1. CONTEXTUALIZAÇÃO

O tema abordado surge da contínua procura – por parte da indústria – por soluções para reptos lançados por mercados em constante modificação e evolução. A persistente busca para reduzir custos, prazos e para rentabilizar recursos das empresas – num mercado cada vez mais global e competitivo –, ao mesmo tempo que se aposta na inovação ou aperfeiçoamento de bens, obriga a que a forma como os produtos são produzidos e transportados seja continuamente alterada e aperfeiçoada. Esta mudança reflete-se na alteração dos tempos de ciclo e quantidades de produção, formas e diversidades dos produtos, entre outros. Obriga, igualmente, a que a própria maquinaria e os sistemas usados para produção, embalamento e transporte de bens, também evoluam.

Neste cenário, a robótica assume-se, claramente, como um meio para tentar fazer frente a estes desafios, dado apresentar um elevado grau de adaptabilidade e robustez para a execução de diversas tarefas. Entre estas, incluem-se, por exemplo, a pintura, a soldadura ou as operações de manipulação. Estas últimas funções apresentam-se como predominantes no mercado Europeu, conforme é possível verificar na Figura 2, a qual exibe os valores estimados das unidades robóticas industriais instaladas na Europa (distribuídas segundo função) nos anos de 2011 a 2013 [5].

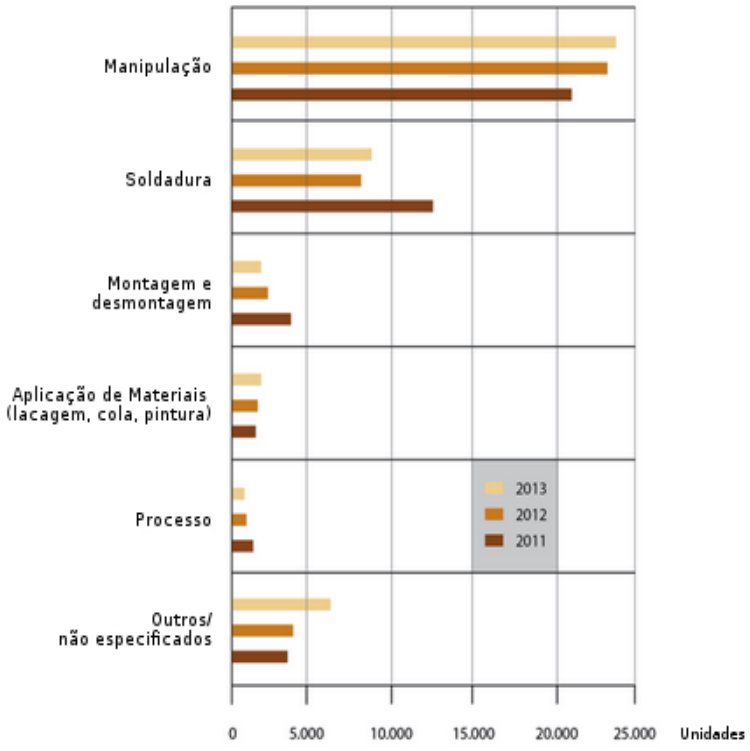


Figura 2 Estimativa de unidades robóticas industriais na Europa por tarefa (adaptado de [5]).

De forma geral, é expectável que esta distribuição se mantenha durante os próximos anos, conforme afirma o recente estudo "European Industrial Robotics Market 2016-2020" [6], com as operações de manipulação a liderarem e preservarem a maior quota no mercado europeu de robôs industriais. Quota essa que o estudo ainda prevê que cresça e atinja um valor superior a 47% em 2020.

As operações de manipulação efetuadas por robôs incluem, entre outras, embalamento, manipulação em salas limpas (*clean rooms*), *pick and place*, e paletização e despaletização. É possível, por isso, afirmar que as funções de manipulação se encontram presentes na quase totalidade da cadeia de logística interna, podendo ser recursos alocados à receção ou armazenamento de materiais, à produção e ao embalamento final para posterior expedição.

Centrando-se no embalamento final, mais precisamente na paletização, o presente trabalho pretende oferecer uma possível solução para os desafios atuais ou apresentar-se como um melhoramento dos métodos hoje em prática, fazendo uso das vantagens oferecidas pela robótica. Vantagens essas que poderão passar, por exemplo, pela redução dos tempos de paragem para reprogramação do *software* ou para reconfiguração/ajuste dos equipamentos (*setup*) no caso de mudança do objeto a paletizar.

1.2. OBJETIVOS

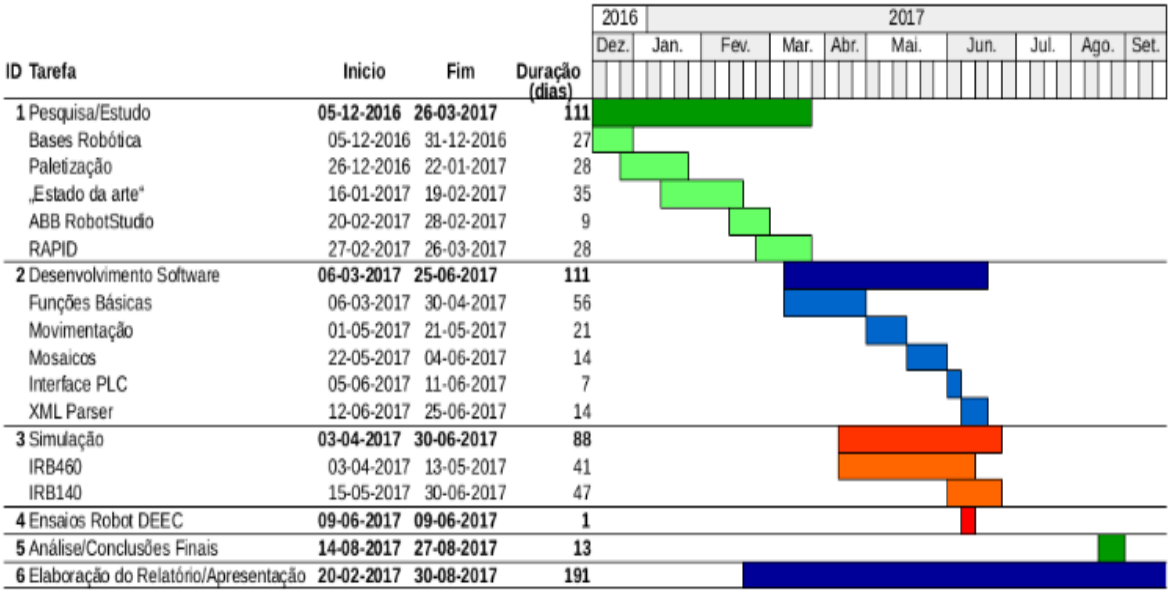
O objetivo principal desta Tese, como o próprio nome indica, é a criação de uma aplicação para a programação automática de robôs afetos a tarefas de paletização. Esta aplicação deverá correr em ambiente base do RobotStudio, fazendo uso das potencialidades que o programa oferece como aplicação para simulação de células robóticas e da virtualização do controlador.

O *software* deverá ser escrito na linguagem nativa da própria marca e ser transversal à gama de produtos da marca, ou seja, ser independente do sistema operativo em uso e do próprio manipulador.

Como meta adicional (não quantificável), deverá esta aplicação ser capaz de gerar a execução da paletização com o número mínimo possível de parâmetros introduzidos pelo utilizador, ou seja, deverá requer o mínimo de interação com o utilizador.

1.3. CALENDARIZAÇÃO

A Figura 3 apresenta a calendarização proposta para elaboração do trabalho aqui apresentado com as devidas etapas e tarefas.



ID	Tarefa	Inicio	Fim	Duração (dias)
1	Pesquisa/Estudo	05-12-2016	26-03-2017	111
	Bases Robótica	05-12-2016	31-12-2016	27
	Paletização	26-12-2016	22-01-2017	28
	„Estado da arte“	16-01-2017	19-02-2017	35
	ABB RobotStudio	20-02-2017	28-02-2017	9
	RAPID	27-02-2017	26-03-2017	28
2	Desenvolvimento Software	06-03-2017	25-06-2017	111
	Funções Básicas	06-03-2017	30-04-2017	56
	Movimentação	01-05-2017	21-05-2017	21
	Mosaicos	22-05-2017	04-06-2017	14
	Interface PLC	05-06-2017	11-06-2017	7
	XML Parser	12-06-2017	25-06-2017	14
3	Simulação	03-04-2017	30-06-2017	88
	IRB460	03-04-2017	13-05-2017	41
	IRB140	15-05-2017	30-06-2017	47
4	Ensaios Robot DEEC	09-06-2017	09-06-2017	1
5	Análise/Conclusões Finais	14-08-2017	27-08-2017	13
6	Elaboração do Relatório/Apresentação	20-02-2017	30-08-2017	191

Figura 3 Calendarização proposta do trabalho

1.4. ORGANIZAÇÃO DO RELATÓRIO

No primeiro capítulo, **Introdução**, é feita uma apresentação aos temas interpelados que pretende enquadrar a temática abordada quer no contexto global, quer no panorama industrial.

No seguinte, **Robótica e Paletização**, são aprofundados os fundamentos das bases e dos assuntos deste trabalho, sucedendo-lhe o capítulo **Programação de Robôs para Aplicações Robotizadas**, que exibe uma sucinta exposição das linguagens de programação e as soluções existentes atualmente, quer sejam comerciais ou não.

O quarto capítulo, **Desenvolvimento e Implementação**, esclarece o processo de desenvolvimento do trabalho, o objeto desta tese, e sua correspondente implementação nos variados ambientes de trabalho.

Segue-se o quinto capítulo, **Testes e Análise de Resultados**, cujo conteúdo engloba os ensaios em simulador e em laboratório, numa unidade física, bem como as ilações retiradas dos mesmos.

As principais conclusões da generalidade deste trabalho, assim como uma ponderação das dificuldades deparadas durante o desenvolvimento do mesmo, e possíveis futuros progressos, encontram-se no último capítulo, **Conclusões**.

2. ROBÓTICA E PALETIZAÇÃO

Os sistemas de paletização são, há vários anos, componentes basilares dos sistemas de logística, de elevada utilização por permitirem agilidade nas operações de carga e, com isto, contribuírem para a diminuição de custos nas empresas, por exemplo. Porém, tal como os demais constituintes das cadeias de logística interna, também estes sistemas têm sido alvo de pressões para o seu melhoramento, dada a contínua evolução e aumento de exigência dos mercados atuais [7]. Os maiores desafios contemporâneos que se colocam, de uma forma geral, incluem a necessidade de lidar com o sempre crescente aumento de quantidades de produção e simultaneamente com a redução e diversidade de lotes, com necessidade de redução dos tempos de ciclo, com a variedade de embalagens e formatos dos produtos, entre outros.

Para fazer frente a estes desafios, os diversos equipamentos necessitam de ser mais rápidos, precisos e de mais fácil reprogramação e reconfiguração para melhor se ajustarem ao formato do objeto, por exemplo. Em suma, a sofisticação ditada, principalmente, pelos desafios na produção, propagou-se ao longo de toda a cadeira, inclusive para os sistemas de paletização, tornando-os mais complexos a nível mecânico e em termos de controlo, mas ao mesmo tempo exigindo maior flexibilidade.

Não será por isto surpreendente que a robótica esteja atualmente a assumir um papel cada vez mais preponderante nos sistemas de paletização, dado dispor de características que respondem precisamente aos desafios apresentados. Apesar de ainda se encontrar em desvantagem em alguns pontos relativamente aos demais sistemas convencionais de paletização, os manipuladores industriais robotizados apresentam atualmente um conjunto de ferramentas específicas que os começam a dotar da flexibilidade necessária para contrapor tais pontos fracos.

Este capítulo pretende expor sucintamente os fundamentos do tema-base do trabalho, incluindo os elementos de base e os auxiliares a processos de paletização, as diferentes abordagens existentes aos mesmos processos e os executantes. Nestes últimos, inclui-se incontornavelmente a robótica industrial, figura de fundo da realização do trabalho, com foco nas operações de paletização.

2.1. ROBÓTICA NA INDÚSTRIA

O termo "robô" deriva da palavra "*robota*" (trabalho forçado) em língua checa e foi cunhado, pela primeira vez em 1920, pelo escritor checo Karel Capek, na peça "Rosumovi Univerzální Roboti" ("Os robôs universais de Rossum"), que a utiliza para descrever os humanoides que prestam serviços à humanidade e posteriormente se revoltam contra a mesma.

Contudo, as bases da robótica industrial contemporânea só foram assentes, em 1954, pelo americano George Devol, que regista, nesse mesmo ano, uma patente de um manipulador programável. Posteriormente, em 1956, Devol funda com outros sócios a firma Unimation que introduz, em 1960, o primeiro robô industrial, à altura com acionamento hidráulico. Este robô, denominado "Unimate", foi vendido no ano seguinte à General Motors (GM), prestando este o atendimento a uma fornalha, manuseando peças de fundição e, posteriormente, executando funções de soldadura das mesmas peças ao chassi de um veículo [8]. Desde então, os robôs têm vindo a melhorar drasticamente as suas características como, entre outras, a repetibilidade, velocidade, capacidade de carga, fruto do desenvolvimento dos acionamentos, materiais e métodos de manufatura dos seus fabricantes. Este contínuo desenvolvimento permitiu que os robôs adquirissem a flexibilidade para realizar diferentes funções e consequentemente fomentou a sua propagação em ambiente industrial.

A Organização Internacional de Normalização – ou em Inglês International Organization for Standardization (ISO) – define, na norma ISO 8373:2012, robô industrial como:

> "Manipulador multifuncional controlado automaticamente, reprogramável, programável em três ou mais eixos, que podem ser fixos num local ou móveis para utilização em aplicações de automação industrial."[1] [9]

Com base na sua definição, um robô industrial é um sistema dotado de lógica programável que consiste, basicamente, num manipulador que pode tomar diferentes formas e configurações, sendo mecanicamente desenhado para posicionar a sua parte terminal movível num determinado ponto dentro de espaço de trabalho delimitado. Parte terminal essa que, concebida para interagir com o ambiente de trabalho, poderá ser uma garra ou uma ferramenta (*end effector*). Enquanto sistema controlado automaticamente, cabe ao robô proceder a funções de verificação, autorregulação, execução de funções programadas pelo utilizador e interface com os periféricos que o rodeiam. Ou seja, para além do manipulador físico em si, é expectável que existam mais componentes que integrem a unidade robotizada.

Apesar dos robôs industriais poderem adotar várias formas, a base constituinte, em termos de estrutura dos componentes, em que consiste um robô industrial permanece geralmente a mesma, sendo composta por [10] :

- Controlador – unidade central de controlo do robô. Contém os modelos de *software* do manipulador e ambiente, assim como o algoritmo de controlo do robô, que usa, aliados à informação dos sensores e do programa do utilizador ("descrição da tarefa"), para comandar a unidade de potência;

- Unidade de potência[2] – seguindo os comandos fornecidos pelo controlador, fornece energia aos atuadores;

- Manipulador – estrutura mecânica constituída por diversos corpos rígidos (elos) ligados em cadeia, movimentados através dos atuadores que se encontram ligados aos mesmos através das respetivas transmissões.

[1] Traduzido do texto original em Inglês: *"Automatically controlled, reprogrammable, multipurpose manipulator, programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications."*.

[2] Vários autores e fabricantes consideram a unidade de potência como parte integrante do controlador. Neste caso optou-se pela unidade de potência individualizada para mais fácil compreensão do robô.

- Sensores – Dividem-se em dois tipos: internos e externos. Os sensores internos transmitem informação ao controlador sobre dados internos do manipulador (velocidade, posição, etc.). Os sensores externos fornecem *feedback* sobre o ambiente, como por exemplo, a deteção de objetos na ferramenta;

O diagrama de blocos presente na Figura 4 pretende ilustrar a forma como os elementos funcionais de um robô, acima descritos, se encontram interligados.

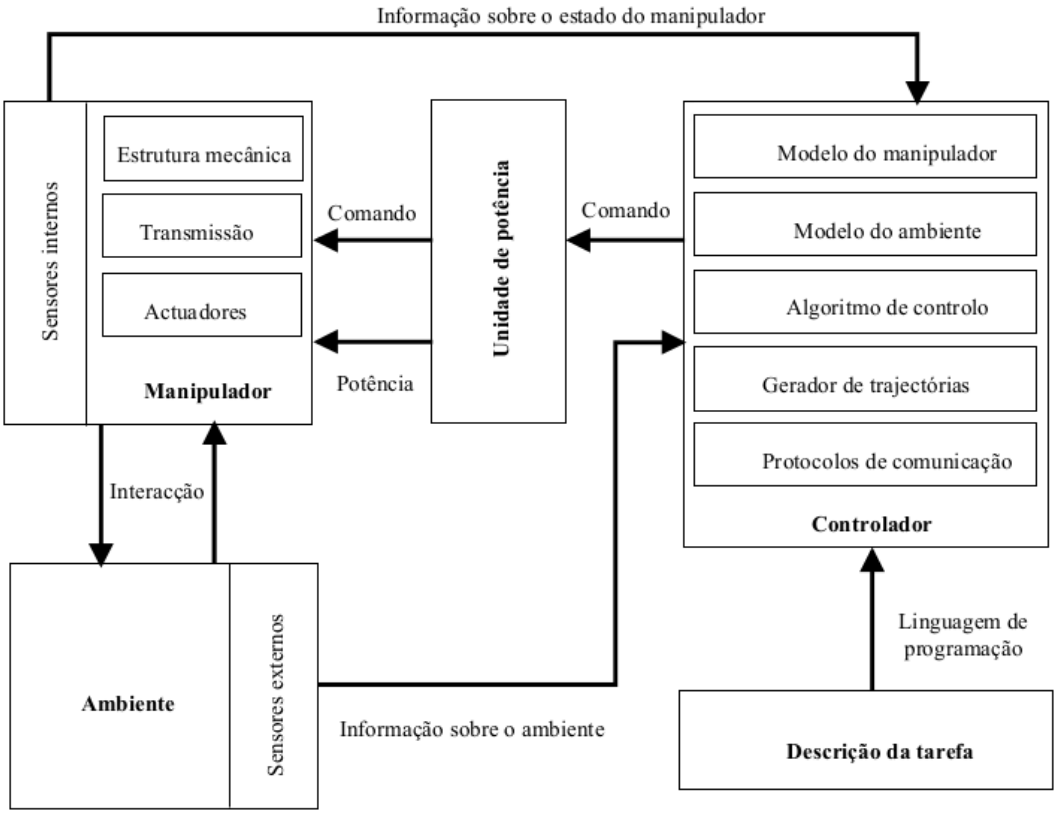


Figura 4 Representação esquemática da estrutura geral de um robô manipulador [10].

Os robôs industriais são, normalmente, classificados segundo a configuração do seu manipulador, sendo as configurações mais comuns a cartesiana, SCARA (*Selective Compliance Assembly Robot Arm*) e a articulada [5]. Existem ainda mais três configurações (cilíndrica, esférica e paralela) perfazendo um total de seis tipos de configurações que se encontram ilustradas na Figura 5. Ainda na mesma figura é possível ver uma representação da região (ou espaço) de trabalho de cada configuração, ou seja, o volume dentro do qual o robô consegue posicionar o atuador final (*end effector)*.

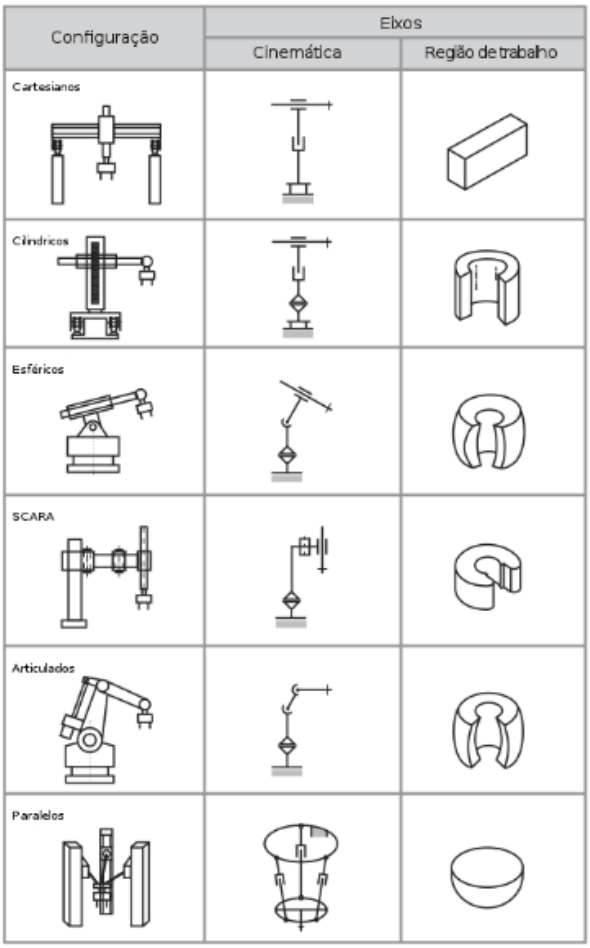


Figura 5 Classificação de robôs industriais (adaptado de [5]).

2.2. PALETIZAÇÃO

Enquadrando-se globalmente nas operações de manuseamento, mais concretamente no empacotamento, a paletização consiste fundamentalmente no ato de dispor (empilhar) camadas de bens sobre paletes com o objetivo de facilitar o transporte, normalmente de elevadas quantidades, dos mesmos.

Tradicionalmente manuais, os sistemas de paletização mais arcaicos, estão sobretudo dependentes de operadores. São estes que, sozinhos ou em grupo, agarram os objetos e os colocam, em pilha, numa palete. Estes sistemas têm como grande vantagem a flexibilidade do operador em poder lidar com vários tipos de objetos, com diferentes formas, tamanhos e até pesos. Por outro lado, estão sempre limitados às capacidades humanas. Por ser considerado um processo penoso e repetitivo para o trabalhador, os avanços da eletrónica e da robótica têm vindo a eliminar quase por completo a interação humana na paletização, dando desta forma lugar aos paletizadores automatizados.

De um ponto de vista genérico os paletizadores automatizados poderão ser divididos em duas grandes categorias: os sistemas convencionais e os robotizados. Estes últimos serão mais detalhadamente abordados no decorrer deste trabalho.

No caso dos convencionais, também é possível falar de paletizadores *high level* e *floor level* [11], cuja nomenclatura se prende com a forma como a máquina é alimentada, podendo ser, respetivamente carregada por cima e descarregada por baixo (*high level*) ou alimentada e descarregada ao nível do solo (*floor level*), conforme se pode verificar nas imagens da Figura 6.

Figura 6 Exemplos de paletizadores tipo *high level* (em cima) e *floor level* (em baixo) [11].

Apesar de ambos os tipos de paletizadores convencionais (*high* e *floor level*) terem uma estrutura diferente, as tarefas de base serão as mesmas. Estas tarefas encontram-se adjudicadas a módulos (componentes) que operam formando a pilha sequencialmente durante o processo de paletização, sendo possível, consoante o caso, acrescentar periféricos que executem tarefas adicionais, exemplo dos dispensadores de paletes ou folhas (*interlayers*).

Os componentes mais comuns num sistema de paletização convencional encontram-se ilustrados e numerados no exemplo da Figura 7 e são os seguintes:

- Alimentador (*feeder*)– equipamento que fornece ao sistema de paletização os objetos a paletizar (usualmente uma tela transportadora) (1);
- *Row former* – normalmente uma tela ou banda que orienta e acumula os objetos de forma a formar filas (2);
- Área de formação da camada (*layer forming area*) – área onde as diversas filas são agrupadas para formar uma camada (3);
- Estação da palete (*pallet station*) – área onde a palete é colocada para receber as camadas (7);
- Transportador de descarga (*discharge conveyor*) – transportador de descarga do paletizador das paletes já executadas (4);
- Dispensador de paletes (*pallet dispenser*) – aparelho que é carregado com uma carga de várias paletes e que alimenta o paletizador com as mesmas conforme a necessidade (5);
- Dispensador de cartões de camada (*sheet dispenser*) – mecanismo opcional que alimenta o paletizador com cartões para separação de camadas (8).

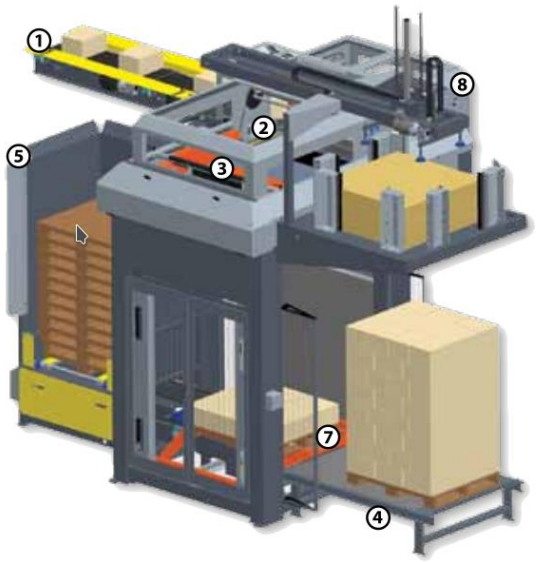


Figura 7 Exemplo de componentes mais comuns de um paletizador convencional *high level* [11].

As versões, ainda que diferentes, apresentam vantagens e desvantagens, inerentes à sua construção. Por exemplo, os paletizadores tipo *floor level*, são, por norma, mais baratos, dado serem de mais simples estrutura e de mais fácil acesso para manutenção. Por outro lado, os *high level* ocupam menos espaço no chão. Refere-se ainda o caso particular dos paletizadores robotizados que oferecem uma maior flexibilidade no momento do desenvolvimento e reprogramação do mosaico do que os seus homólogos convencionais.

Os diferentes tipos de paletizador apresentam também diferentes velocidades de paletização. De salientar, que o tamanho das embalagens e, consequentemente, o mosaico gerado para a palete terão influência sobre estes valores, independentemente do tipo de paletizador. Assim, a velocidade de paletização é medida em número de *cases* (embalagens), baseado num mosaico de camadas de 10 *cases*, independentemente do tipo de paletizador. Os paletizadores tipo *floor level* ou os paletizadores robóticos (*single pick*[3]) poderão atingir velocidade de 80 *cases per minute* (cpm), ou embalagens por minuto, enquanto que os paletizadores *high level* eventualmente conseguirão chegar a velocidades de 200 cpm [11].

2.2.1. ELEMENTOS BASE

O uso generalizado do uso de contentores marítimos, a partir da década de 1980, com vista a reduzir os custos de manuseio, transporte e danos nos bens, rapidamente os tornou num meio de predileção de acondicionamento para deslocação de bens à escala mundial e levou à normalização internacional de diversos equipamentos de transporte. Por sua vez, o facto de os contentores possuírem superfícies inferiores lisas e niveladas, tais como as encontradas nos transportes terrestres (camião ou comboio), que facilitavam a movimentação de objetos sob rodas no seu interior, favoreceu o uso de paletes.

Visto tratar-se de uma estrutura de baixo relevo, lisa, robusta, com capacidade de suportar cargas elevadas e passível de ser movimentada com diversos equipamentos (motorizados ou não), a palete afigurou-se um *standard* de logística. É, atualmente, usada nos mais diversos contextos. A Figura 8 apresenta dois exemplos de movimentação de paletes, à esquerda com recurso a um empilhador, à direita usando um porta-paletes manual.

[3] *Single pick* significa que a ferramenta utilizada pelo robô apenas lhe permite a movimentação de uma *case* de cada vez.

Figura 8 Movimentação de cargas de paletes empilhadas [12]

As paletes podem assumir diversas formas, tamanhos ou materiais consoante a finalidade para a qual foram previstas e de acordo com os mercados que servem. Porém, a tendência atual é para a uniformização das paletes, a nível mundial, considerando que, cada vez mais, o palco das trocas comerciais é um mercado sem fronteiras e global.

A obtenção das dimensões das paletes acarretou a ponderação de diversos parâmetros, dados os inúmeros fins para os quais as paletes podem ser utilizadas. Como expectável, os tamanhos dos contentores marítimos, vagões e camiões foram alguns dos fatores de ponderação, visto que a otimização do espaço disponível durante o transporte tem influência direta nos custos do mesmo. Em alguns casos, o simples facto das paletes poderem passar numa porta foi considerado como fator relevante para as dimensões. Também a sua construção, ou seja, número e dimensão das tábuas e pregos, e os materiais (cartão, madeira, plástico, metal) foram (e são) sujeitos a forte regulamentação, dado poderem ser, por exemplo, uma fonte de possível propagação de epidemias - caso de paletes em madeira.

Na Europa, a *European Pallet Association* (EPAL) assume-se como a entidade reguladora que supervisiona e certifica os fabricantes de paletes, neste caso apelidadas de Europaletes. O programa de troca de paletes colocado em funcionamento pela EPAL, em que na receção de mercadorias se encontram paletes vazias para troca com a palete carregada, permite reduzir o tempo de entrega da mercadoria visto não ser necessário proceder à despaletização da mesma. A EPAL define também os seguintes tamanhos e capacidades de carga para as cerca de 450 milhões de Europaletes em circulação [12], apresentados na Tabela 1.

Tabela 1 Tamanhos normalizados de Europaletes

Tipo de Europalete	Dimensões em milímetros ($C \times L \times A$)	Capacidade de carga (kg.)
EPAL, EPAL 1	800 × 1200 × 145	1500
EPAL 2	1200 × 1000 × 162	1250
EPAL 3	1000 × 1200 × 144	1500
EPAL 6[4]	800 × 600 × 144	1000
EPAL 7[4]	800 × 600 × 160	500

Apesar das diversas vantagens das paletes, existem situações em que o recurso às mesmas se pode tornar oneroso, como é o caso do transporte aéreo, onde qualquer tipo de peso extra é de evitar. Uma solução para tais ocasiões, a par das paletes, e que pode ser usada em conjunto com as mesmas, passa por empilhar os objetos numa folha de cartão (*slip sheet*). Em Inglês, esta operação chama-se *unitizing* e consiste, de grosso modo, no empilhamento sem recurso a uma palete. Neste caso, é frequente o uso de folhas de cartão ondulado, de cartão alveolar, de painel de fibras ou de plástico. Estas folhas são escolhidas – quanto material/tipo – de acordo com a finalidade a que se destinam e são empregadas durante o processo de construção da pilha. As folhas podem, ainda, ter outros usos – para além de serem alternativas a paletes –, tais como o arrefecimento dos objetos empilhados, o reforço estrutural da pilha (imagem esquerda da Figura 9) ou a proteção superior/inferior da pilha (imagem direita da mesma figura).

Figura 9 Dois exemplos de usos para placas separadoras [11]

Para assegurar que a pilha mantém a sua forma durante o transporte, esta é, normalmente, envolta em película extensível e/ou é cintada com fitas de polipropileno ou aço.

[4] Dadas as suas dimensões, os tipos EPAL 6 e 7 são considerados "meia palete".

2.2.2. EMBALAGENS

A evolução das embalagens de bens progrediu ao longo das passadas décadas, não somente por causa de regras impostas por organismos de regulação de atividade, mas também devido a uma mudança de mentalidade por parte dos intermediários e consumidores finais, verificada a nível global. Esta evolução traduziu-se no aparecimento de soluções mais ecológicas, de fácil transporte e manuseamento para toda a cadeia e, em determinados casos, de soluções que permitem transmitir informação ou fazer publicidade nas próprias embalagens.

O empacotamento (*packaging*) pode ser dividido em três grandes tipos: o primário, secundário e terciário, como apresentado na Figura 10. Mediante o casos e produtos, estes grandes tipos poderão estar, ou não, presentes no processo. Isto é, poderá não existir a necessidade de um tipo de empacotamento, dando como exemplo o embalamento de frutos de casca rígida ou bens não perecíveis que poderão dispensar o empacotamento primário.

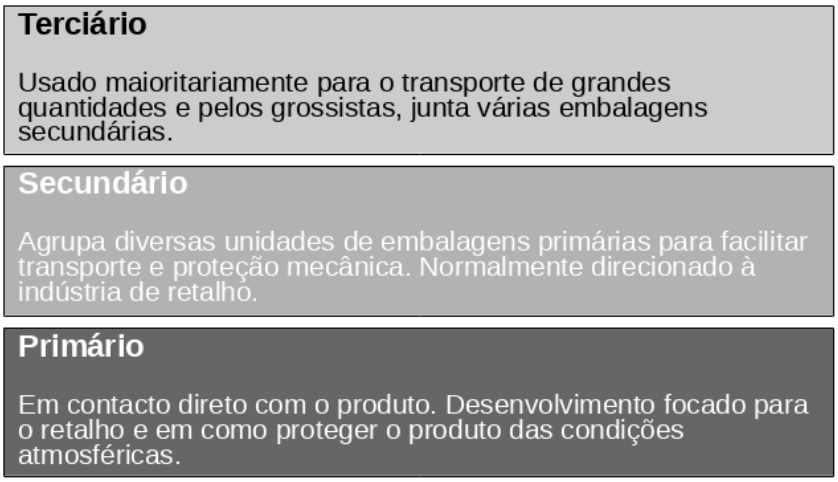


Figura 10 Tipos de empacotamento

Para a paletização, por questões práticas, será sempre preferível optar por um menor número de objetos, pois implicará um menor número de movimentos necessários por parte do paletizador e, consequentemente, uma utilização mais eficiente por parte do mesmo. Por este motivo, e pelo facto de se tratar de uma operação com vista ao transporte de quantidades elevadas, é expectável que a paletização se encontre num nível mais elevado (terciário).

Por consequência, o processo de paletização está dependente do tipo de empacotamento secundário, dado este determinar a configuração de como os produtos poderão ser acondicionados e dispostos nas paletes. Este fator e condicionante deverá, por isso, ser considerado no desenvolvimento de embalagens do tipo secundário.

Tradicionalmente, o empacotamento secundário era feito em grades, tabuleiros (Figura 11) ou caixas de cartão que apresentavam desvantagens consideráveis como serem relativamente pesadas (nos primeiros casos) ou (no caso posterior) absorverem água, se expostas à chuva. Apesar de, presentemente, ainda serem muito utilizadas, existem outras soluções que, em alguns determinados casos, se apresentam mais viáveis do que as convencionais caixas de cartão ou grades plásticas.

Figura 11 Exemplo de grades plásticas e tabuleiros [11]

Uma dessas soluções é o recurso a filme flexível (*bundle wrapping*), cujo exemplo mais conhecido, nos dias de hoje, são as embalagens de garrafas de bebidas. Neste caso, as unidades são envolvidas por película flexível que, não só funciona como método de aglomeração, como providencia uma forma mais fácil e prática de transporte ao consumidor. Dado a película poder ser impressa, serve ainda como meio de publicidade para o distribuidor/produtor. Outro exemplo com recurso a película flexível é o filme termorretráctil (*shrink wrapping*), que, como o próprio nome indica, consiste numa película que se molda aos produtos quando aquecida. Normalmente, este tipo de empacotamento é usado em conjunto com um tabuleiro inferior (em cartão) que oferece maior estabilidade mecânica e suporta itens com peso superior que a película normal não conseguiria suportar (Figura 12).

Figura 12 Exemplos de *bundle wrapping* (topo) e *shrink wrapping* (fundo) [11]

As próprias caixas de cartão convencionais, ainda que muito utilizadas e básicas, também sofreram alguns ajustes e originaram alternativas. Recentemente, devido às exigências de algumas cadeias de supermercados, que pretendiam agilizar a reposição de produtos nas prateleiras, surgiu uma simplificação à clássica caixa de cartão que é conhecida como embalagem de exposição (*display case*). Estas embalagens permitem, após a sua abertura, a visualização de todo o seu conteúdo, eliminando, desta forma, a necessidade de retirar todos os elementos da caixa para reposição nas prateleiras (Figura 13).

Figura 13 Exemplo de embalagens de exposição (*display cases*) [11]

Os exemplos apresentados anteriormente cingem-se a embalagens maioritariamente de forma paralelepipédica semirrígidas ou rígidas. Contudo, existe uma miríade de outros tipos de embalagens passíveis de poderem ser utilizadas nas operações de paletização. Por exemplo, sacos *open mouth*, sacos *stand-up*, sacos de fundo plano, sacos com reforço, latas, bidões, barris, entre outros. Para ilustrar outros tipos de embalagens também usadas na paletização, apresentam-se na Figura 14 dois exemplos de paletes já empilhadas com embalagens, uma com bidões de tinta e a outra com sacos *open mouth* de ração para animais.

Figura 14 Exemplos de paletes de bidões (esquerda) e sacos *open mouth* (direita) [11]

2.2.3. MOSAICOS

Existem diversas abordagens, baseadas maioritariamente em algoritmos, sobre como dispor os bens em paletes. No entanto, o objetivo final permanece o mesmo: encontrar o número máximo possível de produtos que sejam passíveis de carregar na palete sem exceder os seus limites geométricos ou a sua capacidade de carga. Um maior aproveitamento do espaço providenciado pela palete implicará uma redução de custos de armazenamento, transporte e distribuição dos produtos.

Segundo Hongtai *et. al.* [13], os estilos de empilhamento mais usuais são o sobreposto, ou em colunas (*overlap*), e o entrelaçado (*interlaced*). Os dois estilos são considerados os padrões clássicos para a criação de mosaicos e encontram-se representados na Figura 15.

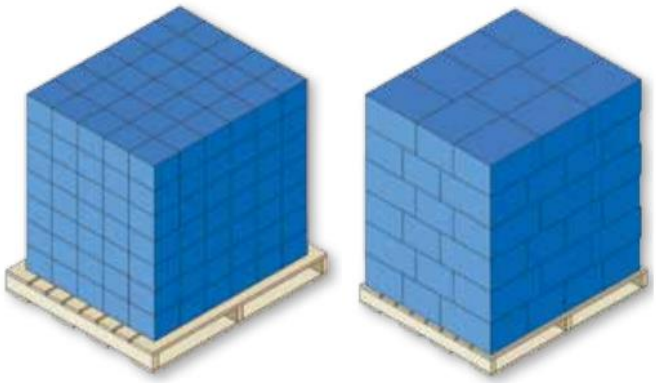


Figura 15 Exemplos de mosaicos em colunas (esquerda) e entrelaçado (direita) [11]

Ambos os casos apresentam vantagens e desvantagens mutuamente. Por exemplo, o padrão em colunas pode originar pilhas não tão estáveis quanto o entrelaçado. Por outro lado, o padrão entrelaçado pode resultar em mais espaço desperdiçado e reduzir significativamente a capacidade de carga, dependendo da embalagem do produto a paletizar.

Os dois exemplos apresentados na Figura 15 consideram que, em cada palete, todas as embalagens possuem o mesmo produto e as mesmas dimensões, caso que nem sempre é aplicável. Tomando como exemplo a paletização de embalagens que possuam as mesmas dimensões, mas cujos produtos são diferentes (ainda que o número de produtos diferentes seja reduzido e na mesma proporção), conforme é apresentado na Figura 16 (lado esquerdo), verifica-se que o mosaico poderá não sofrer alterações. Este caso, apelidado de *display pallets*, permite a paletização de forma muito semelhante ao caso ideal, sendo os produtos divididos em colunas. Caso o número de produtos diferentes seja mais elevado, mas as dimensões permaneçam as mesmas, já poderá ocorrer o caso apresentado ao centro da Figura 16. As paletes, em casos como este último, são conhecidas como *mixed layer pallets*, pois as diferentes camadas que formam a pilha apresentarão um ou mais tipos de produtos. Haverá ainda situações em que as paletes contêm diferentes produtos, de diferentes dimensões originando mosaicos do tipo apresentado na Figura 16 (direita). Estas são as *true mixed pallets*.

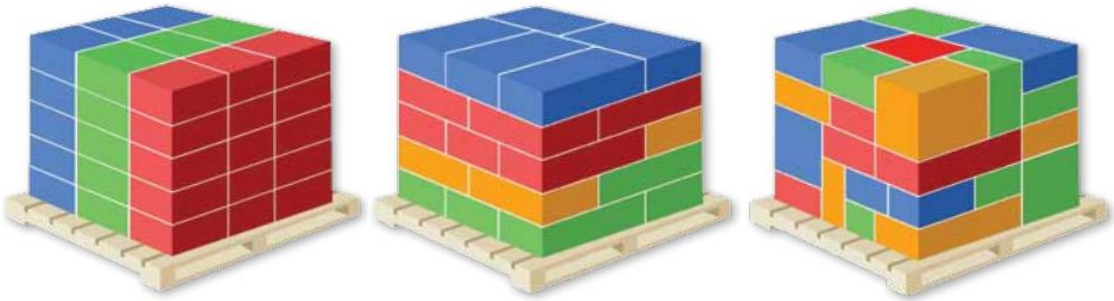


Figura 16 Exemplos de *display*, *mixed layer* e *true mixed pallets* [11]

Os três diferentes tipos de pilhas são a resposta a diferentes exigências de intervenientes distintos na cadeia de fornecimento. Será compreensível que um retalhista necessite de quantidades inferiores, mas uma variedade superior de produtos, quando comparado com um grossista que, pela própria atividade, deverá necessitar de quantidades superiores do mesmo produto.

2.3. PALETIZAÇÃO ROBOTIZADA

Os paletizadores robotizados exibem claras vantagens quando comparados com os paletizadores convencionais, dado um robô poder lidar, simultaneamente, com mais do que uma linha e/ou palete e com diversos tipos embalagens. Além destas, apresentam ainda as vantagens de serem reestruturáveis, permitindo desta forma um largo leque de disposições, das entradas e saídas de produtos ou distribuição dos acessórios, otimizadas para a tarefa em questão.

Contra os paletizadores convencionais pesam, também, os elevados tempos de *setup* (preparação) necessários para as suas reconfigurações em caso de mudança de tamanho ou desenho das embalagens ou de alteração do mosaico de paletização. Para ambos os tipos de paletizadores (convencionais ou robóticos) a possibilidade da reconfiguração, recorrendo a uma aplicação de *software* (que inclusive poderá ter a configuração em memória num computador), surge como uma forma mais expedita de fazer o *setup* sem a necessidade de recorrer a trabalhadores especializados.

Isto leva a um aumento de produtividade e redução de custos, quer com a não contratação dos referidos trabalhadores especializados quer com os transportes e logística. Estes últimos são conseguidos com a otimização do mosaico das paletes, otimização essa que é possível obter recorrendo às aplicações de *software* para fácil e rapidamente alterar o mosaico e o adaptar conforme as necessidades.

Por outro lado, o paletizador robotizado apresenta como principal desvantagem a velocidade com que faz a paletização, e que é manifestamente inferior à de um paletizador convencional *high level*, por exemplo. As desvantagens destes paletizadores (robotizados), contudo, têm vindo a ser, ao longo dos últimos anos, combatidas devido à proliferação dos mesmos, aos avanços nas tecnologias de materiais, eletrónica e acionamentos desenvolvidos para a robótica, fruto do investimento feito por parte dos fabricantes. Acresce que já existem diversas soluções mais imediatas para a diminuição do impacto das desvantagens e que passam pelo desenvolvimento de ferramentas e soluções mais adequadas e específicas (para a tarefa em questão) do que as ferramentas e soluções genéricas. Ilustrado na Figura 17, encontra-se um exemplo de uma solução, onde se podem ver robôs da firma KUKA a paletizar caixas de pão com ferramentas específicas, que manipulam simultaneamente mais do que uma caixa (no total são quatro), aumentado desta forma a velocidade de paletização através da redução do número de movimentos.

Figura 17 Robôs KUKA a paletizar caixas de pão [14]

Dada a sua flexibilidade, a área de influência ou trabalho do paletizador robotizado é, por norma, diferente da do convencional. O paletizador robotizado, normalmente, opera num espaço apelidado de célula que, para além do robô e dos periféricos que auxiliam o processo no qual este se inclui, poderá conter, por exemplo, equipamento de segurança para os operadores humanos. Para a paletização robotizada, a célula deverá incluir a presença dos componentes apresentados no ponto 2.2 deste trabalho, com a exceção, eventualmente, do *row former* e da área para formação da camada. Estes elementos poderão ser dispensáveis pois, dada a própria estrutura do processo de paletização com um robô, o mesmo poderá ser responsável pela formação da fila e da camada na palete. Em alguns casos, poderão robôs executar a formação da camada para, posteriormente, outros robôs procederem ao empilhamento.

Segundo Günthner e Lammer [8] os robôs mais comuns para aplicações de paletização e despaletização são os cartesianos e os articulados. A conhecida versatilidade dos robôs articulados destaca-os dos demais, porém as possíveis aplicações e desempenho deste tipo de robôs encontram-se muito dependentes do órgão terminal (ferramenta) e da tecnologia dos sensores.

2.3.1. TIPOS DE FERRAMENTAS

A enorme variedade de ferramentas que um robô poderá utilizar contribui, de grande forma, para a sua versatilidade. No caso da paletização, as ferramentas (*end effectors*) poderão, conforme a aplicação, ser desenvolvidas para manipular um ou mais produtos, individualmente ou em simultâneo.

Mudando a ferramenta, poderá um paletizador robotizado encarregue da paletização, por exemplo, de caixas de cartão, passar a empilhar sacos, bidões ou outro tipo de embalagens. Algumas ferramentas possuem também utensílios para manusear materiais auxiliares como folhas separadoras ou as próprias paletes.

As ferramentas mais comuns para paletização são do tipo *clamp*, forquilha (ou *fork*), garra (*claw* ou *finger*) e de vácuo [11]. Porém, outras existirão e, apesar da tendência ser para a uniformização das ferramentas existentes no mercado, é de esperar que continuem a surgir possíveis ferramentas customizadas com vista a uma determinada aplicação. Estas surgem necessariamente quando o bem a paletizar não se enquadra em nenhuma das ferramentas disponíveis comercialmente, ou como forma de resolver um problema inerente ao uso das mesmas.

A ferramenta tipo *clamp* (grampo) é, como o próprio nome indica, uma pinça de dois (ou mais) painéis laterais os quais, a comando, se aproximam para prender o objeto. Na Figura 18 podem-se ver dois exemplos deste tipo de ferramenta. A pinça poderá manipular um ou mais objetos e poderá ainda possuir (como é o caso da ferramenta do lado esquerdo), uma lateral com dedos para suportar o objeto por baixo, precavendo a eventualidade de não existir atrito suficiente entre o objeto e a ferramenta, o que levaria ao deslizamento do mesmo. Normalmente, o acionamento que movimenta os painéis é pneumático, com um curso largo que permite ajustar a ferramenta a diversos tamanhos de objetos. É considerada uma ferramenta de uso geral, podendo movimentar diversos tipos de embalagens, apresentando, porém, as desvantagens de comprimir o produto e, dada a espessura dos painéis laterais, deixar brechas entre as embalagens durante a paletização.

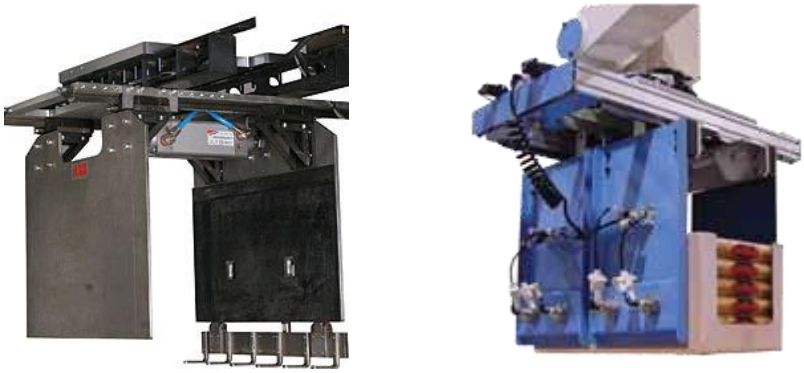


Figura 18 Exemplo de ferramenta tipo *clamp* simples (esquerda) [15] e dupla (direita) [11]

Consideradas ferramentas também de uso geral, as ferramentas do tipo *fork* são utilizadas para a movimentação de embalagens rígidas, semirrígidas e de embalagens não rígidas (flexíveis) que necessitem de total suporte na parte inferior. Consistem em dois painéis centrais ajustáveis que posicionam o objeto para posteriormente este ser fixo pelos dedos movíveis que deslizam por baixo e em cima por uma prensa. O acionamento é, normalmente, pneumático e, tal como as ferramentas *clamp,* ajustam-se a objetos de várias dimensões. Os dedos poderão ter funções secundárias, como por exemplo, movimentar paletes. Pela sua construção, esta ferramenta apenas é adequada para algumas situações específicas, em particular, para elevar objetos que se encontrem em transportadores de rolos ou equipamentos semelhantes.

Figura 19 Exemplo de ferramenta tipo *fork* [11]

Também apelidada por alguns fabricantes como tipo *finger*, as ferramentas tipo *claw* possuem longos dedos, por norma, alinhados em duas filas, e curvados. Para pegar no objeto, estas filas de dedos abrem-se (um pouco à semelhança dos dedos da mão antes de agarrar uma peça) ou afastam-se. Estando o objeto pronto a ser levantado, a ferramenta é posteriormente fechada, suportando-o pelo fundo. Dado o seu baixo perfil é, frequentemente, utilizada para manuseamento de sacos e, como se pode constatar na Figura 20, também poderá ter acionamento pneumático. Dada a forma como pega nos objetos (dedos entrelaçados), necessariamente ser-lhe-á apenas possível recolher embalagens que se encontrem em transportadores de rolos (ou equivalentes). Apresenta, notoriamente, o risco de, se não colocada em posição correta, comprimir o objeto com os dedos, correndo o risco de o perfurar.

Figura 20 Exemplo de ferramenta tipo *claw* [15]

As ferramentas de vácuo consistem num conjunto de ventosas dispostas em linha (ou em linhas paralelas) num único plano. Recorrendo ao vácuo durante o contacto com o objeto, pegam neste pela sua parte superior. Para a descarga do produto, o vácuo é desligado ou é injetado ar na direção oposta. O tipo de ventosas é muitas vezes adequado ao tipo de embalagem que o robô manipulará, variando entre *standard* (de baixo relevo), para superfícies planas, e com fole, para superfícies esféricas. Devido à sua própria construção, são comuns as ferramentas de vácuo que possuem equipamento auxiliar para deteção e movimentação de paletes. Apresentam a desvantagem, nos casos em que a embalagem seja porosa ou irregular, de poderem não produzir eficazmente vácuo e, como consequência, não conseguirem suportar o objeto corretamente.

Figura 21 Exemplo de uma ferramenta de vácuo [16]

Apresentadas algumas ferramentas, bem como alguns dos componentes básicos para o processo de paletização abordados ao longo deste capítulo, justifica-se sublinhar a, por vezes, curta longevidade dos mesmos. Isto deve-se (em grande parte) às atuais tendências e modos de vida que implicam constantes modificações do produto e embalagens. Não será, desta forma, surpreendente que os equipamentos criados para operações de paletização se vejam, muitas vezes, alvo de modificações e aperfeiçoamentos. Além disso, para sistemas mais flexíveis ou de mais fácil reconfiguração, mecanicamente, é expectável que a parte de controlo (ou seja, o *software*) venha a aumentar o seu grau de complexidade e importância. Este tema será abordado no próximo capítulo.

3. PROGRAMAÇÃO DE ROBÔS PARA APLICAÇÕES ROBOTIZADAS

Na revisão dos constituintes de um robô manipulador, na secção 2.1, estabeleceu-se que estes podem ser divididos em quatro componentes, sendo que apenas o controlador carece de *software* interno para operar.

O controlador de um robô inclui o seu sistema operativo, o qual executa as diversas funções de controlo do robô, entre elas o modelo cinemático do manipulador, o algoritmo de controlo e a compilação/interpretação do código fornecido pelo utilizador e que representa a forma como o robô é suposto atuar. Tal como os restantes componentes, o controlador tem sofrido melhoramentos, por causa dos avanços tecnológicos na eletrónica e, como consequência, progressos no *software*. Uma dessas modificações acarretou a possibilidade de programar/configurar robôs, sem ser necessário o robô estar fisicamente disponível (é a apelidada programação *off-line*). Até esse ponto, para programar um robô, era necessário parar o mesmo para a reprogramação e movimentá-lo através de um painel de comandos (programação *on-line*).

A necessidade de executar tarefas mais complexas levou à criação de linguagens de programação específicas. E, mais recentemente, a possibilidade de simular, em ambiente virtual, não só o controlador, mas também os periféricos, acessórios e demais componentes da célula robotizada. Proporcionou também aos utilizadores a hipótese de testarem os seus programas de forma mais fidedigna, tendo em conta um número superior de variáveis. O recurso a estas aplicações informáticas reduziu, também, o nível de especialização necessário por parte do programador e tornou a programação *offline* mais amigável ao utilizador (*user friendly).*

Este capítulo oferece uma descrição sucinta dos tipos de *software* necessários para controlar e programar um robô, assim como, faz a análise à pesquisa efetuada de *software* de simulação e de *software* específico para aplicações de paletização.

3.1. LINGUAGENS DE PROGRAMAÇÃO

De acordo com a definição já acima descrita, a programação *on-line* consiste na movimentação de um robô industrial, de forma manual, recorrendo a uma consola portátil (*teach-pendant*) ou por *manual leadthrough.* O recurso a um *teach-pendant* permite ao utilizador movimentar o robô (de forma assistida) usando as teclas ou *joystick* para o efeito, enquanto que o *manual leadthrough* implica a movimentação física do manipulador por parte do utilizador, tendo este contacto físico com o robô.

A abordagem clássica para a programação de um robô é a sua movimentação feita ponto a ponto. Quando em posição pretendida, são guardados os valores dos eixos do robô e o estado da ferramenta. Isto traduz-se numa sequência de posições do robô e de ações da ferramenta que o robô alcança e repete. Este método, apelidado de *teaching by showing* [17], ainda hoje se revela suficiente para programas que não sejam suscetíveis de alterações frequentes ou para pequenas aplicações repetitivas que não requeiram lógica complexa, como soldadura por pontos, pintura e manipulação (básica). Neste caso, o uso do termo “linguagem” para apelidar esta prática torna-se de difícil emprego, dado se tratar de uma sequência de valores e não de um código estruturado.

As primeiras linguagens dedicadas especificamente para a programação de robôs surgem da necessidade de evolução das aplicações (para, por exemplo, abrangerem operações matemáticas complexas) e, principalmente, da necessidade de se utilizar a informação de sensores durante o programa.

Na Tabela 2 são apresentadas algumas linguagens de programação de robôs industriais. A negrito destacam-se as linguagens que se encontram presentemente em uso. Estas poderão ter sido desenvolvidas de raiz, ser variantes ou alvo de revisões e atualizações.

Tabela 2 Linguagens de programação de robôs, baseado em [17]

Nome	**Desenvolvedor**	**Observações**
AL	Stanford University	desenvolvido em 1974, baseado em WAVE; semelhante a ALGOL[5]
AML	IBM	desenvolvido em 1977, "A Manufacturing Language"
ARLA	ASEA/ABB	de 1981 a 1992, "ASEA programming ROBOT LAnguage"
AS	Kawasaki	-
Inform III	Motoman (Yaskawa)	-
KAREL	Fanuc	desenvolvida em 1981, semelhante a Pascal
KRL	KUKA	semelhante a Pascal, "KUKA Robot Language"
MHI	MIT	desenvolvido em 1960, "Mechanical Hand Interpreter"
PACScript	Denso	baseado em SLIM, "Programmable Automation Controller"
PDL2	Comau	semelhante a Pascal
RAPID	ABB	desde de 1992, sucessora de ARLA
SLIM	Nachi	"Standard Language for Industrial Manipulator"
URScript	Universal Robots (UR)	linguagem de *script*
V+	Adept	baseado em VAL
VAL	Unimation	desenvolvido em 1975, baseado em WAVE, "Variable Assembly Language"
VAL3	Stäubli	baseado em VAL
WAVE	Stanford University	desenvolvido em 1970

[5] ALGOL é a abreviatura de "ALGOrithmic Language" e trata-se de uma família de linguagens de programação de alto-nível, desenvolvida nos meados da década de 50, e usada maioritariamente para operações matemáticas e cientificas [29].

As linguagens iniciais para a programação de robôs são, quase na totalidade, da autoria dos fabricantes destes e, dada a natureza da construção dos mesmos, proprietárias. Como se pôde ver na tabela anterior, apesar da curta duração de vida de algumas, estas formaram a base das linguagens modernas de programação, chegando até aos nossos dias.

Designada como a primeira linguagem de programação para robôs [17], e desenvolvida pelo Massachusetts Institute of Technology (MIT) na década de 1960, a Mechanincal Hand Interpreter (MHI) obteve o seu nome da mão mecânica MH-1 criada no mesmo instituto.

Já na década seguinte viu-se um aumento significativo no número de linguagens desenvolvidas com destaque para a WAVE, cujas bases foram lançadas pela Universidade de Stanford. Trata-se de uma linguagem de baixo nível, na qual diversas outras linguagens se basearam, como por exemplo, a Variable Assembly Language (VAL), ou a AL. Estas por sua vez, serviram de base para linguagens mais recentes como V+ ou VAL3. Ainda nas três décadas de 1970, 80 e 90 surgiram linguagens mais distantes do código máquina, algumas baseadas em Pascal – como PDL (atualmente na sua segunda versão, PDL2), KAREL ou a KUKA Robot Language (KRL) – outras, com fundações distintas das demais contemporâneas – como RAPID ou a Standard Language for Industrial Machines (SLIM).

As diversas linguagens de programação mais recentes especificam diferentes conjuntos de instruções e são, em termos de sintaxe, por vezes, bastante díspares. Contudo, as suas construções apresentam pontos comuns, tendo estas linguagens sido desenhadas para criar estruturas de dados e algoritmos, recorrendo estes a funções e subfunções, centrados nos movimentos do robô. Dependendo do tipo, as linguagens poderão ser executadas sem serem convertidas, neste caso, recorrendo a um programa chamado interpretador, ou poderão ser compiladas para linguagem máquina e posteriormente executadas por um processador. Esta é, respetivamente, a diferença substancial entre linguagens de *script* e as demais.

Quanto à evolução, existem vários pontos de vista sobre que curso a programação de robôs industriais deverá tomar. Em alguns casos, os fabricantes optam por oferecer interfaces para linguagens de alto-nível mais comuns e conhecidas, como por exemplo C++ ou java (caso da KUKA). Esta abordagem abre novas possibilidades de complexidade à programação. Isto, dado se tratarem de linguagens muito divulgadas, flexíveis e presentes em diversos tipos de plataformas, para diversos fins.

Uma outra abordagem, conhecida como programação *task level***,** intenta a substituição do conjunto necessário de movimentos por objetivos para o robô atingir um fim [17]. Ou seja, a programação *task level* (a um nível de tarefa, como indica o nome) pretende tornar a programação independente do robô: sem posições ou percursos (*paths*) que dependam da geometria e cinemática do manipulador, sendo estas variáveis substituídas por um conjunto de instruções empíricas para um determinado intento. Isto proporciona a possibilidade de criar *software* de complexidade notavelmente superior (em termos de lógica), dada a abstração das tarefas de controlo da movimentação do robô.

3.2. AMBIENTES DE PROGRAMAÇÃO *OFFLINE* E SIMULAÇÃO

A interface de programação de um qualquer robô industrial tenderá a ser feita num computador, relegando as consolas táteis para operações básicas de acesso rápido, necessárias durante o arranque do robô ou ensaios. Atualmente, um comum ambiente de programação *offline* é tipicamente uma aplicação informática, que pode adotar diversas aparências e ter uma variedade de diferentes funcionalidades, consoante o fabricante e as marcas de robôs que suporta. O código é construído nas aplicações usando o método mais conveniente ao programador, simulado e posteriormente descarregado para o robô. Os objetivos são sempre claros: com melhor qualidade do código e recorrendo a simulações com uma maior exatidão pretende-se aumentar a eficiência e facilidade da programação assim como a dos correspondentes arranques, consequentemente reduzindo o *downtime* e, no global, os custos associados ao processo.

Partindo da base de programação *offline,* os programas oferecem interfaces para a linguagem de programação e correspondentes ferramentas necessárias para correção ou ajuda à escrita (corretor automático, *snippets,* verificação da sintaxe, entre outras). O código poderá incluir, os pontos que o robô atingirá, as velocidades de movimentação, as tarefas da ferramenta, a interação com sensores internos e externos ou de visão, os periféricos e a interface com o controlador da célula, entre outros. A introdução de alguns destes parâmetros poderá também ser feita através da interface gráfica.

Os ambientes gráficos dos programas servem várias funções, simultaneamente. O objetivo principal do recurso a computação tridimensional é flexibilização e simplificação da interface com o utilizador.

Enquanto função de apoio à programação, a visualização gráfica permite ao utilizador ter uma maior perceção do que poderão significar os valores ou comandos que introduz em linhas de código. Um exemplo, visível na Figura 22, são as posições e orientações de *targets* e sistemas de coordenadas, assim como os respetivos valores. A mesma figura apresenta um exemplo deste tipo de ambiente, onde é possível ver uma peça importada – modelo *Computer Aided Design* (CAD) –, numa simulação de uma operação de soldadura por um robô.

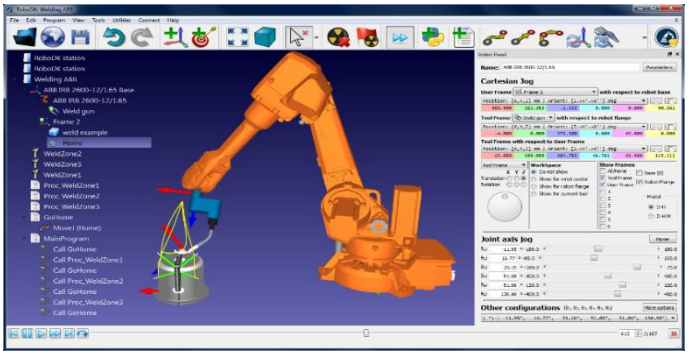


Figura 22 Simulação de soldadura com modelo CAD importado em ambiente RoboDK [18]

Recorrendo a modelos tridimensionais pré-definidos do equipamento da célula, o utilizador pode criar e otimizar de forma intuitiva o *layout* da mesma. Estes *layouts* englobam, normalmente, modelos tridimensionais dos robôs, controladores, ferramentas, equipamentos de segurança e demais auxiliares. Nas aplicações dos fabricantes, estes modelos são representações do equipamento comercializado pela própria marca ou parceiros. No entanto, os programas incluem também a possibilidade de criar objetos recorrendo a sólidos básicos, como esferas, cilindros, cubos, pirâmides, entre outros. A inclusão da possibilidade de importação de ficheiros em formato de CAD possibilita aos utilizadores usarem diretamente modelos de programas externos de peças, ferramentas e equipamento mais complexos, em vez de os terem de criar recorrendo aos sólidos básicos, o que, por vezes, se revela limitativo e, muitas vezes, impreciso.

Uma outra função do ambiente gráfico é a visualização da simulação. Atualmente, a programação *offline* é acompanhada pela possibilidade de simular a execução do código.

As vantagens são evidentes, sendo porventura a maior, a verificação do comportamento do código escrito sem ser necessário recorrer ao robô ou ao seu controlador, fisicamente. Dependendo do fornecedor, a execução e verificação do código introduzido poderá passar por algoritmos criados para o efeito ou, no caso de alguns fabricantes, por uma virtualização do controlador do robô.

Durante a simulação do código, este é executado pelo simulador (passo-a-passo ou em sequência), atualizando o ambiente gráfico, isto é, a posição e orientação do robô ou respetivos periféricos, conforme as instruções.

A simulação da movimentação do robô é, por sua vez, processada recorrendo a modelos matemáticos de cinemática direta ou inversa, consoante as entradas se tratem das posições das juntas ativas ou da posição final do órgão terminal, respetivamente. Englobada na movimentação, em alguns simuladores, encontra-se a opção de (*automatic path solving*) que permite evitar erros – incluindo, singularidades, limites de juntas e de alcance.

Outras funcionalidades presentes em programas de simulação são a visualização da consola de programação, verificação de colisões (útil para evitar danos ao equipamento), a possibilidade de integração de componentes que apresentem um comportamento definido e implementado pelo utilizador que possam interagir com o robô e/ou periféricos (por exemplo, *Smart Components* no caso da ABB) e o recurso a modelos matemáticos para a simulação de dinâmica e física. Este último permite ao programador avaliar a interação de objetos da sua simulação com, por exemplo, a gravidade.

Algumas operações dos robôs são constantes e transversais a diversas indústrias, como a pintura, soldadura, maquinação ou manipulação. Apesar de o *software* de base ser capaz de lidar com estas operações, diversos fabricantes oferecem soluções de *hardware* e *software* específicas, e com funções próprias e mais avançadas do que as genéricas, para tais operações. Estas funcionalidades são, normalmente, comercializadas como *add-ins* ou *plugins* ao programa de base.

O maior desafio do uso de ambientes virtuais é, possivelmente, a "calibração" entre o mundo virtual e a realidade, ou seja, fazer corresponder o mundo virtual simulado à realidade. Após a programação e durante o arranque, deve o técnico ter em atenção as diversas variáveis envolvidas na programação/simulação (posições, velocidades, pressões, entre outras) de forma a evitar erros ou mesmo danos ao equipamento.

Por muito precisos que sejam os modelos matemáticos utilizados na simulação, estes revelam-se infrutíferos se os parâmetros de entrada em ambiente simulado diferirem da realidade.

3.2.1. *SOFTWARE* PROPRIETÁRIO

A totalidade de fabricantes de robôs industriais encontrada, durante a pesquisa efetuada para este trabalho, revelou possuir *software* de programação *offline* e de simulação, de base. Apresentada na Tabela 3, está a lista de alguns fabricantes de robôs, a qual inclui também o nome do seu *software* de simulação.

Tabela 3 *Software* de programação *offline*/simulação por fabricantes de robôs

Fabricante	***Software***
ABB	RobotStudio
Adept	Adept ACE
Comau	Robosim Pro
Denso	EMU
Fanuc	ROBOGUIDE
Kawasaki	K-ROSET
KUKA	Kuka.Sim
Motoman (Yaskawa)	MotoSim
Nachi	OnDesk
Stäubli	Stäubli Robotics Suite
Universal Robots (UR)	URSim

Em determinados casos, o *software* de programação/simulação poderá ser incluído em pacotes mais completos ou na oferta global dos produtos de uma determinada marca. Tomando um exemplo em concreto, constatou-se que o MotoSim, da marca Yaskawa (previamente Motoman até à sua aquisição) poderá ser adquirido – num só pacote de *software* de uma só linguagem – juntamente com outros produtos de automação que visam, por exemplo, a programação de autómatos, servos e variadores de velocidade. [19].

As vantagens do *software* fornecido pelo fabricante são inerentes ao facto do mesmo possuir o *know-how* de todo o sistema e conseguir, com isso, funcionalidades difíceis de alcançar por parte de empresas externas. A este propósito pode referir-se o exemplo do RobotStudio, que permite a simulação do *software* da consola tátil tal como a presente no controlador do robô, ou do caso do Stäubli Robotics Suite, que permite sincronizar o *software* do computador com o do robô, permitindo que o utilizador veja em tempo real os valores das variáveis que se encontram no controlador do robô.

3.2.2. *SOFTWARE* GENÉRICO

Apesar dos fabricantes comercializarem o *software* necessário para a virtualização dos seus produtos, existem correntemente no mercado diversos fornecedores de *software*, dito genérico. Estes fornecedores oferecem programas que permitem a programação *off-line* e/ou simulação de mais do que um modelo de robôs de diferentes fabricantes. Os mais comuns são os programas que atuam como "tradutores" para uma série de linguagens, transpondo as ações geradas nos ambientes gráficos tridimensionais ou em *scripts* para linguagens suportadas pelos controladores, através de um *post-processor*. Neste caso, o nível de abstração do *hardware* é, inevitavelmente, mais elevado do que em aplicações proprietárias. Por este motivo, muitos fornecedores optam por utilizar linguagens de *scripts* ou de alto-nível como, por exemplo, Python, Lua, C++ ou C#. A Tabela 4 apresenta um resumo da pesquisa efetuada por *software* genérico e respetivos fabricantes suportados, expondo os mais divulgados.

Tabela 4 Principais fabricantes de *software* de simulação e programação genérico

Desenvolvedor	***Software***	**Fabricantes suportados**
Coppellia Robotics	V-Rep	não produz código para o robô
RoboDK	RoboDK 2.7	ABB, Fanuc, KUKA, UR, Yaskawa, entre outros
WAT Solutions	Workspace 5	ABB, Fanuc, KUKA, UR, Yaskawa, entre outros
Octopuz	Octopuz	ABB, Fanuc, KUKA, UR, Yaskawa, entre outros
Easy-Rob	EASY-ROB	ABB, Fanuc, KUKA, Yaskawa, entre outros
Siemens	RobotExpert	ABB, Fanuc, KUKA, entre outros
Delfoi	Delfoi Robotics	ABB, Fanuc, KUKA, Yaskawa, entre outros
Dassault Systèmes	Delmia Simulation	ABB, Fanuc, KUKA, Yaskawa, entre outros
Visual Components	Visual Components	ABB, Adept, Fanuc, KUKA, Yaskawa, entre outros
Artiminds	ArtiMinds RPS	Denso, KUKA, UR
Energid	Actin Simulation	UR

Note-se que a Tabela 4 indica os fabricantes de robôs com os quais o *software* é compatível em vez de enumerar a linguagem, individualmente. Isto deve-se ao facto de alguns fornecedores de *software* divulgarem quais os fabricantes com os quais são compatíveis, mas não especificarem qual a linguagem, em particular.

Também da leitura da Tabela 4, evidencia-se um *software* que não se enquadra no contexto global, apelidado de V-REP: este *software* não produz diretamente código para interpretação nem para ser corrido diretamente pelo controlador do robô. A particularidade do V-REP é que permite programar robôs *offline* e comunicar com os restantes periféricos, inclusive com o robô, através de diversas interfaces, das quais se destacam dois tipos: Remote Application Programming Interface (API) ou Robot Operating System (ROS) [6] Interface. O diagrama da Figura 23 permite uma melhor compreensão da arquitetura deste *software*.

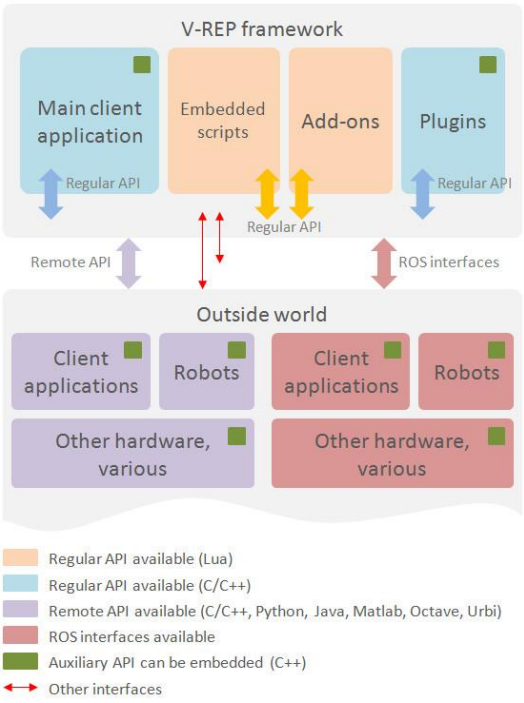


Figura 23 Diagrama de blocos da arquitetura do *software* V-REP [20]

O programa engloba o *software* necessário para a comunicação com a *framework*, sendo necessário, para tal, integrar as API indicadas no sistema operativo do robô ou na aplicação desejada. O recurso às interfaces ROS permite ao V-REP possuir uma interface mais comum e generalizada do que as API do próprio fabricante.

[6] ROS é uma *framework* flexível para a escrita de *software* para robôs, que fornece uma coleção de ferramentas, bibliotecas e convenções, e cujo propósito é a criação de comportamentos complexos e robustos em diversas plataformas robóticas[35]. Aplica-se à globalidade dos robôs e não apenas aos manipuladores industriais, sendo que estes dispõem de uma versão própria apelidada de ROS-Industrial [36].

O código do *software* (V-REP) tem também a vantagem de ser aberto (*open-source*), o que possibilita a leitura e modificação parcial do programa para servir as necessidades do utilizador ou integrador, caso seja necessário. Este tipo de solução, independente do fabricante, revela-se valiosa para utilizadores que lidem com mais do que uma gama de robôs ou que procurem soluções mais avançadas, flexíveis e que englobem mais opções e maior abertura a novos recursos do que as fornecidas pelos fabricantes. Esta solução, claramente, estende-se a mais tipo de robôs do que a manipuladores industriais.

3.3. *SOFTWARE* DE PALETIZAÇÃO

No seguimento do contexto deste capítulo, e à semelhança do que foi feito para o *software* para programação de robôs, procurou-se encontrar e compilar numa lista o *software* para operações de paletização. Os resultados da mesma revelaram-se, porém, escassos, e encontram-se listados na Tabela 5, onde, para além do nome do programa, se encontra o nome do desenvolvedor e página da internet.

Tabela 5 Programas de paletização externos

Software	**Fabricante**	**Website**
Cape Pack	Esko	https://www.esko.com/en/products/cape-pack
Cube Designer	Logen Solutions	http://www.logensol.com/
IrisPallet	IRIS Srl	http://www.iris-it.com/en/index.html
Packer3D	Packer3D	http://www.packer3d.com
Pallet Stacking	Pallet Stacking	http://www.palletstacking.com/
Quick Pallet Maler	Koona	http://www.koona.com/qpm/
Stack Builder	treeDim	https://github.com/treeDiM/StackBuilder

Da lista apresentada na Tabela 5, apenas o IrisPallet (da firma IRIS Srl.) apresenta um programa para uso com um robô KUKA. Alguns outros programas, como o caso do Packer 3D ou Quick Pallet Maker, produzem ficheiros de saída passíveis de serem interpretados por *software* para robôs (XML), porém, esta solução requer *software* adicional do lado do controlador do robô. Em ambos os casos, os ficheiros produzidos pelos programas listam as diversas posições dos objetos na pilha desde um ponto inicial definido, além de indicarem algumas informações adicionais (como o número de camadas, altura das pilhas, entre outras).

É, também, importante focar que o tema da paletização se encontra intrinsecamente ligado à logística. Como tal, muitos programas, que se intitulam para paletização, centram-se, simultaneamente, em operações de apoio à elaboração de cargas para transporte e correspondente otimização das mesmas, quer no processo de paletização em si. Um exemplo disto são os programas Packer 3D ou Cube Designer que, para além das funções de cálculo de palete, fornecem também a opção de cálculo de otimização de espaço em camiões ou contentores marítimos com as paletes criadas.

3.3.1. *SOFTWARE* PARA APLICAÇÕES ROBOTIZADAS

Conforme mencionado anteriormente no ponto 3.2, alguns fabricantes oferecem soluções de *software* para as operações mais comuns a serem executadas pelos seus produtos. Entre as operações mais comuns está a paletização. Porém, das marcas apresentadas previamente na Tabela 3, apenas três fabricantes apresentam *software* dedicado e respeitante a operações de paletização. As outras várias marcas fornecem a possibilidade de incluir as aludidas operações como *add-in* (componentes de *software* que estendem as funcionalidades do principal), conforme se pode verificar na Tabela 6.

Tabela 6 Programas de paletização para programação de robôs

Software	**Fabricante**	**Tipo**
Palletizing PowerPack	ABB	*add-in* - RobotsStudio
Palletizing Motion	Comau	*add-in* – Robosim Pro
K-Sparc	Kawasaki	*add-in* – K-ROSET
PalletPRO PalletTool	Fanuc	independente
PalletTECH	KUKA	independente
PalletSolver	Motoman (Yaskawa)	independente

Uma análise mais aprofundada aos requisitos dos programas apresentados na Tabela 6 revelou que a grande maioria aceita, sensivelmente, as mesmas entradas. Ou seja, a criação do mosaico e correspondente criação da pilha são obtidas primariamente a partir de informações do produto a paletizar (dimensões e tipo de embalagem), da palete (dimensões e capacidade de carga), da ferramenta (tipo e respetivas variáveis), do tipo de robô, da pilha pretendida (altura e mosaico para as camadas) e da configuração da célula robótica (localização das zonas de levantamento e entrega do produto).

Recorrendo a estas informações (introduzidas pelo utilizador), é possível a criação de código ou de valores para a introdução em código previamente desenvolvido

Tomando como exemplo o programa K-SPARC da Kawasaki, verifica-se que este utiliza grande parte da informação [21]. A Figura 24 apresenta o método de como o utilizador deverá proceder para obter o programa de paletização. No total, segundo a marca, são apenas seis passos, iniciando-se pela escolha do modelo de robô, seguindo-se a seleção de um de doze possíveis *layouts*, a introdução do tamanho do produto e do mosaico para as camadas. Os dados gerados serão posteriormente descarregados para o controlador do robô e, recorrendo à consola de interface com o utilizador, será ainda possível proceder a alguns ajustes, como por exemplo, a alteração dos pontos iniciais para levantamento e entrega dos objetos a paletizar.

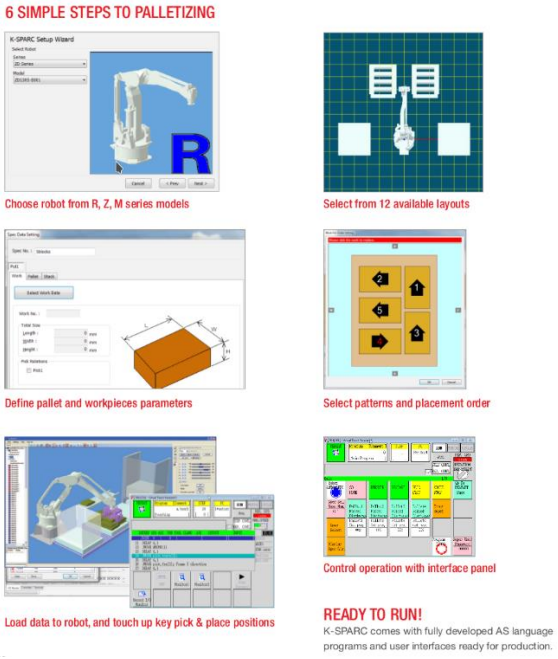


Figura 24 Os seis passos necessários para paletização do programa K-SPARC [21]

Em suma, e perante o descrito, o *software* para aplicações robotizadas, no contexto geral, encontra-se numa direção de maior abertura e flexibilidade. Esta última implica modificações e, porventura, a evolução das ferramentas ao dispor dos programadores, dado os desafios correntes serem mais exigentes. Porém, e ao mesmo tempo, torna-se evidente que existe um esforço, por parte dos fabricantes, em permitir uma mais simples e menor interação dos utilizadores com os equipamentos. No seguimento desta abordagem, e tendo este ponto como um dos objetivos, foi desenvolvido o programa, criado no âmbito desta prova, e descrito e no capítulo seguinte.

4. DESENVOLVIMENTO E IMPLEMENTAÇÃO

O objetivo do presente trabalho é a criação de uma aplicação para a programação automática de robôs afetos a operações de paletização. Por esta razão, foi necessária a escolha de um *software* de base para o desenvolvimento e simulação da aplicação, assim como de um objeto de estudo concreto, ou seja, um robô e respetiva ferramenta para operações de paletização.

A aplicação desenvolvida no âmbito deste trabalho, e à qual foi atribuído o nome de Automatic Robot Palletizer (ARP), consiste num programa principal (*main*), com funções auxiliares para diversos fins, desenvolvida em conformidade com as diretrizes do fabricante dos equipamentos. A sua escolha para a execução deste trabalho, deveu-se a ser a marca de um robô disponibilizado em laboratório que servirá, posteriormente, para validação do programa desenvolvido.

Esta aplicação poderá também interagir diretamente com controladores de robôs ou, caso seja desejada a simulação prévia, com o *software offline* de programação de robôs. A interface com o utilizador poderá ser feita através do ficheiro de inicializações, disponibilizado para o efeito.

O presente capítulo tem como propósito a explanação das bases e da realização desta parte do trabalho, passando pela apresentação do ambiente de desenvolvimento do *software* (RoboStudio), apresentação do robô IRB460 (escolhido para as simulações e ensaios) e, brevemente, do fabricante de ambos, a Asea Brown Boveri (ABB).

4.1. ARQUITETURA DO SISTEMA

A aplicação desenvolvida, no âmbito deste trabalho, tem como entradas um conjunto de valores iniciais, introduzidos pelo utilizador no respetivo ficheiro de inicializações, que variam conforme o caso apresentado (por exemplo, serão em função do tamanho e localização do objeto a paletizar). Este ficheiro, em conjunto com os módulos que compõem a aplicação, após serem descarregados para um controlador, serão os responsáveis pelo controlo das operações do robô.

Exposta na Figura 25, encontra-se uma apresentação global do sistema e as suas ligações a módulos ou programas externos. O sistema assenta, essencialmente, sobre três grandes blocos:

- "Programa ARP": aplicação desenvolvida para a programação automática;
- "*Software* programação": o *software* de programação *offline* e simulação que servirá como intérprete entre a aplicação desenvolvida e o controlador do robô;
- "Célula robótica": que contém as unidades de controlo e o robô físico, assim como todo o equipamento necessário para as tarefas em questão.

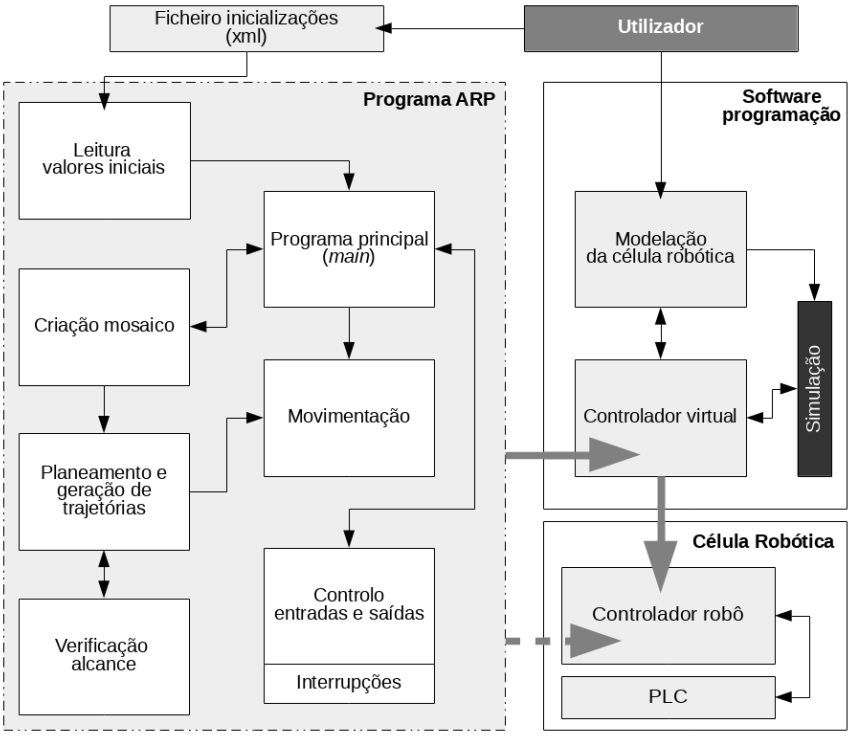


Figura 25 Diagrama de blocos da arquitetura do sistema

Internamente, a aplicação é constituída por um conjunto de blocos que têm como coordenador o programa principal (*main*). Ainda dentro do bloco do programa ARP, podem encontrar-se as tarefas de base – consideradas mínimas para a execução da paletização – e que correspondem, nos módulos do programa, a funções. Entre as mesmas funções verificam-se as dependências conjuntas, como por exemplo, a necessidade do programa principal em trocar informação com a função de criação do mosaico de paletização ou com o bloco responsável pela movimentação do robô. A informação é passada entre funções recorrendo a variáveis globais ou parâmetros.

De salientar ainda na Figura 25, que também foi contemplada uma interface ao autómato, ou *Programmable Logic Controller* (PLC), que gere a célula de paletização e controla o sincronismo entre o transportador, robô e sistema de colocação/remoção das paletes, além de, eventualmente, gerir possíveis interfaces a sistemas externos.

A modelação da célula robótica e a sua respetiva simulação ficaram a cargo do *software* desenvolvido pelo próprio fabricante, de forma a fazer uso das potencialidades oferecidas pelo mesmo. Isto permite ao utilizador antever possíveis perigos ou erros na construção da célula robótica.

Pela leitura da Figura 25 é ainda possível inferir que se pretende a menor interação possível por parte do utilizador com o programa, estando esta restrita à introdução dos valores iniciais, à modelação e à simulação da célula robótica (caso necessárias). Encontra-se também prevista a possibilidade de não ser necessário recorrer novamente à modelação da célula e correspondente simulação para serem feitas pequenas alterações a uma situação já em funcionamento (como por exemplo, a alteração do tamanho do objeto a paletizar). Neste caso será necessário apenas alterar o respetivo ficheiro dos valores e descarregar o mesmo para o controlador do robô. Apesar da praticabilidade desta possibilidade, a mesma acarreta os riscos inerentes à geração de código para uma qualquer máquina sem a prévia simulação, tais como, a eventualidade do robô não gerar o mosaico esperado por falha na introdução dos valores iniciais.

4.2. ABB - ASEA BROWN BOVERI

A Asea Brown Boveri (ABB) nasce em Zurique, na Suíça, em 1988, da fusão de duas empresas: Allmänna Svenska Elektriska Aktiebolaget (ASEA) e a Brown, Boveri & Cie. (BBC). A sua área de negócio assenta, maioritariamente, no setor eletrotécnico, mais concretamente, na tecnologia de componentes elétricos para o sector terciário e indústria, em redes elétricas, em automação industrial e robótica [22]. A última será, porventura, a área pela qual é mais conhecida.

A gama de robôs industriais, que atualmente comercializa, contém um considerável número de modelos (superior a trinta) que abrangem funções genéricas ou específicas. Tal como a grande maioria dos fabricantes de robôs – conforme mencionado previamente no ponto 3.2.1–, também a ABB disponibiliza *software* para configuração, programação e simulação dos seus produtos, apelidado de RobotStudio.

4.2.1. CÉLULA ROBÓTICA

A célula robótica simulada apresentará, para além do manipulador e correspondente controlador, uma tela transportadora que servirá como *infeed* (ponto de entrega das embalagens) ao sistema. O modelo da tela usado é disponibilizado pela ABB, na sua biblioteca de componentes ("Conveyor 400").

O ponto de entrega será representado por uma Europalete (tipo EPAL 1), colocada ao nível do solo para possibilitar a construção de uma pilha mais alta (modelo também fornecido pela ABB). Seguindo a recomendação do fabricante, foi colocado um pedestal sob o robô de cerca de 450 mm, conforme o indicado na documentação do robô [23]. A interface para o controlador central da célula (autómato) também foi contemplada durante o desenvolvimento da mesma e será simulada recorrendo a sinais existentes na própria célula.

Para a simulação do programa gerado, foi escolhido um robô da linha média da ABB, de entre os indicados para operações de paletização. A opção prendeu-se, não só com a especificidade da tarefa a ser executada, mas também com as características medianas, como por exemplo, a capacidade de carga, o alcance e o tamanho [16].

O robô articulado de 4 eixos IRB460 apresenta uma capacidade de carga de 110 kg, um alcance de 2,4 m e uma repetibilidade de 0,2mm. É, segundo a ABB e à data, o robô de paletização mais rápido do mundo [24]. Na Figura 26 apresenta-se uma foto do IRB460 durante um processo de paletização com uma ferramenta do tipo *Clamp*.

Figura 26 Robô ABB IRB460 com uma ferramenta tipo *Clamp* [16].

Como ferramenta de trabalho, foi selecionado o *Gripper* de vácuo disponibilizado pela ABB, por esta se coadunar com as funções de paletização necessárias e apresentar adicionalmente a opção de movimentação de paletes. Esta ferramenta permite manusear objetos com peso até 40 kg e dimensões máximas de 1200 × 500 × 300 mm (sendo estas o comprimento, a largura e a altura, respetivamente). Esta ferramenta encontra-se ilustrada na Figura 27.

Figura 27 Ferramenta de vácuo escolhida [15]

O facto da maior parte dos bens a serem paletizados serem caixas [25] contribuiu também para a escolha da ferramenta, uma vez que esta apresenta dimensões e geometria apropriadas a esta tarefa. Um ponto positivo deste tipo de ferramenta é que os vários segmentos de válvulas de vácuo podem ser atuados de forma independente (em relação aos restantes segmentos) e, desta forma, permitem transportar bens de diferentes dimensões. Estas ferramentas apresentam também a vantagem de agarrar a peça pelo topo, reduzindo, desta forma, a probabilidade de colisão da ferramenta com a pilha executada.

Em ambos os casos (robô e ferramenta), a ABB disponibiliza os modelos tridimensionais para utilização em ambiente RobotStudio. Este foi, igualmente, um fator que influenciou na seleção do modelo de ensaio.

4.2.2. O CONTROLADOR IRC5

Atualmente na sua quinta versão, o controlador da ABB Industrial Robot Controller (IRC) é utilizado para o controlo de toda a gama de robôs industriais da ABB [26]. O seu sistema operativo é denominado "RobotWare" e encontra-se, presentemente, na sua versão 6, sendo a linguagem de programação do robô denominada RAPID.

Como o próprio nome indica, o controlador tem como função a supervisão/atuação dos diversos componentes que compõem robô e a sua interação com os periféricos, aquando da execução de diversas ações que são programáveis num computador ou através da consola disponibilizada para o efeito.

O IRC5 apresenta quatro versões – selecionáveis conforme a aplicação final do robô – assim como acessórios para a interação entre o utilizador e o robô. Entre estes, destacam-se a consola de programação ("Flex Pendant") ou o dispositivo para movimentação básica ("T10 Jogging Device") [26]. A Figura 28 apresenta alguns dos modelos do IRC5.

Figura 28 Diferentes variantes do Controlador IRC 5 [26].

As diversas variantes do IRC5 apresentam, entre outras, diferenças no tamanho do quadro, na tensão de alimentação, no número de entradas/saídas digitais e na inclusão de *software* específico no sistema operativo, por exemplo, para um acréscimo de segurança do robô na célula ("SafeMove*"*) ou para otimização da movimentação do robô ("QuickMove / TrueMove"). O controlador encontra-se também presente de forma virtual no RobotStudio ("ABB Virtual Controler") em termos de lógica funcional e em modelo tridimensional gráfico para melhor simular possíveis ambientes e a própria lógica de trabalho.

4.2.3. ROBOTSTUDIO

Desenvolvido pela ABB com o intuito de proporcionar programação *offline* aos seus produtos, o RobotStudio – versão 6 – apresenta-se como um programa de modelação, simulação e programação que permite que a mesma seja feita de forma independente da implementação física.

O RobotStudio apresenta uma série de funcionalidades que permitem ao utilizador criar uma simulação o mais realista possível da sua célula de trabalho, incluindo bibliotecas de objetos e equipamento base ou a*dd-ins* (pacotes de *software*) específicos para um determinado tipo de operação, como pintura ou soldadura. O ambiente de trabalho do RobotStudio, após a abertura de um projeto existente, está ilustrado na Figura 29.

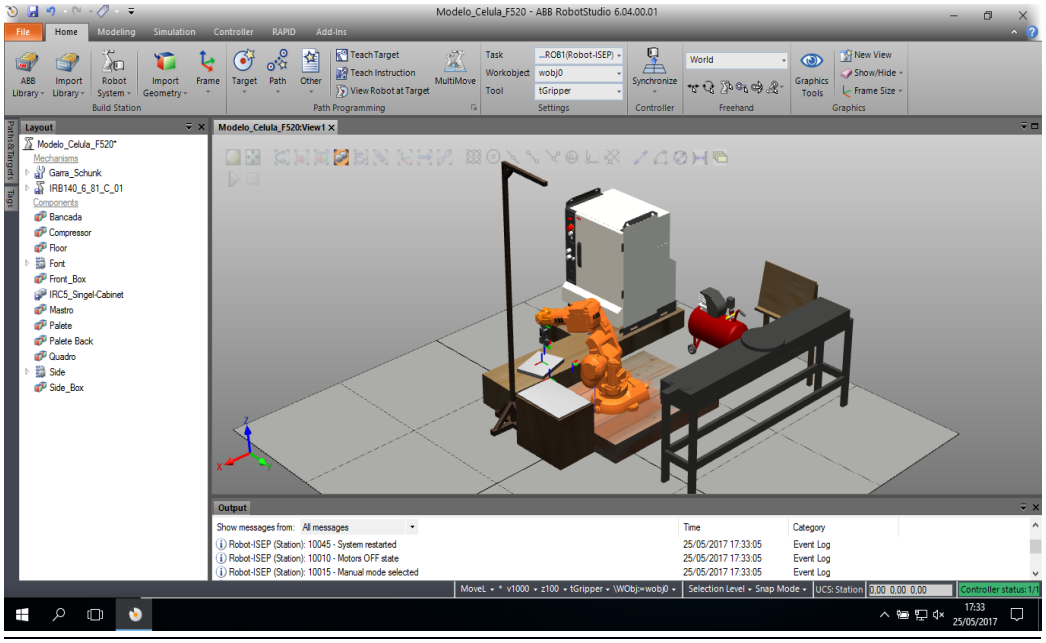


Figura 29 O ambiente ABB RobotStudio

Entre as funcionalidades oferecidas pela aplicação encontram-se, por exemplo, a importação de ficheiros CAD, a criação de representação de objetos (caixas, esferas, cones, etc.) ou o recurso a *Smart Components*, como descrito no ponto 3.2. Numa fase posterior da programação, aquando da movimentação do robô, é possível verificar se o mesmo alcança o ponto desejado e a reação a um programa em RAPID, entre outras. Estas verificações são feitas recorrendo a simulações, executando o código RAPID no controlador virtual, o qual realiza as operações de cinemática inversa necessárias para a movimentação do robô. O RobotStudio oferece também a possibilidade de simular componentes externos ao robô, com lógica comportamental, e verificar possíveis colisões durante a movimentação do robô. Após a simulação e consequente depuração de erros, o mesmo programa poderá ser descarregado para o controlador real. A ABB disponibiliza também conjuntos de diversas rotinas, protocolos e ferramentas – API – para a criação de a*dd-Ins* para uso em ambiente RobotStudio, abrindo-o desta forma, a programas externos como, por exemplo, o Microsoft Visual Studio [27].

4.2.4. SISTEMAS DE COORDENADAS

Para a construção do ambiente simulado, é necessário definir, no espaço, a localização dos intervenientes. Como tal, o RobotStudio utiliza diversos sistemas de coordenadas interligados, alguns inerentes ao programa, outros definidos pelo utilizador. Representados na Figura 30 encontram-se os principais sistemas de coordenadas.

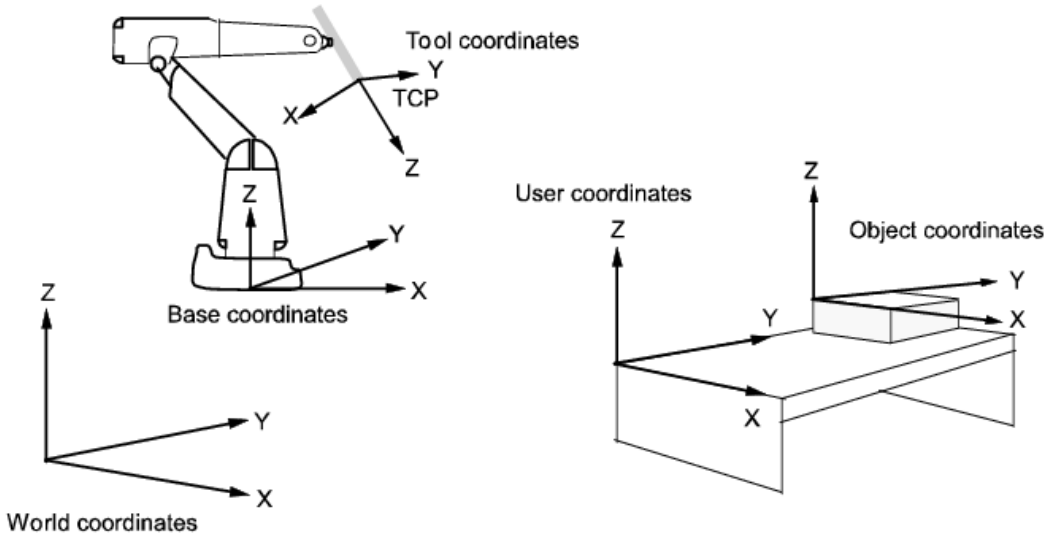


Figura 30 Principais sistemas de coordenadas do RobotStudio [28].

Esta estrutura que agrega diversos sistemas de coordenadas é hierárquica, ou seja, a origem de cada sistema de coordenadas é definida como uma posição relativa ao seu antecessor. No topo encontra-se o sistema de coordenadas mundo (*World Frame*) que, tal como o nome o indica, é a referência base à qual os restantes estarão referenciados, direta ou indiretamente. A exceção à regra deste caso é o sistema de coordenadas do punho do robô, que se encontra posicionado de acordo com a flange que acopla a ferramenta.

Cada robô encontra-se dotado de um sistema de coordenadas na base (*Base Frame*) que pode estar referenciado diretamente ao sistema mundo ou estar previamente relacionado com uma *Task Frame* (sistema de coordenadas de tarefa). A função das *Task Frames* é relacionar diversos mecanismos que executem uma tarefa comum, ou seja, providenciar um sistema de coordenadas para uma determinada tarefa. A par com os robôs, cada ferramenta de trabalho é provida de um sistema de coordenadas (*Tool Frame*) cuja origem é denominada de *Tool Centre Point* (TCP).

O sistema de coordenadas da ferramenta encontra-se, por sua vez, referenciado ao sistema de coordenadas do punho do robô, cuja origem coincide com o centro da flange, na qual a ferramenta será montada. O eixo dos *xx*' aponta no sentido do furo de controlo da mesma e o eixo dos *zz*' tem a mesma direção que o eixo que passa através do furo central da flange, apontando na direção contrária ao robô, conforme ilustrado na Figura 31.

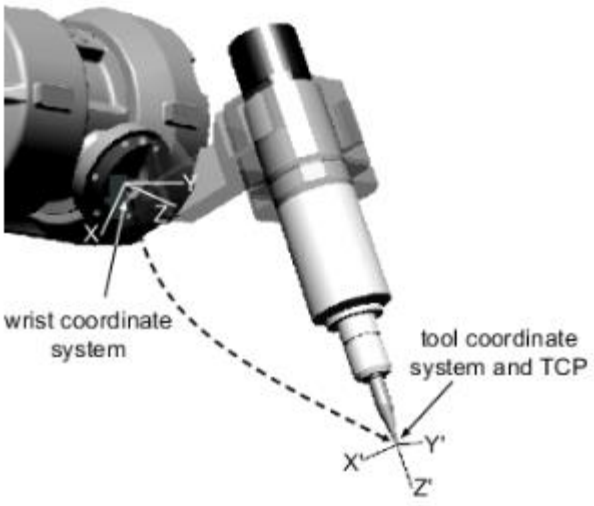


Figura 31 Sistema de coordenadas do punho do robô [28].

O RobotStudio proporciona a possibilidade ao utilizador de criar o seu próprio sistema de coordenadas ou *User Coordinate System* (UCS), cuja vantagem é facilitar a programação, através do (re)posicionamento destes sistemas de coordenadas em pontos estratégicos, se necessário. Desta forma, poderá o utilizador programar, por exemplo, uma movimentação ou rotação do robô referenciada a este sistema de coordenadas.

Os restantes objetos de trabalho, apelidados de *work objects*, podem ainda conter sistemas de coordenadas, os quais são, normalmente, definidos pelo utilizador. Os *work objects* são constituídos por duas *frames*: a *User Frame* e a *Object Frame*, encontrando-se a segunda referenciada à primeira. Quando se programa a movimentação do robô para um determinado ponto (*Target*), este estará sempre referenciado a um *work object* (por omissão será o *wobj0*, que coincide com a base do robô).

A utilização de *Targets* referenciados a *work objects*, que não o definido por omissão, apresenta a vantagem de não existir a necessidade de reajustar os *Targets* caso o *work object* se mova. Por outras palavras, permite facilmente ajustar posições sem perder as trajetórias (*Paths*) previamente programadas [27].

4.2.5. *TARGETS* E *PATHS*

Um *Target* representa um ponto que o robô poderá atingir no espaço. Contém, também, informação sobre a orientação do robô (relativamente ao *work object* a que se encontra referenciado) e sobre a configuração dos eixos do robô quando este o atingir.

Quando descarregados para o controlador do robô, os *Targets* são convertidos para o tipo Robtarget que, por sua vez, inclui informação adicional sobre eixos externos.

Assim, um *Target* poderá ser definido da seguinte forma:

```
CONST robtarget p10:=
        [[x,y,z],[q1,q2,q3,q4],[cf1,cf4,cf6,cfx],[eax_a,eax_b,..,eax_f];
```

onde:

- [*x, y, z*] representam a posição;
- [q_1, q_2, q_3, q_4] representam a orientação (expressa num quaternião);
- [cf_1, cf_4, cf_6, cf_x] são os valores da configuração dos eixos (1, 4 e 6) do robô no *Target* (cf_x é dependente do tipo de robô)*;*
- [*eax_a, eax_b, ..., eax_f*] são as posições dos eixos adicionais (caso existentes).

A utilização dos valores de configuração dos eixos dependerá do tipo de robô. Como exemplo, e de acordo com o manual de referência técnica da ABB, para robôs de quatro eixos apenas o valor do eixo cf_6 será tomado em consideração [28].

A movimentação entre uma determinada sequência de *Targets* é denominada *Path* (percurso). Independentemente do modo usado para a criação dos mesmos – automático ou manual – a movimentação do robô ao longo de um *Path* requer sempre uma instrução de movimento (*Move*).

Um exemplo de uma instrução de "*Move"* poderá ser:

```
MoveL p10, v1000, z10, ttool\WOBj := wref;
```

onde:

- "MoveL" : tipo de movimento – interpolação linear (MoveL), interpolação no espaço de juntas (MoveJ) ou interpolação circular (MoveC), entre outros;
- "*p10*" : ponto de destino;
- "*v1000*" : velocidade de deslocamento do robô;
- "*z10*" : tamanho da zona de aproximação;
- "*tool \Wobj := wref*": ferramenta em uso e *work object* do ponto de destino (parâmetro opcional).

4.2.6. LINGUAGEM RAPID

A programação do robô/célula recorre a uma linguagem de programação, também criada pela ABB, apelidada de RAPID, que tem como bases a extinta ASEA programming Robot Language (ARLA) e C. Através desta linguagem é possível definir o fluxo do programa, controlar a movimentação do robô, a sua interface de entradas e saídas, etc. Tal como qualquer linguagem de alto-nível, também o RAPID segue uma hierarquia lógica para o seu uso, apresentando 3 tipos de rotinas [28]:

- *procedures*: os procedimentos são usados como subprogramas;
- *functions:* as funções retornam um valor de um determinado tipo que é usado como argumento de uma instrução;
- *trap routines*: estas rotinas providenciam meios para detetar erros no programa e podem também ser associadas a uma interrupção específica.

Estas rotinas são compostas por um conjunto de instruções que descrevem o comportamento do robô. Entre as instruções destacam-se as instruções para controlo do fluxo do programa, para movimentação do robô, para manipulação de entradas e saídas, para comunicação e para interrupções. Normalmente, as instruções têm um determinado número de parâmetros associados que são usados como valores de entrada/saída.

A informação em RAPID é guardada sobre a forma de dados que, conforme a aplicação, podem ser globais ou locais (estes serão apenas para uso da própria rotina). Estes dados podem tomar diferentes formas, sendo, por exemplo, dados numéricos (contadores, iterações, etc.) ou dados de ferramentas (peso, localização do seu TCP, etc.). O agrupamento destes dados gera diversos tipos de informação que descrevem posições, cargas, etc.

Os dados são definidos em três tipos distintos:

- constantes: como o próprio nome indica, são dados que são fixos e apenas podem ser alterados modificando o código-fonte;
- variáveis: são dados que podem ser alterados com o decorrer do programa;
- *persistent*: podem ser descritas como variáveis "persistentes", ou seja, o seu valor assume o último valor adquirido, mesmo no caso de reinicialização do programa.

O RAPID inclui, também, outros tipos de funcionalidades, consideradas normais para qualquer linguagem de alto-nível, como as operações aritméticas, o *multi-tasking* ou a forma de lidar, automaticamente, com erros ocorridos.

4.3. ESTRUTURA DO PROGRAMA "ARP"

Com o objetivo de uniformizar a elaboração de aplicações em RAPID, a ABB disponibiliza diretrizes para a criação das mesmas [29]. Estas linhas gerais incluem a metodologia para elaboração de documentação e preparação das ditas aplicações. Assim, foram criados quatro módulos, que contêm as rotinas principais e auxiliares, e um ficheiro eXtensible Markup Language (XML), o qual contém os parâmetros iniciais para a execução das rotinas.

Os ficheiros foram desenvolvidos de forma a serem transparentes ao tipo do robô, isto é, apenas dependentes do tipo de controlador que interpretará o código RAPID, mas sem instruções ou comandos que sejam específicos da versão/tipo de robô.

Na Tabela 7 é exibida uma listagem dos módulos desenvolvidos, assim como, a sua extensão, tipo e uma breve descrição do conteúdo dos mesmos.

Tabela 7 Listagem dos módulos desenvolvidos

Nome	Extensão	Tipo	Descrição
ARPPalletizer	.mod	Módulo de programa	Programa principal que contém a execução do programa e dados suplementares à mesma
ARPMovement	.mod	Módulo de programa	Módulo que lida com as movimentações do robô
ARPBasics	.sys	Módulo de sistema	Contém todas as funções que, apesar de serem necessárias à execução do programa, não pertencem diretamente à execução do programa
ARPInit	.xml	Ficheiro de inicialização	Ficheiro que contém os valores iniciais do programa

Dos módulos apresentados na tabela, destaca-se o módulo "ARPPalletizer", que coordena a execução do programa principal, usando funções ou procedimentos existentes nos outros módulos para a execução das tarefas necessárias. Este módulo é, também, o que lida com as interrupções geradas pelos sinais externos (entradas digitais) e é onde se encontram definidas as variáveis globais.

Ainda tendo em consideração as recomendações da ABB, para a criação e desenvolvimento de operações de manuseamento, foi construído um diagrama de fluxo (ver Figura 32) para a execução do programa principal.

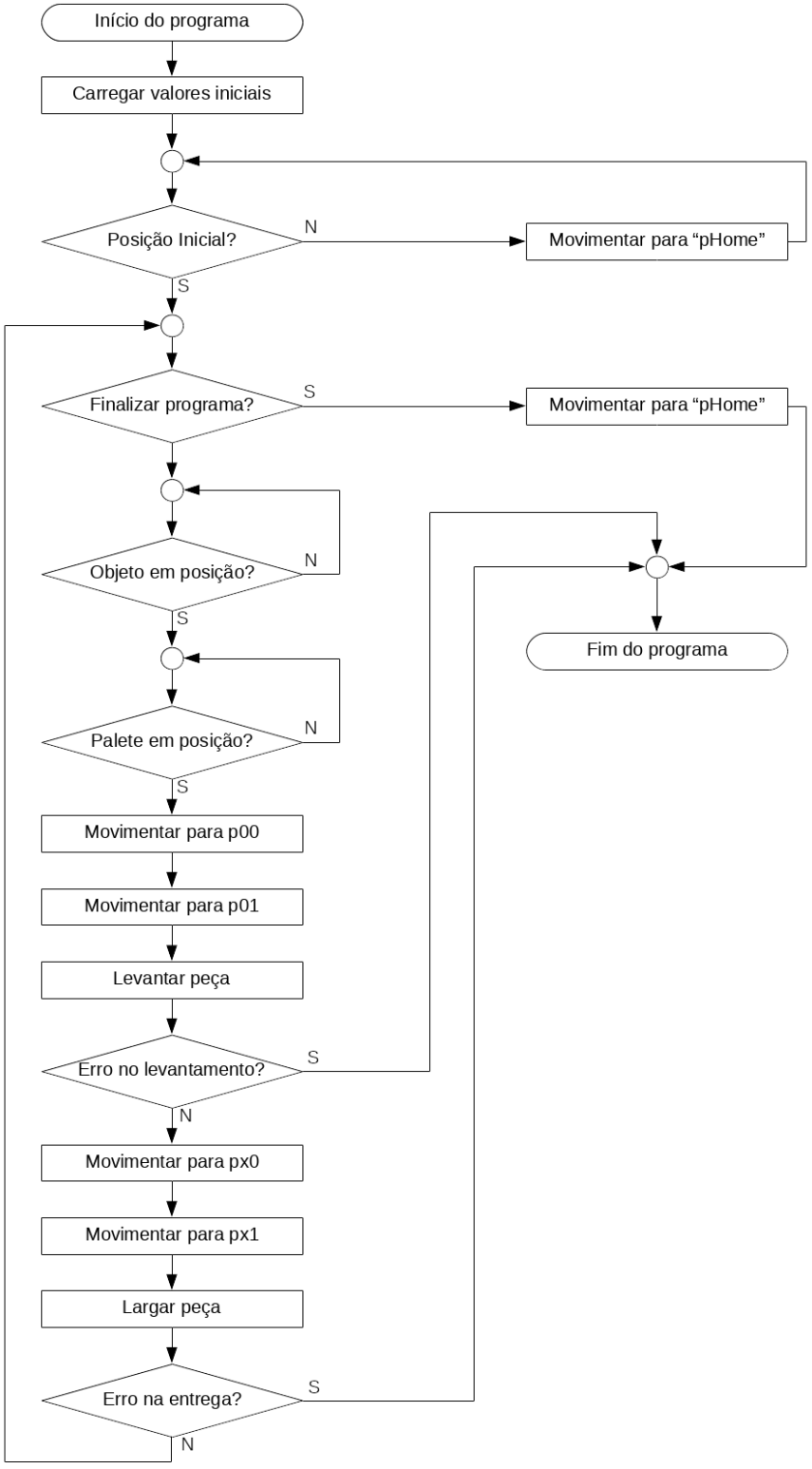


Figura 32 Fluxograma da execução do programa principal.

O fluxograma apresentado resume a execução do programa principal e, para tal, ilustra apenas as etapas principais da dita execução. É de referir que uma observação geral e direta do diagrama de fluxo permite depreender que à leitura dos valores iniciais se seguirá um subciclo; este subciclo movimentará o robô para as posições determinadas e, consequentemente, realocará os objetos.

Nos processos de levantamento e entrega das peças encontram-se previstas duas posições por cada ponto. A posição preliminar (de aproximação) tem como propósito permitir que o robô se movimente de forma mais expedita, sendo colocada diretamente por cima da posição efetiva. A posição final (ou efetiva) facilita a colocação ou levamento da peça, de forma mais precisa embora com menor velocidade, num movimento linear na direção vertical. Exemplos destas posições encontram-se ilustrados na Figura 33 (juntamente com a posição pHome que indica o ponto inicial/final da movimentação do robô).

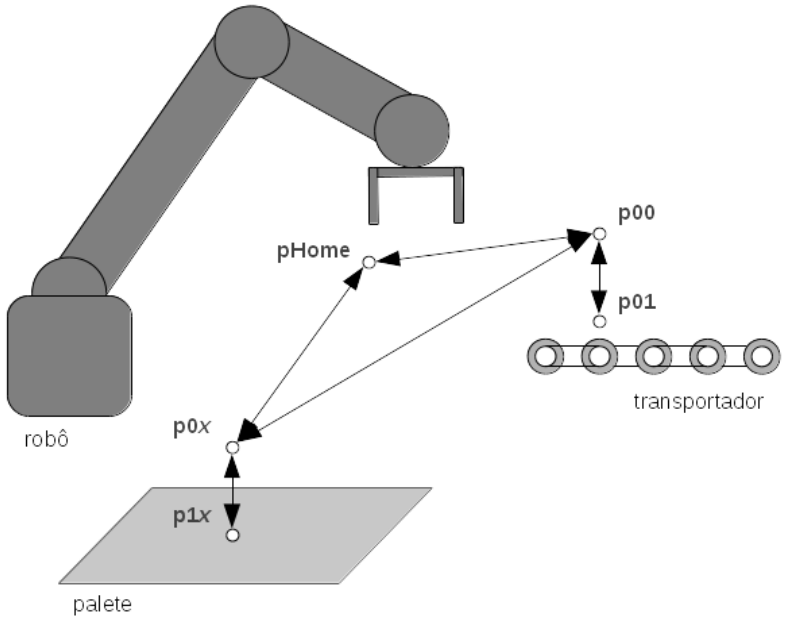


Figura 33 Ilustração esquemática do processo.

As posições p00 e p01 correspondem às posições intermédia e final, respetivamente, no local de levantamento do objeto, representado aqui por um transportador. No ponto de entrega, representado na Figura 33 por uma palete, as posições variam consoante o número e a posição pretendida dos objetos a paletizar. Daí, as posições encontrarem-se identificados como p*x*0 e p*x*1, respetivamente posição de aproximação e final, onde *x* é um número inteiro superior a 0.

Como já mencionado, as operações típicas de paletização ocorrem em ambiente industrial e envolvem o empilhamento de caixas de produtos para transporte. Perante isto, torna-se clara a escolha (para a simulação do programa ARP em ambiente RobotStudio) de um transportador de rolos como ponto de levantamento dos objetos e de uma palete como ponto de entrega. De forma a facilitar e flexibilizar a programação, foram criados dois *work objects* que representam as posições e orientações das áreas de levantamento e entrega, designados, respetivamente, por *wobj_conveyor* e *wobj_pallet.* A estes *work objects* serão referenciadas as posições e orientações finais dos objetos a paletizar ou a levantar.

Na Figura 34 é possível ver os *work objects* e as suas orientações. Note-se que os diversos sistemas de coordenadas têm o eixo dos *xx*' ilustrado a vermelho, o eixo dos *yy*' a verde e o dos *zz*' a azul. Ainda, na mesma figura encontra-se representado o sistema de coordenadas da ferramenta (*tGripper)*, o qual se encontra definido, com a mesma orientação que o sistema de coordenas do punho.

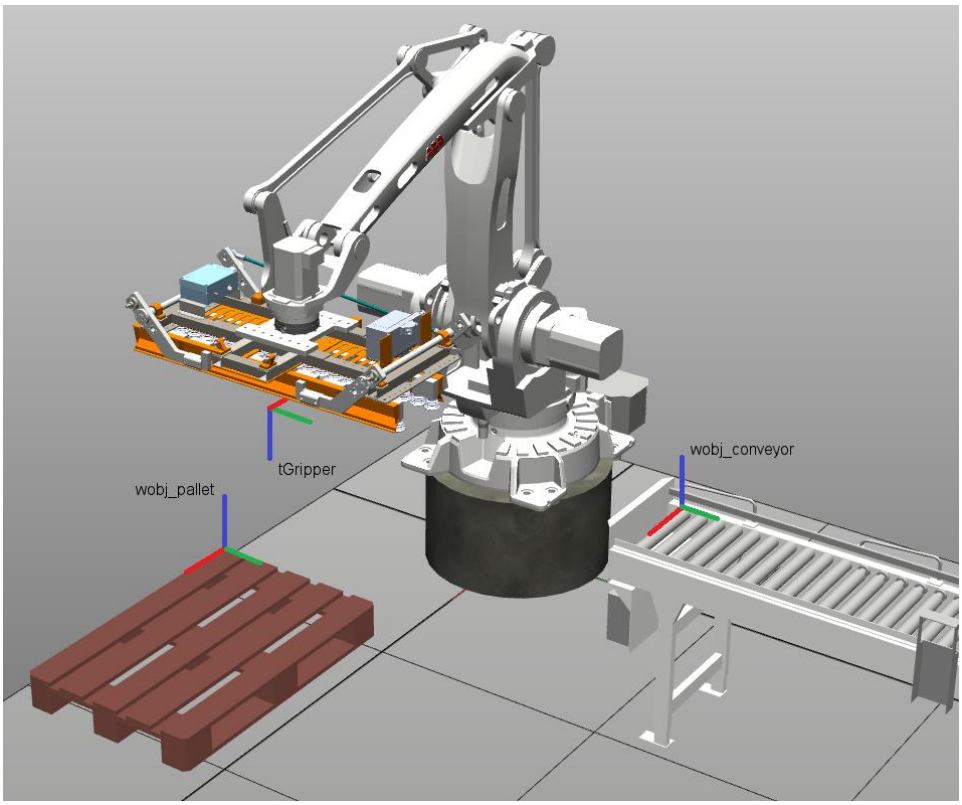


Figura 34 Posicionamento dos *work objects* em ambiente RobotStudio.

O número máximo de possíveis posições de entrega de peças na palete é condicionado pelas áreas das próprias paletes e pelas dimensões máximas da peça. Após o cálculo ou introdução (através dos valores iniciais) das posições das peças na palete, as mesmas são dispostas num *array* (*psPalPos*) para tornar mais fácil o seu armazenamento e posterior manipulação. Analogamente, foi criado um *array* suplementar (*orPalPos*) que contém as orientações das peças correspondentes aos pontos, representadas em quaterniões. As posições preliminares não são guardadas de igual forma, dado que são facilmente calculadas a partir das posições finais, subtraindo um valor incluído nos valores iniciais (ao eixo dos *zz*').

O procedimento principal do programa executará um ciclo que tem como número de iterações o número total de objetos a paletizar. Este ciclo consiste em ler um valor do *array* de posições (de acordo com a iteração), calcular o *Target* correspondente para ambas as posições, final e intermédia, e movimentar o robô para as posições compassadamente. Antes de qualquer movimentação é verificado se o robô consegue atingir o ponto com a configuração pré-determinada. Se não for possível, o programa alterará a configuração dos eixos (*cfx*) e tentará de novo [28]. Este processo pode ser repetido até quatro vezes (dadas as restrições por parte do RobotStudio). Findo este número de tentativas, e caso não tenha sido possível determinar uma configuração aceitável, será gerado um erro e o programa abortado.

Os erros são divididos em dois tipos: os passíveis de serem previstos e os imprevistos (ou de sistema). Durante a execução do programa desenvolvido, este procurará, em algumas etapas, verificar se existiu algum erro e em caso afirmativo, mostrará no painel uma mensagem indicativa e terminará o programa. Para isso, foram contempladas algumas funções que têm como objetivo evitar erros por parte do utilizador. É disto exemplo, a função de verificação do tamanho do objeto a paletizar. No caso de surgir um erro não contemplado no programa principal, este será detetado pelo próprio RobotStudio, que exibirá a mensagem correspondente e executará uma ação pré-determinada pelo fabricante. Exemplos deste caso são as *"corner path failures"* que indicam que não foi possível ao robô atingir a posição desejada dentro do tamanho definido para a zona de aproximação.

Durante a execução do ciclo principal, o programa poderá ser interrompido para atendimento a interrupções geradas pela modificação de estado das entradas. Estas interrupções visam notificar o programa da modificação de alguma condição externa ao IRC, quer esta modificação tenha sido originada pelo controlador de processo, pelo utilizador ou por um fator externo (por exemplo, um sensor). Analogamente, em determinadas alturas durante o processo, as saídas digitais serão ativadas/desativadas.

No geral, o programa executará as movimentações e posicionamentos necessários para concretizar uma palete completa. No final, a saída correspondente será ativada de forma a notificar o controlador que uma palete se encontra concluída e que aguarda nova palete, reiniciando todo o processo de paletização após ter recebido os sinais necessários para o efeito. O programa principal continuará este processo indefinidamente até que seja dada ordem de finalização por parte do controlador, na respetiva interface, independentemente do momento do processo em que o robô se encontre.

4.3.1. LEITURA DOS VALORES INICIAIS

De forma a minimizar a interação do utilizador com o programa e, tendo em vista possíveis futuros desenvolvimentos, foi criado um modelo de ficheiro de inicialização em formato XML, apelidado de "ARPInit". A escolha deste formato prendeu-se com as propriedades desta linguagem, como o facto de ser inteligível quer para máquinas quer para utilizadores humanos, ser um formato aceite em diversas plataformas e a sua simplicidade e flexibilidade [30].

Apesar de manter afinidades com os ficheiros do mesmo tipo existentes no RobotStudio, o ficheiro XML de inicializações referido foi criado em específico para o programa "ARP".

Tratando-se, fundamentalmente, de um *parser* (analisador de sintaxe), a função "*ReadInitValues*" foi desenvolvida em RAPID e encontra-se no módulo de sistema "ARPBasics", sendo chamada durante a rotina de inicializações. Através desta função são inicializadas diversas variáveis fornecidas pelo utilizador.

As inicializações encontram-se descritas conforme as regras da linguagem XML. Ou seja, sob um formato hierarquizado tendo como níveis superiores os seguintes:

- "*System*" – onde se encontram dados gerais relativos ao próprio sistema (por exemplo, a posição "*Home*", altura máxima atingível pelo robô "*Max_Height*" ou os valores da posição do TCP);

- *"Object"* – apresenta os valores de posição, orientação e tamanho dos objetos (por exemplo, a palete ou a caixa);

- *"Layout"* – contém subníveis que inicializam parâmetros relativos ao mosaico de paletização (por exemplo, orientação das camadas pares e ímpares ou as posições e orientações das caixas na palete).

O último grupo possibilita a interface a um programa externo para a realização de mosaico de paletização, dado que contém as possíveis inicializações dos *arrays* de posições e orientações com os quais são criados os *Targets* do robô. Apesar de, presentemente, limitado a dez entradas, o grupo pode ser aumentado adicionando mais entradas sob o mesmo formato das presentes, visto que o *parser* se encontra preparado para ler um número indeterminado de posições/orientações dos objetos. O seguinte extrato de código exemplifica o formato referido, exibindo parte de dois níveis superiores e respetivos subníveis.

```
  <System>
    <Home_Position>
      <trans x="300" y="0" z="300" />
      <rot q1="0" q2="0" q3="1" q4="0" />
    </Home_Position>
       <Max_Height>
      <val num="712" />
    </Max_Height>
    <Int_Dist>
      <val num="80" />
    </Int_Dist>
  </System>
  <Object>
    <Box>
      <size x="80" y="160" z="80" />
      <trans x="100" y="-700" z="125" />
      <rot q1="1" q2="0" q3="0" q4="0" />
    </Box>
...
  </Object>
```

4.3.2. CRIAÇÃO DO MOSAICO

Para a colocação dos itens sobre a palete foi necessário escolher uma forma de empilhar os mesmos. Como abordado no ponto 2.2.3 (Mosaicos), os estilos de empilhamento mais comuns são o sobreposto (*overlap*) e o entrelaçado (*interlaced*), apresentando, o primeiro, os objetos empilhados com o mesmo estilo (posição/orientação) e, o segundo, alterando a orientação consoante a camada da pilha. Sendo um estilo mais fácil para a contagem do número de objetos, para ser manuseado e programado, o estilo sobreposto foi, inicialmente, o estilo escolhido. Posteriormente, foi adicionado ao programa o segundo estilo (entrelaçado). Este estilo tem a vantagem, sobre o primeiro, de ser mais estável em termos de estrutura física, contudo é de mais difícil programação.

Assumindo um tamanho constante dos bens a paletizar, e sabendo as dimensões da área a preencher e do próprio bem, foram calculadas as restantes posições dos bens a serem colocados numa camada. As posições – referenciadas a um ponto fixo – foram determinadas por meio de fórmulas matemáticas, relativamente a esse ponto, conforme descrito abaixo.

A orientação das camadas foi deixada como parâmetro de entrada (no ficheiro de inicializações), tornando desta forma possível ao utilizador realizar ambos os estilos perante quatro orientações possíveis (0°, 90°, 180° e 270°). As diversas orientações possibilitam, caso os objetos tenham uma marcação ou etiqueta de um dos lados, que esta fique para fora, por exemplo. Por questões de simplificação da programação, as orientações definidas foram agrupadas em pares: 0°/180° e 90°/270 °. O critério torna-se evidente quando se toma em consideração as dimensões e a forma dos objetos a paletizar que, normalmente, são paralelepipédicas ou cúbicas.

Para o cálculo destas orientações foi definida uma posição inicial (0°) – que se encontra representada na primeira linha da Figura 35 – a partir da qual foram determinadas as restantes posições rodando o objeto, no destino (*wobj_pallet*), sucessivamente 90°. É assumido que o objeto, no ponto de levantamento, se mantenha com posição e orientação constante na altura de *pickup*. Da mesma fora, é possível verificar na Figura 35 que a orientação do objeto, no ponto de entrega, será dada pela rotação da ferramenta (respetivamente, pelo *tool coordinate system*), em torno do seu eixo dos *zz*'. Independentemente da localização em que os *work objects conveyor* e *pallet* se encontrem ou da orientação que tenham, estas relações valor ângulo/orientação sistema de coordenadas, manter-se-ão fixas para o programa.

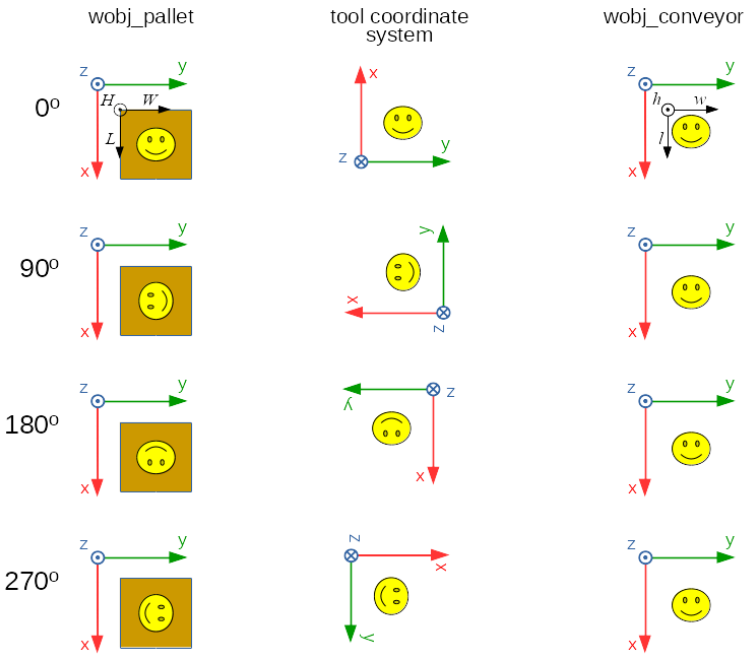


Figura 35 Representação das orientações dos diversos eixos

Ainda na Figura 35 pode ver-se a forma como foram consideradas as dimensões dos objetos, mediante a orientação dos seus *work objects*. Ou seja, independentemente da orientação dos *work objects*, as dimensões permanecerão associadas a um eixo no seu sistema de coordenadas. Tomando o objeto a paletizar como exemplo, verifica-se que o comprimento (*l*) foi associado ao eixo dos *xx*', a largura (*w*) ao eixo dos *yy*' e a altura *(h)* ao eixo dos *zz*'. Estas associações são em tudo equivalente às da palete ($L \times W \times H$). Um resumo destas dimensões está visível na Tabela 8, apresentada em baixo.

Tabela 8 Variáveis das dimensões

	Largura	**Comprimento**	**Altura**
Palete	L	W	H
Objeto	l	w	h
Pilha	L_{stack}	W_{stack}	H_{stack}

Para a criação de uma pilha completa de objetos sobre uma palete torna-se necessário conhecer também a altura máxima alcançável pelo robô (H_{max}) e a diferença (distância) entre o ponto intermédio e o final (D_{int}). Obtendo estes valores é possível calcular o número inteiro máximo de camadas a perfazer (N) segundo a seguinte fórmula:

$$N = trunc\left(\frac{H_{max} - H - D_{int}}{h}\right) \quad (1)$$

De notar que o número inteiro (N) é obtido recorrendo à truncatura do mesmo a zero casas decimais. A altura efetiva da pilha será então obtida por:

$$H_{stack} = (N * h) + H \tag{2}$$

4.3.3. CAMADAS

A orientação que cada camada tomará é essencial para a determinação das diversas posições dos objetos nessas camadas. Na Figura 36 estão esquematizadas as duas possíveis orientações que poderão ser executadas pelo programa desenvolvido – à esquerda 0°/180° e à direita 90°/270°. De forma a facilitar a exemplificação das mesmas, foi escolhida a forma retangular para o objeto a paletizar. Os objetos encontram-se numerados pela ordem em que as suas posições serão calculadas e posteriormente inseridas no *array* correspondente.

Para a elaboração de uma camada serão calculados, inicialmente, os limites necessários por eixo, isto é, o número de itens e o deslocamento (*offset*) de cada objeto. Neste caso, por deslocamento entende-se a distância entre os objetos e a origem do referencial, *wobj_pallet*, a partir do qual se consideram as medidas da palete.

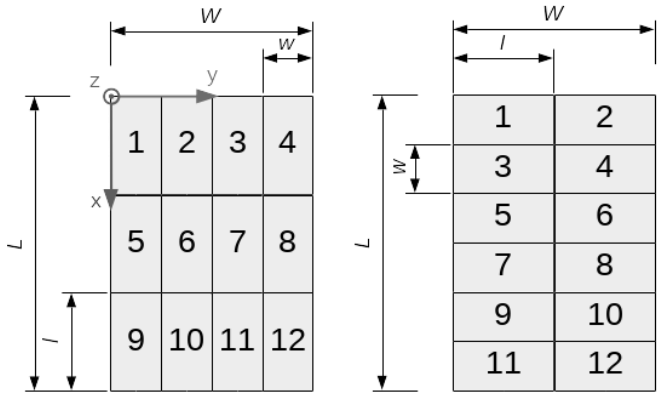


Figura 36 Exemplos de orientações de camadas sem deslocamento (caso ideal)

Quando a soma das dimensões dos vários objetos, ao longo de um eixo, não coincidir com a dimensão da palete, nesse eixo, serão feitos ajustes ao posicionamento dos objetos. A distribuição dos itens na palete pode ser feita de duas formas, designadamente, por (o que neste trabalho foi apelidado de) distribuição uniformizada ou não uniformizada. No primeiro caso, após a obtenção do número inteiro máximo de objetos, o espaço restante será dividido em duas partes iguais e posicionado nas extremidades, entre os objetos e os limites da palete. Na distribuição não uniformizada, o espaço restante será colocado nas extremidades e entre os itens de forma equitativa. A forma de distribuição é selecionável nos valores iniciais.

Assim:

$$N_x = trunc\left(\frac{L}{l}\right) \tag{3}$$

$$N_y = trunc\left(\frac{W}{w}\right) \tag{4}$$

$$O_x = \frac{L - (N_x * l)}{N_x + 1} \tag{5}$$

$$O_y = \frac{W - (N_y * w)}{N_y + 1} \tag{6}$$

onde:

Tabela 9 Variáveis dos limites

	Nº Itens	***Offset***
Eixo *x*	N_x	O_x
Eixo *y*	N_y	O_y

Se salientar que, caso a distribuição seja uniformizada o denominador das equações (5) e (6) será "2". De notar, ainda, que as fórmulas aqui descritas apresentam apenas o caso 0°/180°, ou seja, o caso presente do lado esquerdo da Figura 36.

A posição de cada caixa pode, então, ser individualmente calculada segundo as seguintes fórmulas:

$$Pos_x = n_{aux} * O_x + \left((n_x - 1) * l\right) \tag{7}$$

$$Pos_y = n_{aux} * O_y + \left(\left(n_y - 1\right) * w\right) \tag{8}$$

$$Pos_z = H + \left((n_z - 1) * h\right) \tag{9}$$

onde $Pos_{x/y/z}$ é a posição do objeto por eixo e $n_{x/y/z}$ a iteração por eixo. Pos_x é obtida fazendo n_x variar de um até ao número limite de itens do eixo (N_x). Da mesma forma, n_y varia de um até N_y, sendo incrementada quando n_x atingir o seu valor máximo, indicando desta forma a conclusão de uma linha ou coluna. Para a determinação de Pos_z, a iteração n_z é incrementada após n_x e n_y terem atingido os seus valores máximos, o que significa que uma camada se encontra concluída.

Caso a distribuição seja uniformizada, n_{aux} será igual a "1", caso contrário, será igual ao respetivo número da iteração do eixo (n_x ou n_y, conforme o caso). Este processo é utilizado para as posições dos objetos nos *arrays* de posições e orientações (*psPalPos e orPalPos*).

Assim, a posição absoluta do objeto pode ser obtida por:

$$P_{objecto} = \left[Pos_x, Pos_y, Pos_z\right] \tag{10}$$

O segundo caso, isto é, o caso apresentado do lado direito da Figura 36, é em tudo análogo ao descrito, sendo, no entanto, necessário, para o cálculo dos limites e posições finais, trocar as variáveis correspondente às dimensões do objeto (*l* por *w* e vice-versa).

Para a criação da camada, é também imperativo determinar a orientação dos objetos, sendo esta estabelecida pela ferramenta, aquando da deposição na palete. Para o RobotStudio, a forma mais comum da representação de uma orientação é sob a forma de quaternião, porque se manifesta relevante para a criação de *Targets*, numa fase posterior. Assim, segundo Raković *et al*. [31], uma matriz rotacional pode ser convertida usando o seguinte procedimento[7]:

Seja **T** uma matriz rotacional, onde:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \tag{11}$$

A sua representação em quaternião (q_1, q_2, q_3, q_4) é dada pelo seguinte algoritmo:

$$Se\ (t_{11} + t_{22} + t_{33}) > 0$$
$$\begin{cases} s = 2\sqrt{(t_{11} + t_{22} + t_{33} + 1)} \\ q_1 = 0.25 * s \\ q_2 = t_{32} - t_{23}/s \\ q_3 = t_{13} - t_{31}/s \\ q_4 = t_{21} - t_{12}/s \end{cases} \tag{12}$$

[7] Durante a elaboração do trabalho foi necessário recorrer a estes métodos de conversão de matrizes rotacionais em quaternião, para calcular as orientações dos objetos nas camadas. A função foi implementada no programa em RAPID e foram feitos vários testes para comprovar que a função se encontrava a funcionar. No entanto, a mesma não é utilizada no decorrer do programa, dado não se ter revelado necessária.

$$Senão\ se\ (t_{11} > t_{22})\ e\ (t_{11} + t_{33}) > 0$$

$$\begin{cases} s = 2\sqrt{(t_{11} - t_{22} - t_{33} + 1)} \\ q_1 = t_{32} - t_{23}/s \\ q_2 = 0.25 * s \\ q_3 = t_{12} - t_{21}/s \\ q_4 = t_{13} - t_{31}/s \end{cases} \quad (13)$$

$$Senão\ se\ (t_{22} > t_{33})$$

$$\begin{cases} s = 2\sqrt{(t_{22} - t_{11} - t_{33} + 1)} \\ q_1 = t_{13} - t_{31}/s \\ q_2 = t_{21} + t_{12}/s \\ q_3 = 0.25 * s \\ q_4 = t_{32} + t_{23}/s \end{cases} \quad (14)$$

$$Senão$$

$$\begin{cases} s = 2\sqrt{(t_{33} - t_{11} - t_{22} + 1)} \\ q_1 = t_{21} - t_{12}/s \\ q_2 = t_{13} + t_{31}/s \\ q_3 = t_{32} + t_{23}/s \\ q_4 = 0.25 * s \end{cases} \quad (15)$$

Aplicando o algoritmo ao contexto apresentado na Figura 35, para a obtenção da camada, e tomando novamente como base a linha que corresponde a uma rotação de 0°, verifica-se que os eixos do *wobj_conveyor* e o sistema de coordenadas da ferramenta estão relacionados da seguinte forma:

$$\begin{cases} x_{tool} = -x_{conveyor} = (-1,0,0) \\ y_{tool} = y_{conveyor} = (0,1,0) \\ z_{tool} = -z_{conveyor} = (0,0,-1) \end{cases} \quad (16)$$

O que equivale à matriz rotacional:

$$\mathbf{T} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{17}$$

Ao empregar o método previamente descrito, verifica-se que este caso recai sobre a equação (14) e, aplicando a mesma, que se obtém a seguinte representação em quaternião:

$$\begin{cases} q_1 = 0 \\ q_2 = 0 \\ q_3 = 1 \\ q_4 = 0 \end{cases} \tag{18}$$

A totalidade das orientações foram calculadas aplicando o método, acabado de descrever, aos restantes casos apresentados na Figura 35.

De forma semelhante, é possível determinar a configuração dos eixos do robô. Assumindo como ponto de partida os 0°, o valor da configuração do eixo 6 (cf_6) será 0 para 0°, ou seja, não apresentará rotação relativamente ao *Tool Frame*. Para 90° e 180° os valores das configurações serão de 1 e –1, respetivamente, que corresponderão ao quadrante em que se encontram após a rotação (0° a 90° e 0° a –90°) [28]. Finalmente, pela mesma lógica, uma rotação de 270° corresponderá ao quadrante 2. Os restantes valores de configuração dos eixos ficaram a 0 dado o robô selecionado ser de quatro eixos.

A Tabela 10 apresenta um resumo das orientações dos objetos e configurações dos eixos determinadas.

Tabela 10 Orientações dos objetos e configuração dos eixos

	Orientação objetos	**Configuração eixos**
0°	[0, 0, 1, 0]	[0, 0, 0, 0]
90°	[0, –0.707107, 0.707107, 0]	[0, 0, –1, 0]
180°	[0, 1, 0, 0]	[0, 0, 2, 0]
270°	[0, 0.707107, 0.707107, 0]	[0, 0, 1, 0]

4.3.4. INTERFACE COM O AUTÓMATO

Para a interface com o autómato (ou PLC) de controlo da célula foram definidos os sinais necessários para informar o estado do robô ao PLC (e vice-versa) e definidos os sinais de *hand-shake* entre os dois controladores, os quais indicam pedidos/presença dos objetos, entre outros. De forma idêntica, para algumas funções da ferramenta, foram também contemplados sinais para operar a válvula de vácuo ou o *feedback* quando o objeto se encontra alojado na ferramenta. A Tabela 11 apresenta um sumário dos sinais considerados.

Tabela 11 Resumo das entradas/saídas do robô

Tipo	**Sinal**	**Variável Interna**	**Descrição**
Entrada	di1	*diCycleStart*	Início do ciclo de paletização
Entrada	di2	*diCycleRestart*	Reinicialização do ciclo de paletização
Entrada	di3	*diCycleStop*	Finalização do ciclo de paletização
Entrada	di4	*diBoxReady*	Item encontra-se à disposição para ser recolhido
Entrada	di5	*diPalletReady*	Palete encontra-se pronta para receber itens
Entrada	di6	*diObjectSensor*	Sensor de deteção da presença de objeto na ferramenta – opcional
Saída	do1	*doRobotReady*	Robô pronto
Saída	do2	*doVacuumOn*	Ligar vácuo
Saída	do3	*doVacuumOff*	Desligar vácuo
Saída	do4	*doRequestBox*	Pedido de item para paletizar
Saída	do5	*doRequestPallet*	Pedido de nova palete

O RobotStudio permite ainda a criação de ficheiros de configuração ".cfg" que gravam, entre outros parâmetros, os sinais resumidos na Tabela 11. Assim, foi criado um ficheiro ("EIO.cfg"), segundo as especificações da ABB, para evitar a tarefa de ter que se definir, a cada projeto novo, os sinais na configuração, ficando apenas a ligação à unidade física ou virtual das entradas/saídas como procedimento manual. Estes sinais serão, posteriormente, ligados – no programa ARP – à respetiva variável interna para mais fácil manipulação. Na Figura 37 e na Figura 38 encontram-se ilustradas secções distintas do mesmo fluxograma E/S (Entradas/Saídas) que pretende demonstrar as etapas dos vários ciclos do programa principal e dos estados das correspondentes entradas e saídas definidas para cada etapa. É de notar que as figuras se encontram interligadas. Tomando como exemplo a movimentação para a posição pHome por parte do robô (Figura 37) verifica-se que, após essa movimentação, o passo seguinte encontrar-se-á na Figura 38 (salto 1).

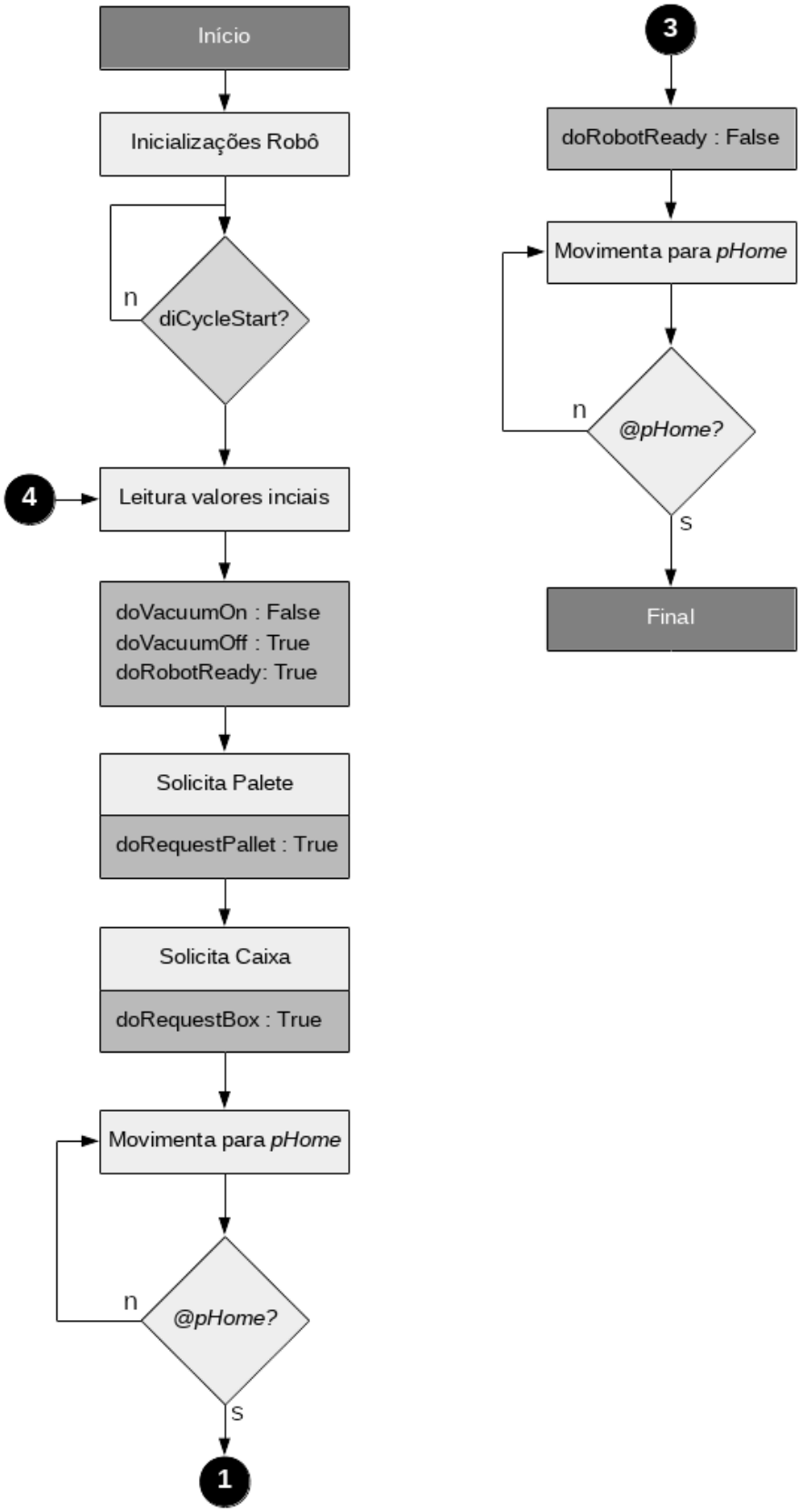


Figura 37 Secções do início e final do ciclo principal do fluxograma E/S programa ARP

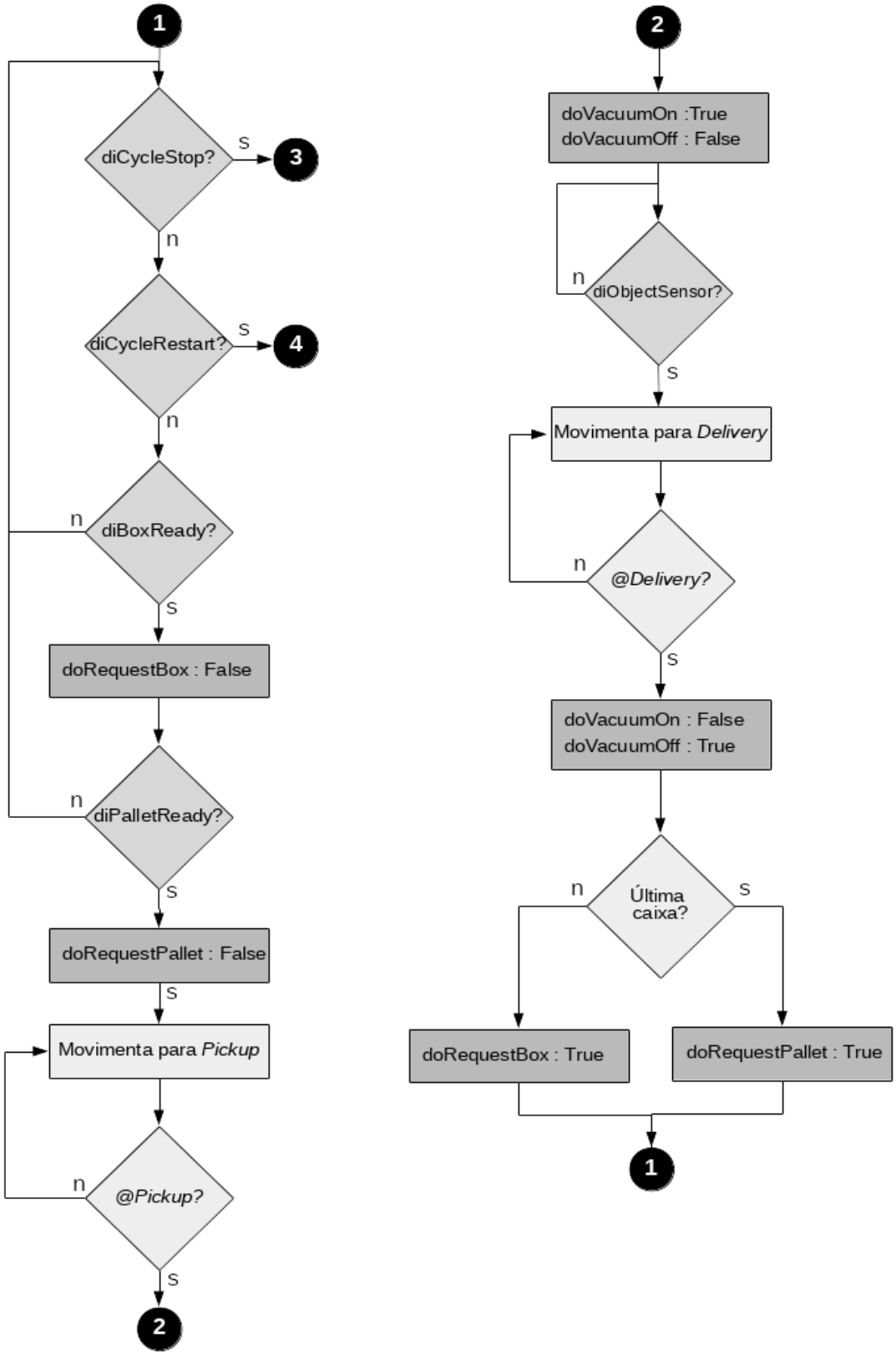


Figura 38 Secções dos ciclos principais de controlo do fluxograma de E/S do programa ARP

O desenvolvimento, aqui descrito, do programa ARP foi executado em concordância com os manuais e linhas gerais de desenvolvimento de *software* colocados à disposição pelo fabricante do equipamento, a ABB. Após a escrita dos módulos do programa, recorreu-se ao verificador de sintaxe do RobotStudio para a averiguação de possíveis erros de escrita. Estabelecido que o programa se encontrava sem erros de sintaxe e passível de ser compilado, foram executados os testes das diversas funções constituintes do programa, individualmente. O passo seguinte traduziu-se nos testes e ensaios do programa como uma unidade, em ambiente virtual, contando com os modelos do manipulador e controlador do robô, assim como o ambiente da célula.

O capítulo seguinte apresenta, de forma resumida, esses mesmos testes e a metodologia empregue nos mesmos, assim como uma análise dos resultados obtidos.

.

5. TESTES E ANÁLISE DE RESULTADOS

O presente capítulo tem como finalidade explanar todas as simulações, ensaios e posteriores alterações efetuadas durante a última etapa do trabalho. Após o desenvolvimento do programa ARP, recorreu-se ao simulador presente no ABB RobotStudio para a validação do aludido programa. Para tornar a simulação o mais realista e correta possível, foram conduzidos múltiplos testes, os quais requereram etapas e definições adicionais. Com a experiência adquirida nas simulações foi possível, com os necessários ajustes ao ficheiro de inicializações, correr o programa desenvolvido em simulação num robô real.

5.1. METODOLOGIA E PARÂMETROS

A versão do RobotStudio selecionada para proceder às simulações foi a mais recente, à data da elaboração do trabalho, e que se encontrava disponível no site da ABB. Esta versão – RobotStudio 6.04 – fez-se acompanhar pela mesma versão do controlador do robô (RobotWare).

Necessária para a simulação (devido a conter o controlador virtual), a criação da célula robótica, cujas características foram abordadas no Capítulo 1, assumiu-se como a primeira etapa para os ensaios.

A célula encontra-se ilustrada na Figura 39, onde se destacam os pontos de levantamento e a zona de entrega, que no caso são a tela transportadora e palete, respetivamente.

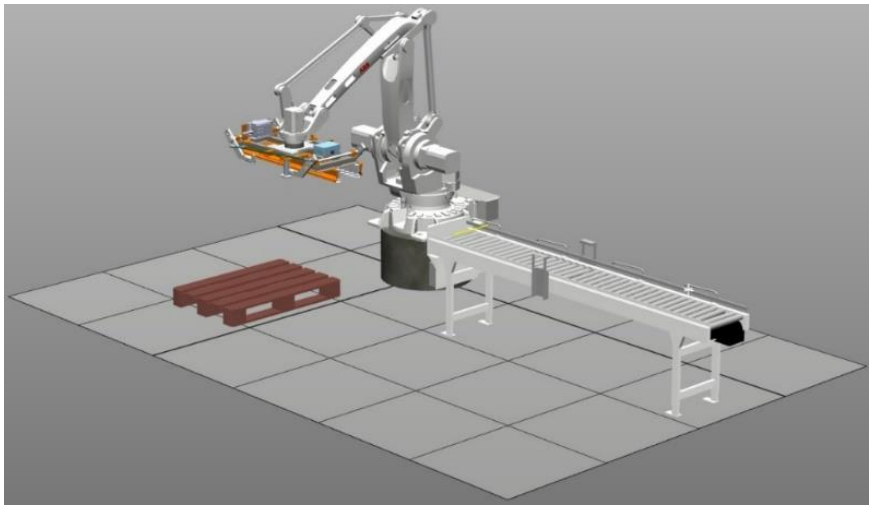


Figura 39 Célula robótica desenvolvida para simulação

Ainda na mesma figura é possível verificar que o robô se encontra dotado de uma ferramenta de vácuo da ABB (“FlexGripper”) e que este foi colocado em cima de um pedestal. Esta decisão teve como base a Figura 40, exibida na ficha técnica do robô IRB460. De forma a tentar utilizar a altura máxima alcançável pelo robô (para a criação da pilha), este foi elevado em 452 mm na sua base. Horizontalmente, as distâncias ao *World Frame,* ou seja, ao ponto central da base do robô foram, também, levadas em conta durante o posicionamento da palete e tela transportadora.

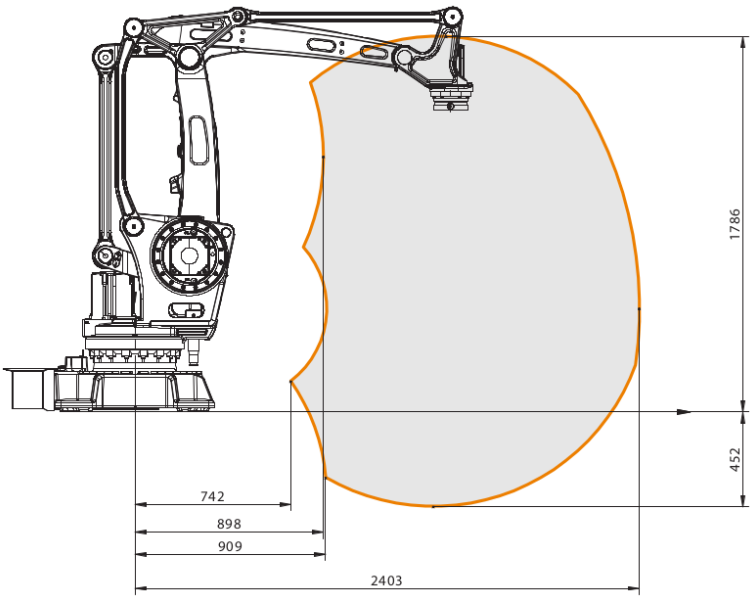


Figura 40 Corte da representação do volume de trabalho do robô IRB460 [24]

Os diversos módulos do programa ARP foram criados num editor de texto, em RAPID, e posteriormente importados para o RobotStudio, onde foram objeto de verificação de sintaxe, tendo a mesma sido corrigida nos pontos devidos. Após este processo e após a verificação das funções constituintes, os ficheiros foram então carregados para o controlador virtual.

Seguindo-se os testes ao código, estes centraram-se no processo de utilização dos *targets* e na movimentação do robô, ou seja, no módulo de movimentação do programa "ARPMovement.mod". Numa fase inicial, foi delimitada a movimentação do robô apenas a deslocações entre o ponto de levantamento e o de entrega, fazendo uso da verificação do alcance do robô. Apesar de ter sido ativada e configurada, o RobotStudio não permitiu a simulação das rotinas de supervisão do movimento ("Motion Supervision"), que salvaguarda possíveis colisões.

O programa fez uso de dois tipos de movimentação: a de interpolação nas juntas, para os movimentos de aproximação menos exigentes, e de interpolação linear, para os de precisão. No primeiro caso, a velocidade escolhida foi a v5000, ou seja, uma velocidade do TCP de 5000 mm/s [28]. Esta foi escolhida de entre o leque disponibilizado pela ABB, atendendo a uma presunção inicial de mil unidades paletizadas por hora. A zona de aproximação selecionada, para este caso, foi a z15. Isto significa que os eixos terão de se encontrar a cerca de 15 mm da posição desejada antes de se avançar para a próxima posição. Para os movimentos cuja meta é a posição final, a velocidade foi reduzida para 1000 mm/s (v1000) e a zona para o valor *fine*. Utilizando estas velocidades, foi possível, em teste ulteriores, atingir as dezoito unidades por minuto (aproximadamente), perfazendo mil e oitenta itens por hora.

A etapa seguinte focou-se na geração de mosaicos, fazendo uso das funções desenvolvidas para o efeito no módulo "ARPBasics.sys". Os parâmetros de entradas das funções foram inseridos de forma manual e as respetivas saídas controladas. Isto é, foram introduzidas diversas dimensões de paletes e de objetos e verificado se as funções geravam o número de elementos, limites, as posições e orientações, corretamente, assim como se produziam o preenchimento dos respetivos *arrays*.

Assentes os elementos centrais das operações com o robô, a fase subsequente passou por criar a lógica comportamental necessária para o RobotStudio correr a simulação, continuamente.

Para simular alguns dos comportamentos verificados no mundo físico, utilizaram-se modelos dos mesmos como, por exemplo, o movimento da tela que transporta os objetos. Este movimento pretendeu simular a cadência com que as caixas são entregues ao robô pela tela, durante o normal funcionamento. A velocidade escolhida para as caixas foi de 1150 mm/segundo, de forma a que uma caixa se encontrasse disponível, no ponto de levantamento, para o robô, a cada dois segundos (limite consideravelmente mais alto do que o necessário para as mil unidades por hora).

Tendo sido alterado diversas vezes durante as descritas simulações, o módulo principal "ARPPalletizer.mod" assumiu a sua versão final nos testes globais, nos quais o objetivo passava pela interação entre todos os constituintes do programa. Nesta fase foi também possível confirmar que as rotinas de atendimento às entradas (*Trap*) se encontravam em funcionamento e atuavam sobre as respetivas variáveis. De forma análoga, foram testados os sinais de saída do controlador.

A simulação do código teve como etapa final os ensaios à leitura do ficheiro XML dos valores iniciais e correspondente atribuição dos mesmos às variáveis internas. A aplicação foi testada, na totalidade, recorrendo a diversas simulações, com diferentes tamanhos de caixas e de paletes para garantir que o programa reagiria de forma adequada aos diversos cenários. As simulações foram divididas em ensaios, consoante a variação das dimensões de entrada. No início ou durante a execução das mesmas, foram também alterados diversos parâmetros, como a orientação dos objetos ou o tipo de mosaico, para analisar o comportamento do programa a variações. Para além dos valores necessários para o normal funcionamento da paletização, foi também simulada a atribuição das posições dos objetos na palete, sem fazer uso das funções internas do programa ARP, recorrendo às entradas designadas para o efeito, no ficheiro XML.

5.2. LÓGICA DA SIMULAÇÃO

A lógica da simulação (por vezes apelidada de lógica da estação) serve para a reprodução do comportamento dos diversos intervenientes da célula robótica. O RobotStudio permite a programação de lógica comportamental para estes mesmos intervenientes que se traduz em possíveis eventos e ações que, durante a simulação, possibilitam uma melhor análise da célula robótica, nos casos em estudo.

Durante a execução das simulações, revelou-se necessário a introdução da dita lógica em diversas situações como, por exemplo, perante a presença de um objeto no ponto de levantamento, que necessita de ser reportado ao controlador. Por este motivo, foi introduzido um sensor, na tela transportadora.

Não sendo possível simular o controlador da célula robótica (autómato) no RobotStudio, surgiu a necessidade de recorrer a alguns outros sinais para simular o comportamento esperado por parte do autómato. De salientar que, nas simulações iniciais (ao código), estes sinais eram gerados de forma manual pelo utilizador. Posteriormente, este processo manual foi substituído por um conjunto de ligações de sinais, entre os blocos, na lógica da estação. A Figura 41 apresenta o diagrama da estrutura criada. Na mesma, é possível identificar os diferentes blocos que correspondem a componentes físicos com lógica associada ("IRB_460", "LineSensor"), as ações ou eventos ("Source", "Attacher", "Detacher", "LinearMove", "Queue", "SimulationEvents") e as operações binárias ("NOT", "AND"). As propriedades dos blocos (objetos) e as respetivas ligações (*bindings*) foram omitidas para facilitar a leitura e compreensão do diagrama.

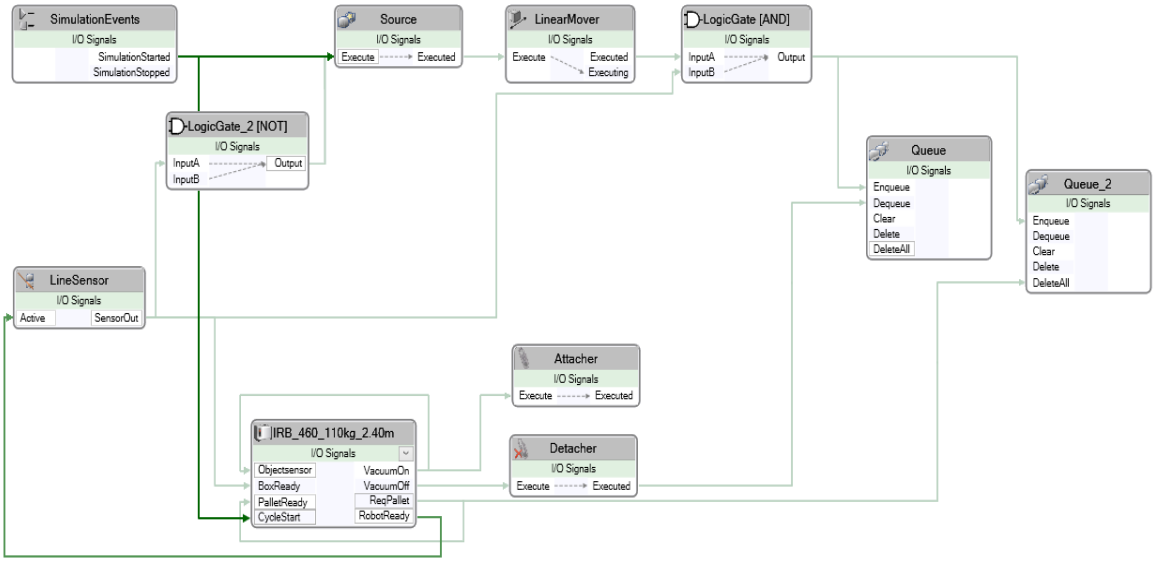


Figura 41 Esquema da lógica da estação

Ainda da leitura da Figura 41, destaca-se o sinal de início da simulação, no bloco "SimulationEvents". Este sinal é o de um impulso fornecido à entrada do controlador do robô, que dá início ao arranque do programa (ARP), através da entrada "CycleStart" do controlador. O mesmo sinal atua, ainda, sobre a entrada de uma fonte ("Source"), gerando uma cópia de um objeto previamente selecionado. Neste caso, o objeto é a caixa a paletizar.

De igual forma, e durante o normal desenrolar da simulação, caso o sensor da tela transportadora (“LineSensor”) não detete a caixa presente na tela, será executada a cópia de uma nova. Este processo de cópia encontra-se associado a um movimento linear (“LinearMove”), que levará a caixa de uma extremidade da tela à outra, onde será levantada pelo robô. No final do movimento da caixa na tela, e caso o sinal do sensor se encontre presente, o objeto será registado em duas *queues* (ou filas) diferentes. Cada uma das filas representa uma “memória”. A primeira contém a indicação do objeto a empilhar, sendo o item removido da *queue* quando colocado na pilha. A segunda *queue* (“Queue_2”) contém todas as cópias criadas de objetos desde o início, sendo estas eliminadas quando o robô termina a execução da pilha e requer uma nova palete.

A ação de prender um objeto à ferramenta de vácuo encontra-se aqui representada pelo bloco “Attacher”. Resultará, no ambiente gráfico, na fixação da caixa à ferramenta, simulando o que aconteceria na realidade, caso a saída de vácuo fosse ligada (“VacuumOn”). De igual forma, ligando a saída para desligar o vácuo (“VacuumOff”), a caixa deixaria de estar acoplada à ferramenta (“Detacher”) e, por consequência, de estar na fila (“Queue”) da mesma. A realimentação do sinal de ligar vácuo à entrada de presença de objeto (“ObjectSensor” – opcional) imita a existência de um objeto na ferramenta.

Central a este processo encontra-se a representação do bloco do controlador do robô (“IRB_460_110kg_2.40m”), no qual se situam as entradas e saídas digitais para comunicação com o controlador da célula. Como exemplo da lógica estabelecida para as entradas e saídas presentes no bloco do robô, encontra-se a saída “ReqPallet”. A saída corresponde ao aviso de que a pilha está concluída e ao pedido do robô para uma nova palete vazia. Esta saída encontra-se ligada à entrada de remoção de todos os itens da “Queue_2” e à entrada do robô “Pallet_Ready”. Num cenário real, corresponderia à remoção da palete concluída, à inserção da nova e ao ligar do sinal de resposta. Da mesma forma, o sinal de saída “RobotReady”, que pretende indicar ao controlador que o robô se encontra pronto para paletizar, liga o sensor do transportador. Com isto pretende-se que o sensor só se encontre ativo se o robô estiver pronto, evitando a geração de objetos aleatoriamente.

Embora não esteja ilustrada na Figura 41, a entrada do controlador “CycleRestart” está prevista e foi simulada de forma manual, dada a sua própria natureza – reiniciar o controlador após alteração dos dados de entrada.

5.3. COLISÕES

No decorrer do desenvolvimento do programa ARP procuraram-se formas de detetar colisões entre o robô (ou a sua ferramenta) e os demais objetos a paletizar ou o equipamento da célula, quer em ambiente real quer em simulação. Para a primeira hipótese, foram tomadas as medidas possíveis para incluir a deteção de colisões, as quais passaram por colocar ativas as funcionalidades de supervisão de movimento "Motion Supervision". Contudo, esta opção não é simulável em ambiente RobotStudio.

A simulação de deteção de colisões para certas situações encontra-se, porém, contemplada pela ABB, que disponibiliza um *add-in*, em versão "PowerPack", para o *software* de base do RobotStudio. Uma vez que isto implicaria a utilização de um *software* adicional, em conjunto com a aplicação desenvolvida, o que goraria as bases do trabalho, o recurso a este *add-in* foi evitado.

No caso do pacote base do RobotStudio, a simulação existente requer que sejam criados, manualmente, dois grupos para a deteção prévia ("CollisionSets"). A criação de "Collision Sets" foi também colocada de parte, uma vez que só no decorrer da simulação os elementos seriam criados (dinamicamente) e, em diversos casos, estes ascenderiam às dezenas de unidades, inviabilizando assim a referida tarefa manual.

Outra opção está disponível na lógica da célula, onde seria possível recorrer a um "sensor de colisão". Contudo as entradas do sensor apenas permitem a verificação das mesmas entre um objeto e o restante mundo ou entre dois objetos, o que logo à partida travou a continuação dos testes, visto que a ferramenta, sempre que pegava num objeto, "colidia" com este ou, sempre que um objeto era colocado sobre a palete ou sobre um outro item, também o objeto "colidia".

5.4. SIMULAÇÕES

Como já mencionado, foram efetuadas inúmeras simulações, variando diversos parâmetros de entrada, para os testes da aplicação.

De forma a facilitar a análise dos dados, estas simulações centraram-se nas principais variáveis de entrada e foram divididas em dois grupos designados por ensaios. Esta separação pretendeu focar os testes ou nas dimensões do objeto a paletizar ou nas dimensões das áreas de paletização (respetivamente, o primeiro e o segundo ensaio).

Durante os ensaios, foram ainda alterados outros parâmetros e verificado se o comportamento do programa era o desejado. No final de cada ensaio foram reunidos os dados, resultantes dos mesmos, e resumidos numa tabela.

5.4.1. PRIMEIRO ENSAIO

A Tabela 12 apresenta os oito casos, considerados de maior relevância, que foram escolhidos para constituir o primeiro ensaio, nos quais foram utilizados diferentes tamanhos de objetos a paletizar (neste caso, caixas). Para todas as simulações deste ensaio foi mantido o tamanho *standard* de uma Europalete tipo EPAL 2 como área de empilhamento.

Tabela 12 Tamanho das caixas simuladas no primeiro ensaio

Caso	Comprimento (mm)	Largura (mm)	Altura (mm)	Volume (litros)
1	200	200	125	5,0
2	150	150	400	9,0
3	225	325	200	14,6
4	400	250	300	30,0
5	400	400	400	64,0
6	750	350	550	144,4
7	500	600	700	210,0
8	800	600	600	288,0

Através dos vários ensaios foi possível delimitar as dimensões dos objetos, as quais devem oscilar, aproximadamente, entre os valores apresentados no primeiro e no oitavo caso. Pretendeu-se, com isto, encontrar limites aceitáveis para evitar erros que pudessem colocar em causa a execução dos testes ou que fugissem à realidade.

As medidas resultam da constatação que se fossem usadas dimensões muito grandes – ou seja, muito superiores às do oitavo caso –, estas seriam desproporcionais relativamente às dimensões do robô e da ferramenta. Da mesma forma, dimensões pequenas – muito inferiores às do primeiro caso – significariam um número muito grande de posições, que ultrapassaria o limite do *array* correspondente (de posições) e impediria a execução do programa.

Uma das primeiras simulações no início dos trabalhos – não indicada na tabela e na qual os objetos tiveram as dimensões de 100 × 100 × 100 mm – revelou precisamente este último erro. No caso, o programa reagiu com a geração de uma mensagem de aviso da falha e concludentemente com a paragem da simulação. A Figura 42 apresenta um *screenshot* da simulação do Flex Pendant do controlador, desse momento.

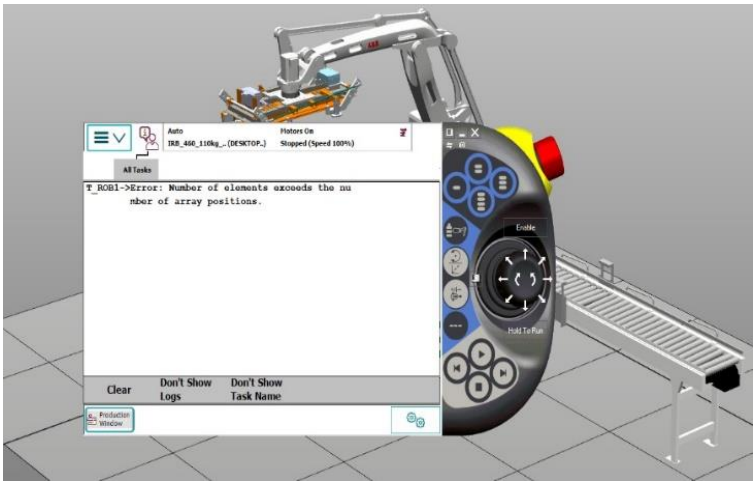


Figura 42 Erro de cálculo gerado durante simulações iniciais

Um outro erro presente nas simulações iniciais permitiu verificar que o posicionamento indevido da palete (mais precisamente do objeto *wobj_pallet*) poderia, também, interromper as simulações, visto que o robô não atingiria determinados pontos. Isto é, certos pontos estariam fora do espaço de trabalho do robô. A mensagem de erro encontra-se ilustrada na Figura 43.

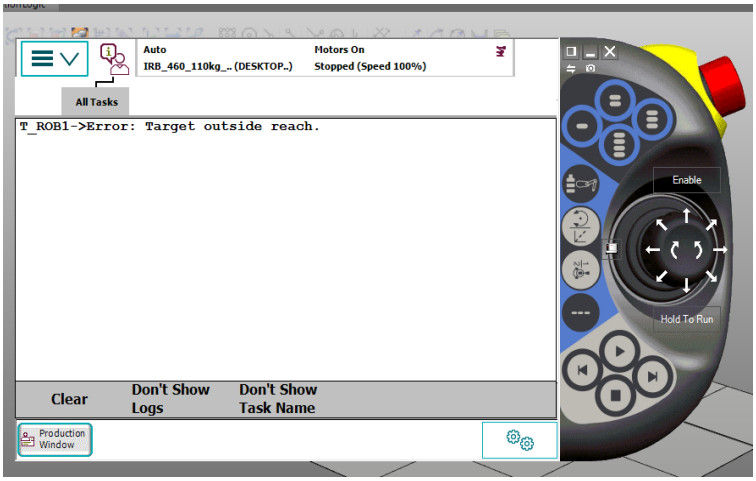


Figura 43 Erro de posicionamento gerado durante simulações iniciais

Quanto aos casos deste primeiro ensaio, no primeiro teste foram produzidas as pilhas com maior número de elementos e, consequentemente, as que mais tempo demoraram a simular. Isto deve-se ao facto de os objetos a paletizar terem as menores dimensões de todos os casos. A Figura 44 mostra uma imagem de uma das simulações efetuadas para este caso. Dada a geometria da peça, as simulações produziram o mesmo padrão, apesar dos diferentes estilos de camadas (em colunas e entrelaçados) e das diferentes orientações.

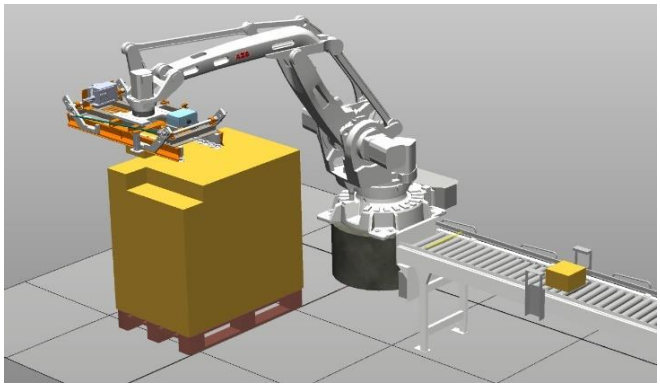


Figura 44 Simulação do caso 1/ensaio 1

Sublinha-se que, durante a fase inicial da simulação deste caso, observou-se que o número de camadas previsto, nos cálculos iniciais, não conseguia ser atingido. Este caso particular, segundo cálculos e considerada a altura máxima atingível, referida nas folhas de dados do fabricante, deveria executar quatro camadas. Contudo, tal altura apenas é atingível numa zona restrita do topo da região de trabalho.

Após repetidos ensaios (que também incluíram, por exemplo, a movimentação da palete para sítios diferentes), e tendo em conta que o limite superior exterior do volume de trabalho do robô é curvo (conforme se pode verifica na Figura 40), optou-se por reduzir a altura máxima do robô em cem milímetros (como margem de segurança), e refazer a simulação. Prevendo esta limitação durante a execução das simulações, adotou-se esta medida para os casos seguintes.

O segundo caso deste ensaio focou-se na paletização de um item com dimensões aproximadas às de uma caixa para garrafas, apresentado na Figura 45. A pilha foi criada em colunas, ou seja, com orientações iguais em todas as camadas (imagem do lado esquerdo) e em padrão entrelaçado (lado direito). Tal como no caso anterior, devido à geometria do objeto, os resultados produzidos não divergiram em termos de forma final da pilha.

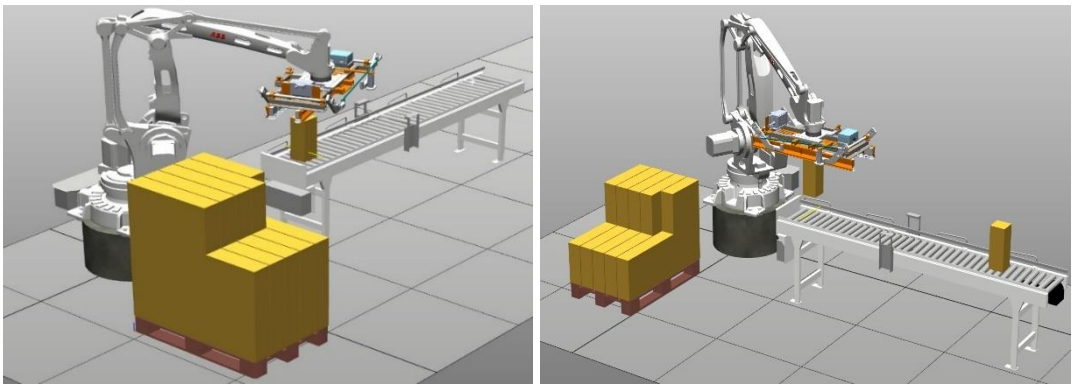


Figura 45 Simulação do caso 2/ensaio 1

A Figura 46 apresenta duas simulações feitas para o terceiro caso. Uma grande diferença entre as duas simulações reside na distribuição dos itens na palete (ver subsecção 4.3.3), a qual foi não uniformizada (lado esquerdo da figura) ou uniformizada (lado direito).

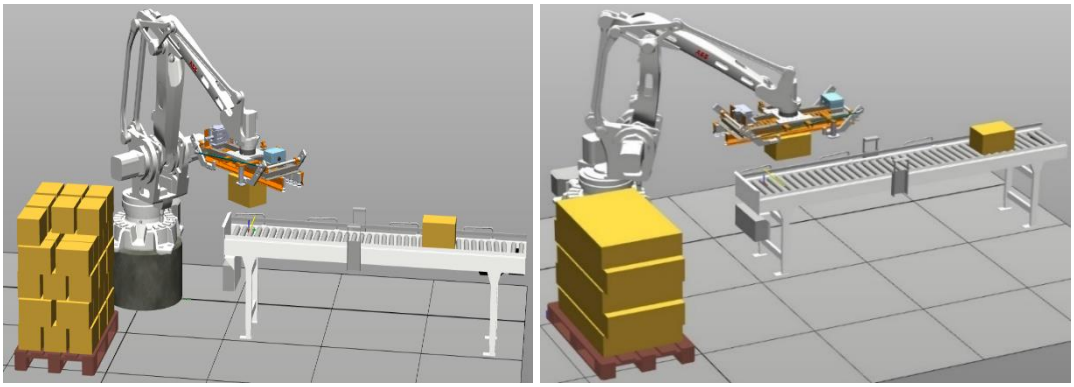


Figura 46 Simulação do caso 3/ensaio 1

Na Figura 47 é possível ver, ainda, um pormenor da criação da pilha, durante a execução da mesma simulação, que não é muito percetível na figura anterior. Este detalhe é referente à distribuição do espaço (não utilizado) pelas laterais da palete e que, na Figura 47, se encontra assinalado a vermelho. Como se comprova pela observação da figura, o espaço entre os limites da palete e os objetos é maior se não houver espaço restante entre estes.

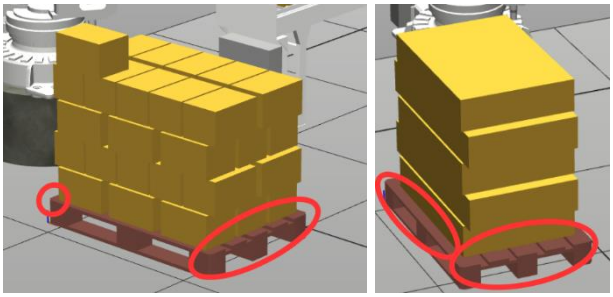


Figura 47 Pormenores da simulação do caso 3/ensaio 1

Presente na Figura 48, a simulação do quarto caso envolveu também a elaboração de um padrão em colunas (lado esquerdo da figura), tal como o caso anterior.

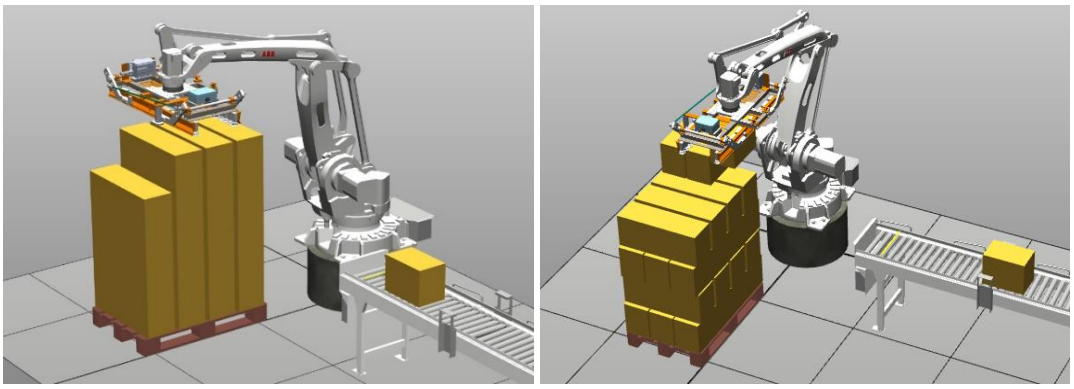


Figura 48 Simulação do caso 4/ensaio 1

Analogamente, foi executada uma simulação adicional com o mosaico do tipo entrelaçado para comparação dos resultados, a qual é visível na mesma figura (lado direito). A escolha das orientações dos objetos foi, porém, diferente da do caso anterior, apresentando as camadas pares uma rotação de 180° e as impares 270°, neste caso. A diferença na posição da ferramenta está patente na comparação das imagens da figura. Nestas, é possível verificar a rotação da ferramenta de vácuo, observando a posição da caixa de junção azul presente no topo da ferramenta (na imagem do lado esquerdo a posição está para rotações de 0°, na do lado direito, a rotação é de 270°).

Seguindo a lógica das simulações anteriores, o quinto caso testado produziu uma pilha com padrão em colunas (na primeira simulação) e um padrão entrelaçado (na segunda simulação). Neste caso, a aparência da palete permanece a mesma independentemente do estilo e orientação, dada a geometria dos objetos (cubos), conforme se pode confirmar pela análise da Figura 49.

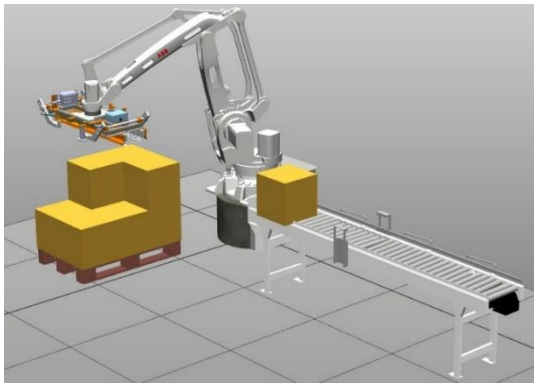


Figura 49 Simulação do caso 5/ensaio 1

Presentes na Figura 50, os resultados da sexta simulação do primeiro ensaio mostraram a importância da criação e utilização de mais do que um estilo de empilhamento. Como se pode verificar na imagem do lado esquerdo da Figura 50, utilizando o estilo em colunas foi possível proceder à paletização de seis caixas. Porém, para o estilo entrelaçado (imagem do lado direito da figura), mantendo o mesmo tamanho da palete, apenas foi possível colocar cinco caixas empilhadas. Isto significa uma diferença de 16%, da carga empilhada, entre ambas as pilhas, com significativa vantagem para o estilo em colunas, neste caso.

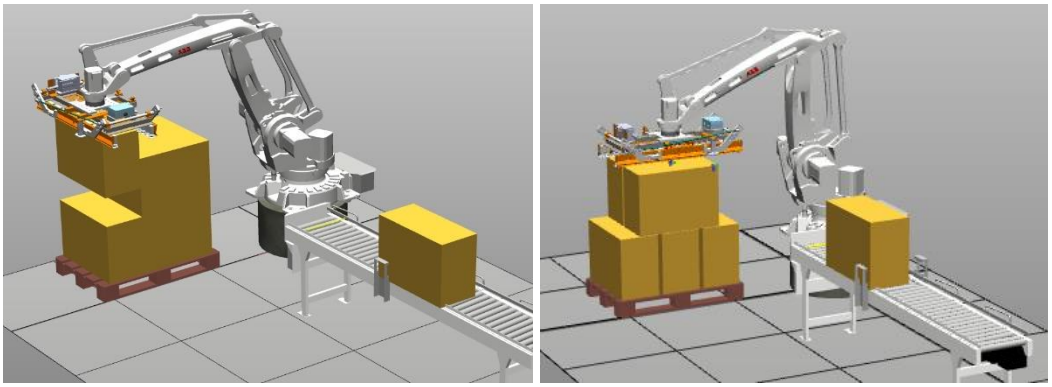


Figura 50 Simulação do caso 6/ensaio 1

O sétimo caso realçou a referida importância das camadas, mencionada no caso anterior. Como se pode depreender das imagens da Figura 51, apenas alterando o estilo de empilhamento, poderão criar-se paletes que, em princípio e por questões geométricas, serão mais estáveis. No caso em concreto, pela leitura da imagem do lado direito da Figura 51, verifica-se que o estilo entrelaçado levou à criação de uma camada superior que se estende para além dos limites da camada inferior. Esta afigura-se menos estável do que a sua homóloga, apresentada no lado esquerdo da mesma figura, onde a pilha toma a aparência de um bloco inteiriço.

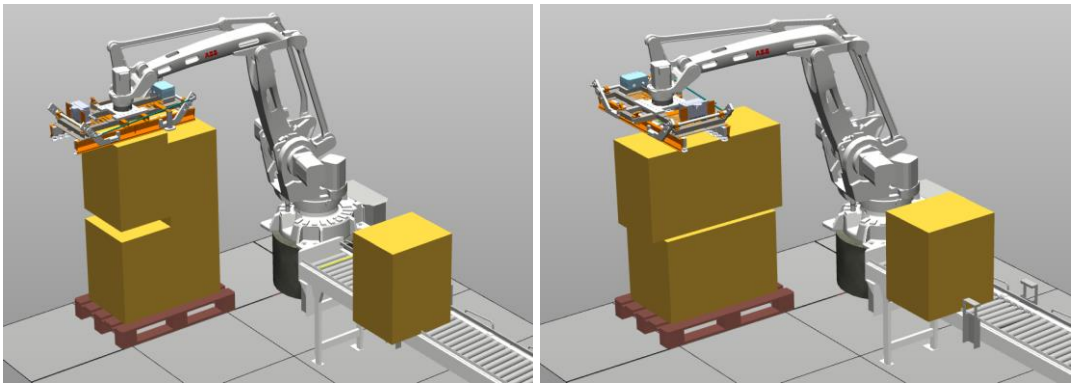


Figura 51 Simulação do caso 7/ensaio 1

No último teste deste ensaio, por causa do tamanho escolhido para os itens, a pilha foi de quatro caixas para o estilo em coluna e de apenas três para o entrelaçado. Na Figura 52 é possível ver duas imagens do final de cada um dos ciclos de paletização, durante duas das simulações deste caso.

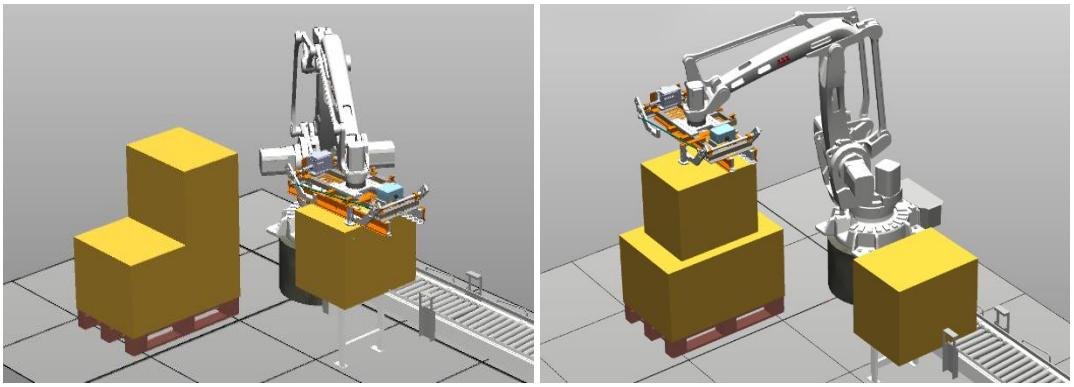


Figura 52 Simulação do caso 8/ensaio 1

Os resultados das simulações dos oito casos encontram-se resumidos na Tabela 13. Para ser possível uma comparação eficaz dos resultados, todos os casos foram simulados (por várias vezes) usando os estilos de mosaico em colunas e entrelaçado.

Tabela 13 Resultados das simulações do primeiro ensaio

Caso	**Número de elementos**		**Número de Camadas**	**Altura da pilha (metros)**	**Duração (segundos)**	
	Colunas	Entrelaçado			Colunas	Entrelaçado
1	288	288	12	1,645	965	965
2	120	120	3	1,345	403	403
3	57	57	6	1,345	196	196
4	42	42	5	1,645	132	137
5	18	18	3	1,345	62	62
6	6	5	2	1,245	22	18
7	4	4	2	1,545	16	16
8	4	3	2	1,345	15	11

Durante as simulações dos diversos casos, foram testadas, também, as possíveis orientações da ferramenta, variando as orientações dos objetos no ficheiro de inicializações. Isto permitiu avaliar se o robô conseguiria atingir os pontos necessários com a ferramenta em vários ângulos. Tendo em conta os resultados das diversas simulações, também se verificou que a duração da tarefa de paletização é independente do estilo de mosaico executado.

5.4.2. SEGUNDO ENSAIO

No segundo ensaio as simulações foram executadas com tamanhos variáveis de áreas de paletização – paletes ou folhas de cartão – e, no total, foram considerados dois casos de interesse. Ambos se encontram descritos na Tabela 14. Destaca-se o primeiro caso, que não faz uso de uma palete e é, como tal, considerada uma operação de *unitizing* onde a espessura do cartão (que serve de base) é desprezada.

Tabela 14 Tamanhos e tipos das paletes simuladas do segundo ensaio

Caso	Comprimento (mm)	Largura (mm)	Altura (mm)	Tipo
1	1200	800	0	*unitizing*
2	800	600	144	EPAL 6

O tamanho dos objetos foi mantido constante, ao longo do ensaio. As dimensões deste são exatamente as mesmas que as caixas têm no terceiro caso do primeiro ensaio (225 × 325 × 200 mm). Tal como no ensaio anterior, foram testados ambos os tipos básicos de empilhamento (em colunas e entrelaçado).

As primeiras simulações (caso 1) encontram-se ilustradas na Figura 53. Tal como no ensaio anterior, foram conduzidas várias simulações, alterando parâmetros que têm influência direta na forma de distribuição das caixas na palete. Ainda na Figura 53 é possível ver imagens das simulações, com recurso a estilo entrelaçado, onde se destaca a distribuição não uniformizada no lado esquerdo e uniformizada no lado direito.

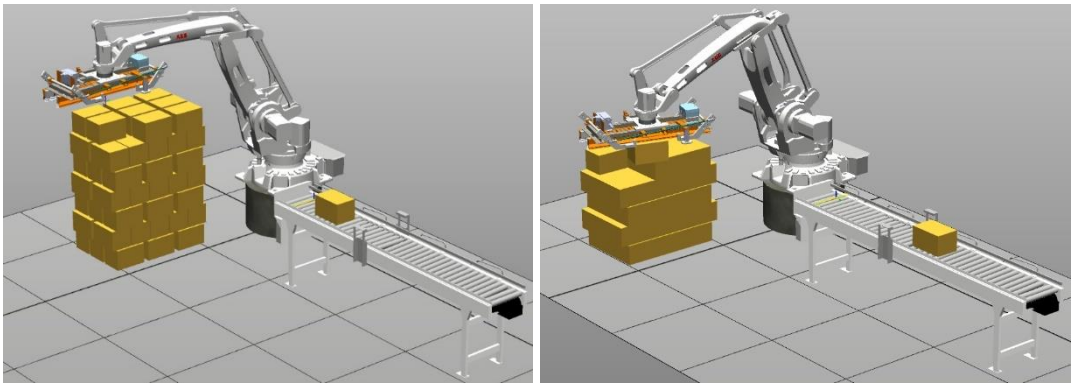


Figura 53 Simulação do caso 1/ensaio 2

O segundo caso consistiu na simulação de meia palete e a distribuição escolhida foi a uniformizada, conforme se verifica na Figura 54. De forma a ser mais evidente a diferença na área de criação da pilha, foi mantido o modelo de palete completa, como é possível ver na figura.

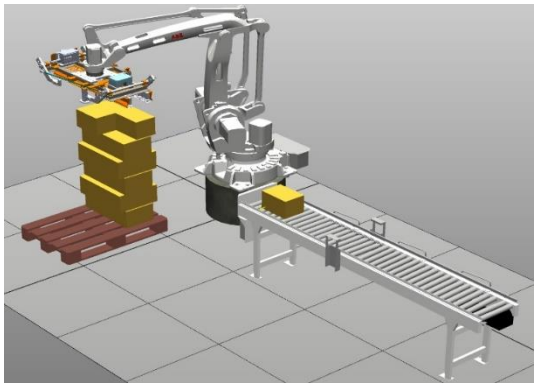


Figura 54 Simulação do caso 2/ensaio 2

Os resultados obtidos durante a execução das diferentes simulações encontram-se resumidos na Tabela 15. Incluído no fundo da tabela encontra-se o terceiro caso do ensaio anterior, cuja área de paletização corresponde a uma palete.

Tabela 15 Resultados das simulações do segundo ensaio

Caso	**Número de elementos**		**Número de Camadas**	**Altura da pilha (metros)**	**Duração (segundos)**	
	Colunas	Entrelaçado			Colunas	Entrelaçado
1	66	66	7	1,600	226	226
2	21	21	6	1,344	74	74
	57	57	6	1,345	196	196

Optou-se por incluir o mencionado caso do ensaio anterior por este permitir comparações com o segundo caso deste ensaio, dado que as dimensões dos objetos se mantêm iguais e ambos fazem uso de paletes. Quando comparados, diretamente, estes dois casos, verificou-se que, apesar da área da palete do ensaio anterior ser o dobro da área para a paletização do caso do corrente ensaio, o número de elementos empilhados (no caso do ensaio anterior) é superior ao dobro do caso deste ensaio. Concluiu-se assim, que o espaço disponibilizado na palete no caso do ensaio anterior foi mais rentabilizado do que neste caso, ou seja, para estes métodos de formação de mosaicos houve um melhor aproveitamento da palete no caso do ensaio anterior. O número de camadas, porém, manteve-se igual em ambos os casos e a diferença de alturas entre os dois casos justifica-se pela diferença de alturas das paletes.

Tal como no ensaio anterior (para o caso com o mesmo tamanho das caixas), as modificações no estilo do mosaico, em ambos os casos, não trouxeram alterações ao número de elementos, nem à duração da execução da pilha.

5.5. TESTES EM LABORATÓRIO COM ROBÔ IRB140

Os testes em laboratório envolveram o equipamento disponível no ISEP, o qual pode ser visto na Figura 55, e que consiste no robô (1), controlador (2) e ferramenta (3). Na mesma, ainda é possível observar a zona de levantamento dos objetos (o "transportador" nas simulações principais) e a zona de entrega dos mesmos ("palete") que, no caso, consistem em placas de esferovite branca com *stencils* de figuras geométricas sobrepostos. Ainda presente na célula (mas não na foto) encontra-se um compressor de ar que abastece o atuador da garra.

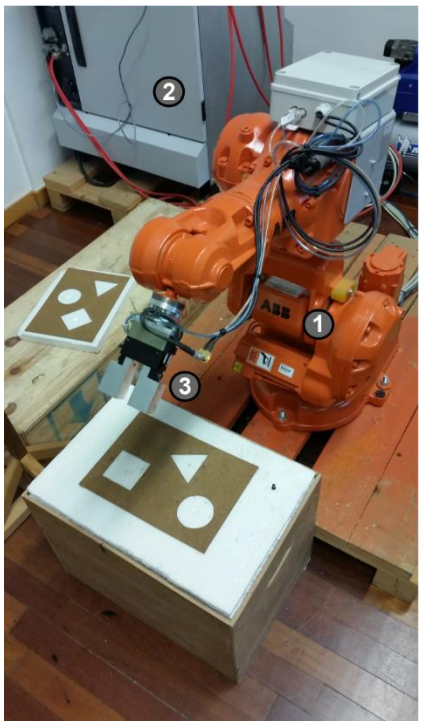


Figura 55 Célula robótica do laboratório de controlo do ISEP

O robô apresentado é da marca ABB, modelo IRB140, controlado por um controlador IRC5 cujo sistema operativo instalado é o RobotWare 5.15.1091. Trata-se de um robô um tanto díspar do simulado anteriormente (IRB460), dado se tratar de um robô de seis eixos com uma capacidade de carga de 6 kg [32], por exemplo. O modelo do controlador é, no entanto, o mesmo que o simulado, previamente, apesar de dispor de uma versão do sistema operativo anterior.

O controlador inclui, também, um Flex Pendant (consola para interface com o utilizador) que foi utilizado para carregar o programa ARP para o mesmo controlador. A consola permitiu dar início à execução do ciclo e, durante os ensaios, alterar a velocidade e as dimensões do objeto a paletizar, e reiniciar o ciclo quando necessário.

A ferramenta que se encontra anexada ao robô é uma garra de dois dedos paralelos com acionamento pneumático, da marca Schunk, modelo PGN-64. Quando aberta dispõe de um volume entre os dedos de dimensões 20 × 24 × 65 mm (comprimento, largura, altura). De salientar que esta garra é apropriada para operações de *pick and place* e não de paletização, sendo, contudo, a única disponível para testes/ensaios.

5.5.1. SIMULAÇÃO

Antes dos ensaios em laboratório foram efetuadas simulações do comportamento do robô, em ambiente RobotStudio, numa célula robótica previamente disponibilizada.

O procedimento foi mais breve do que o apresentado anteriormente (secção 5.1 – Metodologia e parâmetros), dado que os passos iniciais dos testes ao código já haviam ocorrido, aquando dos ensaios na célula robótica do robô IRB460. Acresce que, como o controlador do robô foi o mesmo em ambos os casos, não foram necessários ajustes ao mesmo.

Os *work objects* necessários ao funcionamento do programa – *conveyor* e *pallet* – foram colocados em duas áreas, as quais estão assinaladas com os números um (caso do *conveyor*) e dois (*pallet*) na Figura 56. Na mesma figura ainda se destaca a posição da *frame* do TCP da garra – especificada com o número três – definida ao centro da abertura entre os dedos no plano inferior dos mesmos.

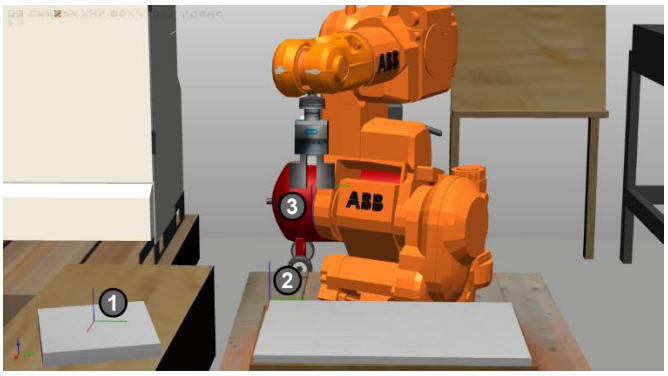


Figura 56 Localização dos *work objects* e TCP para a simulação do robô do laboratório

A lógica comportamental foi mantida e, quase na sua globalidade, inalterada (relativamente ao robô anterior). Foi, porém, removida a função de movimentação do objeto ao longo da tela transportadora e do sensor de presença de objeto, no final da tela, visto esta ser inexistente.

Foi também criada uma cópia do ficheiro de inicializações, onde foram alterados os valores necessários para o ajuste do tipo de ferramenta (TCP, posição pHome), do robô (altura máxima alcançável, distância intermédia) e do objeto (palete e objeto). O ficheiro foi posteriormente copiado para a pasta de raiz ("HOME") do controlador virtual do robô, tendo sido alterado conforme necessário.

Tal como no caso das simulações do robô IRB460, também para o IRB140 foram efetuadas simulações com objetos diferentes e, neste caso, zonas de entrega de diferentes dimensões. Os diferentes tamanhos dos objetos testados tiveram como limitação a abertura máxima dos dedos da garra e a largura dos mesmos.

Apesar de, segundo o fabricante [32], a altura máxima atingível pelo robô ser 712mm, esta foi restringida, neste caso, a 375 mm. Isto foi devido às diferenças de altura entre a base do robô e o topo da caixa que contém a área de paletização, no modelo da célula fornecido.

5.5.1.1. PRIMEIRO ENSAIO

A Tabela 16 apresenta as dimensões dos dois objetos de teste para o primeiro ensaio. Os tamanhos escolhidos foram os de um cubo de açúcar e de uma peça de dominó, respetivamente, no primeiro e segundo caso. No primeiro ensaio, o tamanho da zona de entrega para estas simulações foi mantido constante e consistia numa área quadrada, de 50 × 50 mm, tendo sido a altura desprezada.

Tabela 16 Tamanho das caixas simuladas no primeiro ensaio do robô IRB140

Caso	Comprimento (mm)	Largura (mm)	Altura (mm)
1	16	16	11
2	20	40	10

As simulações para estes ensaios, tal como para os ensaios anteriores para verificação do comportamento do programa ARP, utilizaram os dois padrões descritos para criação de camadas.

Assim, cada caso apresentado neste ensaio foi alvo de, pelo menos, duas simulações cujo conteúdo divergiu no tipo de empilhamento. Da mesma forma, também as orientações foram alteradas, com o decorrer dos ensaios, para confirmação do alcance dos pontos, por parte do IRB140.

A Figura 57 apresenta duas imagens do primeiro caso. Nas mesmas é possível verificar a diferença de padrões criados (à esquerda, em colunas, e à direita, entrelaçado). O padrão em colunas, apresentado na imagem da direita, teve, neste caso, como orientação 270°.

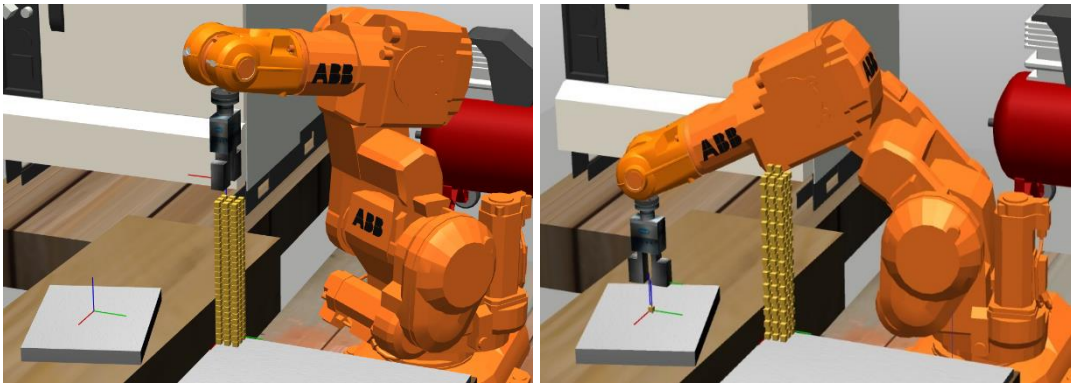


Figura 57 Simulação do caso 1/ensaio 1 – IRB140

Durante a simulação deste caso, tornou-se evidente que não seria possível conduzir o teste em ambiente real, devido à geometria da garra. A largura dos dedos da garra é impeditiva dos tipos de paletização efetuados neste trabalho, visto a garra colidir com a pilha em execução, aquando da colocação do objeto na pilha, como se verifica na Figura 58.

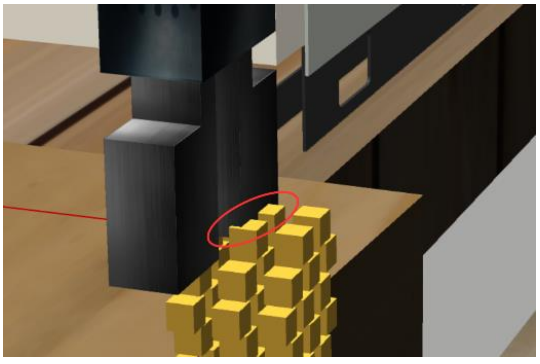


Figura 58 Pormenor da simulação do caso 1/ensaio 1– IRB140

Para evitar este problema, os objetos teriam que ser distanciados, no mínimo, à largura de um dedo da garra, para ambos os lados, ou ter-se-ia de mudar a ferramenta. As alterações primeiramente sugeridas, porém, inviabilizariam a geração de uma pilha, visto a distância entre objetos ser superior às dimensões do mesmo.

Também, dada ser a única ferramenta disponível em laboratório, a substituição desta não pôde ser equacionada. Por este motivo, optou-se por continuar com as simulações (com o objetivo de testar o programa no robô existente, com a garra disponível) embora reconhecendo a limitação.

No segundo caso deste ensaio foram, novamente, testados os estilos básicos de criação de mosaicos e as simulações contaram, novamente, com várias alterações das orientações das camadas, visíveis na Figura 59. Na mesma figura, no lado esquerdo, é possível observar uma pilha criada com um padrão em colunas, uniformizado, com orientação de 180°. No lado direito, o padrão escolhido foi o entrelaçado e, neste caso, não uniformizado.

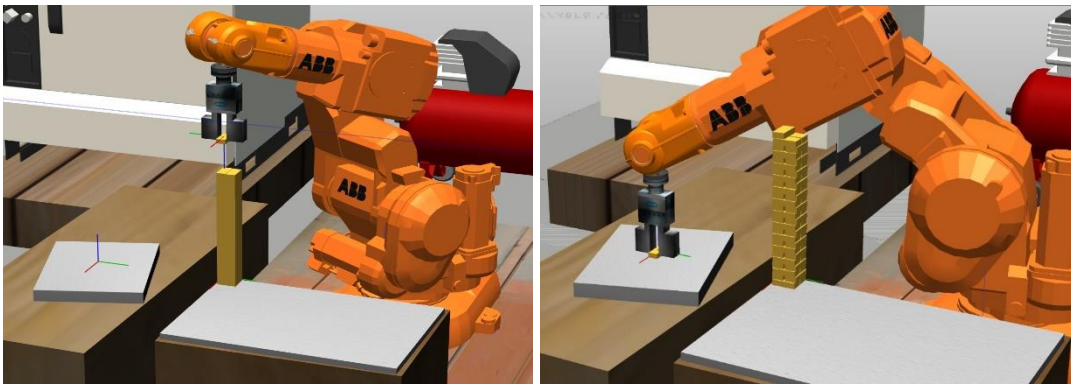


Figura 59 Simulação do caso 2/ensaio 1 – IRB140

Expectavelmente, e tal como no primeiro caso, a ferramenta colidia com a pilha no momento da deposição da peça, impossibilitando, também, que este caso fosse testado no robô real.

Os resultados dos dois casos testados durante este ensaio encontram-se resumidos na Tabela 17. Da leitura da tabela é possível inferir que, tal como para o robô anterior, a modificação do estilo de construção da pilha não tem influência sobre a duração do processo para estes casos em estudo. Poderá, eventualmente, ter influência no número de itens a paletizar, dado que o aproveitamento da superfície de paletização varia consoante o mosaico escolhido.

Tabela 17 Resultados das simulações do primeiro ensaio do robô IRB140

Caso	**Número de elementos**		**Número de Camadas**	**Altura da pilha (metros)**	**Duração (segundos)**	
	Colunas	Entrelaçado			Colunas	Entrelaçado
1	234	234	26	286	421	421
2	58	58	29	290	115	115

5.5.1.2. SEGUNDO ENSAIO

Seguindo a mesma lógica utilizada para o IRB460, o ensaio que se seguiu manteve fixo o tamanho do objeto a paletizar, tendo sido alteradas as medidas das áreas de deposição. No entanto, perante as dimensões do robô e da ferramenta acoplada ao mesmo, não se justificou a adição de um elemento suplementar para colocar os objetos. Por este motivo, a altura das áreas de paletização na Tabela 18 é inexistente.

Tabela 18 Tamanhos das áreas de paletização para o segundo ensaio do IRB140

Caso	Comprimento (mm)	Largura (mm)
1	40	80
2	120	100

O primeiro caso foi alvo de várias simulações que, não obstante a diferença de orientações nos objetos e no estilo das camadas, geraram os mesmos resultados. Dadas as dimensões das peças, no caso do empilhamento em colunas, estas encaixaram perfeitamente na área designada para o efeito. Como tal, para este caso, a escolha do tipo de distribuição (uniformizada ou não) revelou-se indiferente. É possível verificar o efeito mencionado, assim como o padrão produzido na Figura 60.

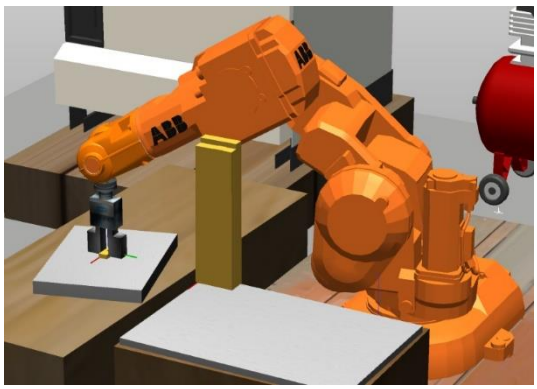


Figura 60 Simulação do caso 1/ensaio 2 – IRB140

Na primeira simulação, para o estilo entrelaçado, foram usadas as orientações de 0° nas camadas pares, tendo sido este valor alterado para 180° na segunda e terceira simulação. Relativamente às camadas impares estas iniciaram com o valor de 90°, na primeira e segunda simulação, o qual foi posteriormente alterado para 270°, na terceira simulação. Pretendeu-se com estas alterações verificar o comportamento do robô à alteração de uma das orientações.

O segundo caso deste ensaio abarcou também várias simulações que produziram os mesmos resultados. A Figura 61 ilustra um momento de uma simulação onde é possível ver duas das diversas pilhas produzidas. No lado esquerdo da Figura 61 apresenta-se o início da execução de uma pilha formada em colunas, não uniforme. Neste caso em particular, a uniformização e a orientação não alteraram o aspeto final da pilha. Isto, dado que as dimensões dos itens permitiram encaixar, precisamente, um número inteiro de objetos por eixo. Ainda na mesma figura (lado direito), vê-se a pilha formada, que teve como padrão o entrelaçado e uma distribuição não uniformizada.

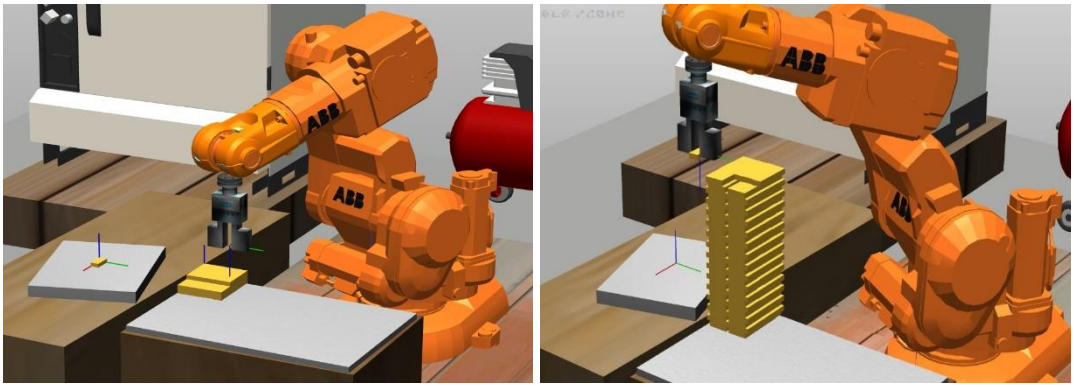


Figura 61 Simulação do caso 2/ensaio 2 – IRB140

Resumidos na Tabela 19 estão os resultados das simulações dos dois casos deste ensaio. É de salientar que a alteração de padrão, no segundo caso, provocou um decréscimo de itens de 42 unidades, do padrão entrelaçado relativamente ao padrão em colunas. Consequentemente, esta mudança conduziu a um aumento da duração da simulação, mas não teve influência nem na altura nem no número de camadas produzido.

Tabela 19 Resultados das simulações do segundo ensaio do robô IRB140

Caso	**Número de elementos**		**Número de Camadas**	**Altura da pilha (metros)**	**Duração (segundos)**	
	Colunas	Entrelaçado			Colunas	Entrelaçado
1	116	116	29	290	210	210
2	435	393	29	290	792	712

5.5.2. TESTES FÍSICOS

Os ensaios finais do programa ARP foram efetuados com recurso ao robô disponibilizado em laboratório. Estes testes iniciaram-se com o carregamento dos ficheiros dos módulos do ARP e, consequentemente, do ficheiro de inicializações. Este processo recorreu à interface do Flex Pendant existente no controlador do robô que permitiu copiar os ficheiros de uma memória USB para o mesmo controlador.

Neste caso, tal como em simulações prévias, não foi utilizada uma palete, dadas as limitações presentes pelo tamanho do robô e da ferramenta. A velocidade máxima foi reduzida para 25%, inicialmente, devido à imperfeita ancoragem do robô, tendo sido progressivamente aumentada até ao máximo (pontualmente) com o desenrolar dos ensaios, observando sempre as condições de segurança. Na Figura 62 é possível observar o robô na sua posição de levantamento de objetos a paletizar, durante a execução dos testes.

Figura 62 Robô IRB140 no ponto de levantamento de objetos a paletizar

Previsivelmente, e tal como mencionado no ponto anterior, existiram restrições ao uso de objetos durante os ensaios com o robô real. Estas limitações acabaram por inviabilizar o recurso a objetos físicos durante os ensaios. Assim, foram efetuados diversos testes, desta feita, sem objeto a paletizar, mas sendo sempre verificado se o robô produzia o número de elementos, pontos e orientações desejados através de sucessivas e aleatórias paragens e arranques do robô e consequentes medições e contagens.

Conforme se verifica na Figura 63 e na Figura 64, o robô atingiu os diversos pontos com as configurações esperadas, em termos de orientação e posição dos objetos. Os diversos ensaios procuraram reproduzir simulações anteriores – do primeiro ensaio reproduziu-se o caso dois, e do segundo ensaio, o caso um. Os resultados foram em tudo idênticos aos simulados com a exceção dos tempos de ciclo, que foram superiores, proporcionalmente, à velocidade do robô.

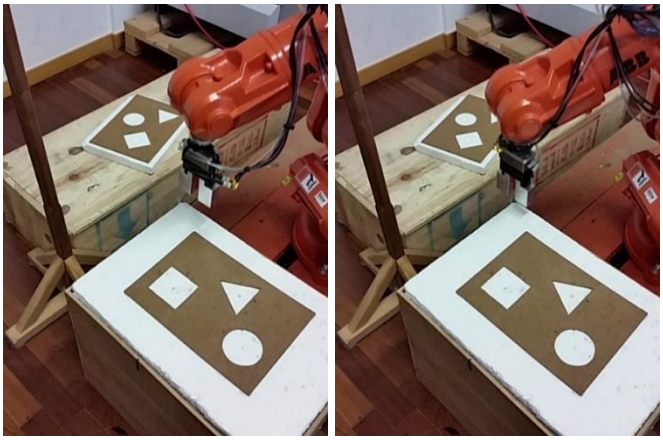


Figura 63 IRB 140 na zona de entrega com a garra a 0° e 180°

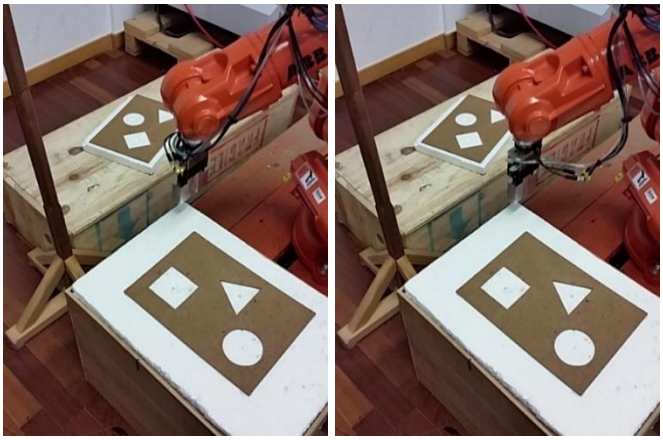


Figura 64 IRB 140 na zona de entrega com a garra a 90° e 270°

Resumidamente, pela análise aos resultados, foi possível atestar alguns dos resultados obtidos em simulação e verificar algumas diferenças. Rapidamente se tornou claro que o processo é influenciado pela ferramenta utilizada bem como pela escolha do mosaico de paletização. As principais ilações retiradas da elaboração dos ensaios, e em suma do trabalho na sua generalidade, encontram-se descritas no capítulo que se segue.

6. CONCLUSÕES

O desenvolvimento da aplicação objeto deste trabalho acarretou um estudo alongado de diversas temáticas, relacionadas com a paletização e robótica, constatando-se que há uma crescente procura de robôs, incluindo paletizadores robotizados.

Em termos gerais, a oferta de equipamentos para funções de paletização robotizada encontra-se, significativamente, difundida. A quase totalidade dos fabricantes de robôs tem à disposição pelo menos um robô específico para tais funções. Acresce que não existem muitas opções de *software* genérico dedicado à paletização (ou seja, produzido por não fabricantes de robôs). Será de deduzir, por isso, que a oferta existente de *software* proprietário – possivelmente aliado à conveniência do fornecedor para *software* e *hardware* ser só um – esteja a cobrir, atualmente, as necessidades do mercado.

Da análise ao *software* proprietário disponível poderá retirar-se a ilação que, apesar de este prever diversas funcionalidades e ser muito evoluído, a tentativa de cobrir o máximo possível de casos/situações para o processo de paletização levou a que alguns programas tomassem proporções complexas, em termos de opções colocadas à disposição do utilizador.

Independentemente desta complexidade, e ainda que alguns fabricantes continuem a optar por linguagens de programação específicas para robôs, verifica-se também que um número crescente de fabricantes tem criado formas de integrar linguagens mais comuns e divulgadas no seu *software*. Isto surge como forma de tentar minimizar a necessidade de qualificação técnica para a programação (ou configuração) de robôs, em específico, tentando reduzir deste modo os custos de programação e da mão-de-obra, em geral.

Em termos de *software* de programação *offline* da marca ABB, destaca-se a facilidade de utilização do RobotStudio para a elaboração e simulação de células robotizadas. No RobotStudio, é ainda fornecido um método para simular o controlador de um robô e, separadamente, o ambiente que o rodeia como, por exemplo, equipamentos que se encontrem na célula ou os próprios objetos de trabalho.

Relativamente à simulação de colisões no RobotStudio, e estando estas alocadas à simulação do ambiente e não à simulação do controlador, não foi possível encontrar uma forma que supervisionasse satisfatoriamente as mesmas, sem recorrer especificamente ao desenvolvimento personalizado da célula e de uma complexa lógica adjacente à estação. E, não obstante as várias tentativas, também não foi possível encontrar um consenso na interação da simulação das colisões com o programa ARP. É de sublinhar que, neste mesmo programa, a opção "Motion Supervision" foi ligada no módulo de movimentação do robô na função "MoveRob". Esta opção visa detetar colisões, em ambiente real, contudo não é simulável.

Os resultados obtidos das simulações e ensaios realizados permitiram tirar diversas conclusões. Entre estas destaca-se a franca dependência da ferramenta do robô selecionada para o processo de paletização, bem como a importância da escolha do mosaico. Uma ferramenta inadequada ao processo poderá levar a erros durante a programação das tarefas do robô e a produções menos eficientes. Da mesma forma, a seleção do mosaico pretendido afigura-se crucial para a obtenção de melhores resultados. Como na criação do mosaico são possíveis ajustes, estes poderão levar a acréscimos de eficiência por poderem, por exemplo, significar um melhor aproveitamento do espaço quer nos meios de transporte quer em armazenamento. Outra conclusão foi que o tempo de processamento de uma palete é independente da orientação com que os objetos são colocados, visto que a rotação dos mesmos será feita durante a movimentação até ao ponto desejado.

Durante as simulações constatou-se ainda que, para a criação da pilha, é importante saber qual a região (ou espaço) de trabalho do robô. Isto é, só será possível construir a pilha, por completo, caso todos os pontos que o robô tenha que atingir se encontrem dentro da região de trabalho.

Apesar de não figurar como um objetivo principal para a execução do trabalho, a escolha de uma linguagem de marcação (XML) para o ficheiro de inicializações de interface com o utilizador, trouxe a vantagem de permitir que a plataforma desenvolvida opere autonomamente. Por outras palavras, dado ser um ficheiro inteligível por máquinas (para além de por humanos), a geração ou alteração do ficheiro de inicializações poderá ser feita por uma outra aplicação de *software* sem que seja necessária a interação humana. Para tal bastará que o ficheiro de inicializações seja alterado/regenerado e, usando as entradas disponibilizadas para o efeito, o ciclo reiniciado.

6.1. PERSPETIVAS FUTURAS

Assentes as bases para a programação automática, a inclusão de possíveis periféricos acessórios ao processo de paletização (como dispensador de paletes e de folhas) seria o passo seguinte na evolução do trabalho. Conjuntamente, deveria ser estudado o desenvolvimento de uma interface mais atrativa (para o utilizador) para a introdução dos valores iniciais.

Aumentando de complexidade, e tendo em vista os exemplos que foram analisados ao longo do processo de pesquisa, a possibilidade de manipulação de mais do que uma linha e mais do que uma palete apresentar-se-ia como uma mais-valia, em termos de função, para aplicações mais avançadas.

Também contemplada, durante a execução deste trabalho, foi a hipótese de se vir a desenvolver um programa externo que crie o mosaico para a paletização. Este programa poderia conter definições e algoritmos mais avançados para a criação de um ficheiro de saída com os pontos e as orientações dos objetos na palete, proporcionando deste modo a hipótese de se usarem *true mixed pallets*. A possibilidade de importação já se encontra disponível no programa APR, tendo sido testada durante as simulações (conforme mencionado acima), com a introdução dos pontos e respetivas orientações no ficheiro de inicializações, de forma manual.

Referências Documentais

[1] "robótica in Dicionário infopédia da Língua Portuguesa com Acordo Ortográfico," Porto Editora, 2003-2013. [Online]. Available: https://www.infopedia.pt/dicionarios/lingua-portuguesa/robótica. [Acedido em 27 maio 2017].

[2] L. Willcocks e M. Lacity, ""Will robots replace Humans? - Society, media and science"," London School of Economics and Political Science, outubro 2015. [Online]. Available: http://www.lse.ac.uk/researchAndExpertise/researchHighlights/societyMediaAndScience/Will-robots-replace-humans.aspx. [Acedido em junho 2017].

[3] International Federation of Robotics, "World Robotics 2016 - Results, Forecast, Trends," Carate Brianza - Itália, 2016.

[4] H. Kagermann, W. Wahlster e J. Helbig, "Securing the future of German manufacturing industry: Recommendations for implementing the strategic initiative INDUSTRIE 4.0," ACATECH – German National Academy of Science and Engineering, 2013.

[5] F. Molzow-Voit, M. Quandt, M. Freitag und S. Georg, Robotik in der Logistik: Qualifizierung für Fachkräfte und Entscheider, Wiesbaden: SpringerGabler, 2016.

[6] Technavio, "European Industrial Robotics Market 2016-2020 | Market research reports," [Online]. Available: https://www.technavio.com/report/europe-robotics-industrial-robotics-market. [Accessed junho 2017].

[7] M. Argenti, D. Burati, D. L. Rizzini e S. Caselli, "An Integrated Tool Suite for Simulation and Programming," em *International Symposium on Robotics*, Munique, Alemanha, 2010.

[8] W. A. Günthner e U. Lammer, "Funktionsvereinigung in der Lagertechnik," fml – Lehrstuhl für Fördertechnik Materialfluss Logistik, Munique, Alemanha, 2009.

[9] International Organization for Standardization, "ISO 8373:2012 - Robots and robotic devices — Vocabulary," 2012. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en. [Acedido em 27 maio 2017].

[10] A. M. Lopes, "Modelação Cinemática e Dinâmica de Manipuladores de Estrutura em Série," FEUP - Faculdade de Engenharia da Universidade do Porto, Porto, 2001.

[11] R. A. Popple, The Science of Palletizing, Vancouver: Columbia Machine, Inc., 2009.

[12] EPAL - European Pallet Association e.V., "Pallet Overview," EPAL - European Pallet Association e.V., 2017. [Online]. Available: https://www.epal-pallets.org/eu-en/load-carriers/overview/. [Acedido em junho 2017].

[13] C. Hongtai, X. Yunjie, L. Yumei e H. Lina, "Design of Palletizing Algorithm Based on Palletizing Robot Workstation," em *Conference on Real-time Computing and Robotics*, Angkor Wat, Cambodja, 2016.

[14] KUKA Roboter GmbH, *Factory Automation with industrial robots for palletizing food products like bread and toast at a bakery in Germany, robotics,* Augsburg: KUKA Roboter GmbH, Bachmann, 2005.

[15] D. Liu, *FlexGripper/Palletizing Grippers Clamp, Claw, Vacuum,* ABB Robotics, 2011.

[16] ABB Robotics, *Robots for packaging industry/Robot based packaging automation,* ABB Robotics, 2012.

[17] A. Hoffmann, *Serviceorientierte Automatisierung von Roboterzellen,* Augsburgo, Alemanha: Universidade de Augsburgo, 2015.

[18] RoboDK, "RoboDK Tips," RoboDK, 2015. [Online]. Available: https://robodk.com/help. [Acedido em julho 2017].

[19] D. Elkins, "Speaking One Language is Better Than Two," Yaskawa, março 2016. [Online]. Available: https://www.motoman.com/blog/index.php/speaking-one-language-is-better-than-two/. [Acedido em julho 2017].

[20] Coppellia Robotics, *V-REP User Manual,* Coppellia Robotics, 2017.

[21] Kawasaki Robotics (USA), Inc., "Programming Tool - K-SPARC Palletizing software," 2014.

[22] ABB Robotics, "ABB Corporate Web-Site - History," ABB Robotics, março 2017. [Online]. Available: http://new.abb.com/about/abb-in-brief/history. [Acedido em março 2017].

[23] K. Bengtsson, *PalletPack 460/Palletizing Function Package,* ABB Robotics, 2012.

[24] ABB Robotics, *IRB 460 Industrial Robot Datasheet,* ABB Robotics, 2011.

[25] S. Yu, S. Lim, M. Kang, C. Han e S. Kim, "Off-line Robot Palletizing Simulator Using Optimized Pattern and Trajectory Generation Algorithm," em *IEEE/ASME International conference on advanced intelligent mechatronics*, Zurique, Suiça, 2007.

[26] ABB Robotics, *IRC5 Industrial Robot Controller,* ABB Robotics, 2014.

[27] ABB Robotics, *RobotStudio Operating Manual,* 3HAC032104-001 Revision T ed., ABB Robotics, 2016.

[28] ABB Robotics, *Technical reference manual/RAPID Instructions, Functions and Data Types,* 3HAC050917-001 Revision D ed., ABB Robotics, 2016.

[29] ABB Robotics, *Application Manual/RAPID development guidelines for handling applications,* ABB Robotics, 2013.

[30] The World Wide Web Consortium (W3C), "Extensible Markup Language (XML) 1.0 (Fifth Edition)," W3C, 2008. [Online]. Available: https://www.w3.org/TR/REC-xml/. [Acedido em março 2017].

[31] M. Raković, B. Borovac, M. Nikolić, M. Jovanović, B. Tepavčević e M. Papović, “Design and Fabrication with Industrial Robot as Brick-laying tool and with Custom Script Utilization,” em *23rd International Conference on Robotics in Alpe-Danube Region,* Castelo de Smolenice, Eslóvaquia, 2014.

[32] ABB Robotics, *Product Specification IRB140,* ABB Robotics, 2017.

[33] J. M. P. Waveren, “From Quaternion to Matrix and Back,” Id Software, 2005.

[34] R. Sebesta, Concepts of Programming Languages, Third Edition, Addison Wesley Press, 1996.

[35] Open Source Robotics Foundation, “ROS.org | Powering the world's robots,” julho 2017. [Online]. Available: http://www.ros.org. [Acedido em julho 2017].

[36] Open Source Robotics Foundation, “ROS - Industrial,” setembro 2017. [Online]. Available: http://rosindustrial.org/. [Acedido em setembro 2017].