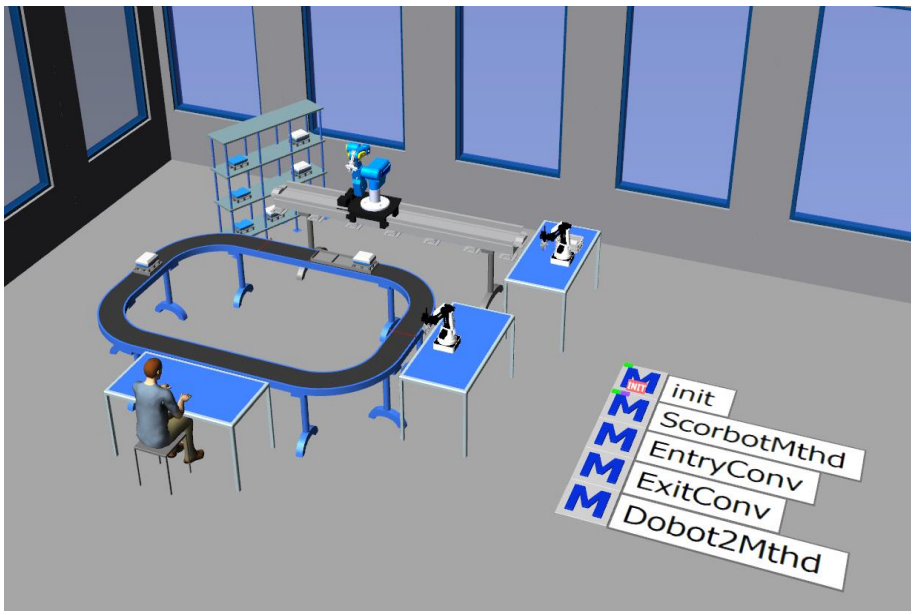
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Mecânica

ISEL



Simulação, Monitorização e Comando de um Sistema CIM

PAULO FILIPE DA COSTA VARELA

(Licenciado em Engenharia Mecânica)

Trabalho Final de Mestrado para obtenção do grau de Mestre
em Engenharia e Gestão Industrial

Orientadores:

Doutor Francisco Mateus Marnoto de Oliveira Campos
Doutor Fernando Paulo Neves da Fonseca Cardoso Carreira

Júri:

Presidente: Doutor António João Pina da Costa Feliciano Abreu
Vogais:

Doutor Paulo António da Silva Ávila
Doutor Francisco Mateus Marnoto de Oliveira Campos

Janeiro 2022

Agradecimentos

Gostaria de agradecer ao professor Francisco Campos e ao professor Fernando Carreira pela orientação e apoio prestados no desenvolvimento deste projeto.

Um agradecimento à minha família e amigos pelo apoio e motivação prestados durante todo o meu percurso académico.

Por fim, agradeço ao ISEL e todos os professores por me darem o conhecimento e condições para atingir todos os objetivos pretendidos.

Resumo

A simulação, monitorização e o controlo de processos são ferramentas em constante desenvolvimento sendo o seu principal suporte o avanço das tecnologias de informação. É possível verificar a correlação no desenvolvimento destas tecnologias observando o progresso dos algoritmos de simulação nomeadamente das ferramentas de simulação desde o 2D até ao 3D e com a inclusão de realidade virtual. Este progresso apenas tem sido possível graças ao maior poder de processamento dos computadores, o qual se traduziu num impacto significativo na indústria na forma de sistemas de supervisão, aquisição e controlo (SCADA) totalmente integrados que constituem uma peça importante na revolução da Indústria 4.0.

A presente dissertação consiste no desenvolvimento de ferramentas de monitorização, simulação e controlo da célula de fabrico robotizada existente no Laboratório de Robótica do Departamento de Engenharia Mecânica do ISEL. Para o desenvolvimento do modelo de simulação recorreu-se ao *software* Tecnomatix Plant Simulation disponibilizado pela Siemens e para o controlo do mesmo foi utilizado o Matlab. Para validação do projeto foi definido um ciclo de manufatura que compreende os diversos equipamentos disponíveis para utilização no laboratório.

Este trabalho apresenta o processo do desenvolvimento das ferramentas acima mencionadas, contextualizadas na revisão bibliográfica dos temas relevantes ao mesmo. Perspetiva-se também que no futuro o sistema seja usado na monitorização e controlo em tempo real da célula de fabrico.

Palavras-chave: Simulação, SCADA, CIM, Indústria 4.0, Integração de sistemas

Abstract

The process simulation, monitoring and control are tools in constant development being the information technology evolution their main support. It is possible to verify the correlation between the development of these technologies looking at the progress of simulation algorithms, specifically the path from 2D to 3D simulation and the inclusion of virtual reality. This process has only been possible thanks to the greater computing processing power, which had a significant impact in modern industry in the form of supervising control and data acquisition systems (SCADA). The total integration of SCADA has played a big role in the Industry 4.0 revolution.

The current dissertation consists in the development of monitoring, simulation and control tools for the robotized manufacturing cell installed in ISEL's Mechanical Engineering Departments' Robotics Lab. In order to develop the simulation model, Siemens's Tecnomatix Plant Simulation was used and for the controller the Matlab *software* was used. Using all the equipment available in the lab, a manufacturing cycle was also created so that the project could be tested.

This dissertation presents the process of developing the above-mentioned tools as well as the state of the art of the relevant topics. It is expected that in the future, the system will be used for monitoring and control of the real manufacturing cell.

Keywords: Simulation, SCADA, CIM, Industry 4.0, System Integration

Índice

Lista de Figuras

Lista de Tabelas

Lista de Listagens

1. Introdução

Neste capítulo serão abordadas a motivações que impulsionaram a realização da presente dissertação com o tema "Simulação, monitorização e comando de um sistema CIM" e definidos os seus objetivos. Será ainda feita uma breve descrição da estrutura utilizada no restante documento, para facilidade de leitura.

1.1 Motivação

Um sistema CIM (*Computer Integrated Manufacturing*), como o nome sugere, integra o software e hardware necessários para conceção de produto, incluindo modelação CAD (*Computer Aided Design*) e visualização gráfica e para a realização de todas as atividades de produção como o planeamento, programação, otimização, etc. O CIM aborda todas as etapas desde o desenvolvimento de conceito até à distribuição do produto final. Assim, são abrangidas áreas como: programação CNC (*Computer numerical control*), controlo adaptativo de parâmetros de produção, automação de MRP (*Manufacturing Resource Planning*) e de sistemas de montagem, planeamento de processos **criação de sistemas flexíveis de manufatura, controlo de manufatura celular e de robôs** etc.[1]. A presente tese foca-se nestas três últimas áreas referidas consistindo na modelação, monitorização e controlo de uma célula flexível de fabrico. Uma vez que os trabalhos desenvolvidos nesta tese se restringem a estes níveis de controlo, usar-se-á ao longo do documento preferencialmente os termos *célula de fabrico*, em lugar de *sistema CIM*.

Os sistemas de controlo e monitorização, assim como a utilização de simulações, são fundamentais para automação de um processo de produção industrial. A simulação é uma ferramenta de suporte utilizada para reproduzir em computador o funcionamento de um processo ou sistema real de forma a obter informações acerca do seu desemprenho. Desta forma, o ambiente simulado permite programar a automatização de um sistema, ou testar possíveis otimizações, auxiliando a tomada de decisão do projetista, sem recorrer ao sistema real.

Os sistemas de aquisição, controlo e supervisão SCADA (*Supervisory Control And Data Acquisition*) são compostos por componentes físicos e digitais. Têm como função principal o controlo e monitorização de sistemas industriais, permitido controlar o processo, recolher informação e emitir alertas em caso de emergência. Os sistemas SCADA têm inúmeras

vantagens o que torna a sua utilização na monitorização e controlo de processos industriais bastante apelativa e largamente adotada. No entanto, muitas vezes a aquisição deste tipo de sistemas demostram-se dispendiosa e a sua implementação e personalização de difícil execução. É também comum que estes sistemas não permitam a simulação *offline* de alterações ao processo e a avaliação das mesmas. Por sua vez, os programas desenvolvidos para simulação de processos raramente permitem a monitorização e controlo do sistema real uma vez que não têm implementadas funcionalidades de interoperabilidade, o que torna difícil a sua integração com outras aplicações e limita a utilização dessas ferramentas.

Neste trabalho prático pretende-se colmatar a limitação referida, desenvolvendo ferramentas de controlo e monitorização que também permitam a simulação, servindo de suporte ao planeamento e otimização de fluxos, recursos, *layouts*, etc. Como caso de estudo, propõe-se a utilização da célula de fabrico existente no laboratório de robótica do ISEL, passível de ser integrada num sistema CIM, constituída por um braço robótico, um armazém e um tapete transportador com três estações de trabalho. À célula de fabrico existente foram adicionados alguns equipamentos (braços robóticos) e programas de livre utilização disponibilizados também por parcerias do ISEL.

1.2 Objetivos

Na presente dissertação pretende-se controlar e monitorizar uma célula de fabrico e utilizar o modelo visual desenvolvido para simular alterações de processo. Para cumprir este propósito será necessário:

- Selecionar o *software* de controlo e o *software* de simulação;
- Desenvolver o modelo 3D da célula de fabrico em Tecnomatix;
- Desenvolver o controlador em Matlab;
- Efetuar a integração dos dois sistemas: controlo e monitorização.

Além dos objetivos mencionados, o sistema a desenvolver deve cumprir os seguintes requisitos:

1) deve haver uma integração entre o *software* de controlo e o de monitorização;
2) o modelo de monitorização deve ser efetuado em 3D, com a representação geométrica o mais fidedigna possível do sistema real, de modo a melhor a experiência do utilizador;

3) todo o sistema deverá ser flexível, permitindo alterações de fluxo ou *layout*, no caso de ser necessário efetuar ajustes no processo;
4) o sistema deve ficar apto para a integração com o sistema real, funcionando como um gémeo digital.

1.3 Estrutura

Esta dissertação é composta por seis capítulos. No primeiro pretende-se apresentar o enquadramento do tema, a motivação para a realização do trabalho e definir os objetivos a atingir.

No capítulo 2 será apresentada a revisão bibliográfica relativa aos temas abordados, onde serão introduzidos os conceitos que permitem uma melhor compreensão da componente prática deste projeto.

Em seguida, nos capítulos 3 e 4, são apresentadas as componentes do trabalho prático. No capítulo 3 é feita a apresentação do sistema a modelar e no 4 capítulo é apresentada uma solução de monitorização e simulação.

No capítulo 5 é apresentada a solução de controlo desenvolvida. Para tal foi desenvolvido um controlador em Matlab. Devido à necessidade de introduzir no modelo variáveis responsáveis pela comunicação como o controlador, esta solução obrigou também à criação de um novo modelo de monitorização e simulação.

No capítulo 6 será feita a discussão dos resultados obtidos e do trabalho futuro a desenvolver.

2. Enquadramento Teórico

2.1 Evolução tecnológica

Para o enquadramento deste trabalho torna-se indispensável apresentar detalhadamente os desenvolvimentos tecnológicos que permitiram a criação das ferramentas de suporte à indústria que temos hoje disponíveis. Este desenvolvimento tecnológico foi assinalado por evoluções tecnológicas em áreas específicas que revolucionaram o processo produtivo e, portanto, ficaram conhecidas como revoluções industriais. Na Figura 1 é possível observar as revoluções industriais ocorridas até ao momento e as respetivas áreas nas quais o desenvolvimento foi mais significativo.

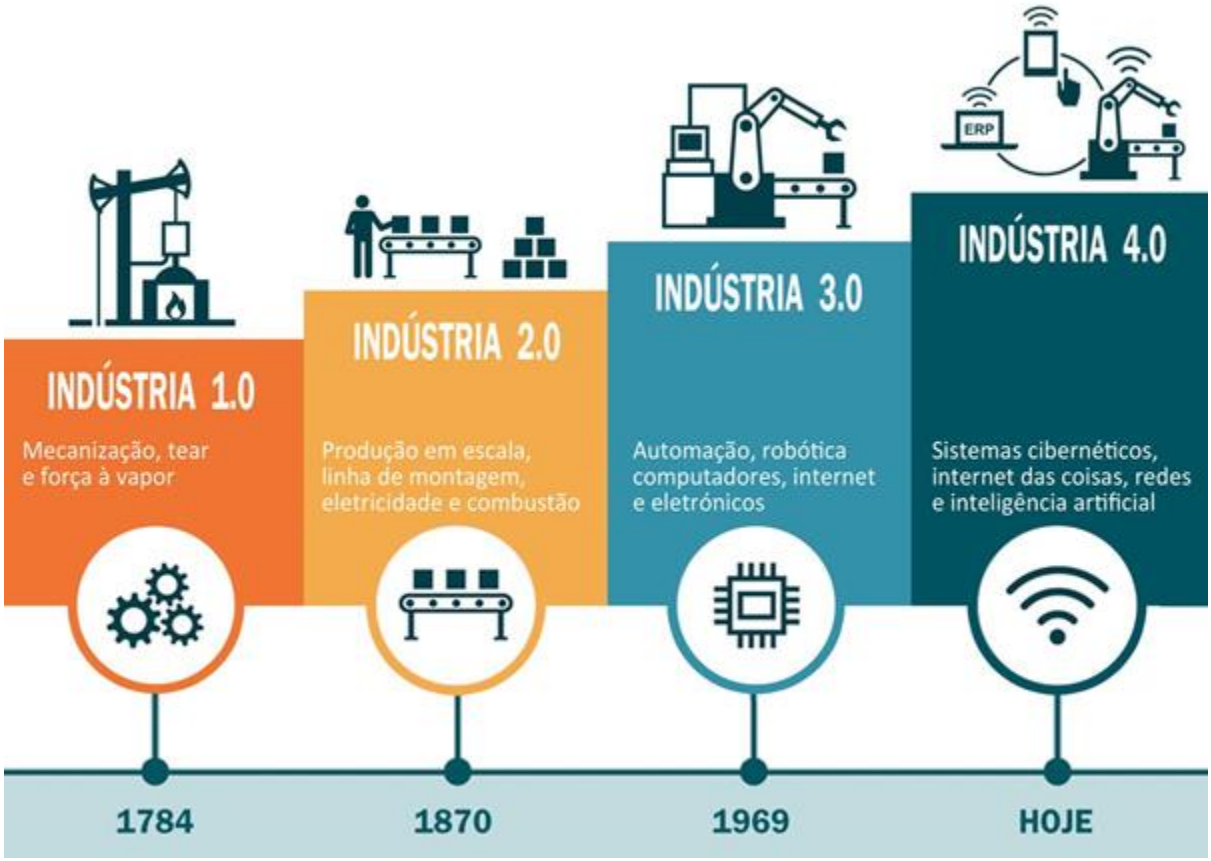


Figura 1 - Cronologia das revoluções industriais e respetivas áreas de maior desenvolvimento [2].

Na história ficaram registadas até ao momento quatro revoluções industriais dando origem a quatro períodos distintos, que se abordam de seguida. Até ao século XVIII a indústria era caracterizada por ser maioritariamente manual. A revolução industrial realizada nesse século foi marcada pela mecanização e pela introdução do motor a vapor de James Watt, resultando num crescimento nas indústrias têxteis e siderúrgicas. Foi necessário cerca de um

século para se voltar a registar uma mudança significativa na indústria. Desta vez, a eletrificação e a implantação de produção em massa, popularizada por Henry Ford, marcariam a nova geração. A produção em massa permitiria ainda a construção de ferrovias e desenvolvimentos na área da química. O final deste período ficou assinalado pela Primeira Guerra Mundial, que abradou o desenvolvimento tecnológico. Após o término da Segunda Guerra Mundial (1945) a evolução tecnológica ganhou força e, na década de 70, surgiram os computadores, controladores lógicos e equipamentos de telecomunicação que permitiram a criação de uma nova geração industrial. Os princípios da internet, automação e robótica também foram estabelecidos na segunda metade do século XX.

Atualmente, as evoluções tecnológicas originaram um novo período designado por Indústria 4.0. Esta designação teve origem no projeto realizado numa parceria entre as entidades privadas *Robert Bosch GmbH, German Academy of Science and Engineering* e o Governo alemão, com o objetivo de desenvolver um plano de informatização da manufatura do país. O relatório final que delineava os princípios da Indústria 4.0 ficou finalizado em 2013 [3].

Segundo Kagermann a Indústria 4.0 define-se como: "*Embedded systems, production, logistics, engineering, coordination and management processes as well as internet services, which by courtesy of sensors directly collect physical data and utilizing actuators influence physical procedures. These systems are interconnected via digital networks, use worldwide available data and services and are equipped with human-machine interfaces.*" [3]. Nesta descrição é possível identificar as 2 tecnologias mais marcantes da Indústria 4.0: os Sistemas Ciber-Físicos (CPS) e a Internet of Things (IoT) [4],[5].

Segundo a National Science Foundation (NSF) os CPSs podem ser definidos como: *"The systems where physical and software components are deeply intertwined, each operating on different spatial and temporal scales, exhibiting multiple and distinct behavioural modalities, and interacting with each other in a myriad of ways that change with context."* [6].

Embora a ligação de dispositivos através da internet já existisse de forma primitiva, a IoT promoveu o desenvolvimento de sistemas mais eficazes para a deteção de falhas e monitorização de processos remotos para a realização de manutenção corretiva e preventiva. Esta capacidade gerou o conceito de "supervisão e controlo de sistemas através da *web*", que permite aos utilizadores finais aceder a dados em tempo real e controlar equipamentos através

de um *web browser*, funcionalidades que caracterizam a última geração de sistemas SCADA [7].

Atualmente, alguns autores já anteveem a próxima geração da indústria, a indústria 5.0. Embora o conceito ainda esteja em aberto, duas áreas de foco se destacam. A primeira área foca-se em potenciar a criatividade e a capacidade de decisão humana na colaboração com os equipamentos potentes, inteligentes e precisos já existentes, trazendo a sensibilidade humana de volta à manufatura (*human-centricity*) [8],[9]. Na segunda área são abordadas questões como a sustentabilidade, tanto a nível social como ambiental [10]. É esperado que a Indústria 5.0 promova a integração da produção com elevada eficiência das máquinas com o pensamento crítico dos humanos. Um exemplo desta integração será o desenvolvimento de *cobots* (robôs colaborativos). A interação de humanos e máquinas no processo produtivo permitirá também uma personalização em massa dos produtos de acordo com o gosto e as necessidades dos clientes. Segundo Saeid Nahavandi [8] a Indústria 5.0 impulsionará melhorias na qualidade da produção pela atribuição das tarefas monótonas, repetitivas e de elevado esforço a robôs/máquinas, enquanto as tarefas que requerem maiores capacidades cognitivas são reservadas aos humanos. Na Figura 2 [11] apresenta-se a tipologia dos operadores desta nova geração. Esta tipologia tem como objetivo de expandir as capacidades dos operadores com recursos a novas ou existentes tecnologias.

Figura 2 - Tipologia de operadores.

Devido à atual situação ambiental, o respeito pelo planeta tornou-se uma prioridade. Assim, torna-se essencial desenvolver processos sustentáveis, envolver processos circulares, reciclar e reutilizar, de modo a reduzir a produção de resíduos. Também a redução do consumo de energias provenientes de fontes não renováveis e emissão de gases de estufa terão de ser reduzidos. A gestão dos recursos energéticos deve ser realizada de forma ponderada, privilegiando os renováveis. Nesta vertente, os acordos internacionais e leis ambientais, a inteligência artificial e a manufatura aditiva assumem um papel essencial para atingir os objetivos propostos [10].

Na Figura 3 é possível observar algumas das tecnologias e inovações que, integradas com o fator humano, podem ajudar uma indústria a melhorar a sua eficiência atingindo os objetivos propostos. Mais informação acerca destas tecnologias pode ser consultada em [12].

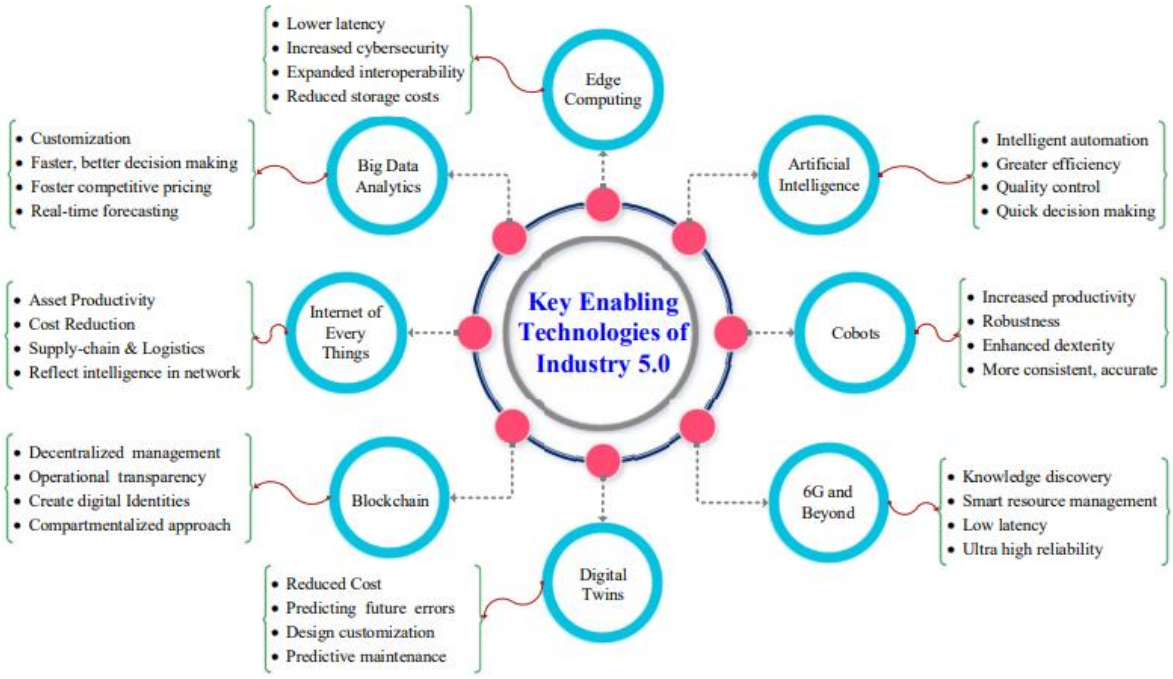


Figura 3 - Tecnologias e inovações potenciadoras da Indústria 5.0 [12].

2.2 SCADA

Os sistemas SCADA (*Supervisory Control and Data Acquisition*), como o nome indica, são sistemas de controlo, supervisão e aquisição de dados utilizados para a monitorização e/ou controlo de diversos sistemas industriais, processos de fabrico e seus equipamentos assim como de infraestruturas: redes de distribuição de energia, estações de tratamento de água, oleodutos etc. Os sistemas SCADA têm como principais funções a redução de *downtimes*, emissão de alertas e minimização de consequências em situações críticas, a recolha de dados para posterior análise e o suporte à decisão e controlo remoto de equipamentos [13].

A pirâmide da automação (Figura 4) consiste numa representação gráfica dos diferentes níveis de automação existentes numa indústria [14] e está segmentada em 5 níveis: campo, controlo, supervisão, planeamento e gestão. Esta divisão facilita a compreensão da complexidade dos sistemas de automação assim como dos componentes e das suas funções [15]. Os sistemas SCADA integram os três níveis inferiores da pirâmide da automação.

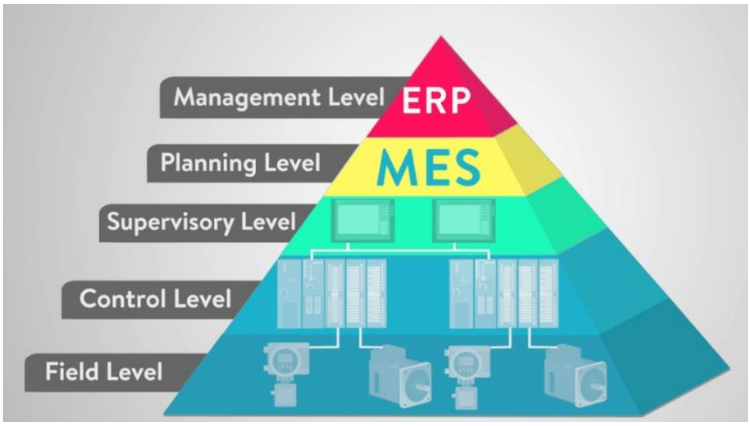


Figura 4 - Pirâmide da automação [16].

- **Nível 0 – Nível de campo**

O nível de campo é constituído pelos diversos dispositivos presentes no chão de fábrica que têm contacto direto com processo a desempenhar. É possível identificar dois grupos principais: os dispositivos de entrada que emitem os *inputs* para o controlador ou equipamento de monitorização tais como sensores de pressão, temperatura ou posição, etc., e os dispositivos de saída ou comando, os quais são responsáveis pela execução de ações de controlo ou resposta como válvulas, motores, luzes, alarmes, etc.

- **Nível 1 – Nível de controlo**

Este nível é composto por elementos de controlo, habitualmente PLCs (*Programmable Logic Controller*). Os controladores recebem os *inputs* do nível de campo (ex: sensores, botões, manípulos, etc.) e produzem os sinais de *output* para os atuadores e outros dispositivos de saída.

- **Nível 2 – Nível de supervisão**

O nível de supervisão é composto por todos os sistemas de monitorização existentes. Este nível abrange a monitorização dos sistemas SCADA, normalmente compostos por um computador mestre que executa o *software* SCADA e por terminais que permitem aos utilizadores a monitorização e controlo dos processos, usualmente conhecidos por HMI (*Human-Machine Interface*).

- **Nível 3 – Nível de planeamento**

Este nível é responsável pelo planeamento de produção e as atividades acessórias à mesma, como logística, aprovisionamento, manutenções, etc. A partir deste nível, a automação baseia-se maioritariamente em *software* instalado em computadores ligados à rede.

Além de enviar dados de planeamento para os níveis inferiores, o nível de planeamento também recebe dados recolhidos da monitorização de forma a ajustar o planeamento efetuado.

- **Nível 4 – Nível de gestão**

Este nível baseia-se num sistema ERP (*Enterprise Resource Planning*), para permitir que os gestores conheçam o estado de todas a secções da empresa: produção, vendas, aprovisionamento, contabilidade e fianças, recursos humanos, etc. A integração de um ERP permite uma maior organização a nível de gestão, tanto da empresa como a nível de fábrica.

2.2.1 Elementos do sistema SCADA

Os sistemas SCADA são compostos por diversos componentes dos três primeiros níveis: campo (ou "chão de fábrica"), controlo e supervisão. No entanto, trata-se de um sistema centralizado, uma vez que toda a informação é direcionada para um computador central. Um sistema SCADA típico de manufatura é composto pelos seguintes componentes: computador mestre (ou conjunto de computadores responsáveis pelas mesmas funções), PLC (*Programable*

Logic Controller), HMI (*Human Machine Interface*), sensores, atuadores e elementos de suporte à comunicação [13] (Figura 5).

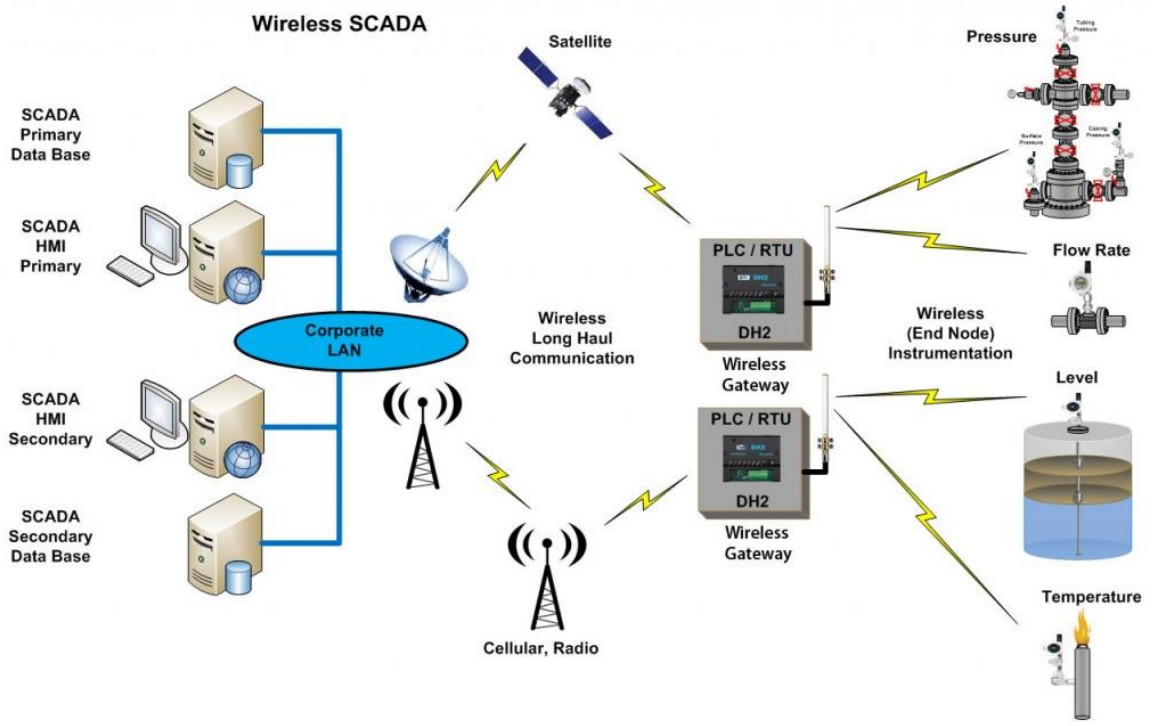


Figura 5 - Exemplo da arquitetura de um sistema SCADA [17].

Habitualmente, os sistemas SCADA são fornecidos por integradores num formato "chave na mão". O integrador pode ser o próprio fabricante dos equipamentos (Siemens, ABB, GE, etc.), um representante ou um especialista que trabalhe com diferentes marcas. A integração e programação é, usualmente, adaptada às necessidades do cliente.

- **Computador mestre**

O computador mestre é o componente central do sistema SCADA estando direta ou indiretamente conectado a todos os restantes componentes. É neste computador que o *software* SCADA é instalado e executado, sendo este o responsável pela recolha, processamento e monitorização de dados em tempo real. O computador mestre é normalmente utilizado como interface direta por utilizadores experientes ou pelos desenvolvedores do sistema [7]. Em sistemas distribuídos estas funções podem estar repartidas por diversos computadores (desenvolvido na secção 2.2.2).

- **PLC**

Usualmente, os PLCs são utlizados nos sistemas SCADA para desempenhar funções avançadas de controlo. O PLC é um equipamento eletrónico composto por vários módulos como processador, memórias, módulos de entradas e saídas, comunicação, etc. Este equipamento recebe sinais dos *inputs* e produz os *outputs* de acordo com um programa desenvolvido pelo utilizador, para efetuar o controlo de processos. A utilização de PLC é comum em ambientes industriais, pois são facilmente configuráveis e garantem uma elevada fiabilidade a preços acessíveis [18].

Com a evolução da tecnologia, os equipamentos tornaram-se mais diferenciados de forma a dar resposta à elevada variedade de aplicações e controlo de processos remotos. O RTU (*Remote Terminal Unit*), paralelamente ao PLC, é um dispositivo com microprocessador utilizado na monitorização de processos para receber sinais elétricos de sensores e enviar sinais para os atuadores. Os sinais são processados pelo sistema central. O RTU destaca-se pela capacidade de funcionamento sem ligação à rede elétrica, no entanto, fica aquém relativamente às capacidades de controlo do PLC. É tipicamente utilizado na monitorização de fugas, níveis de reservatórios, sistemas de irrigação, etc. [19].

- **Interface Homem-máquina**

A HMI é um dispositivo responsável pela apresentação dos dados, de forma gráfica e simplificada, podendo ser personalizada de acordo com os requisitos do utilizador e níveis de acesso. Além dos dados, geralmente também apresenta indicações, animações, estados e alarmes relevantes do processo a controlar. A interface é normalmente suportada por um computador, *touch screen* ou página online para acesso remoto [7]. A Figura 6 apresenta o exemplo de uma HMI de um sistema de armazenamento automático – ASRS (*Automatic Storage and Retrieval System*).

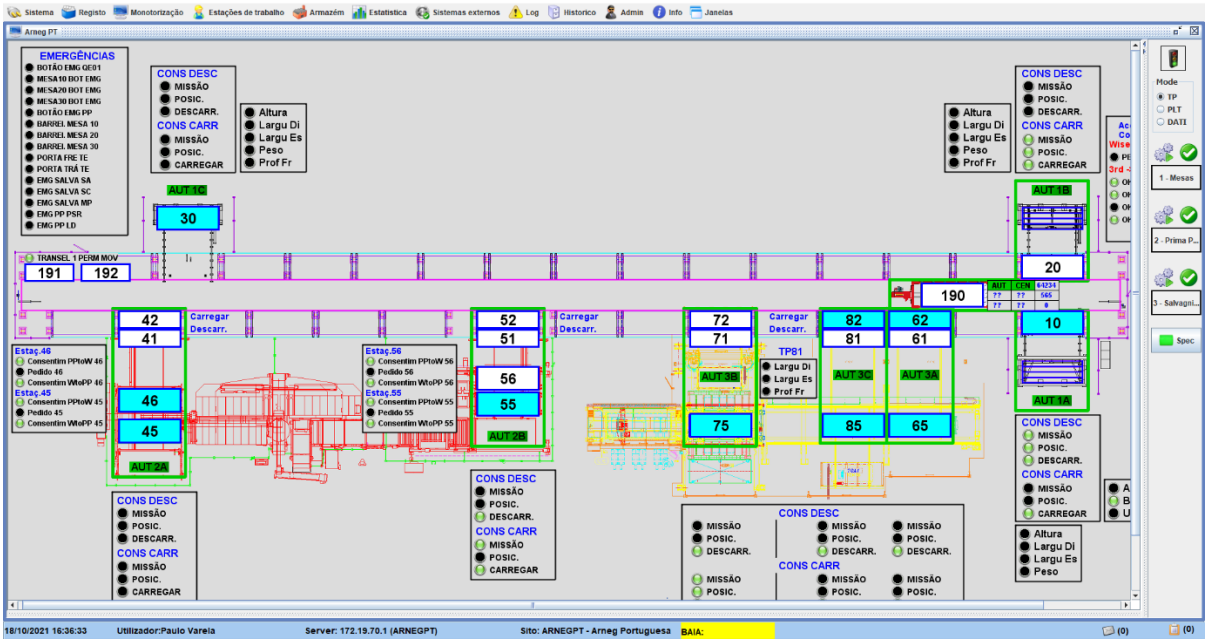


Figura 6 – Imagem de uma HMI de um sistema de armazenamento automático.

- **Sensores e atuadores**

Usualmente são utilizados sensores para recolher a informação do processo e atuadores para efetuar ações de comando. Existem diversos tipos de sensores como barreiras fotoelétricas, balanças, sensores de temperatura, pressão e posição que transformam uma alteração de estado numa alteração da diferença de potencial, resistência ou frequência, cujos sinais são usados como *inputs* no PLC ou num computador. Nos sistemas SCADA que desempenham a função de controlo existem atuadores como válvulas, atuadores lineares, motores, etc. que intervêm no processo, de acordo com a programação que executa ciclicamente [7].

- **Elementos de suporte à comunicação**

Atualmente existem diversos protocolos industriais de comunicação que são utilizados para integrar todos os componentes, como por exemplo *profinet*, *ethercat*, etc. A partir da centralização da informação no computador mestre, é possível acedê-la ou alterá-la a partir de qualquer parte do mundo através de ligações remotas via internet - WAN (*Wide Area Network*) - ou através de rede local - LAN (*Local Area Network*)[7].

2.2.2 Evolução dos sistemas SCADA

Desde o seu aparecimento em 1950, os sistemas SCADA apresentaram desenvolvimentos suportados por avanços tecnológicos a nível de *hardware* (computadores, RTU/PLC, etc.) e *software* (protocolos, linguagens de programação, etc.). Esta evolução ficou marcada em 4 gerações.

- **1ª geração**

Na primeira geração de sistemas SCADA, a informação era centralizada num computador mestre do tipo *mainframe* que requisitava a informação dos diversos RTUs (*Remote Terminal Units*) a ele associados (Figura 7). Uma vez que as redes de comunicação eram praticamente inexistentes, o sistema SCADA era incapaz de comunicar com outros sistemas tornando-o independente. A aquisição de dados era feita na forma de sinais analógicos no local para um RTU que emitia os *outputs* para controlo do processo [13].

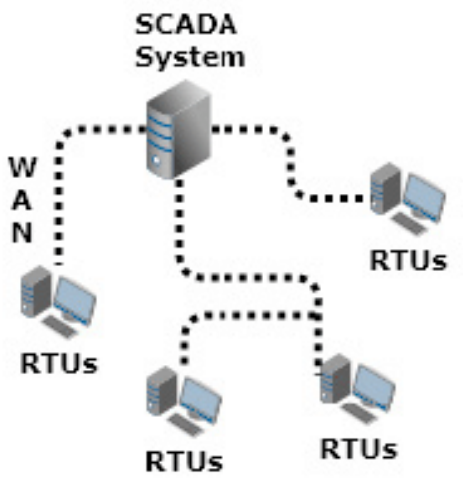


Figura 7 - Representação gráfica de um sistema baseado em *Monolithic* SCADA [7].

As redes dos sistemas SCADA desta geração utilizavam protocolos de comunicação proprietários (baseados em linhas de rádio e telefone), o que limitava a quantidade e velocidade de dados partilhados, assim como a compatibilidade entre componentes. Comparativamente com os sistemas atuais, a monitorização de processos nesta geração era bastante limitada uma vez que era efetuada através interfaces gráficas, que funcionavam à base de indicadores luminosos e com uma latência significativa (Figura 8).

Figura 8 – HMI primitiva utilizada para monitorização de uma estação de distribuição de energia elétrica.

- **2ª geração**

O desenvolvimento de computadores de dimensões e custos mais reduzidos e o desenvolvimento das LAN (década de 1970) impulsionaram uma nova geração, designada por *Distributed* SCADA. Esta designação reflete a possibilidade existente nestes sistemas de distribuir tarefas por diversas estações do sistema através da LAN. Durante esta geração foram também introduzidos os PLCs, permitindo assim o controlo baseado em *software.*

A existência de uma LAN permitiu que a comunicação passasse a ser feita em tempo real uma vez que tem a capacidade de transmitir dados com uma velocidade de comunicação superior [7]. Esta melhorias nas comunicações permitiu ligar estações como HMI, processadores, bases de dados, etc. As comunicações dos restantes equipamentos que se encontravam com uma distância física superior permaneceram suportadas por WAN. Apesar da maior flexibilidade existente nesta geração, os protocolos de comunicação mantiveram-se proprietários, só sendo possível a sua utilização pelo próprio fabricante.

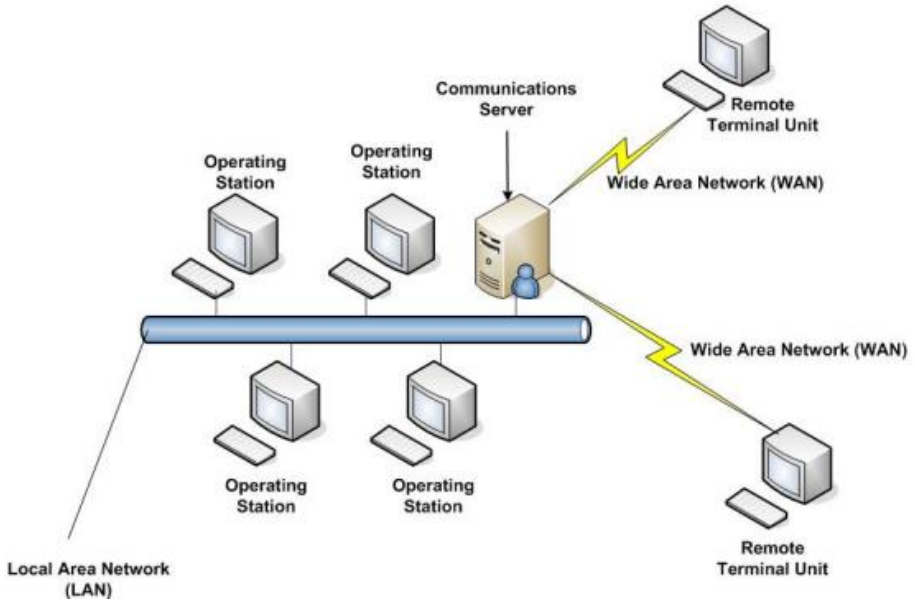


Figura 9 - Representação gráfica de um sistema baseado em *Distributed* SCADA [7].

- **3ª geração**

A 3ª geração, designada por *Networked* SCADA, foi uma geração muito semelhante à anterior, diferindo principalmente na flexibilidade da arquitetura do sistema (Figura 10). Nesta geração o sistema deixou de ser proprietário, passando-se a adotar um sistema aberto, com protocolos *standard*, possibilitando a ligação de dispositivos de diversos fabricantes ao sistema SCADA. Nesta época foram estabelecidos alguns protocolos, como DNP, IEC 61580 e TCP/IP (*Transmission Control Protocol/Internet Protocol*), tornando normalizadas as comunicações entre componentes [13].

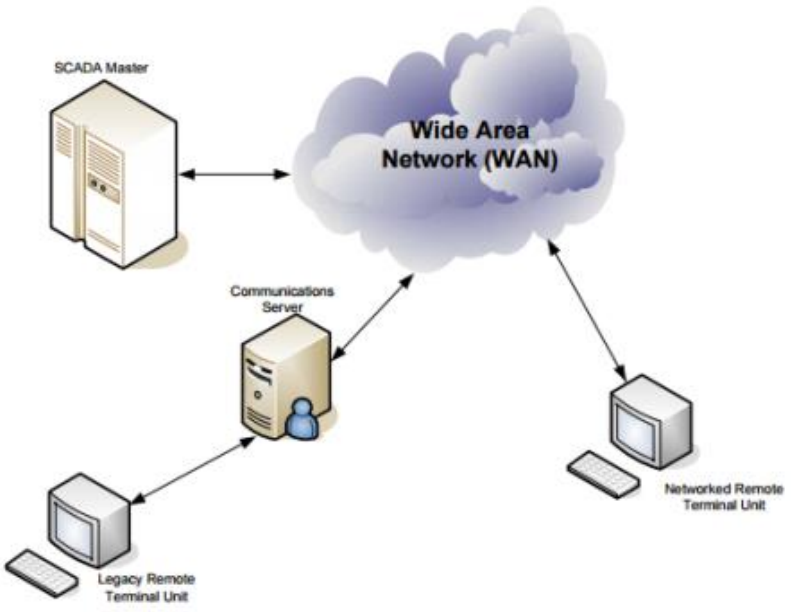


Figura 10 - Representação gráfica de um sistema baseado em *Networked* SCADA [7].

Com a normalização dos protocolos de comunicação, os fornecedores de sistemas SCADA passaram a concentrar-se no desenvolvimento de *software* deixando o *hardware* para produtores de equipamentos. Uma das consequências desta estratégia foi o desenvolvimento das interfaces gráficas muito personalizadas adaptadas à realidade de cada cliente ou utilizador (Figura 11).

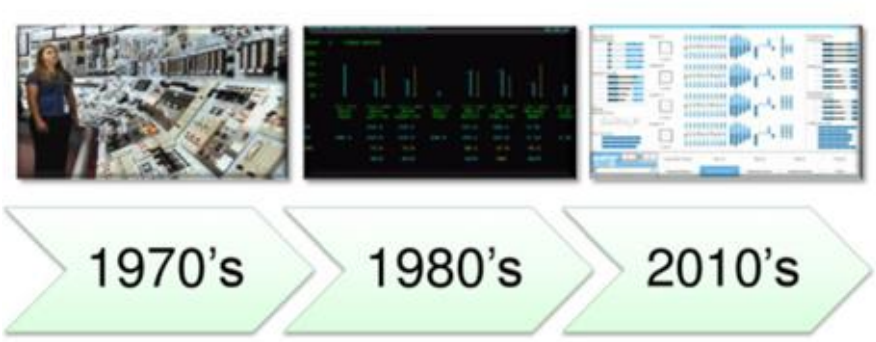


Figura 11 - Evolução das HMIs.

- **4ª geração**

A 4ª geração (*Web-based*), adotada na última década, trouxe aos sistemas SCADA a vertente Web/IoT (*Internet of Things*), permitindo a monitorização e controlo de processos a partir de qualquer parte do mundo, através do protocolo de comunicação Websocket. Com esta tecnologia tornou-se possível aceder às HMIs do sistema utilizando um explorador de internet a partir de um computador, *laptop* ou *tablet*. Nos últimos anos, esta tecnologia transitou também para *smartphones*, através do desenvolvimento de aplicações e da integração em *cloud* dos sistemas SCADA. Com a mobilidade conferida aos sistemas nesta geração, a segurança de dados e da informação tornou-se uma preocupação, pois os sistemas passaram a estar diretamente ligados à internet e, portanto, sujeitos aos perigos de ataques e vulnerabilidades existentes (Figura 12) [7].

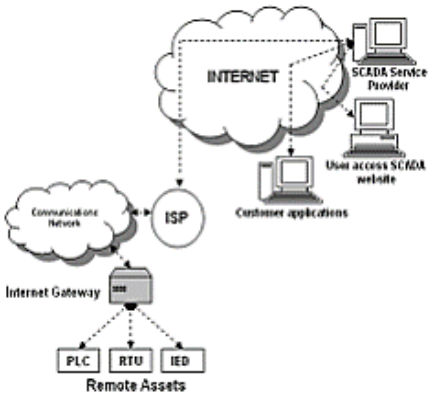


Figura 12 - Representação gráfica de um sistema baseado em Web SCADA [20].

2.3 Simulação de Sistemas

O ato de simular consiste na reprodução de operações de processos reais ou sistemas ao longo do tempo [21]. A simulação de um processo requer a utilização de modelos que contêm as características essenciais e comportamentos do sistema real, representando a sua evolução ao longo do tempo.

Atualmente, devido às elevadas capacidades de processamento dos computadores, a simulação é maioritariamente realizada em *software*, acelerando assim a obtenção de resultados. As simulações também podem ser feitas no sistema real, como em caso de treinos ou testes. No entanto, esta abordagem costuma obrigar à paragem do sistema real, o que traz inconvenientes para a organização.

2.3.1 Técnicas de simulação

As simulações podem ser utilizadas nas mais diversas áreas como a otimização de processos, segurança, educação, visualização de funcionamento de sistemas mecânicos, estudos económicos, etc., permitindo o estudo da situação atual de uma forma segura e sem perturbar o sistema real [22]. As técnicas de simulação mais utilizadas são as seguintes:

- Simulação de eventos discretos -DES (*Discrete Event Simulation*);
- Simulação virtual em ambiente 3D;
- Dinâmica de sistemas -SD (*System Dynamics*);
- Modelação baseada em agentes -ABM (*Agent-Based Modelingl*);
- Simulação Inteligente (baseada em inteligência artificial);
- Rede de Petri;
- Simulação de Monte Carlo -MCS (*Monte Carlo Simulation*).

A Tabela 1 apresenta alguns exemplos da aplicação das técnicas referidas na manufatura [23]. No Anexo 1 é possível consultar a tabela completa.

Tabela 1 – Exemplos de aplicações de técnicas de simulação em manufatura. [23]

Técnica de Simulação	Exemplo
DES	Balanceamento de linhas
DES, SD, Monte Carlo, Rede de Petri	Planeamento de capacidade, aumento de recursos, melhoria de processos

Simulação virtual 3D	Comparação de *layout* de células de manufatura
DES, ABS, Rede de Petri	Planeamento de rotas, gestão logística, gestão de tráfego
DES, Monte Carlo	Gestão de stocks, custo de armazenamento, stocks de segurança, quantidade ótima de encomenda
DES	Design de sistemas Kanban, *just in time*
DES, SD, ABS, Monte Carlo, Rede de Petri	Melhoria de processos, resolução de problemas de equipamentos, design de infraestruturas, medição de performance
DES, ABS, Distributed	Planeamento de inventário, *stocks* de segurança, quantidade ótima de encomenda, *bottlenecks*
DES	Alocação de recursos para melhoria de fluxo de materiais
DES	Capacidade, fiabilidade, sequenciamento de operações, planeamento de produção, minimização de tempos de paragem
DES, SD	Gestão da qualidade, controlo e melhoria, qualidade de fornecedores, qualidade total e *lean*

Relativamente às técnicas de simulação, a simulação de eventos discretos representa 40% dos artigos analisados na "Simulation in manufacturing and business (2009)" [23] onde se conclui que o DES é a técnica mais utilizada na indústria (Figura 13). Como observado na Tabela 1, o DES tem uma ampla variedade de aplicações ao nível de gestão, planeamento e engenharia, devido ao seu suporte na tomada de decisão, tática e operacional. A sua utilização proporciona um elevado nível de detalhe do processo, utilização de recursos e tempos de espera numa simulação de curto prazo [24].

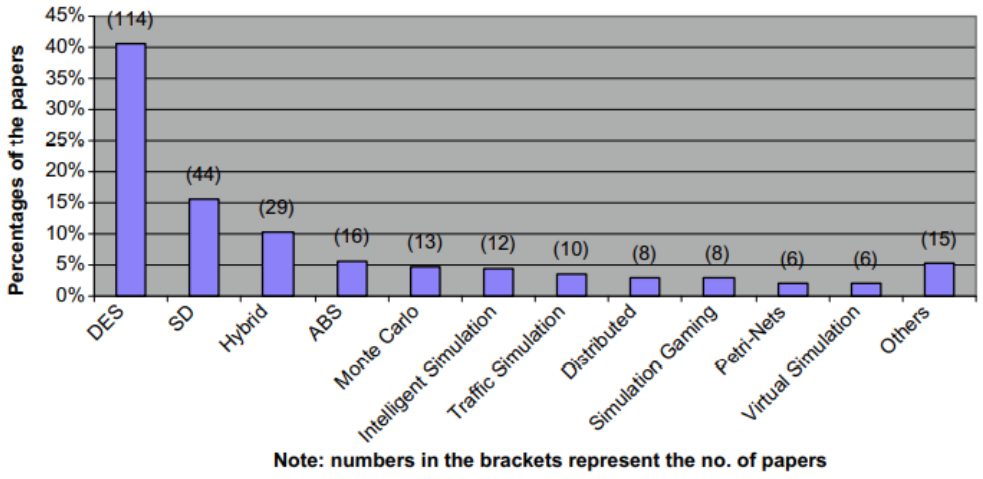


Figura 13 - Número de artigos analisados por técnica de simulação [23].

O caso prático em estudo pretende desenvolver uma ferramenta de simulação para suporte ao planeamento, controlo, monitorização e otimização de uma célula de manufatura. Devido à natureza da simulação, do tipo sequência de acontecimentos para obter uma boa visualização do processo, a simulação virtual 3D de eventos discretos (simulação DES em ambiente 3D) torna-se a melhor opção.

Uma simulação de eventos discretos modela o funcionamento de um sistema como uma sequência de eventos no tempo tendo como referência o funcionamento dos ponteiros dos segundos de um relógio [25]. Com este objetivo, é possível adotar uma de duas abordagens. Na primeira, a cada tique do relógio ocorre um evento marcando assim uma mudança de estado no sistema; caso nenhum evento ocorra é considerado um evento nulo e não existe mudança no sistema. Na segunda abordagem, a qualquer instante não conhecido previamente, o evento anuncia o seu acontecimento marcando a mudança de estado do sistema [26].

2.3.2 Análise e avaliação de *software* de simulação

Para a simulação de DES, sobretudo em ambiente 3D, torna-se importante a escolha do *software* a utilizar. Diversos autores (Hlupic [27], Paul, Nikoukaran [28]) propõem métodos para a seleção de um *software* de simulação de sistemas, no entanto, as ligeiras diferenças entre eles não são significativas. Assim, de seguida são apresentados os passos para o método segundo Bard [25]:

1) **Estabelecer requisitos de modelação**

   Os requisitos para a modelação devem ser definidos pela organização ou equipa responsável, abrangendo aspetos como: 1) o número e tipo de utilizações (para uso geral ou por pessoal especializado), 2) aplicação geral ou específica e 3) se é para modelação simples e rápida ou complexa e detalhada, requerendo diferentes níveis de flexibilidade.

2) **Pesquisar e listar programas**

   Tendo estabelecido os requisitos gerais, é necessário definir uma lista de potenciais programas através de diversas fontes desde websites, vendedores, consultores especializados, literatura ou até conferências. Deve-se garantir que todos os programas selecionados se enquadram nos requisitos previamente estabelecidos. Nesta etapa é necessária especial precaução pois os programas são ferramentas em constante atualização.

3) Estabelecer critérios de avaliação

De forma a avaliar as opções obtidas no passo anterior, é necessário definir critérios de avaliação. Estes critérios devem ser selecionados e adaptados de acordo com as necessidades e requisitos, de forma idêntica ao passo 1. A Tabela 2 apesenta alguns critérios que podem ser utilizados.

Tabela 2 - Exemplo de critérios para seleção de *software* de simulação.

Critérios
Características gerais
Compatibilidade
Integração com outros sistemas
Segurança de dados
Velocidade de simulação
Eficiência
Idiomas
Aspetos Visuais
Ícones
imagens de fundo
Objetos 3D
Animações 3D
Qualidade visual
Realidade virtual
Aspetos lógicos
Modelos incorporados
Programação
Atributos e variáveis
Regras de fluxo
Regras de fila de espera
Input
Input de dados
Output
Exportação de dados
Facilidade de utilização
Interface de utilizador
Construção de modelos
Visualização em tempo real
Aplicação para dispositivos móveis
Suporte

Informação disponível
Capacidade do vendedor
Custos
Custo de aquisição
Implementação
Custo de manutenção
Extras

4) Avaliar os programas em relação aos critérios estabelecidos

Depois de escolhidos os critérios, todos os potenciais programas selecionados devem ser avaliados em relação a estes. Aconselha-se a utilização de uma escala, de 0 a 5 por exemplo, de forma a indicar o nível de reposta ao critério em questão. Neste passo é comum surgirem dúvidas e questões. Nesta situação é aconselhado esclarecimento através de manuais/documentação, vendedores, ou demonstrações do *software*.

5) Seleção de *software*

O *software* pode ser selecionado com base na avaliação efetuada no passo anterior. Esta escolha pode ser subjetiva se for apenas baseada na comparação entre avaliações ou objetiva se for possível calcular uma pontuação para cada *software*, selecionando-se o que obtiver melhor pontuação. Deve-se, nestes casos, efetuar a ponderação dos critérios para que os pesos sejam tidos em conta na avaliação. Esta ponderação pode ser definida através de percentagem, médias ponderadas ou AHP (*Analytic Hierarchy Process*) por exemplo.

Esta metodologia será aplicada na seleção do software de simulação para utilização no caso de estudo (secção 4.1).

3. Caso de Estudo

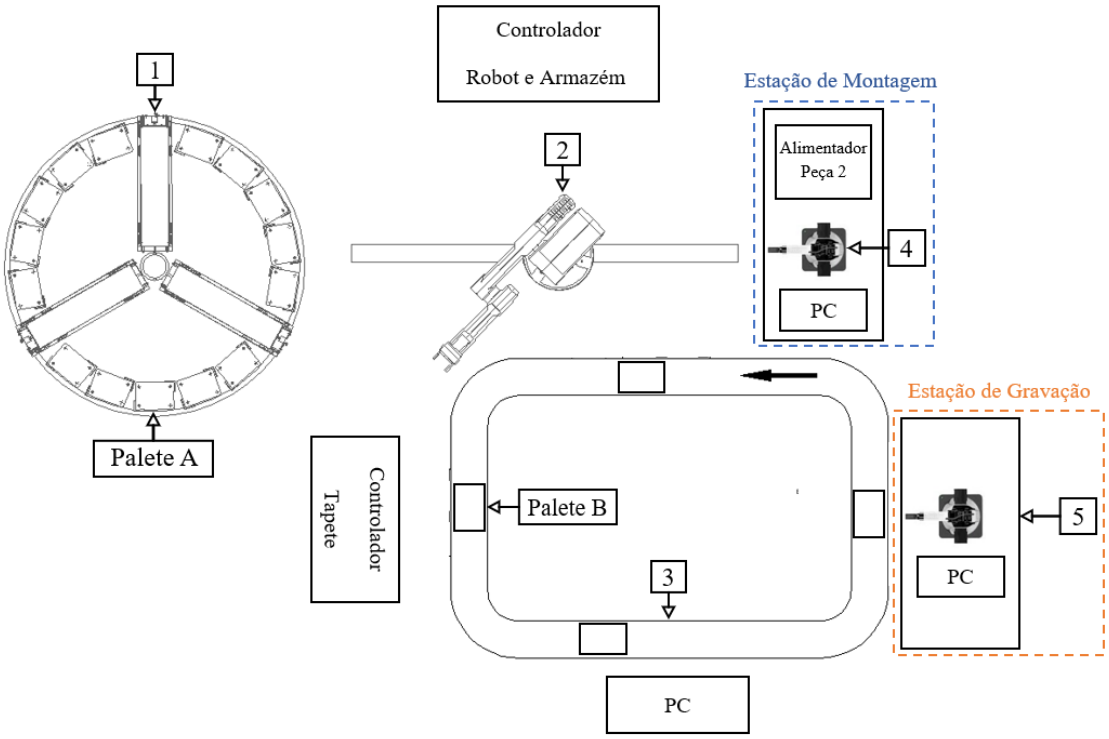
Este trabalho tem como principal objetivo a criação de um modelo virtual representativo da célula de fabrico automatizada existente no Laboratório de Robótica do Departamento de Engenharia Mecânica (DEM) do ISEL. Esta célula de fabrico foi produzida pela Eshed Robotec [29] e instalada em 2000. Tem como principal objetivo servir de suporte educacional aos alunos do DEM, oferecendo uma formação *hands-on* na área da robótica industrial e automação. O sistema é bastante flexível, permitindo o desenvolvimento de trabalhos de programação, estudo de processos de manufatura, manutenção industrial e otimização de sistemas, *layout* e utilização e recursos. Além da célula Eshed, foram ainda incluídos dois Dobots de forma a alargar os recursos disponíveis.

Pretende-se que o modelo a desenvolver reproduza as funcionalidades dos diversos equipamentos da célula de fabrico e possa ser configurado com vista a simular diferentes processos de fabrico, usando os recursos existentes na célula. Para além da simulação *offline*, pretende-se também obter um modo de operação em que o simulador fique apto a comunicar com o sistema real por forma a fazer-se a visualização e monitorização do sistema real em execução, funcionando como um gémeo digital.

Neste capítulo faz-se a apresentação dos diversos componentes que constituem a célula de fabrico, focando-se as suas principais características técnicas, princípios de operação, interfaces de comunicação, entre outros.

3.1 Componentes da célula de fabrico

A Figura 14 representa o conjunto dos componentes que integram o sistema e a sua disposição no espaço de trabalho (*layout*). A designação de cada componente utilizada na Figura 14 será considerada como a sua identificação ao longo deste documento.



1. Armazém 2. Scorbot 3. Tapete
4. Dobot 1 5. Dobot 2

Figura 14 – *Layout* da célula de fabrico.

A Figura 15 apresenta uma imagem real da célula flexível de fabrico que se pretende modelar. Nesta figura não se encontram os Dobots uma vez que a implementação do layout não se encontra concluido.

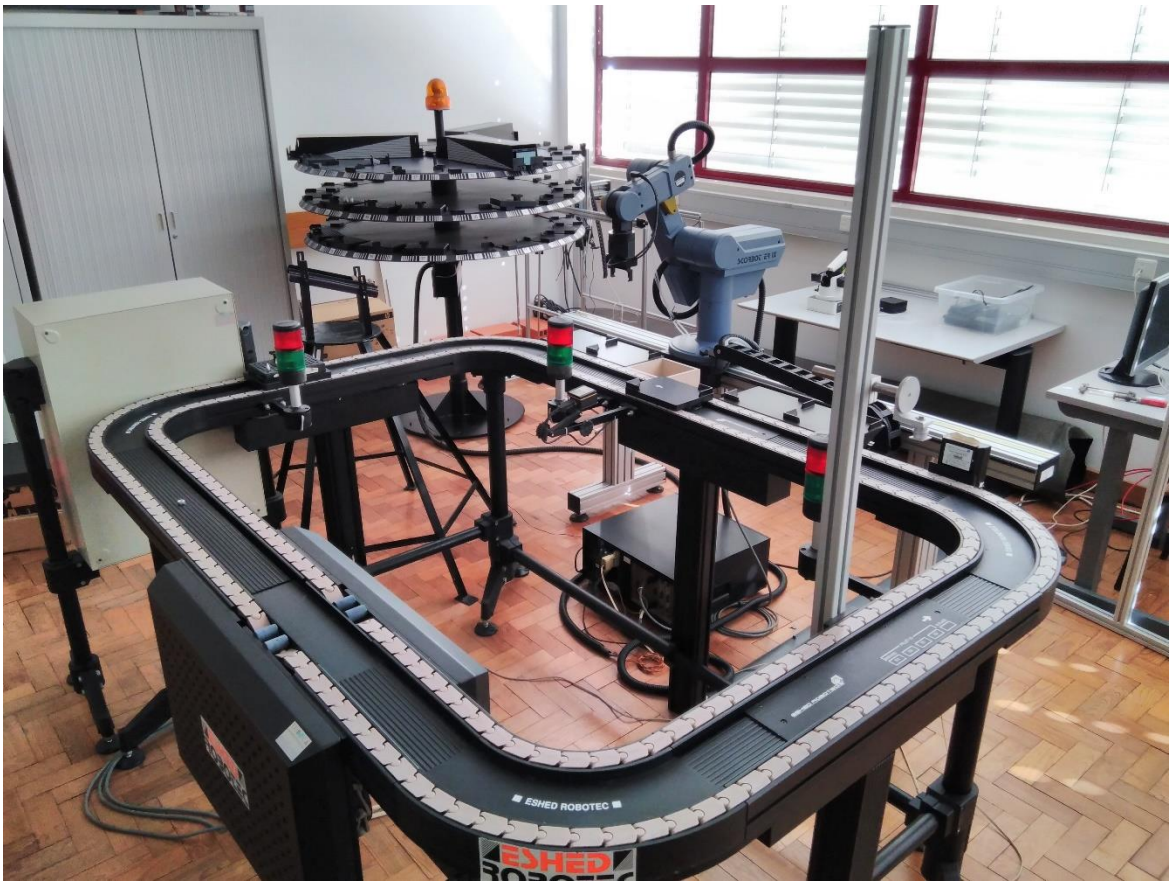


Figura 15 - Célula de fabrico real.

3.1.1 Braço manipulador Scorbot ER-IX

O Scorbot ER-IX (Figura 16) é um robô manipulador do tipo braço articulado vertical com 5 eixos de rotação, que pode ter como elemento terminal uma garra elétrica ou pneumática. Na configuração presente neste caso de estudo, o robô foi instalado sobre um eixo linear com 1,8 metros de comprimento, que expande consideravelmente o seu espaço de trabalho. A sua utilização permite que o robô possa manipular materiais e componentes entre várias estações. O controlo deste eixo linear é realizado pelo controlador do robô.

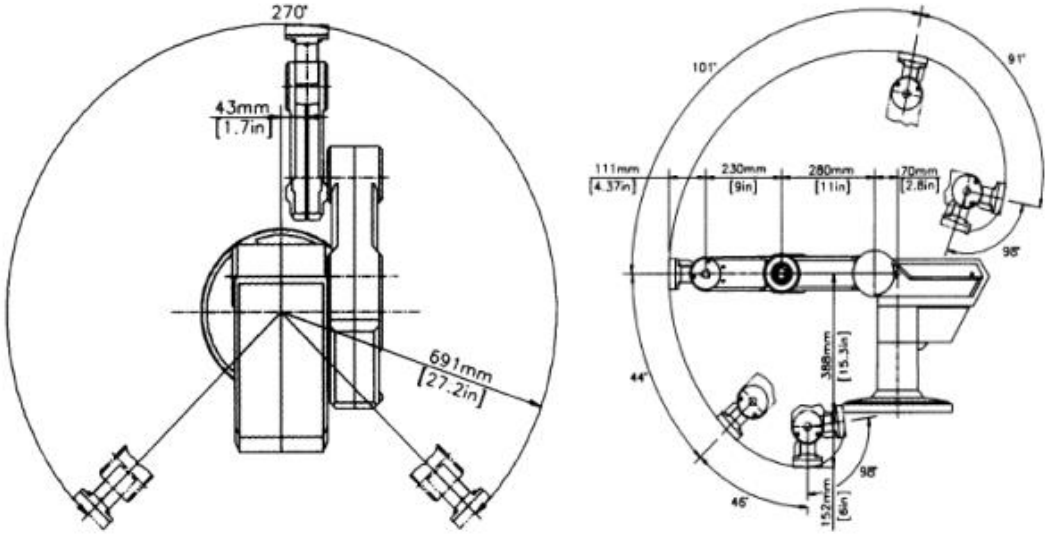


Figura 16 – Scorbot.

3.1.2 Armazém Rotativo ASRS

O armazém ASRS (*Automatic Storage and Retrieval System*), como o nome indica, é um componente que permite o armazenamento de materiais, transportados em paletes (Figura 17).

Figura 17 - Armazém automático.

O armazém existente é do tipo circular, com 3 andares e um eixo de rotação automático. Através da rotação deste eixo, é possível aceder às diferentes células do armazém, usando uma única posição de carga/descarga. Os andares 1 e 2 (numerados de baixo para cima) estão divididos em 36 células que podem receber paletes do tipo A (Figura 18), que circulam pelo sistema fazendo o transporte de materiais.

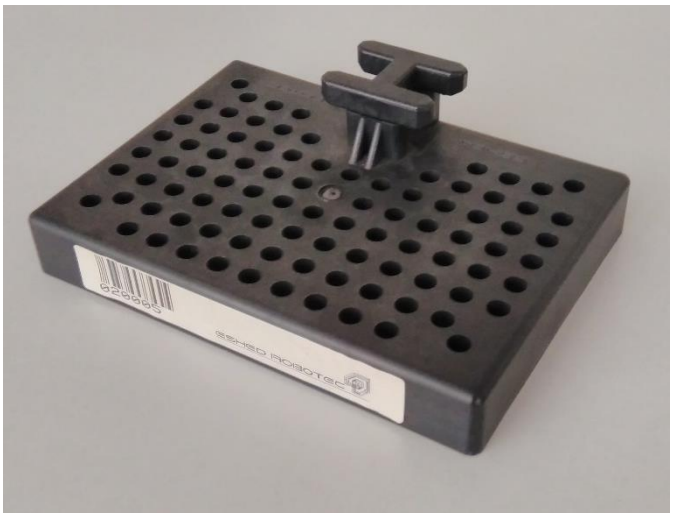


Figura 18 - Palete tipo A.

O último andar contém a alimentação de matéria-prima. Esta configuração é personalizável. A carga/descarga do armazém é feita exclusivamente pelo Scorbot ER-IX, o que exige a movimentação dos eixos de ambos os sistemas: robô e armazém. No sistema instalado, o controlo do eixo do armazém também é realizado pelo controlador do robô, sendo considerado como o 6º eixo do sistema.

3.1.3 Tapete transportador

O tapete transportador utilizado é construído de forma modular com base em troços lineares e curvos. Cada módulo é constituído por uma estrutura metálica que inclui duas calhas paralelas onde deslizam duas correias.

O tapete (Figura 19) está desenhado para transportar 4 paletes do tipo B que assentam sobre as correias e aderem simplesmente pelo atrito estático entre as superfícies. Isto significa que podem ser intercetadas individualmente em cada estação sem necessidade de parar o tapete. Para esse efeito existe, em cada estação, um pino de movimento automático (de atuação

pneumática) que quando levantado impede o avanço da palete. Nessas circunstâncias as correias deslizam por baixo da palete bloqueada. O tapete é modular para permitir a alteração da sua geometria e está equipado com um sistema de *tracking* através da leitura de códigos magnéticos existentes nas paletes do tipo B que nele circulam.

Figura 19 - Tapete Eshed.

As paletes do tipo B estão desenhadas com vista a transportarem as paletes do tipo A como demonstrado na Figura 20.

Figura 20 - Conjunto palete tipo A + palete tipo B.

3.1.4 Dobots

Na célula de fabrico existem também dois robôs do tipo Dobot Magician [30], que se trata de um braço robótico de 4 eixos desenvolvido com o objetivo de proporcionar todas as vantagens de uma Indústria 4.0 num ambiente didático. Isto é evidenciado não só pela conectividade (Bluetooth, WIFI, ligação por fio, controlo por voz), mas também pelas 13 portas de comunicação e o suporte de 20 linguagens de programação. Este robô oode ainda ser controlado por PLC, microcontrolador ou Arduíno.

Existem diversas ferramentas que podem ser acopladas ao robô, através das quais é possível executar funções como:

- Movimentação de cargas;
- Gravação laser;
- Impressão 3D;
- Escrita/Desenho.

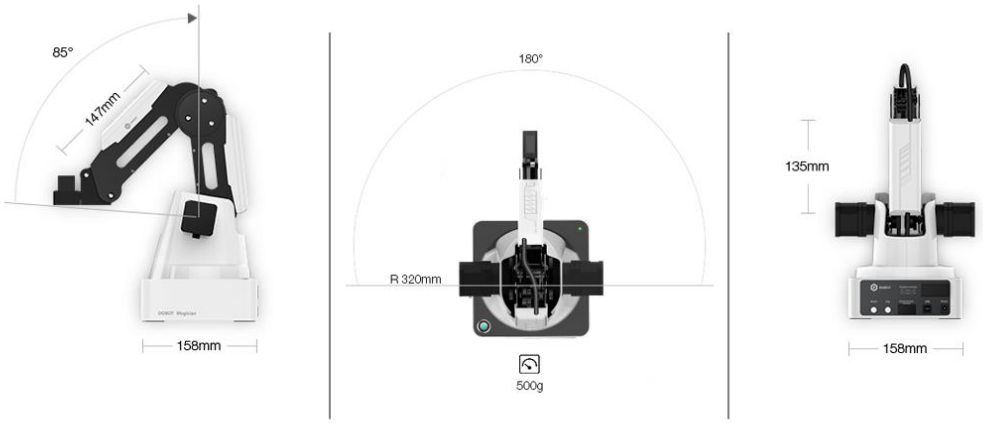


Figura 21 - Dobot Magician.

No sistema descrito na Figura 14, o Dobot 1 utiliza uma ventosa como ferramenta para a montagem de peças. Esta ventosa pode estar equipada com um eixo extra para rotação. Por sua vez, o Dobot 2 utiliza uma ferramenta de gravação de laser.

3.2 Arquitetura de Controlo e Supervisão

Apresentada a configuração física da célula, de seguida descreve-se a estrutura hierarquizada de controlo do sistema, que se apresenta na

Figura 22.

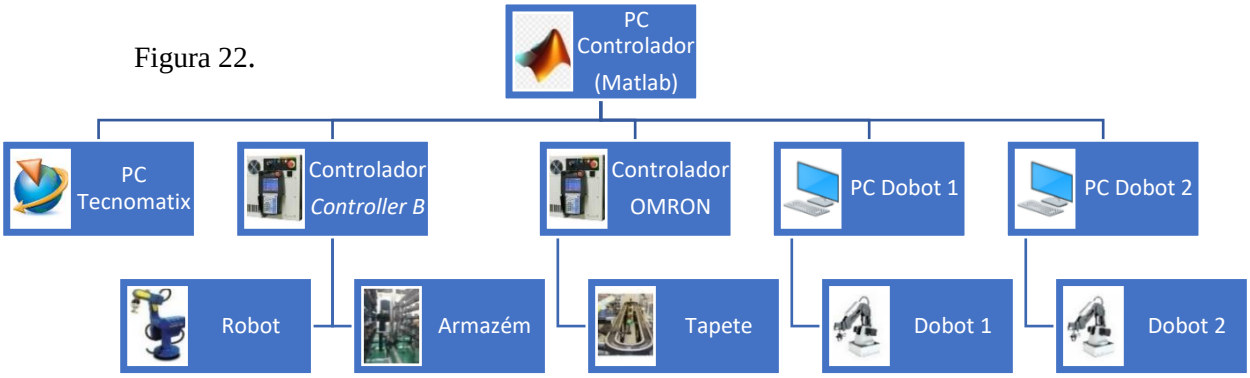


Figura 22 - Hierarquia de controlo da célula automatizada.

No nível de controlo de topo encontra-se um computador que executa um programa de supervisão e comando da célula especificamente desenvolvido em linguagem Matlab no âmbito deste trabalho. Este computador comunica com os diversos controladores que se encontram no nível de controlo abaixo. Cada um dos Dobots é controlado por um computador, com *software* específico desenvolvido pelo fabricante. O controlo do tapete é realizado por um PLC OMRON, responsável pela recolha da informação relativa às paletes B. O controlador "*Controller B*", fornecido pelo fabricante do robô Scorbot ER-IX é responsável pelo controlo desse braço manipulador, da base linear e do armazém, estando definidos como eixos adicionais do sistema robótico.

O Anexo 2 apresenta o esquema elétrico da célula Eshed Robotec, onde é possível verificar que as ligações entre o computador e os controladores assim como dos controladores aos componentes é feita através do protocolo RS232. Para comandar os Dobots, a ligação entre os computadores e os controladores será efetuada por Universal Serial Bus USB. Finalmente, a ligação entre os vários computadores do sistema é realizada através de LAN.

3.3 Descrição do processo

Para efeitos de simulação e de teste do modelo, foi definido um processo de fabrico compatível com os recursos disponíveis na célula de fabrico do caso de estudo (Secção 3.1). Na escolha deste processo procurou-se utilizar todos os equipamentos disponíveis e considerou-se um processo que fosse possível de implementar numa situação real.

O processo envolve a montagem de 2 peças, seguida da gravação a laser de um código de modelo do produto. Neste sentido, foram definidas duas peças, designadas por peça tipo 1 e peça tipo 2. A primeira é passível de ser transportada pelas paletes do tipo A, que se admite conterem guias para receber e acomodar estas peças. As peças do tipo 2 são montadas sobre a peças do tipo 1. A forma como as peças estão acopladas sobre as paletes de transporte é ilustrada na Figura 23.

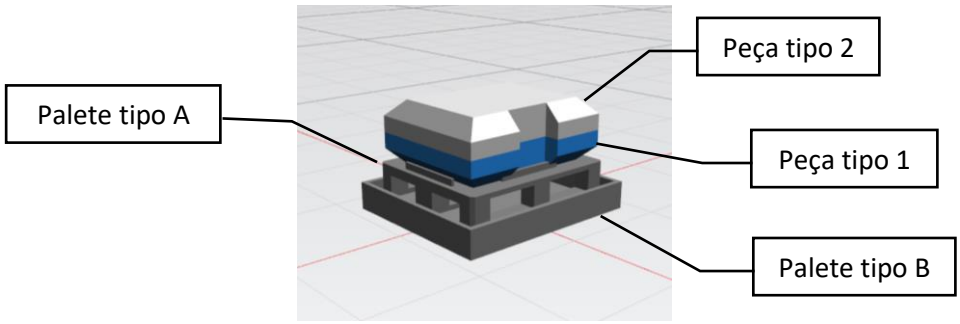


Figura 23 - Paletes e peças do sistema.

O processo descrito em seguida admite que existe em armazém um conjunto de paletes do tipo A com peças do tipo 1. O ciclo inicia-se com o levantamento em armazém de uma palete com peça do tipo 1 e transporte para a estação Dobot 1, por meio do Scorbot. Na estação de montagem é feita a montagem da peça tipo 2 sobre a peça tipo 1. As peças tipo 2 são fornecidas por um alimentador próprio e a montagem está a cargo do Dobot 1. De seguida, a palete tipo A é transferida da estação Dobot 1 para a estação de entrada do tapete, sendo colocada sobre uma palete tipo B. Através do tapete esta palete é transportada para a estação de gravação onde é parada e é feita a gravação a laser pelo Dobot 2. Finalmente, a palete tipo B é levada à estação de entrada do tapete e o conjunto finalizado palete tipo A + peça tipo 1 + peça tipo 2 é transferido para o armazém pelo Scorbot.

Resumindo, a Tabela 3 apresenta a sequência de operações do processo descritas anteriormente e na Figura 24 é possível visualizar os movimentos realizados pelas peças na

célula. Cada movimento ilustrado na Figura 24 está numerado de acordo com a Tabela 3, servindo de referência em capítulos seguintes:

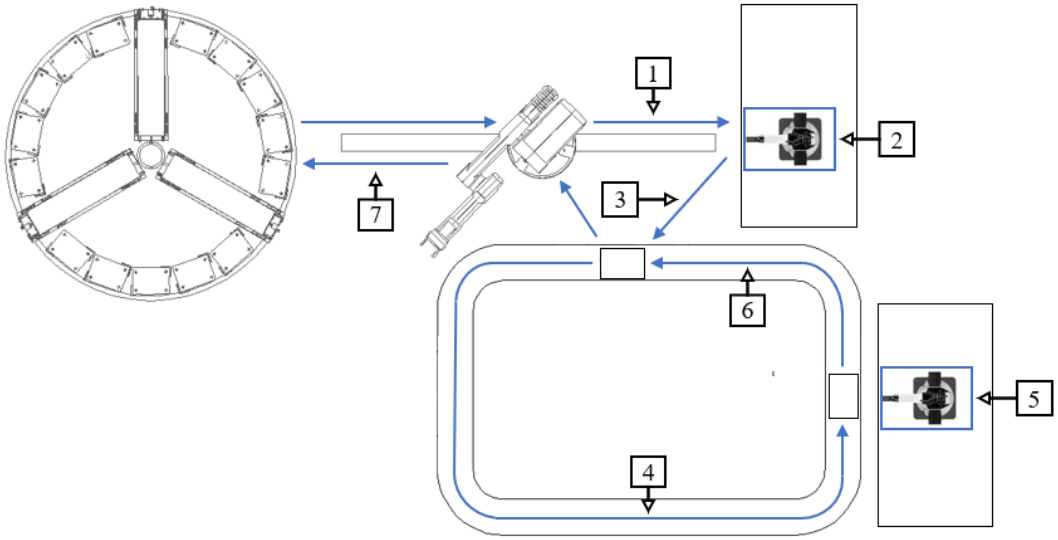


Figura 24 - Representação gráfica das etapas do processo de fabrico.

Tabela 3 - Etapas do processo de fabrico.

Nr.	Agente	Descrição	Origem	Destino
1	Scorbot	Transporte de palete A com peça 1	Armazém	Estação de Montagem
2	Dobot 1	Montagem de peça 2		
3	Scorbot	Transporte do conjunto	Estação de Montagem	Palete tipo B Entrada do Tapete
4	Tapete	Transporte do conjunto	Entrada do Tapete	Estação de Gravação
5	Dobot 2	Gravação laser		
6	Tapete	Transporte do conjunto	Estação de Gravação	Entrada do Tapete
7	Scorbot	Armazenamento do conjunto	Palete tipo B Entrada do Tapete	Armazém

4. Simulação da célula de fabrico

4.1 Seleção do *software*

A seleção do *software* para realizar a simulação foi realizada com base na metodologia apresentada na secção 2.3.2[25]:

1) Estabelecer requisitos de modelação;
2) Pesquisar e listar programas;
3) Estabelecer critérios de avaliação;
4) Avaliar os programas em relação aos critérios estabelecidos;
5) Seleção de *software*.

Assim, a abordagem realizada em cada etapa do método de seleção foi a seguinte:

1) Estabelecer requisitos de modelação

Para o desenvolvimento do modelo de simulação foram definidos os seguintes requisitos:

1) a representação visual do modelo deve ser feita em 3D;

2) o *software* da simulação a desenvolver deve ter a capacidade de comunicar com um *software* de controlo;

3) o modelo deve ser capaz de fazer uma reprodução visual e funcional do sistema real de forma fidedigna.

2) Pesquisar e listar programas

Utilizando a Internet como principal ferramenta de pesquisa e tendo em consideração os requisitos estabelecidos no passo anterior, foram selecionados os seguintes programas:

- Siemens Tecnomatix Plant Simulation [31];
- Anylogic [32];
- FlexSim [33];
- Visual Components [34].

3) Estabelecer critérios de avaliação

Selecionados os programas candidatos, o passo seguinte foi a definição dos critérios a utilizar na escolha do *software* a utilizar com vista a atingir os objetivos deste trabalho, que se descrevem de seguida:

- Facilidade de acesso (demonstração, gratuito, licença);
- Facilidade de modelação;
- Número máximo de objetos;
- Disponibilidade da informação de suporte;
- Possibilidade de simular eventos discretos;
- Flexibilidade de programação/suporte;
- Suporte para criar modelos personalizados;
- Baixo custo.

4) Avaliar os programas em relação aos critérios estabelecidos

Após a definição dos critérios, foi realizada uma pesquisa de informação sobre cada candidato de forma a avaliar a sua resposta aos critérios em análise. Para facilitar a ponderação dos critérios, foi atribuído um valor quantitativo a cada um, numa escala de 0 a 5, sendo que 0 é considerada uma má resposta ao critério e 5 será uma muito boa resposta. NaTabela 4 é possível observar a resposta dada a cada critério, assim como a respetiva avaliação.

Tabela 4 - Avaliação de programas de simulação.

software	**Tecnomatix**	**Anylogic**	**FlexSim**	**Visual Components**
Facilidade de acesso (demo, licença)	Demo (4)	Licença (2)	Licença (2)	Licença (2)
Facilidade de modelação	Difícil (1)	Acessível (4)	Intermédio (3)	Intermédio (3)
Número máximo de objetos	80 Objetos (4)	Não existente (5)	30 Objetos (3)	Não existente (5)
Informação disponível/ Suporte	Limitada (2)	Médio (3)	Médio (3)	Limitada (2)

Simulação de eventos discretos	Sim (5)	Sim (5)	Sim (5)	Sim (5)
Flexibilidade de programação	SimTalk (2)	Java (5)	Fluxograma (5)	Python (5)
Suporte para modelos personalizados	Sim (5)	Sim (5)	Sim (5)	Sim (5)
Custo	Demo Gratuita (5)	Pago (0)	Pago (0)	Pago (0)

5) Seleção de *software*

Uma vez que os critérios não têm todos o mesmo grau de importância na seleção do *software*, foram definidos fatores de ponderação (pesos), que serão utilizados para calcular a pontuação final.

Tabela 5 - Pesos atribuídos aos critérios de avaliação

Critério	**Peso**
Facilidade de acesso (demo, licença)	20%
Facilidade de modelação	20%
Número máximo de objetos	10%
Informação disponível/ Suporte	5%
Simulação de eventos discretos	5%
Flexibilidade de programação	10%
Suporte para modelos personalizados	10%
Custo	20%

A pontuação final atribuída a cada *software* foi obtida somando as pontuações de cada critério, ponderada com o respetivo peso, como se segue:

$$P = \sum_{i} W_i E_i$$

Sendo:

- P – Pontuação final;
- W_i – Peso do critério i;
- E_i – Pontuação do critério i;

tendo-se obtido os seguintes resultados para cada *software*:

- Siemens Tecnomatix Plant Simulation: 3,45;
- Anylogic: 3,10;
- FlexSim: 2,45;
- Visual Components: 2,85.

Avaliados os resultados, optou-se pelo *software Tecnomatix Plant Simulation* desenvolvido pela Siemens. O Tecnomatix responde forma positiva a todos os requisitos e oferece uma versão livre para estudantes com a única restrição de os modelos estarem limitados a um máximo de 80 objetos, o que se verificou ser adequado às necessidades de modelação deste trabalho. O Tecnomatix é um *software* genérico para a modelação e simulação de sistemas de eventos discretos, adequando-se também à modelação de sistemas de produção.

4.2 O Tecnomatix

O Tecnomatix é um *software* de modelação baseado na programação por classes e objetos. Este tipo de programação é sustentado nos conceitos de classes e objetos com atributos ou propriedades. Os objetos contêm a informação necessária para a utilização de métodos que, por sua vez, determinam o funcionamento do sistema [35].

Uma classe pode ser definida como o conjunto da informação de formatação, visual e funcional, que caracteriza um determinado tipo de objetos. A uma instância de uma classe é dado o nome de objeto.

O Tecnomatix é pré-programado com uma livraria de classes pré-definidas (*Class Library*), como ilustra a Figura 25. A existência desta livraria permite que um modelo simples possa ser elaborado através das classes pré-existentes. Em casos mais complexos, onde não é possível fazer-se a modelação recorrendo apenas às classes *standard*, é possível a criação de classes personalizadas.

Cada classe pré-definida no *software* Tecnomatix possui uma lista de métodos que, num sistema simples, funcionam de forma autónoma, baseada na configuração gráfica do modelo. Para o controlo de um sistema mais complexo é possível a criação de métodos personalizados utilizando por base os métodos de cada classe. Os métodos são procedimentos ou rotinas desenvolvidas em código que, quando invocados com o respetivo *input*, manipulam os dados e objetos da forma obter o *output* pretendido.

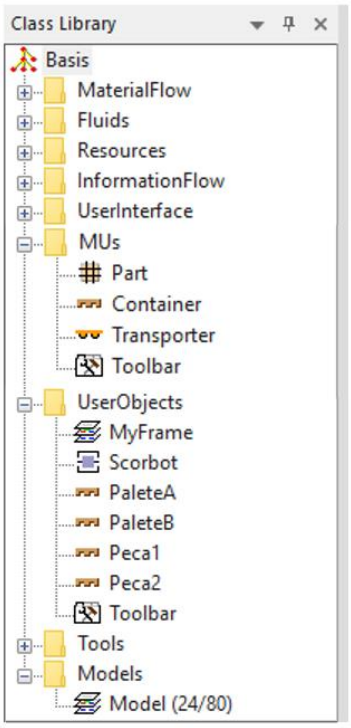


Figura 25 - *Class Library.*

O Tecnomatix permite criar classes de objetos personalizadas a partir das classes pré-definidas, tanto a nível gráfico como paramétrico (como alterar a cor de um objeto ou o tempo de processamento). Esta funcionalidade poder ser útil quando existe a necessidade de criar múltiplas entidades de um objeto muito personalizado. Também é possível criar-se uma classe que herde as configurações da classe original e assuma as alterações efetuadas, ao invés de se personalizar os objetos individualmente. Quando inserido no modelo, o objeto ou instância herdará as características da nova classe. As classes criadas pelo utilizador serão sempre colocadas na livraria *User Objects*.

Em caso de necessidade é possível identificar os endereços de cada objeto através da *Class Library*, sabendo a que classe este pertence. Estes endereços tornar-se-ão relevantes na programação e configuração do modelo, que se irá descrever na secção 4.4.

Ao iniciar um novo projeto no Tecnomatix, é apresentado ao utilizador o ambiente de trabalho onde serão colocados os objetos, designado por *Frame* (Figura 26). Geralmente, todos os objetos inseridos no projeto antes do início da simulação ficarão endereçados ao *Frame Model*. Um projeto pode ser composto por vários *Frames* interligados entre si.

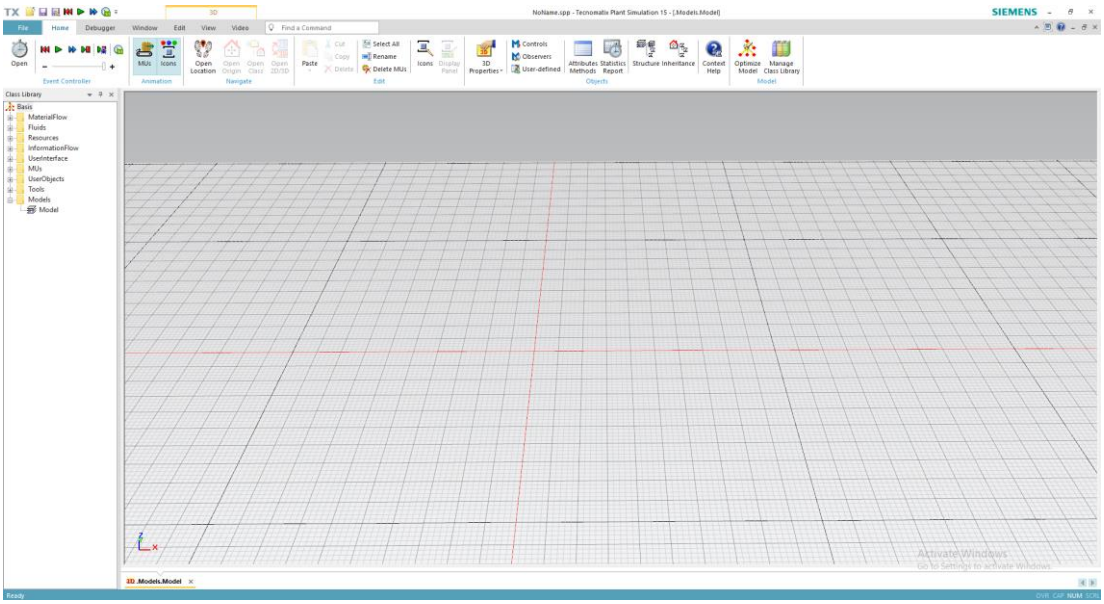


Figura 26 – Janela inicial do Tecnomatix.

O projeto foi abordado de forma faseada, tendo uma primeira fase consistido no desenvolvimento dos objetos que efetuam a representação visual e uma segunda na elaboração do código (métodos) utilizado para o controlo desses objetos.

4.3 Desenvolvimento do ambiente 3D

Esta secção descreve o desenvolvimento do ambiente 3D a utilizar na simulação do Tecnomatix, tendo-se iniciado pela modelação dos componentes da célula de fabrico. Neste processo, procurou-se maximizar a utilização das classes *standard* do Tecnomatix. Contudo, tendo-se verificado que seriam limitadas para cumprir o propósito, optou-se pela criação de classes personalizadas. Apesar de este processo não ser trivial, permite ao utilizador uma maior personalização do sistema como será descrido ao longo da secção.

Como foi referido na secção 4.2, uma classe é caracterizada por um conjunto de informações, associadas na forma de parâmetros e atributos. O Tecnomatix contém algumas classes pré-configuradas, dando ao utilizador um formato base com diversos parâmetros ajustáveis. Assim, para a configuração da classe são apresentadas pelo menos duas janelas, uma dedicada à componente paramétrica e outra à componente gráfica.

4.3.1 Scorbot

O trabalho de modelação iniciou-se pelo Scorbot por ser o elemento central principal da célula de fabrico, tendo sido desenvolvidos vários modelos até se obter o final.

Inicialmente foi utilizada a classe standard *Pick and Place* (Figura 27) que pode ser configurada com apenas alguns parâmetros como os ângulos relativos às posições das estações onde é feita a recolha/entrega de objetos (Figura 28). O Scorbot, aqui representado por um objeto *Pick and Place*, tem a função de efetuar os movimentos n.º 1, 3 e 7 descritos na Figura 24, de acordo com as referências definidas na Tabela 3 (secção 3.3).

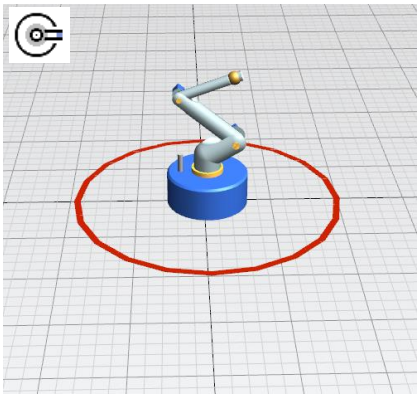


Figura 27 - Modelo gráfico do objeto *Pick and Place.*

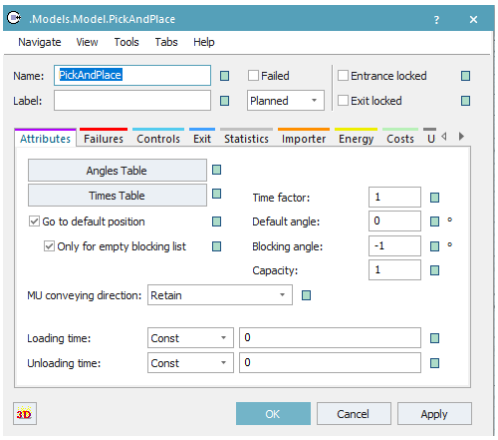


Figura 28 - Parâmetros do objeto *Pick and Place*.

O objeto *Pick and Place* pode ser configurado através dos ângulos relativos às posições das estações onde é feita a recolha/entrega de objetos e, caso este seja associado por ligações a outros objetos, esta configuração é efetuada automaticamente. No entanto, existem ainda outros parâmetros que podem ser definidos manualmente tais como:

- Tempos de transferência entre estações;
- *Pull Control*;
- *Target Control*;
- Tempo de carga e descarga;
- Capacidade;
- Tempos de falhas;
- Custos de funcionamento.

Não recorrendo a programação, o fluxo de peças é definido através de ligações () e gerido de forma autónoma pelo Tecnomatix. Este é o caso representado num modelo de testes que foi desenvolvido e que se apresenta na Figura 29. Uma vez iniciada a simulação, o *Pick and Place*, automaticamente, inicia a arrumação de peças nas prateleiras. Conforme demonstrado pelas setas, as peças partem das caixas onde são geradas para serem arrumadas no armazém. Para tal requerem a utilização do *Pick and Place*.

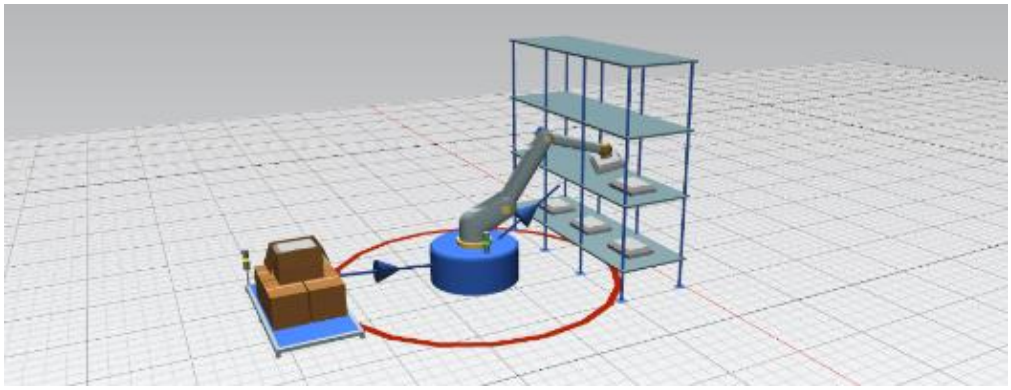


Figura 29 - Modelo de teste do objeto *Pick and Place* desenvolvido.

Para um controlo mais personalizado, o fluxo de peças pode ainda ser controlado através das rotinas *pull control* e *target control*, sendo que:

- *pull control* é uma rotina de código do objeto *Pick and Place* que restringe as entidades móveis aceites para entrada no *Pick and Place* [36]*;*
- *target control* é uma rotina de código do objeto *Pick and Place* que define o alvo de destino, dependendo da peça nele contida [37].

Uma vez compreendido o funcionamento do objeto *Pick and Place*, foram inseridos outros componentes de forma a testar a modelação e configuração do sistema. O modelo obtido encontra-se apresentado na Figura 30. A modelação e configuração dos componentes tapete, entrada de peças e saída de peças serão abordados nas próximas secções.

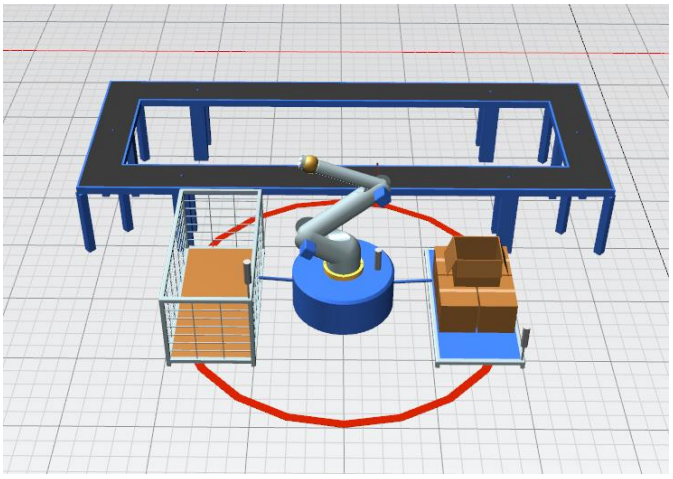


Figura 30 - Primeira iteração do modelo.

Esta abordagem apresentou problemas na sua implementação porque, apesar da existência dos controlos (*pull* e *target*), não foi possível definir o tapete, simultaneamente, como entrada e saída de instâncias (etapas números 1 e 3 do ciclo). Seria expectável que dependendo da origem das peças que chegam ao *Pick and Place*, fosse possível definir o seu destino, no entanto, este objetivo não foi possível para as peças que atingem a posição de saída no tapete. Além deste problema, esta abordagem não permite adicionar o eixo linear ao braço robótico. Por estas razões, procurou-se uma forma alternativa de modelar o Scorbot.

O desenvolvimento do novo modelo do Scorbot passou pela criação de uma classe personalizada, obtida a partir da classe pré-existente *Process* (Figura 31). Para se modelar o objeto Scorbot a partir da classe *Process* é necessário alterar:

- Modelo, estrutura gráfica e animações;
- Programação e parâmetros.

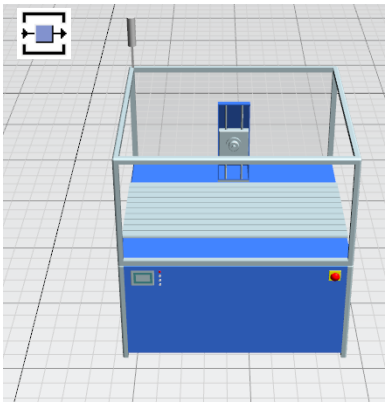


Figura 31 - Modelo gráfico do objeto *Process*.

Utilizando uma classe personalizada, deixam de estar disponíveis as funcionalidades de movimentação automáticas da classe *Pick and Place*. Por esta razão, a movimentação do Scorbot e das peças, passou a ser efetuada recorrendo à elaboração de um algoritmo (código), como se descreve na secção 4.4. Apesar de a programação ter como inconveniente o maior tempo de desenvolvimento da configuração do modelo, permite um nível maior de personalização, controlo e flexibilidade.

Tal como na classe *Pick and Place*, a classe *Process* também tem parâmetros configuráveis (Figura 32). Para utilizar a classe na sua forma mais simples (sem alteração do modelo visual 3D) apenas é necessário configurar o tempo de processamento, podendo ter uma

duração fixa ou ser definido através de distribuições matemáticas. Adicionalmente, podem ser definidos os seguintes parâmetros opcionais na classe *Process*:

- Tempos de *set-up*;
- Tempos de falha;
- Custo de operação;
- Controlos de entrada e saída:
- Requisição de recursos (Operadores).

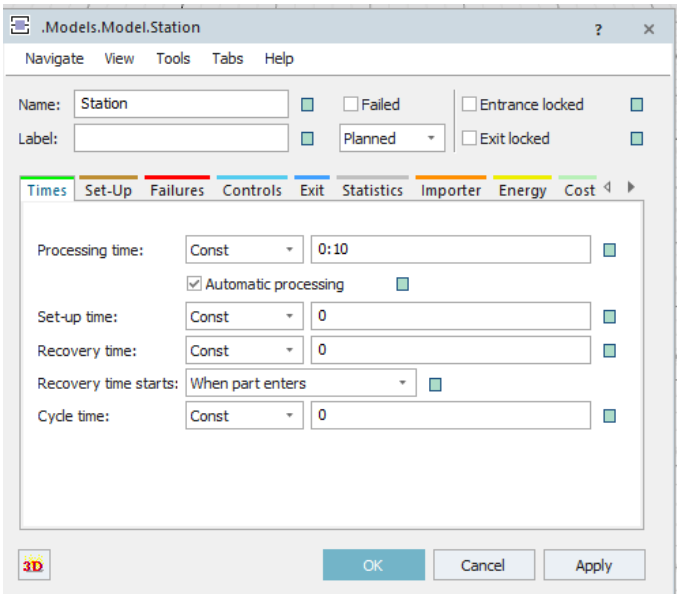


Figura 32 - Parâmetros do objeto *Process*.

Para a realização de testes de funcionamento com recurso à classe personalizada desenvolvida, importou-se um modelo gráfico proveniente de um reportório de modelos complementares ao livro do autor Steffen Bangsow [38] (Figura 33).

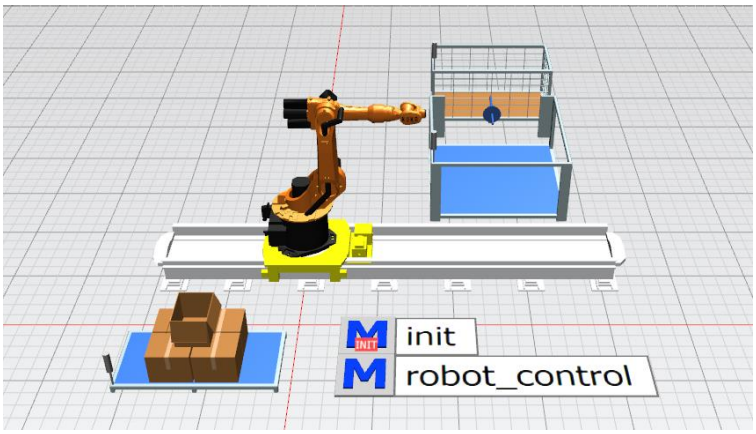


Figura 33 – Modelo robot 7 eixos [39].

Uma vez introduzido o novo modelo e verificado o seu funcionamento concluiu-se que esta abordagem se adequa aos requisitos de modelação do sistema.

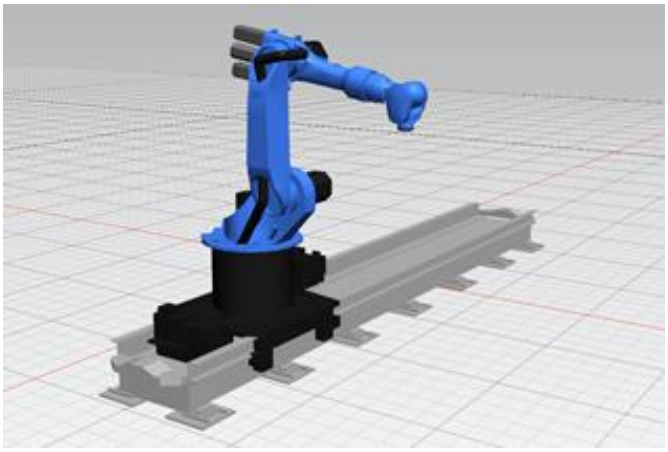


Figura 34 – Segunda iteração do desenvolvimento do modelo do Scorbot.

Confirmado o funcionamento do modelo para o efeito pretendido iniciou-se o desenvolvimento do modelo final com o objetivo de se obter um robô com a mesma aparência do robô real. Através da parametrização de algumas configurações gráficas, como a escala, cores, estrutura, etc., foi possível aproximar a representação visual à aparência real do Scorbot (Figura 34).

Uma vez que não existe documentação explicativa referente à elaboração do modelo, procedeu-se a uma análise do que seria necessário alterar no modelo base e como fazê-lo. Concluiu-se que para elaborar o modelo do Scorbot partindo da classe *standard* (Figura 31),

era necessária a existência de uma estrutura gráfica composta por modelos CAD de cada componente do Scorbot e necessário definir os seus eixos de rotação.

A Figura 35 apresenta o modelo finalizado, onde é possível visualizar a estrutura gráfica (janela da esquerda) e o modelo com todos os componentes e eixos necessários (janela da direita).

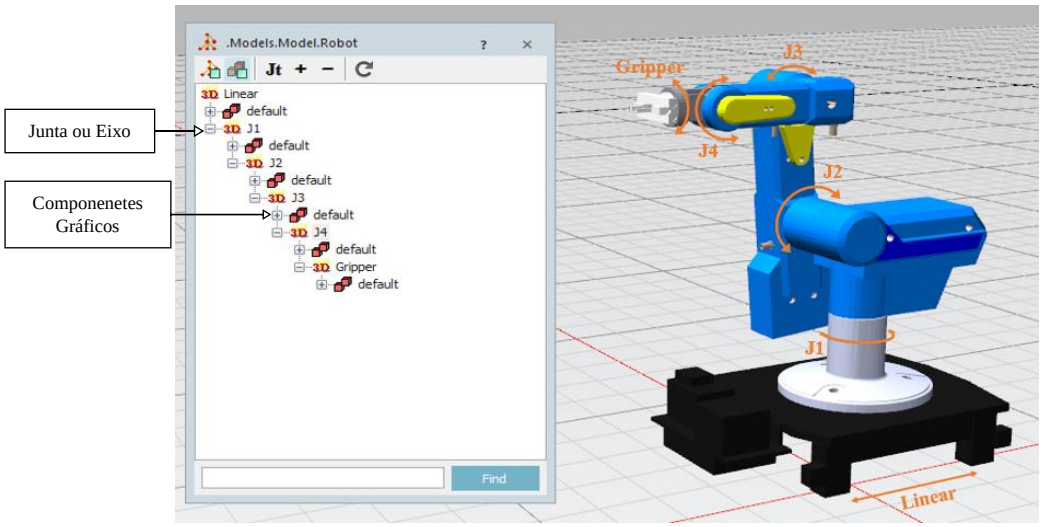


Figura 35 - Estrutura e juntas do Scorbot.

Os passos realizados para se obter o modelo apresentado na Figura 35 serão detalhados de seguida. Após eliminar todos os componentes do objeto original (Figura 31), inseriram-se os componentes do Scorbot, no formato WRL (o único suportado no Tecnomatix), através da função *Import Graphics* (Figura 36).

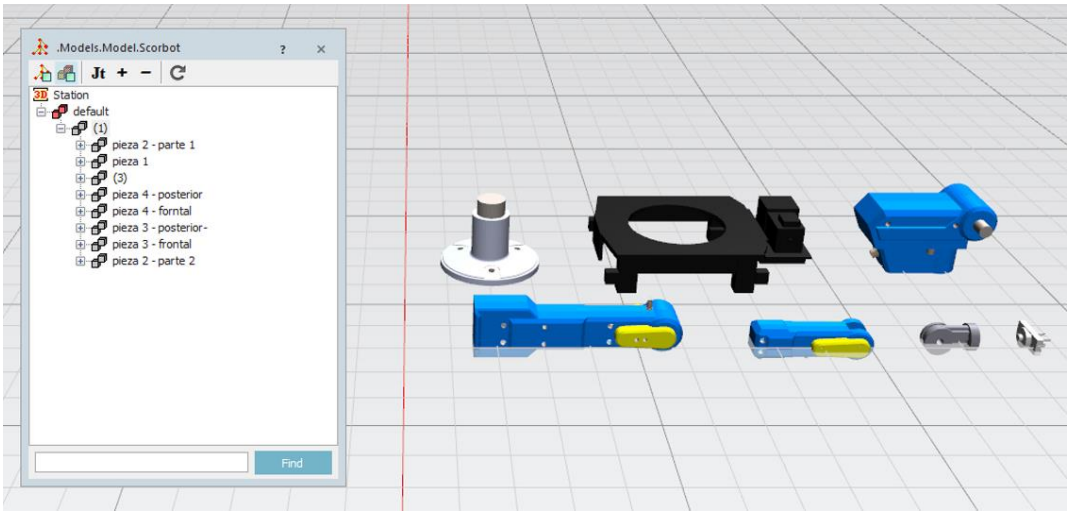


Figura 36 - Componentes do Scorbot.

Depois de inseridos todos os componentes, é necessário organizá-los criando os vários níveis na estrutura gráfica (Figura 37, janela esquerda). A definição da hierarquia da estrutura gráfica é fundamental, uma vez que a posição do elemento terminal (*gripper*) depende do movimento combinado de todos os eixos do Scorbot. Esta dependência só é obtida organizando hierarquicamente as juntas, sendo que se deve associar a cada junta um ou vários componentes que dela dependem. Isto significa que, no caso apresentado, se houver um movimento no eixo linear, esse movimento afetará a posição de todas as juntas seguintes e a dos seus componentes. Por exemplo, caso exista um movimento em J3, apenas J4 e o *gripper* se movimentam também.

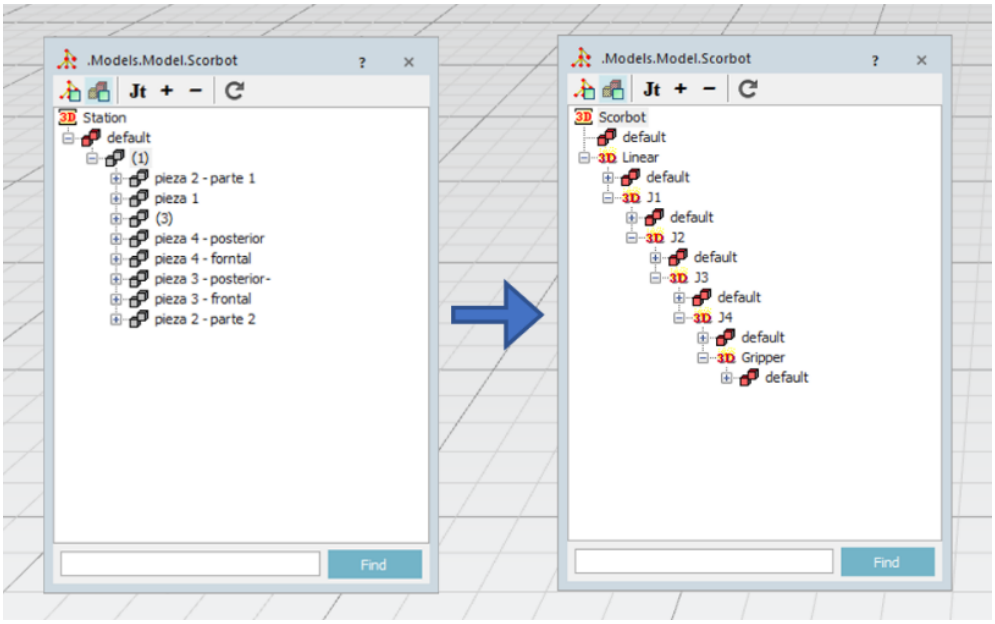


Figura 37 – Criação da estrutura gráfica do modelo do Scorbot.

Uma vez definida a estrutura, tem de ser feita a "montagem" dos componentes, posicionando-os corretamente entre si (Figura 38). Este posicionamento é efetuado manualmente de forma relativa ao componente anterior (superior hierarquicamente). O constrangimento dos movimentos é definido pelos eixos apresentados de seguida.

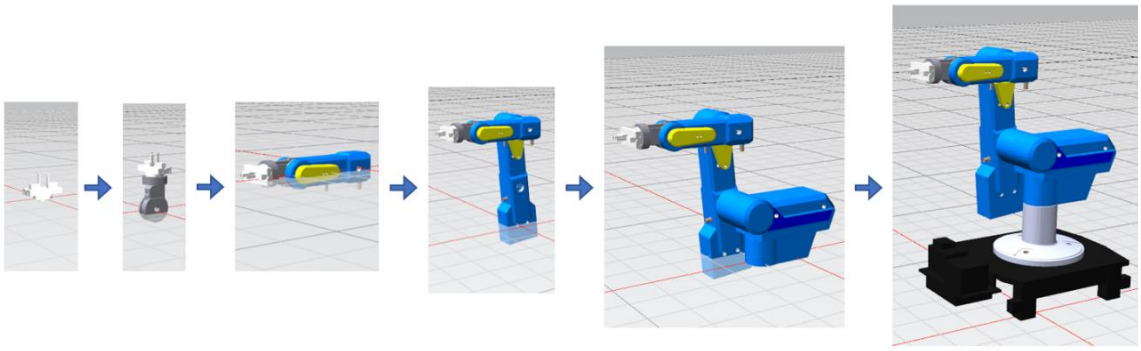


Figura 38 - Montagem do Scorbot.

Na configuração dos eixos, é possível definir o tipo de junta: linear ou rotativa, limitando assim os seus graus de liberdade. É também possível definir a sua posição em repouso, limites e velocidade máxima (Figura 39).

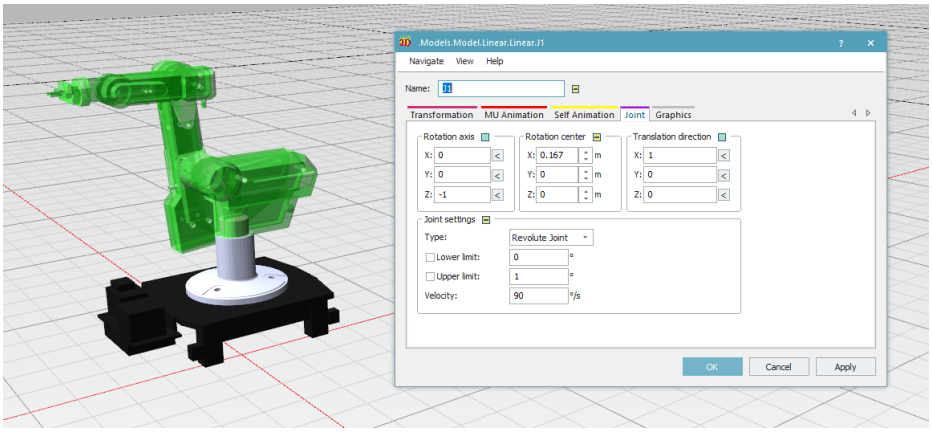


Figura 39 - Configuração dos eixos do Scorbot.

Posteriormente à modelação dos restantes componentes foi necessário definir as posições de interesse do Scorbot, concretamente para os locais onde se pretende que o robô passe, coloque ou recolha paletes. Estas posições são definidas manualmente através da modificação dos valores dos ângulos de cada junta.

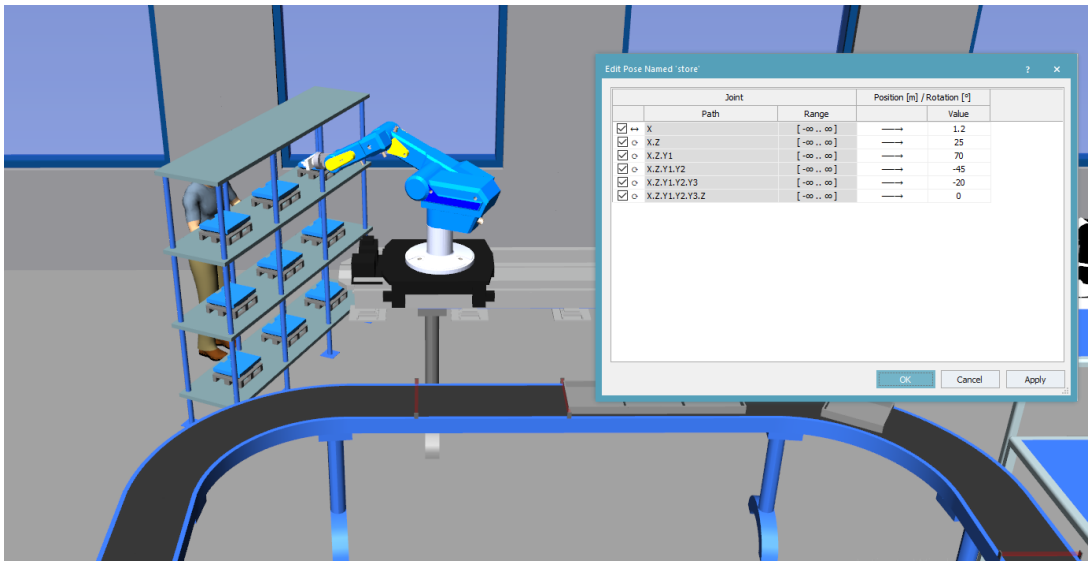


Figura 40 - Criação da posição do Scorbot para acesso ao armazém.

Na Tabela 6 é possível observar todas as posições criadas para o Scorbot. Estas posições serão depois utilizadas na secção 5 para a programação dos seus movimentos.

Tabela 6 - Posições do Scorbot.

Scorbot	
Posição	**Nome**
Aproximação para colocação no tapete	Entry_ap
Colocação no tapete	Entry
Aproximação ao Dobot 1	Dobot1_ap
Colocação no Dobot 1	Dobot1
Aproximação para retirada do tapete	Exit_ap
Retirada do tapete	Exit
Aproximação para colocação no armazém	Store_ap
Colocação no armazém	Store

4.3.2 Tapete Transportador

Esta secção apresenta o trabalho realizado para modelar o tapete transportador que tem a função de transportar as paletes do tipo B nas etapas 4 e 6 apresentadas na Figura 24 (secção 3.3). A modelação recorreu à classe pré-definida *Conveyor* que permite a construção de um tapete virtual, combinando troços elementares igualmente pré-definidos.

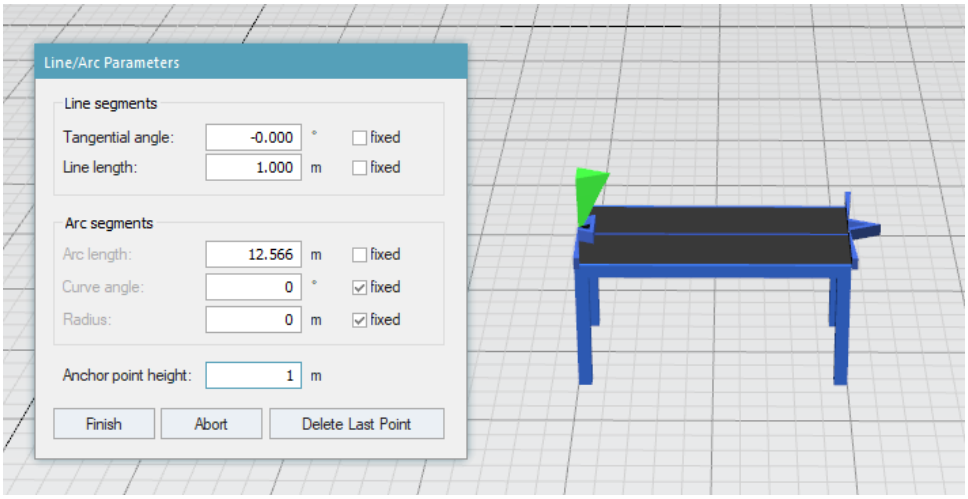


Figura 41 - Menu de modelação de secções de tapete.

Cada troço de tapete tem uma entrada e uma saída utilizadas para, respetivamente, a entrada e saída de objetos do troço. Neste caso, uma vez que se trata de um tapete transportador em circuito fechado, a saída de cada troço corresponde à entrada do seguinte. Para cada troço podem ser definidos, entre outros, os seguintes atributos (Figura 42):

- Largura;
- Comprimento;
- Velocidade;
- Capacidade;
- Tempos de falha.

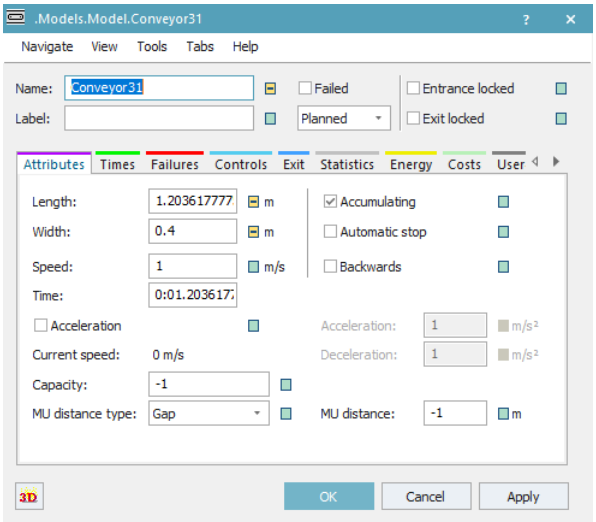


Figura 42 - Parâmetros da classe *Conveyor*.

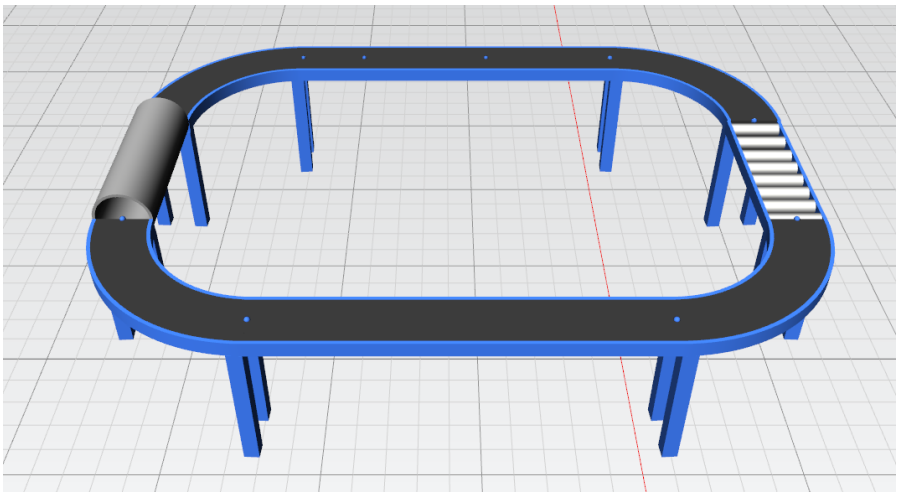


Figura 43 – Exemplo de modelos gráficos do objeto *Conveyor*.

Cada troço pode ainda ter um formato retilíneo ou curvilíneo e pode ser baseado em diferentes tecnologias de movimentação como rolos, cinta, tubo, magnético, etc. (Figura 43).

Ao invés de um troço contínuo de tapete, foi necessária a criação de diversos troços individuais (representados na Figura 44). Esta divisão permite o controlo (paragem e arranque) individual de cada troço, simulando assim o comportamento do tapete real, que permite a paragem de paletes em troços individuais.

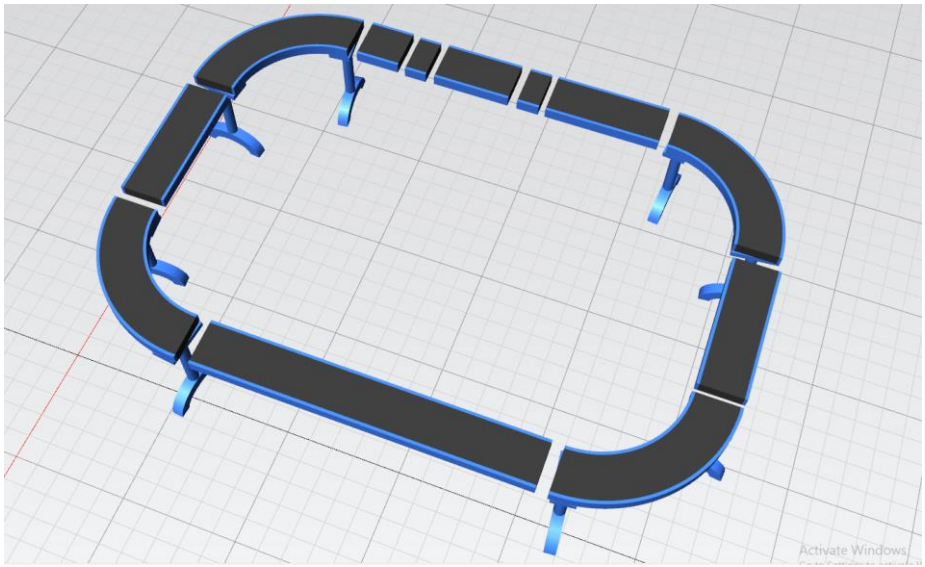


Figura 44 - Troços do tapete.

4.3.3 Alimentador

A classe do tipo *Source* (alimentador) tem como função a criação das instâncias que irão circular pelo sistema: peças ou paletes. No sistema a simular, o único alimentador necessário será utilizado na criação de peças tipo 2 que irão alimentar o Dobot 1. Para tal foi utilizada a classe *Source* e posteriormente alterado o modelo gráfico (Figura 45).

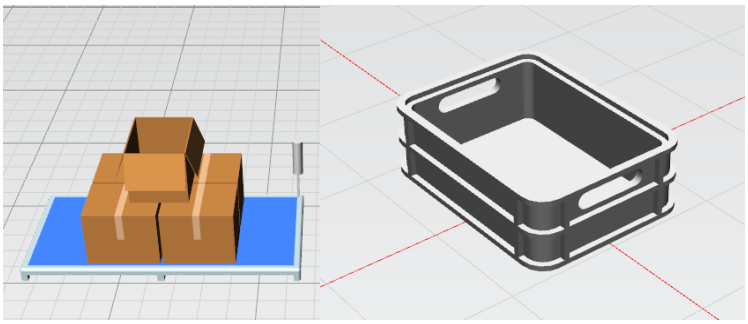


Figura 45 - Modelo gráfico inicial, à esquerda, e final, à direita, do objeto *Source.*

A criação de entidades com a classe *Source* pode ser definida através de um intervalo de tempo fixo, distribuições matemáticas ou tabelas. Podem ainda ser definidos outros parâmetros, tais como (Figura 46):

- Tempos de início e fim de criação de instâncias;
- Tempos de falha MTBF (*Mean Time Between Failures*) e MTTR (*Mean Time to Repair*).

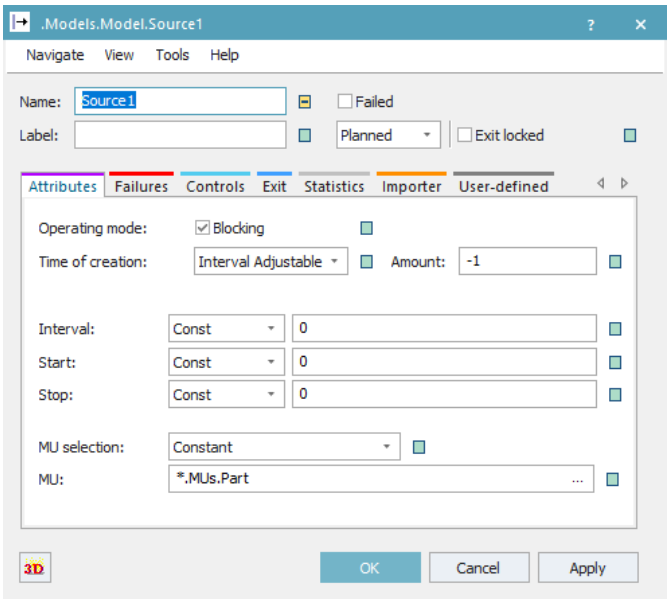


Figura 46 - Parâmetros do objeto *Source*.

4.3.4 Armazém

No processo a simular, o armazém foi modelado no Tecnomatix com recurso à classe *Store*. Esta classe será o ponto de partida das paletes com peças do tipo 1 e onde serão guardadas as paletes com o produto final, de acordo com as etapas 2 e 5 da Figura 24 (secção 3.3). Tal como o tapete, o armazém pode ter várias configurações gráficas pré-definidas, entre elas *rack*/prateleiras ou apenas marcação no chão.

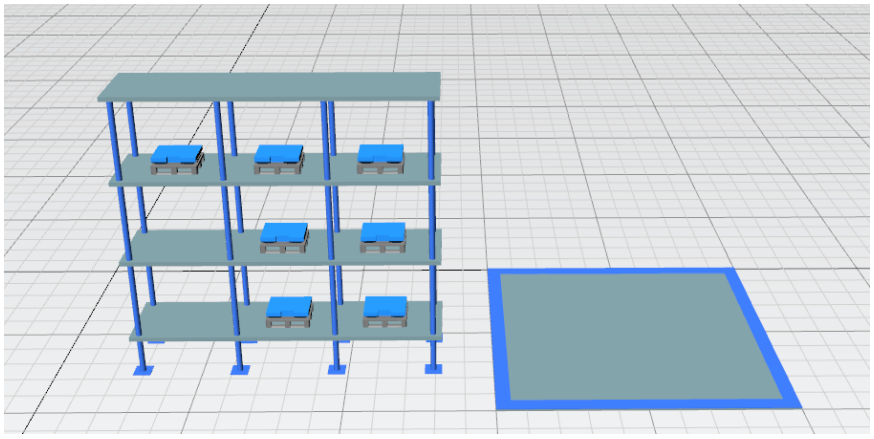


Figura 47 - Exemplo de modelos gráficos do objeto *Store.*

Relativamente à classe *Store* existem diversos parâmetros configuráveis, tais como as dimensões físicas do armazém, o número de células de armazenamento existentes em comprimento (X), largura (Y) e largura (Z) e a sua capacidade de armazenamento (Figura 48).

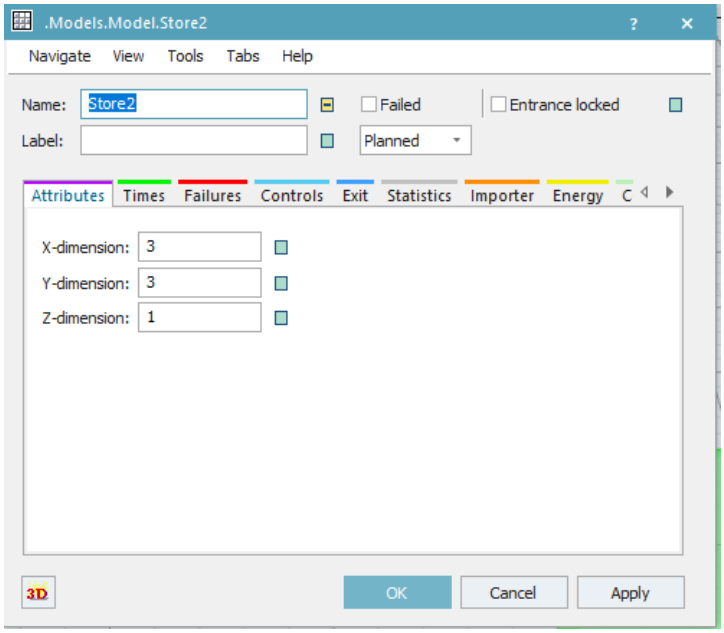


Figura 48 - Parâmetros do objeto *Store*.

4.3.5 Dobots

No processo de fabrico em estudo são utilizados dois braços robóticos Dobot para realizar tarefas de montagem e gravação laser - etapas representadas com os números 2 e 5 da Figura 24 (secção 3.3). A estrutura e a cinemática de ambos os Dobots são iguais, sendo apenas alterado o tipo de ferramenta instalada como elemento terminal do braço.

A modelação destes robôs foi realizada de acordo com a mesma metodologia usada com o Scorbot. Deste modo, a modelação do Dobot foi realizada com base na classe *Process*, tendo sido adaptada a configuração cinemática e melhorada a sua aparência com a colocação do modelo CAD deste robô. Em comparação com o Scorbot, o braço robótico Dobot tem menos um grau de liberdade (menos uma junta de rotação) e não se encontra sobre um eixo linear (menos uma junta de translação) (Figura 49).

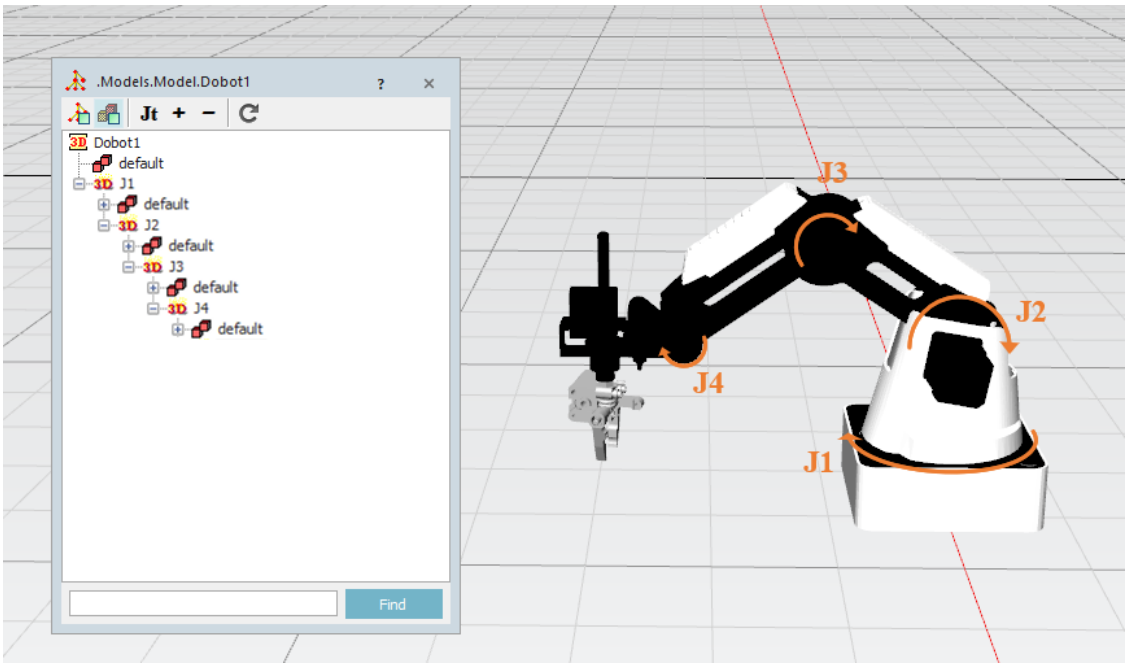


Figura 49 - Estrutura hierarquizada e juntas do modelo do robô Dobot.

Tal como no caso do Scorbot, posteriormente à modelação geométrica e cinemática, foi necessário definir as posições a utilizar no ciclo de funcionamento dos Dobots. Estas posições são definidas manualmente através dos valores dos ângulos de cada junta (Figura 50).

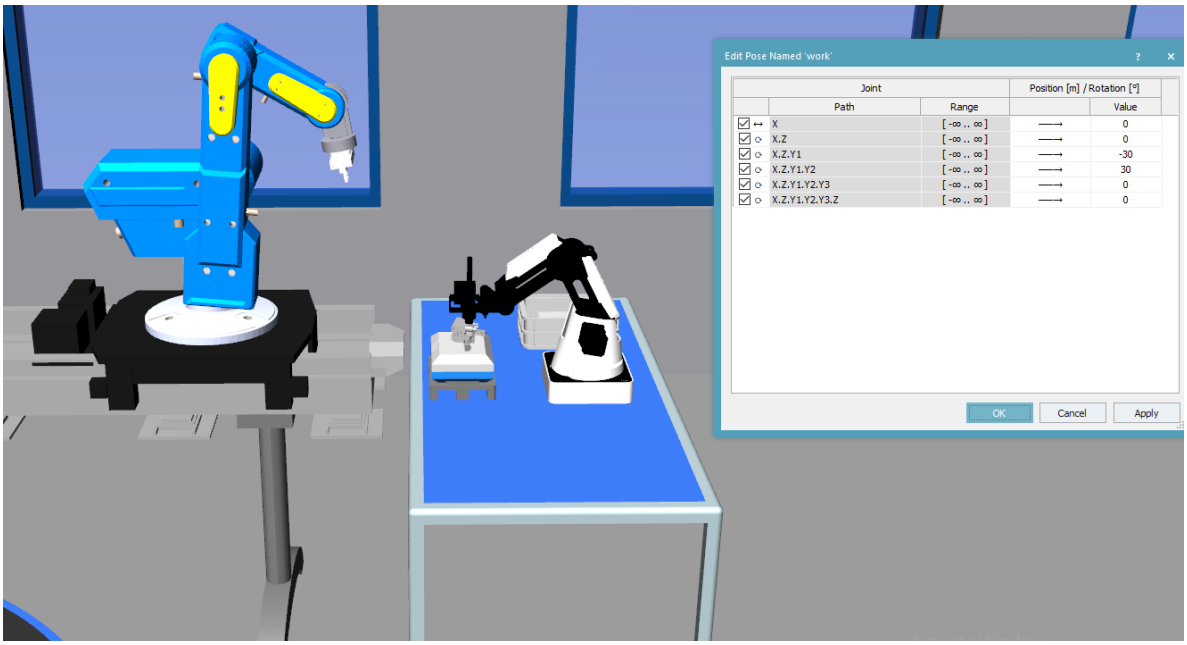


Figura 50 – Criação da posição do Dobot 1 para colocação da peça 2.

Na Tabela 7 e na Tabela 8 é possível observar todas as posições criadas para o Dobot 1 e Dobot 2, respetivamente, as quais serão utilizadas na programação dos movimentos (Secção 4.4).

Tabela 7 - Posições do Dobot 1.

Dobot 1	
Posição	**Nome**
Colocação da peça 2	Work
Recolha de peça 2	Source 2
Repouso/Posição intermédia	Home

Tabela 8 - Posições do Dobot 2.

Dobot 2	
Posição	**Nome**
Gravação	Work
Repouso	Home

4.3.6 Paletes e Peças

As peças e paletes são os únicos componentes modelados com base nas classes *standard* do tipo *MU* (*Mobile Units*) [40]. Os objetos destas classes podem ser possuídos por outros objetos existentes na célula de fabrico e transferidas entre eles. Dentro da categoria de classes *MU* é possível distinguir as classes *Parts*, Containers e *Transporters*:

- As *Parts* () são as *MU* que modelam as peças as serem produzidas, não permitindo transportar outras peças;
- Os *Containers* () são utilizados para modelar o transporte de peças entre estações. Podem ser representados por caixas ou paletes e necessitam sempre de uma Ação externa para se movimentarem;
- Os *Transporters* () são veículos autopropulsionados utilizados para modelar o transporte de peças ou contentores e requerem uma pista para se deslocarem.

Inicialmente, as peças do tipo 1 e 2 foram modeladas utilizando a classe tipo *MU Part*. Contudo, esta abordagem levantou alguns problemas relacionados com a montagem pois o *Tecnomatix* não considera a classe *Part* como transportador e consequentemente não permite que a peça 2 seja montada (transportada) na peça 1. Como alternativa, o problema foi solucionado com a modelação da peça do tipo 1 como um objeto *Container* (Figura 51), podendo, assim, ser utilizado como transportador da peça 2.

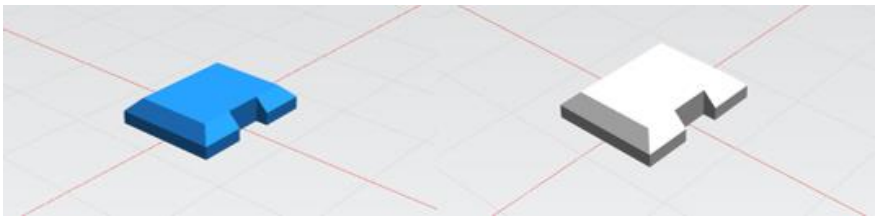


Figura 51 – Modelo das peças a acoplar: à esquerda, Peça 1 (*Container*) e, à direita, peça 2 (*Part*).

O modelo da palete A foi obtido através do objeto pré-definido *Container*, tendo sido alterado através das ferramentas de CAD existentes do Tecnomatix (Figura 52).

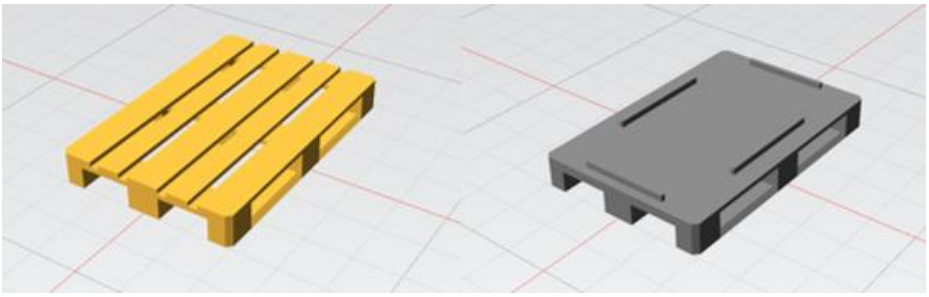


Figura 52 - Modelo inicial e final da palete tipo A.

O modelo da palete do tipo B também foi obtido a partir do objeto pré-definido *Container*, tendo-se realizado o mesmo processo de alteração, através da ferramenta de CAD do Tecnomatix, para se obter um objeto com a forma de uma bandeja (Figura 53).

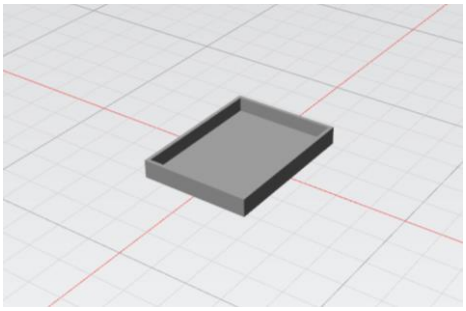


Figura 53 - Modelo da palete tipo B.

4.4 Controlo/Programação

Na secção anterior foram abordados todos os componentes e as suas configurações de forma a obter o modelo base da célula de fabrico (Figura 54). Apesar da presença de todos os objetos, não existe qualquer fluxo de materiais ainda definido, de forma que nesta fase não se pode considerar ainda um modelo de simulação funcional. Neste capítulo será apresentada a solução que foi desenvolvida para o controlo do fluxo na célula usando as ferramentas do Tecnomatix.

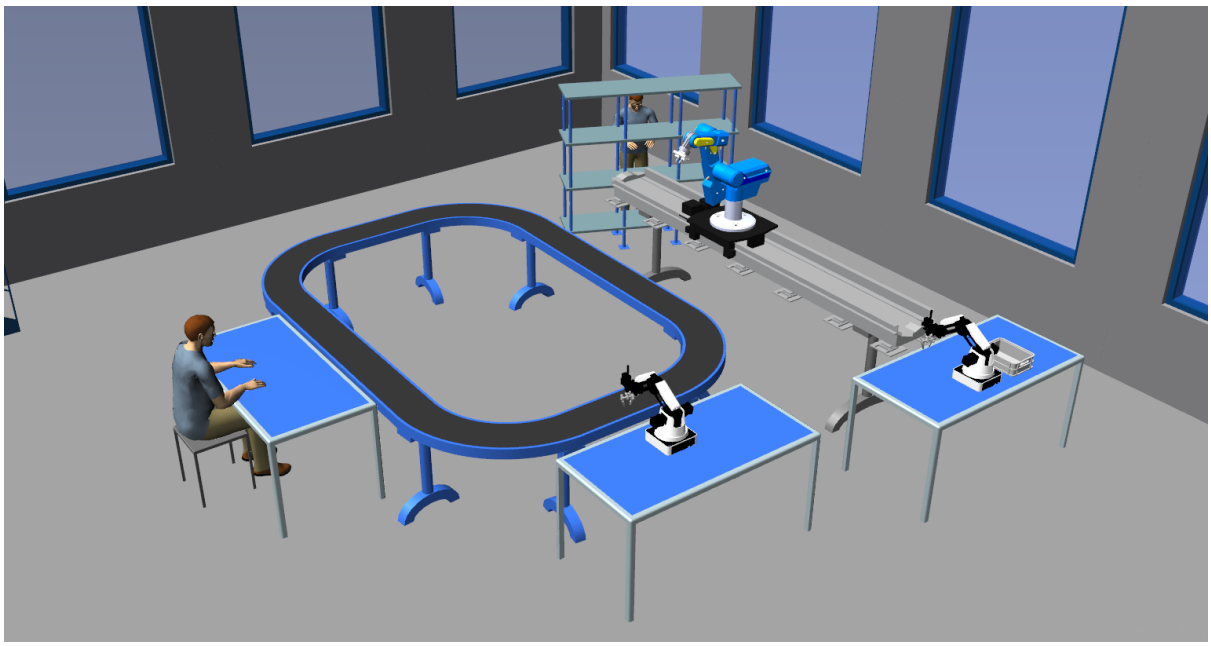


Figura 54 - Modelo gráfico da célula (não funcional).

Para controlar o sistema é necessário recorrer à programação de pequenos blocos de código, denominados por *Methods*, que podem ser ativados quando necessário. Apesar destes blocos de código não representarem objetos físicos, a sua existência adiciona uma representação gráfica no modelo da célula de fabrico, como ilustrado na Figura 55.

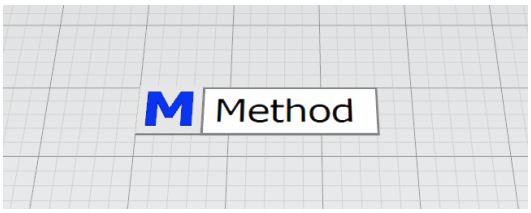


Figura 55 – Modelo gráfico da classe *Method*.

No Tecnomatix, o termo “Método” pode ser utilizado para designar duas ferramentas distintas:

- os objetos que contêm código SimTalk (Figura 55 e Listagem 1);
- as funções específicas a cada classe utilizadas no código SimTalk.

Ao contrário dos restantes objetos, o *Method* não é configurado através de parâmetros, mas sim numa janela de edição de texto, onde é possível escrever código em SimTalk, a linguagem desenvolvida pela Siemens para a programação no Tecnomatix, exclusiva deste *software* . Apesar de ser proprietária, a linguagem SimTalk também implementa estruturas de decisão e repetição que se encontram nas linguagens de programação mais clássicas (ver exemplo de código na Listagem 1).

```
for x:=1 to .Models.Model.Store.XDim
    for y:=1 to .Models.Model.Store.YDim

        if .Models.Model.Store[x,y].empty=false

           if .Models.Model.Store[x,y].cont.state=false
             paletenovapos:=[x,y]
             xf:=x
             yf:=y
             x:=100
             y:=100
           end

        end
    next
next
```

Listagem 1 - Exemplo de programação em SimTalk.

Para o controlo da célula foram criados 5 métodos, podendo cada um deles controlar um ou mais componentes da célula:

- **Método Init** – Criação de conjuntos no armazém constituídos por uma palete do tipo A e uma Peça do tipo 1, assim como criação das paletes do tipo B no tapete *(Set-up*);
- **Método Scorbot** – Controlo dos movimentos dos modelos do Scorbot e do Dobot 1, correspondendo às etapas n.º 1, 2, 3, 4 e 7 (Figura 24 da Secção 3.3). Este método também é também responsável pelo *reset* ao armazém;

- **Método EntryConv** – Controlo do troço de tapete onde é efetuada a retenção e o carregamento das paletes do tipo B vazias, que ocorre na etapa n.º 3 (Figura 24 da Secção 3.3);
- **Método ExitConv** – Controlo do troço de tapete onde é efetuada a retenção e descarga das paletes tipo B processadas, que ocorre na etapa n.º 6 (Figura 24 da Secção 3.3);
- **Método Dobot2** – Controlo do modelo do Dobot 2 e do troço de tapete onde este efetua o processamento, que ocorre na etapa n.º 5 (Figura 24 da Secção 3.3).

4.4.1 Método Init

No Tecnomatix está prevista, quando necessário, a existência de um método designado por **Init** (

Figura 56). O método **Init** é executado quando o botão de início da simulação é acionado e executa o código nele contido antes de qualquer iteração da simulação. Este método tem, por isso, a função de estabelecer as condições iniciais da simulação.

No processo de fabrico definido, o ciclo inicia-se pela recolha no armazém de uma palete tipo A com uma peça tipo 1. Assim, é necessária a criação de vários destes conjuntos no armazém no momento imediato em que a simulação inicia, de forma a alimentar o sistema. O mesmo acontece com as paletes do tipo B que circulam no tapete. Esta operação poderia ser efetuada manualmente. No entanto, uma vez que a criação teria de ser feita manualmente no início de cada simulação, optou-se por automatizar este processo, programando no método Init o código necessário à criação daqueles objetos nas localizações desejadas (Listagem 2).

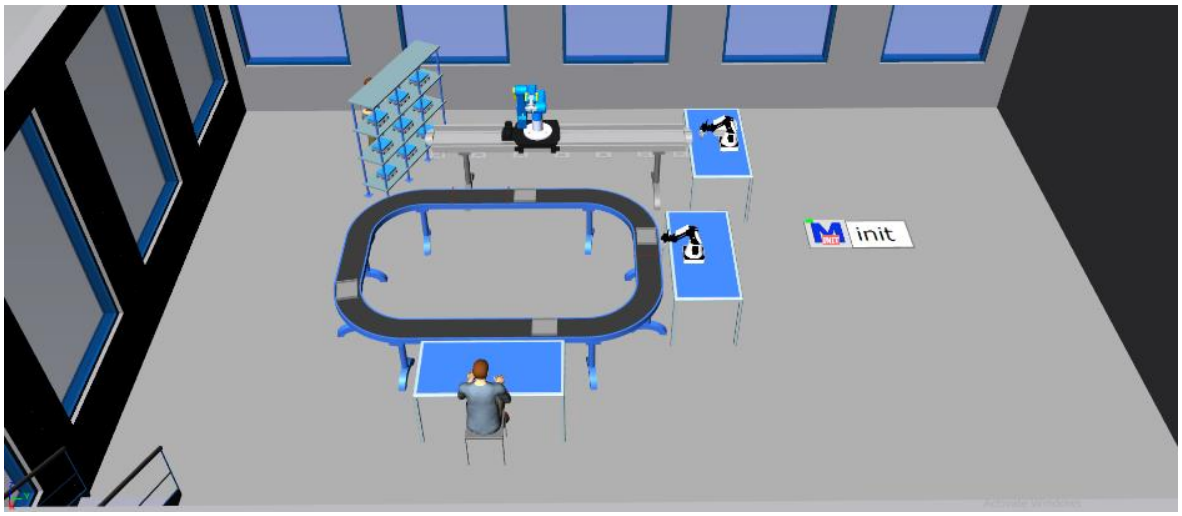


Figura 56 - Modelo com método **Init**.

Abrindo e analisando a lista de métodos e atributos referentes à palete A (Figura 57), observa-se a existência do método *create***.** De acordo com o manual do Tecnomatix [39], o método *create* é o mais indicado para criar instâncias de objetos de classe *MU*, requerendo apenas a definição de duas entidades: 1) a classe de objeto a criar e 2) o local onde criar.

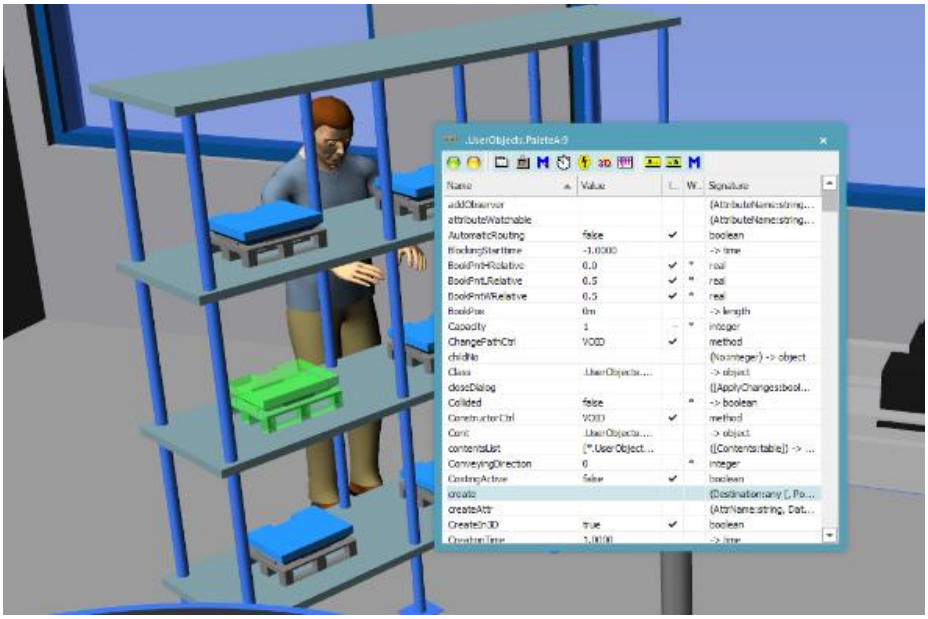


Figura 57 - Atributos da palete A.

De acordo com a sintaxe do SimTalk a criação de uma palete A pode ser feita por:

```
.UserObjects.PaleteA.create(.Models.Model.Store[1,1])
```

Nesta instrução é criada a instância número 1 da classe palete tipo A, na posição X=1 e Y=1 do armazém *Store* inserido no modelo *Model*. Esta instrução foi repetida para os restantes espaços (posições) do armazém.

Foi criada uma peça tipo 1 em cada palete A anteriormente criada utilizando o mesmo método *create*. A instrução da colocação da 1ª peça na 1ª palete apresenta-se de seguida, sendo repetida para as todas as restantes paletes.

```
.UserObjects.Peca1.create(.UserObjects.PaleteA:1)
```

Na última etapa do *set-up*, foi necessário criar no tapete transportador as 4 paletes tipo B. O procedimento é idêntico ao usado para criar as paletes A. Convém ressalvar que não existe um único objeto que represente o tapete na totalidade, mas vários objetos que representam cada um dos troços do tapete. Assim, as paletes serão criadas nos troços do tapete que o utilizador pretender, sendo definida a sua posição no troço (em metros), medida a partir do início do respetivo troço. Se se pretender criar a palete no início do troço, a distância no troço é zero, resultando na seguinte instrução:

```
.UserObjects.PaleteB.create(.Models.Model.Conveyor,0)
```

Na Listagem 2 apresenta-se o código realizado para efetuar o *set-up* do sistema no método **Init**.

```
.UserObjects.PaleteA.create(.Models.Model.Store[1,1])
.UserObjects.PaleteA.create(.Models.Model.Store[1,2])
.UserObjects.PaleteA.create(.Models.Model.Store[1,3])
.UserObjects.PaleteA.create(.Models.Model.Store[2,1])
.UserObjects.PaleteA.create(.Models.Model.Store[2,2])
.UserObjects.PaleteA.create(.Models.Model.Store[2,3])
.UserObjects.PaleteA.create(.Models.Model.Store[3,1])
.UserObjects.PaleteA.create(.Models.Model.Store[3,2])
.UserObjects.PaleteA.create(.Models.Model.Store[3,3])

.UserObjects.Peca1.create(.UserObjects.PaleteA:1)
.UserObjects.Peca1.create(.UserObjects.PaleteA:2)
.UserObjects.Peca1.create(.UserObjects.PaleteA:3)
.UserObjects.Peca1.create(.UserObjects.PaleteA:4)
.UserObjects.Peca1.create(.UserObjects.PaleteA:5)
.UserObjects.Peca1.create(.UserObjects.PaleteA:6)
.UserObjects.Peca1.create(.UserObjects.PaleteA:7)
.UserObjects.Peca1.create(.UserObjects.PaleteA:8)
.UserObjects.Peca1.create(.UserObjects.PaleteA:9)
```

```
.UserObjects.PaleteB.create(.Models.Model.Conveyor31, 0)
.UserObjects.PaleteB.create(.Models.Model.Conveyor5, 0)
.UserObjects.PaleteB.create(.Models.Model.Conveyor3, 0)
.UserObjects.PaleteB.create(.Models.Model.Conveyor4, 1)

&ScorbotMthd.execute
```

Listagem 2 – Método **Init**.

4.4.2 Método Scorbot

O método que comanda as ações do Scorbot designa-se por **Scorbot** e é utilizado nas etapas n.º 1, 3 e 7 do processo (Figura 24) referentes ao Dobot 1. Uma vez que a ação do Dobot1 apenas é condicionada pela execução da etapa n.º 1 do Scorbot, e por simplicidade da programação, foi ainda incluída neste método a etapa n.º 2 referente à montagem com o Dobot 1. De forma a facilitar a compreensão do método **Scorbot**, na Figura 58 é apresentado o seu fluxograma.

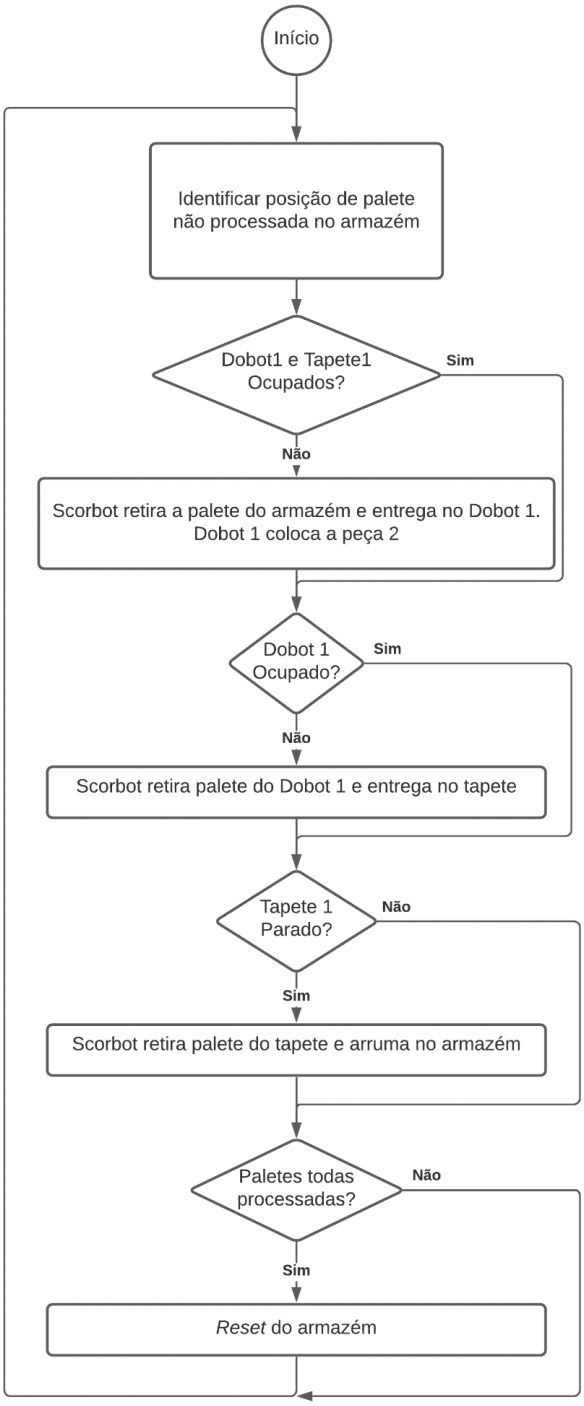


Figura 58 - Fluxograma do método Scorbot.

O método **Scorbot** é iniciado pelo método **Init**, após a definição do *set-up* do sistema. Quando está em execução inicia uma verificação ao armazém de forma a identificar se existem paletes por processar. Para se efetuar a verificação em todo o armazém são utilizadas as variáveis *x* e *y* para definir a posição em análise, desde 1 até ao número máximo de cada linha e coluna do armazém, definidas por *XDim* e *YDim*, respetivamente. Uma vez encontrada uma

palete, o programa verifica se esta se encontra processada. A verificação é feita através da função *cont.state* (estado do conteúdo) que apura a existência de uma peça tipo 2 no conjunto:

```
.Models.Model.Store[x,y].cont.state
```

No caso de o resultado devolvido ser falso (não existe peça tipo 2), a posição é guardada para o processamento [xf,yf]. Na Listagem 3 é possível observar o código SimTalk utilizado no processo de procura de palete por processar descrito anteriormente.

```
for x:=1 to .Models.Model.Store.XDim
    for y:=1 to .Models.Model.Store.YDim
        if .Models.Model.Store[x,y].empty=false
           if .Models.Model.Store[x,y].cont.state=false
             paletenovapos:=[x,y]
             xf:=x
             yf:=y
             x:=100
             y:=100
           end
        end
    next
next
```

Listagem 3 - Código relativo à procura por palete por processar no armazém.

Identificada a palete a processar, deve realizar-se a sua transferência entre o armazém e o Scorbot. O conjunto palete A com peça 1, definida na posição [xf,yf] é movimentada entre os objetos Store e Scorbot através do método *move*, utilizando a seguinte sintaxe:

```
.Models.Model.Store[xf,yf].move(.Models.Model.Scorbot)
```

Após o conjunto palete A com peça 1 ser transferido para o Scorbot, recorreu-se às posições definidas na Tabela 6 para movimentar o *Scorbot* para a posição “Dobot1”.

```
Scorbot._3D.poses.moveTo("Dobot1",2)
```

Quando o Scorbot chega à posição “Dobot1” é novamente utilizado o método *move* para transferir a palete do *Scorbot* para o objeto Dobot1. No entanto, o método *move* apenas moveria o Scorbot para o Dobot e não a palete contida no Scorbot. Para mover a palete é necessário utilizar o atributo *cont*, referenciando assim o conteúdo do Scorbot.

```
Scorbot.cont.move(.Models.Model.Dobot_1)
```

Já na posse do Dobot1, este executa a montagem da peça tipo 2 sobre a peça tipo 1 (conjunto palete A com peça 1) através da seguinte instrução:

```
Dobot1.cont.move(.Models.Model.Dobot_1.MU.MU)
```

Nesta instrução é utilizado Dobot1.MU.MU de forma a indicar que a peça 2 será colocada na peça 1 (uma MU), que está contida na palete A (outra MU), que por sua vez está contida no objeto Dobot_1 (ver código da Listagem 4).

```
Scorbot._3D.poses.moveTo("Store_ap",2)
wait 2
Scorbot._3D.poses.moveTo("Store",2)
wait 2
.Models.Model.Store[xf,yf].move(.Models.Model.Scorbot)
Scorbot._3D.poses.moveTo("Store_ap",2)
wait 2
Scorbot._3D.poses.moveTo("Dobot1_ap",2)
wait 2
Scorbot._3D.poses.moveTo("Dobot1",2)
wait 2
Scorbot.cont.move(.Models.Model.Dobot_1)
wait 2
Scorbot._3D.poses.moveTo("Dobot1_ap",2)
wait 2
Dobot1._3D.poses.moveTo("Home",2)
wait 2
Dobot1._3D.poses.moveTo("Source2",2)
wait 2
.Models.Model.Source.cont.move(Dobot1)
wait 2
Dobot1._3D.poses.moveTo("Home",2)
wait 2
Dobot1._3D.poses.moveTo("Work",2)
wait 2
Dobot1.cont.move(.Models.Model.Dobot_1.MU.MU)
wait 2
Dobot1._3D.poses.moveTo("Home",2)
```

Listagem 4 - Código do Scorbot e Dobot 1 para execução das etapas 1 e 2.

O restante método do Scorbot realiza a entrada e saída do tapete, utilizando as funções até este ponto demonstradas.

Posteriormente a cada movimentação, o método **Scorbot** executa uma verificação a cada posição do armazém de forma avaliar se todas as paletes estão processadas. Esta verificação é efetuada através de dois *if statements*. O primeiro, através da função *empty*, identifica se a posição do armazém está vazia. O segundo, através da função *cont.state*, verifica se a palete contida no espaço em questão tem peça tipo 2, isto significa que a palete já está

processada. Se esta última condição se verificar para todas as paletes, estas são apagadas através do comando *cont.delete* e novas criadas como descrito no método **Init** .

```
if .Models.Model.Store[1,1].empty=false and...
.Models.Model.Store[3,2].empty=false

  if.Models.Model.Store[1,1].cont.state=true and...
  .Models.Model.Store[3,2].cont.state=true
      .Models.Model.Store[1,1].cont.delete
      .Models.Model.Store[1,2].cont.delete
      .Models.Model.Store[2,1].cont.delete
      .Models.Model.Store[2,2].cont.delete
      .Models.Model.Store[1,3].cont.delete
      .Models.Model.Store[2,3].cont.delete
      .Models.Model.Store[3,1].cont.delete
      .Models.Model.Store[3,2].cont.delete
      .UserObjects.PaleteA.create(.Models.Model.Store[1,1])
      .UserObjects.PaleteA.create(.Models.Model.Store[1,2])
      .UserObjects.PaleteA.create(.Models.Model.Store[1,3])
      .UserObjects.PaleteA.create(.Models.Model.Store[2,1])
      .UserObjects.PaleteA.create(.Models.Model.Store[2,2])
      .UserObjects.PaleteA.create(.Models.Model.Store[2,3])
      .UserObjects.PaleteA.create(.Models.Model.Store[3,1])
      .UserObjects.PaleteA.create(.Models.Model.Store[3,2])
      .UserObjects.Peca1.create(.Models.Model.Store[1,1].MU)
      .UserObjects.Peca1.create(.Models.Model.Store[1,2].MU)
      .UserObjects.Peca1.create(.Models.Model.Store[1,3].MU)
      .UserObjects.Peca1.create(.Models.Model.Store[2,1].MU)
      .UserObjects.Peca1.create(.Models.Model.Store[2,2].MU)
      .UserObjects.Peca1.create(.Models.Model.Store[2,3].MU)
      .UserObjects.Peca1.create(.Models.Model.Store[3,1].MU)
      .UserObjects.Peca1.create(.Models.Model.Store[3,2].MU)

  end

end
```

Listagem 5 - Método **Scorbot** - *Reset* do Armazém.

4.4.3 Método EntryConv

No final do método **Scorbot**, a palete A que transporta as peças é transferida do Dobot 1 para o tapete. Na modelação do tapete foi definido um troço para as paletes serem carregadas, cuja função é reter as paletes B vazias, aguardar o seu carregamento e, de seguida, libertá-las. Na Figura 59 ilustra-se o momento em que a palete foi bloqueada na posição de carregamento e o carregamento efetuado pelo Scorbot, impedindo assim o avanço das restantes paletes.

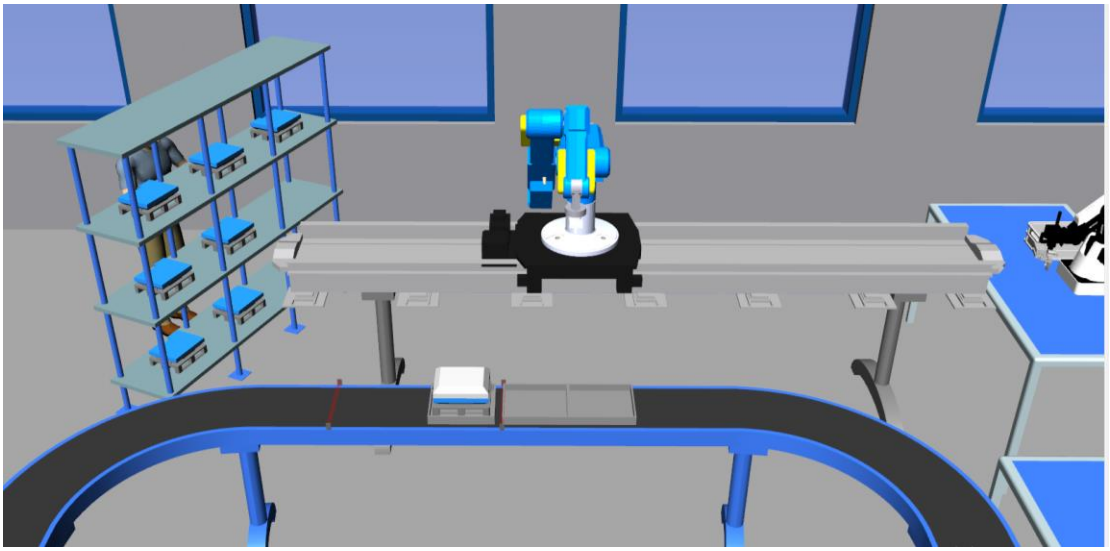


Figura 59 - Etapa 3 - Entrada no tapete.

Cada palete B tem uma propriedade boleana, denominada “Ocupada”, que é utilizada para indentificar o estado das paletes, tomando o valor *true* quando estas estão ocupadas e *false* quando estão vazias. A palete B apenas é retida para carregamento caso a variável “Ocupada” apresente o estado *false*. Depois de carregada, a propriedade “Ocupada” é alterada para *true*. Estas ações devem ser executadas quando a palete passa pela secção de tapete onde as mesmas são retidas e carregadas.

As paletes tipo B devem ser retidas no loal exato do tapete para que possam ser carregadas. Para tal, criou-se nesta secção de tapete, um sensor que simula uma barreira ótica (Figura 60). Quando a barreira é interrompida, o método associado à mesma, o **EntryConv**, é automaticamente executado.

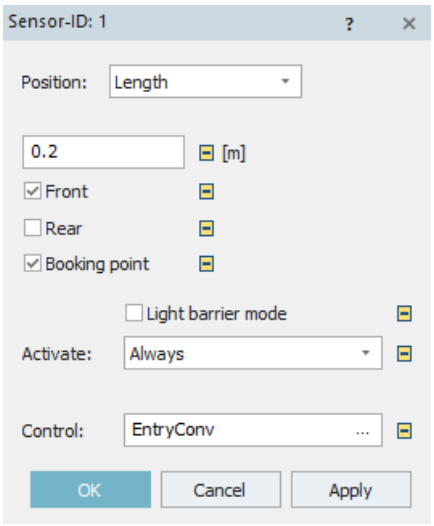


Figura 60 - Configuração do sensor de entrada.

Na listagem seguinte é possível observar o código utilizado no método **EntryConv**. Como descrito aneteriormente, o *if statement* utilizado verifica o estado da variável "Ocupada" de forma a avaliar se para ou não no tapete.

```
if .Models.Model.Conveyor.MU.Ocupada = false
     .Models.Model.Conveyor.stopped := true
     .Models.Model.Conveyor.MU.Ocupada := true
end
```

Listagem 6 - Método **EntryConv**.

4.4.4 Método Dobot 2

Após o carregamento da palete B no tapete transportador, esta segue até chegar à estação em que se encontra o Dobot 2. A chegada da palete B ao posto do Dobot 2 é detetada por um sensor tipo barreira, semelhante ao criado na zona de entrada do tapete, criado no modelo do tapete especialmente para o efeito.

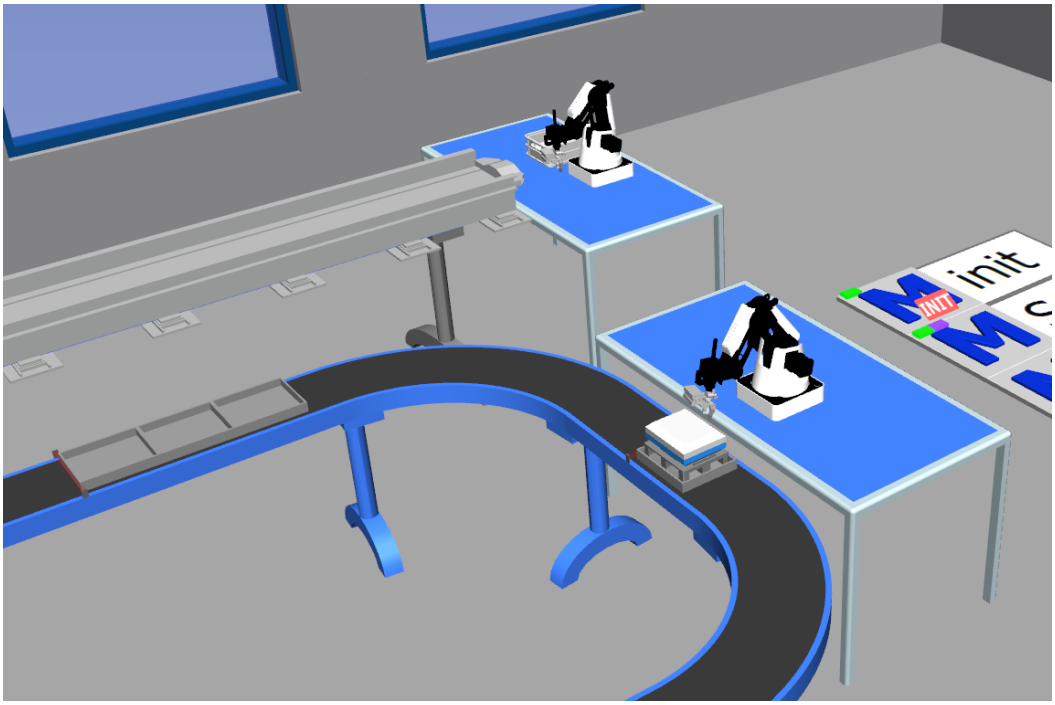


Figura 61 – Etapa 5 - gravação laser executada pelo Dobot 2.

O método **Dobot2**, apresentado na Listagem 7, é executado quando a barreira é interrompida pela chegada da palete B à estação. Este método verifica a propriedade "Ocupada" da palete e, caso apresente o estrado *true*, o tapete é parado e o Dobot 2 efetua a gravação laser. De seguida, é registado na variável "Station" o último local por onde a palete passou, ou seja, o troço do tapete com a estação do Dobot 2. Esta variável será usada para tomar decisões nas estações seguintes para se determinar onde parar a palete.

```
if .Models.Model.Conveyor31.MU.Ocupada =true
     .Models.Model.Conveyor31.stopped := true
     Dobot2._3D.poses.moveTo("Home",2)
     wait 2
     Dobot2._3D.poses.moveTo("Work",2)
     wait (10)
     Dobot2._3D.poses.moveTo("Home",2)
     wait 2
     @.Station := .Models.Model.Conveyor31
     .Models.Model.Conveyor31.stopped := false
end
```

Listagem 7 - Método **Dobot 2**.

4.4.5 Método ExitConv

Terminada a gravação laser, as paletes deixam a estação do Dobot 2 e seguem no tapete até chegarem à zona de saída do tapete, e serem detetadas por outro sensor ótico. As paletes com o conjunto finalizado são retidas na zona de saída do tapete e aguardam que sejam descarregadas pelo Scorbot, sendo depois arrumadas novamente no armazém.

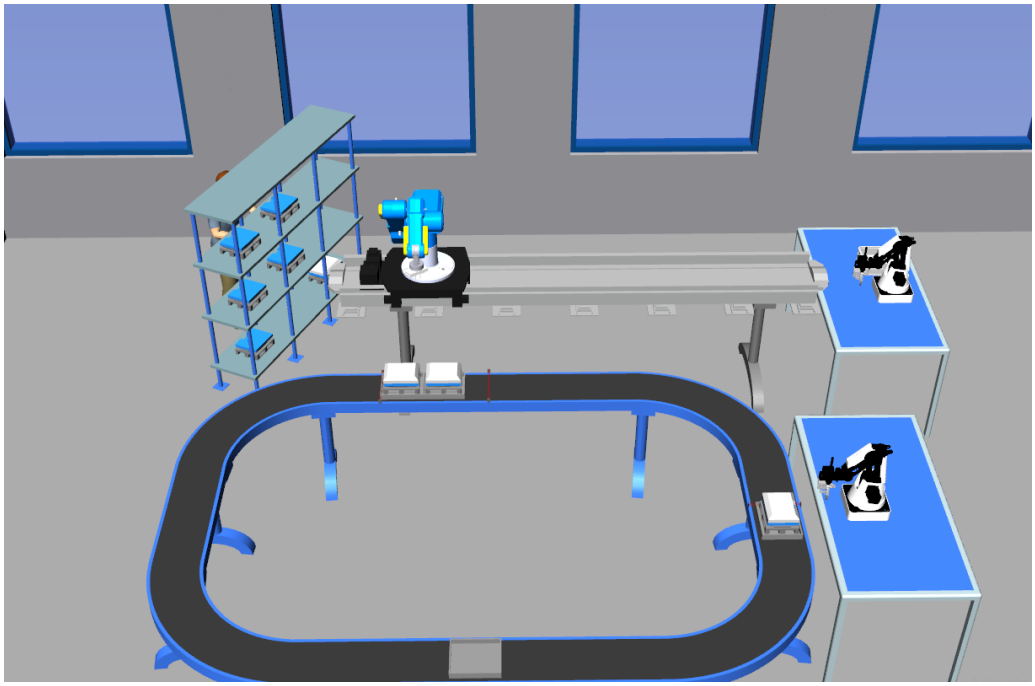


Figura 62 - Etapa 7 - Saída do tapete.

O método **ExitConv** (Listagem 8) é responsável pela paragem das paletes na zona de saída do tapete e da sua libertação após a tansferência. Assim, o tapete é parado quando se verifica que a palete é proveniente do tapete do Dobot 2, utilizando a variável "Station". Durante a descarga da palete, a variável "Ocupada" é alterada para *false*.

```
if @.Station = .Models.Model.Conveyor31
    .Models.Model.Conveyor1.stopped := true
    @.Ocupada:=false
end
```

Listagem 8 - Método **ExitConv**.

5. Comando e Monitorização da Célula de Fabrico

Este capítulo apresenta o desenvolvimento do sistema que comanda e monitoriza a célual de fabrico anteriormente modelada com o Tecnomatix, devendo ter a capacidade de ser utilizado com a célula de fabrico real. Para este efeito houve a necessidade de adaptar o modelo Tecnomatix da célula de fabrico que havia sido desenvolvido anteriormente por forma a cumprir a função de monitorização.

5.1 Aplicação em Matlab

De forma a controlar ambos os processos em simultâneo (célula real e modelo Tecnomatix), foi desenvolvido um controlador independente que é responsável pela emissão das ordens de movimentação que serão recebidas e executadas em simulação, pelo Tecnomatix, e no mundo físico, pela célula real. Para este efeito houve a necessidade de adaptar o modelo Tecnomatix da célula de fabrico que foi desenvolvido anteriormente por forma a cumprir a função de monitorização. Neste ponto optou-se por desenvolver o controlador do modelo Tecnomatix e do processo de sistema real com recurso ao *software* Matlab. As principais razões que fundamentaram esta opção foram as seguintes:

- A programação em Matlab é de fácil aprendizagem, quando comparada com outras linguagens;
- As ferramentas de programação no ambiente Matlab favorecem o rápido desenvolvimento e *debugging* de programas;
- A linguagem Matlab permite a programação por objetos, um paradigma adequado à modelação de sistemas de fabrico;
- O Matlab contém as interfaces de comunicação necessárias à comunicação com os diversos controladores da célula e com o Tecnomatix.

Assim, o controlador foi elaborado em parceria com o professor Francisco e colocado à disposição deste projeto, sendo depois necessário indicar as funções para executar a transferência de informação com o Tecnomatix.

O Tecnomatix disponibiliza algumas funções que, ao serem utilizadas no Matlab, tornam possível o controlo da simulação. De seguida encontram-se listados os comandos utilizados e o seu significado.

- *Actxserver()* – Inicia o Tecnomatix;
- *LoadModel()* – Carrega o ficheiro do projeto;
- *SetVisible()* – Torna a simulação visível, evitando que seja executada em segundo plano;
- *StartSimulation()* – Inicia a simulação;
- *ResetSimulation()* – Para a simulação e repõe o estado inicial;
- *SetValue()* – Envia o valor desejado para uma variável do Tecnomatix;
- *GetValue()* – Lê o valor de uma variável do Tecnomatix.

Na Listagem 9 é possível observar o código utilizado no Matlab para estabelecer a comunicação com o Tecnomatix e iniciar a simulação.

```
PlantSimulation=actxserver('Tecnomatix.PlantSimulation.RemoteControl.15.0';
PlantSimulation.SetVisible(true);
PlantSimulation.LoadModel('C:\Matlab_v3.spp');
PlantSimulation.ResetSimulation('.Models.Model.EventController');
PlantSimulation.StartSimulation('.Models.Model.EventController');
```

Listagem 9 - Código em Matlab responsável pelo início da simulação no Tecnomatix.

De forma conseguir a integração dos dois programas, foi necessário estabelecer uma série de variáveis comuns que são responsáveis pela transferência de instruções. Cada instrução enviada ao Tecnomatix representa a movimentação de uma peça, palete ou conjunto. Assim, tratando-se de uma movimentação, é sempre obrigatório fornecer, através de variáveis, a indicação do ponto de partida e chegada. Além destas, são necessárias outras variáveis de comunicação. Na Figura 63 é possível observar as variáveis utilizadas assim como a função desempenhada por cada uma.

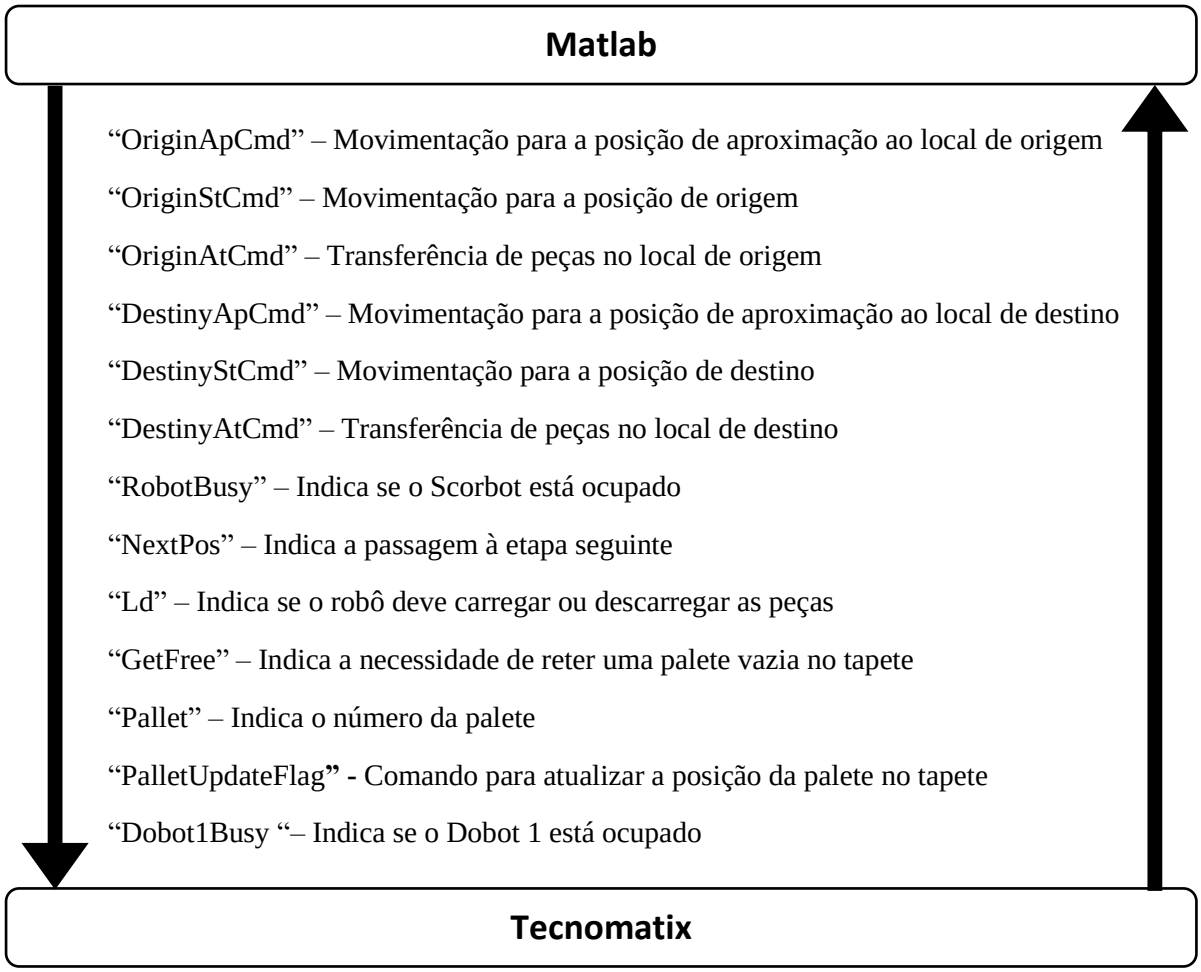


Figura 63 - Variáveis utilizadas na comunicação entre programas.

Depois do início da simulação, o controlador, em Matlab, prossegue enviando e requisitando dados das variáveis do Tecnomatix (Listagem 10), utilizando as funções *SetValue* e *GetValue*, respetivamente. Para este projeto, esta parte do controlador é considerada uma "*black box*", apenas sendo relevantes os *inputs* e *outputs*.

O Matlab envia as instruções com o código SimTalk para o Tecnomatix no formato *string*, de modo a poderem ser facilmente executadas. Uma vez que o Tecnomatix tem a possibilidade de executar instruções definidas no formato *string*, verificou-se que esta forma conferia ao sistema uma maior flexibilidade, colocando do lado do controlador a responsabilidade de definir as *strings* com os comandos, na linguagem SimTalk, adequados à finalidade desejada.

```
PlantSimulation.SetValue('Models.Model.OriginApCmd',ap);
PlantSimulation.SetValue('Models.Model.OriginStCmd',st);
PlantSimulation.SetValue('Models.Model.OriginAtCmd',at);
PlantSimulation.SetValue('Models.Model.NextPos',cmd(1));
PlantSimulation.SetValue('Models.Model.LD',1);
```

Listagem 10 - Exemplo de envio de instruções para o Tecnomatix.

Com este controlador foi também desenvolvido um simulador 2D de forma a verificar a sincronização entre os dois programas. O simulador Matlab mostra todas as posições do armazém, os Dobots, o Scorbot, o alimentador do Dobot 1, o tapete e as paletes tipo B no tapete. As linhas apresentadas na Figura 64 (canto direito) representam as possíveis transferências de entidades entre componentes. Na mesma figura é exibido o momento de uma transferência de um palete com peças do tapete para o armazém. Devido às diferentes geometrias e velocidades de simulação, é possível verificar que a posição das paletes nos dois simuladores não coincide exatamente.

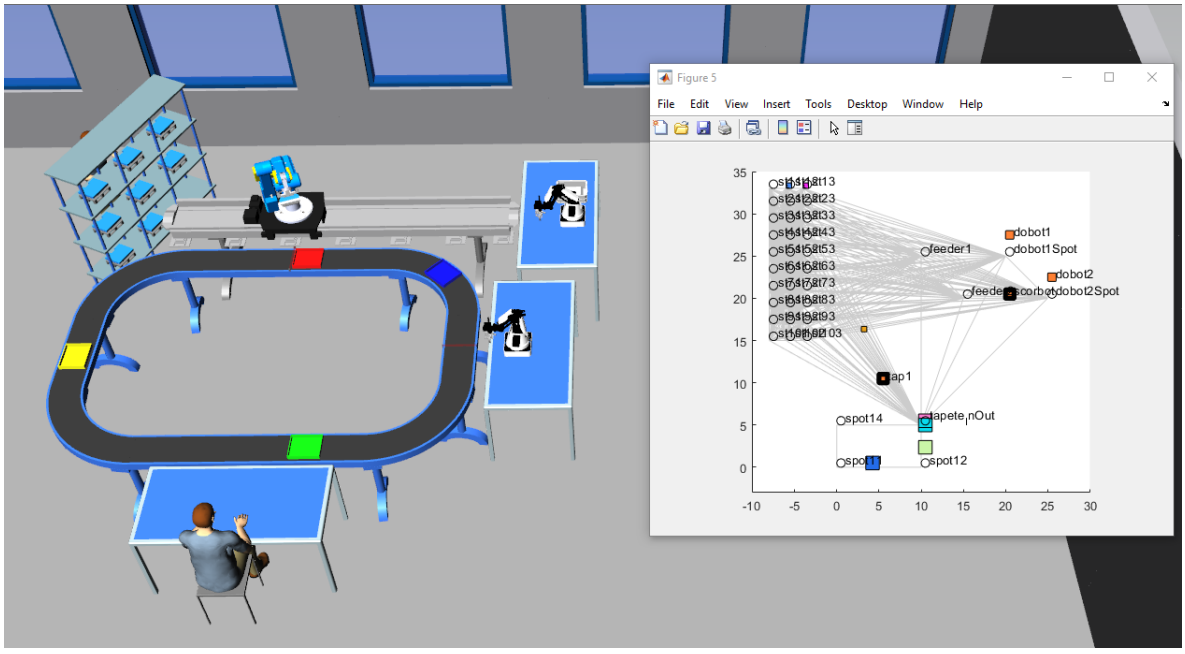


Figura 64 - Momento da transferência de um conjunto de peças no Tecnomatix (esquerda) e no simulador Matlab (Direita).

5.2 Modelo de monitorização em Tecnomatix

Os métodos descritos na secção 4.4 foram desenvolvidos para realizar a simulação no Tecnomatix de forma automática, sem recurso a um controlador externo. No entanto, para realizar a simulação no Tecnomatix controlado através do Matlab, todos os métodos utilizados originalmente no controlo autónomo tiveram de ser alterados ou mesmo removidos.

Para o Tecnomatix receber a informação proveniente do Matlab foram definidos novos métodos, variáveis e geradores (Figura 65). As variáveis definidas contêm os nomes referidos na secção 5.1, podendo ser de 3 tipos, de acordo com o tipo de dados recebido:

- *Real* – Armazena números reais;
- *String* – Variável alfanumérica, armazena caracteres, números e símbolos;
- *Boolean* – Variável binária, armazena verdadeiro (*true*) ou falso (*false*).

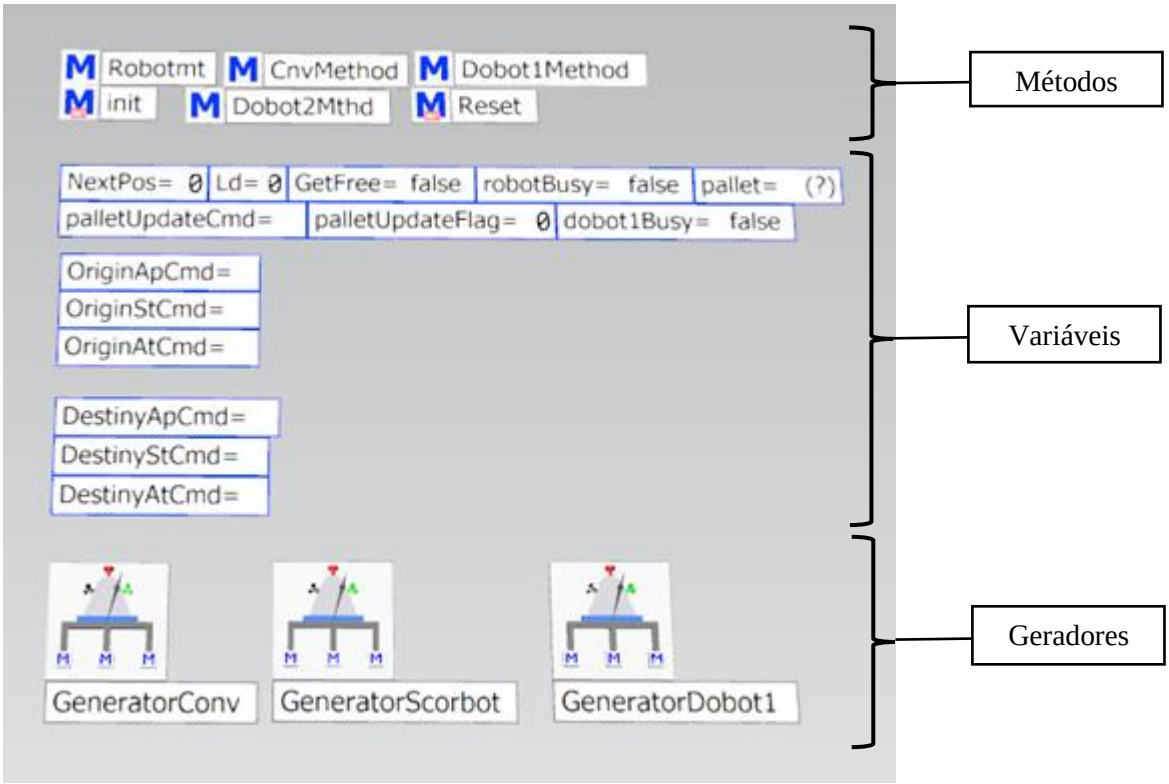


Figura 65 - Métodos, Variáveis e Geradores do modelo integrado.

Analisando a Figura 65 é possível observar os novos métodos desenvolvidos para processar as informações obtidas através do Matlab. Como indicado na secção 5.1, a

informação chega por meio das variáveis, que são posteriormente utilizadas pelos métodos de forma a executar os vários movimentos.

Nesta fase surgiu um problema uma vez que os métodos são rotinas desenvolvidas para serem ativadas apenas por outro objeto do modelo. O problema surgiu, pois a partir do Matlab não é possível ativar os métodos. Assim, foi necessária a criação de geradores (*Generators*) (Figura 66) que têm como função ativar ciclicamente, com uma frequência de 1 Hz os métodos a eles associados. Desta forma, garante-se que ciclicamente o Tecnomatix verifica se existem ações comandadas pelo Matlab e, caso existam, estas são executadas.

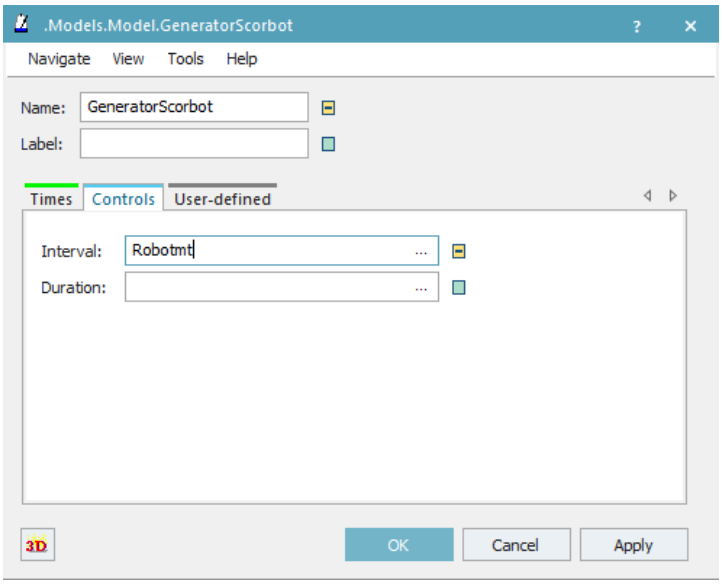


Figura 66 - Definição do Generator.

Uma vez que o Matlab envia as *strings* com os comandos na sintaxe SimTalk, os métodos apenas recorrem à função "*execute*" para executar o código contido na variável designada. Após o Matlab executar a função *SetValue*, o gerador ativa o respetivo método, que por sua vez executa a linha de código contida na variável, ou seja, a movimentação a realizar. Na Listagem 11, é possível observar o método do Scorbot representativo do funcionamento descrito (**Robotmt**). O método faz uso das variáveis "robotBusy" e "NextPos" de forma avaliar se tem alguma movimentação para realizar. Caso as condições se verifiquem, através da função "execute", as instruções provenientes do Matlab são executadas. Entre cada movimento, um intervalo de 0,6 segundos é adicionado através da função *wait* para que não se verifiquem sobreposições de instruções. Por fim, as variáveis de controlo são alteradas de forma a indicar ao Matlab que o ciclo terminou.

```
if robotBusy=false
 if NextPos > 0
   if NextPos < 6
    .Models.Model.robotBusy:=true
    execute(OriginApCmd)
    wait 0.6
    execute(OriginStCmd)
    wait 0.6
    execute(OriginAtCmd)
    wait 0.6
    execute(OriginApCmd)
    wait 0.6
    execute(DestinyApCmd)
    wait 0.7
    execute(DestinyStCmd)
    wait 0.7
    execute(DestinyAtCmd)
    wait 0.7
    execute(DestinyApCmd)
    wait 0.7
    wait 0.7
    .Models.Model.NextPos:=0
    .Models.Model.robotBusy:=false
  end
 end
end
```

Listagem 11 - Método **Robotmt**.

O método do Dobot 1 segue a mesma lógica que o método do Scorbot. Na Listagem 12 é possível observar o código utilizado. Como demonstrado pela listagem, o fator que diferencia este método é o valor permitido para a variável “NextPos”.

```
if dobot1Busy=false
 if NextPos > 5
   if NextPos < 9
    .Models.Model.dobot1Busy:=true
    execute(OriginApCmd)
    wait 0.6
    execute(OriginStCmd)
    wait 0.6
    execute(OriginAtCmd)
    wait 0.6
    execute(OriginApCmd)
    wait 0.6
    execute(DestinyApCmd)
    wait 0.7
    execute(DestinyStCmd)
    wait 0.7
    execute(DestinyAtCmd)
    wait 0.7
    execute(DestinyApCmd)
    wait 0.7
```

```
    .Models.Model.NextPos:=0
    .Models.Model.dobot1Busy:=false
   end
 end
end
```

Listagem 12 - Método **Dobot 1**.

Para o controlo do tapete foi necessário o desenvolvimento de um novo método (**CnvMethod**). Só assim foi possível o sincronismo das paletes entre o controlador e o modelo de monitorização. Este método faz uso de duas variáveis: a "palletUpdateFlag" e a "palletUpdateCmd" para identificar se a palete necessita de ser movida e para receber o comando para efetuar a movimentação respetivamente. O método segue a mesma lógica descrita nos anteriores, ou seja, através da "palletUpdateFlag" é verificado se existe a necessidade de mover a palete. Caso o seja, a instrução na variável "palletUpdateCmd" é executada através da função *execute* (Listagem 13).

```
if palletUpdateFlag > 0
 execute(palletUpdateCmd)
.Models.Model.palletUpdateFlag:=0
 palletUpdateCmd:=""
end
wait 0.02
```

Listagem 13 - Método **CnvMethod.**

Por fim, foi necessário definir o método **Reset**. Tal como o método **Init**, o **Reset** é um método pré-configurado no Tecnomatix que é automaticamente executado quando o botão de terminar a simulação é pressionado. Na Listagem 14 é possível observar o código implementado, através do qual é feito o *reset* de duas variáveis. O comando *deleteMovables* apaga todas as MUs do sistema.

```
.Models.Model.robotBusy :=false
.Models.Model.dobot1Busy :=false
deleteMovables
```

Listagem 14 - Método **Reset**.

6. Conclusões e Trabalho Futuro

O objetivo da presente dissertação consistiu no projeto e desenvolvimento de ferramentas de monitorização, simulação e controlo da célula de fabrico robotizada existente no laboratório de robótica do Departamento de Engenharia Mecânica do ISEL. Foi definido o caso de estudo através do levantamento dos equipamentos que constituem a célula de fabrico. Esta é composta por um braço robótico articulado Scorbot, colocado sobre uma base linear, dois braços robóticos Dobot, um tapete transportador e um armazém. Na fase inicial deste trabalho foram também definidos o *layout* e o processo de fabrico que caracterizam o caso de estudo que será foco dos capítulos posteriores.

Após a definição do caso de estudo, a etapa seguinte consistiu na escolha do *software* de modelação e simulação, utilizando a metodologia descrita na secção 2.3.2. Este processo verificou-se complexo devido à necessidade de testar todos os candidatos para obter as respostas necessárias, sobretudo quando estas não estavam disponíveis na documentação respetiva. Optou-se pelo *software* Tecnomatix, por ser o que melhor correspondia aos requisitos colocados, nomeadamente a capacidade de comunicar com o controlador (Matlab) e de permitir a simulação de modelos 3D, construídos a partir de modelos CAD importados.

O desenvolvimento do modelo 3D da célula de fabrico constituiu a maior porção do trabalho desenvolvido. Além da aprendizagem da utilização do *software*, as principais dificuldades verificaram-se a nível da importação de modelos 3D que não existiam na biblioteca de base do *software* e à reduzida informação de suporte para a programação em SimTalk. Esta dificuldade abrandou o desenvolvimento dos métodos de controlo (SimTalk). Para corresponder aos objetivos propostos foi necessário desenvolver dois modelos: um modelo auto-suficiente (independente do controlador) apenas para simulação e um segundo modelo derivado deste primeiro e que manteve o *layout* da célula, mas que pudesse ser controlado por um *software* externo, o Matlab.

De forma a interagir com o controlador Matlab houve a necessidade de alterar os métodos previamente desenvolvidos para executar a simulação no Tecnomatix sem controlador externo. Apesar de ter sido disponibilizado um controlador previamente elaborado e disponível no Laboratório de Robótica, houve a necessidade de serem programadas as instruções a enviar para o Tecnomatix e os comandos de envio e pedido de informação. Esta integração revelou-se desafiante no sentido de se conseguir estabelecer um modo de funcionamento que garantisse uma comunicação eficaz entre os dois programas (Matlab e Tecnomatix). Verificou-se uma

dificuldade ao tentar executar os métodos do Tecnomatix a partir do Matlab. A solução encontrada foi baseada na implementação de temporizadores que correm os métodos de forma cíclica, sendo necessária a criação de variáveis de controlo para esses métodos. O facto de as ferramentas de *debugging* no editor de código do Tecnomatix não desempenharem as suas funções corretamente dificultou o processo de escrita do mesmo e de depuração da interface com o Matlab.

Apesar de cumpridos todos os objetivos propostos, alguns problemas não obtiveram solução, nomeadamente, alguns *bugs* de simulação e de integração com o Matlab que ficaram presentes, como as simulações não serem totalmente síncronas e algumas movimentações dos robôs serem incongruentes. De salientar, no entanto, que estes problemas nunca inviabilizam a simulação. A continuação natural deste projeto seria a integração com o sistema real, tornando-o um sistema SCADA completo. Para tal será necessário o desenvolvimento da interface entre o controlador Matlab e os diversos equipamentos como apresentado na

Figura 22, de forma que o Matlab comande todo o sistema. Para esta integração, haverá também a necessidade de medir experimentalmente os valores dos parâmetros de funcionamento (velocidades de tapete, tempos de movimentação, etc.), uma vez que, para efeitos de simulação, estes foram arbitrados.

Bibliografia

[1] S. Kalpakjian, S. Schmid, and V. Sekar, *Manufacturing Engineering and Technology*. 2014.

[2] "O que a digitalização de documentos e a indústria 4.0 têm em comum?" https://netscandigital.com/blog/o-que-a-digitalizacao-de-documentos-e-a-industria-4-0-tem-em-comum/ (accessed Nov. 20, 2021).

[3] H. Kagermann, "Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group," 2013.

[4] W. MacDougall, "Germany Trade & Invest," 2014.

[5] K. Schwab, *The Fourth Industrial Revolution*. 2016.

[6] NSF, *Cyber-Physical Systems (CPS)*. 2021.

[7] A. Ujvarosi, "EVOLUTION OF SCADA SYSTEMS," 2016.

[8] S. Nahavandi, "Industry 5.0 – A Human-centric Solution," 2019.

[9] E. H. Østergaard, "WELCOME TO INDUSTRY 5.0 The 'human touch' revolution is now under way," 2018.

[10] M. Breque, L. De Nul, and A. Petridis, "Industry 5.0 - Towards a sustainable, humancentric and resilient European industry," 2021.

[11] M. Breque, L. De Nul, and A. Petridis, "Industry 5.0 Towards a sustainable, human-centric and resilient European industry," 2021.

[12] P. K. Reddy, "Industry 5.0: A Survey on Enabling Technologies and Potential Applications."

[13] S. A. Boyer, *Supervisory Control and Data Acquisition (SCADA) Systems*. 2004.

[14] *IEC 62264*. 2016.

[15] K. Cope, "What is the Automation Pyramid? | RealPars," 2020. https://realpars.com/automation-pyramid/ (accessed Oct. 25, 2021).

[16] "What is the Automation Pyramid? | RealPars." https://realpars.com/automation-pyramid/ (accessed Nov. 16, 2021).

[17] "What is SCADA? Supervisory Control and Data Acquisition." https://oleumtech.com/what-is-scada (accessed Nov. 16, 2021).

[18] M. P. Groover, *Automation, Production Systems, and Computer-Integrated Manufacturing*. 2015.

[19] "Remote Terminal Units – RTUs," Accessed: Jan. 05, 2022. [Online]. Available: https://new.siemens.com/global/en/products/automation/industrial-communication/industrial-remote-communication/telecontrol/rtu-remote-terminal-unit/simatic-rtu-3000c.html.

[20] J. Robles, "Architecture for SCADA with mobile remote components," 2010.

[21] J. Nicol, J. Banks, B. Carson, and D. Nelson, "Discrete-Event System Simulation," 2001.

[22] J. Banks and C. Sokolowski, *Principles of Modeling and Simulation*. 2009.

[23] M. Jahangirian, T. Eldabi, A. Naseer, L. Stergioulas, and T. Young, "Simulation in

manufacturing and business," 2009.

[24] M. I. Kellner, "Software process simulation modeling: Why? what? how?," 1999.

[25] S. Robinson, *Simulation: The Practice of Model Development and Use*. 2004.

[26] C. Cassandras, "Introduction to Discrete Event Systems," 2008.

[27] V. Hlupic, *Simulation Modelling Software Approaches to Manufacturing Problems*. 2011.

[28] J. Nikoukaran, V. Hlupic, and R. Paul, "CRITERIA FOR SIMULATION SOFTWARE EVALUATION," 1998.

[29] "Intelitek," 2020. https://intelitek.com/ (accessed Sep. 22, 2021).

[30] "DOBOT Magician." https://www.dobot.cc/dobot-magician/product-overview.html (accessed Sep. 23, 2021).

[31] "Tecnomatix | Siemens Software." https://www.plm.automation.siemens.com/global/pt/products/tecnomatix/ (accessed Nov. 04, 2021).

[32] "AnyLogic: Simulation Modeling Software Tools & Solutions for Business." https://www.anylogic.com/ (accessed Nov. 04, 2021).

[33] "3D Simulation Modeling and Analysis Software | FlexSim." https://www.flexsim.com/pt/ (accessed Nov. 04, 2021).

[34] "Visual Components - 3D manufacturing simulation software." https://www.visualcomponents.com/ (accessed Nov. 04, 2021).

[35] K. B. Bruce, *Foundations of Object-oriented Languages: Types and Semantics*. 2002.

[36] "Pull control (general description)." https://docs.plm.automation.siemens.com/content/plant_sim_help/15/plant_sim_all_in_one_html/en_US/tecnomatix_plant_simulation_help/objects_reference_help/material_flow_objects/shared_properties_of_the_material_flow_objects/dialog_items_of_the_objects/tab_controls/pull_control_general_description.html (accessed Sep. 23, 2021).

[37] "Target control (pick-and-place)." https://docs.plm.automation.siemens.com/content/plant_sim_help/15/plant_sim_all_in_one_html/en_US/tecnomatix_plant_simulation_help/objects_reference_help/material_flow_objects/pickandplace_robot/dialog_window_of_the_pick_and_place_robot/tab_controls/target_control_pick_and_place.html (accessed Sep. 23, 2021).

[38] "Steffen Bangsow examples." https://www.bangsow.eu/suche_kategorie_en.php?kategorie=7 (accessed Sep. 22, 2021).

[39] S. Bangsow, *Tecnomatix Plant Simulation*. Springer International Publishing, 2020.

[40] "Mobile Objects (MUs)." https://docs.plm.automation.siemens.com/content/plant_sim_help/15.1/plant_sim_all_in_one_html/en_US/tecnomatix_plant_simulation_help/objects_reference_help/mobile_objects/mobile_objects_mus.html (accessed Sep. 22, 2021).

Anexo 1

Aplicações de técnicas de simulação em manufatura

Table 4
Applications of simulation techniques in manufacturing and business.

Application	Simulation technique	Industry sector	A sample of Class A papers	No. of Class A papers	Total no. of papers (Class A + B + C)
Assembly line balancing	DES	Computer hardware	[85]	1	1
	Other hybrid techniques	Optic lens assembly	[142]	1	1
Capacity planning	DES	Generic service industries	–	–	1
		Generic part manufacturing	–	–	5
		Pump production	–	–	1
		Transformer manufacturing	[61]	2	2
		Beverages	[136]	1	1
	SD	Extruded food production	[96]	1	1
	Monte Carlo simulation	Electricity generation	–	–	1
	Petri-net simulation	Generic part manufacturing	–	–	1
	Other hybrid techniques	Financial/insurance and education	–	–	1
Cellular manufacturing	Virtual simulation	Automotive	[94]	1	1
Transportation management	DES	Urban traffic management	[111]	1	1
		Stationary production	[90]	1	1
		Concrete transport	–	–	1
	ABS	Railway transport	[16]	1	1
		General transport	–	–	1
	Petri-net simulation	Generic part manufacturing	–	–	1
	Traffic simulation	Traffic control, freeway traffic control	[17]	9	9
		City logistics	–	–	1
	Hybrid (SD&DES)	Traffic control	–	–	1
	Other hybrid techniques	Highway traffic control, urban traffic control	[66]	4	5
	Other techniques	Airport management	[130]	1	1
		Travel navigation	[62]	1	1
Facility location	Other hybrid techniques	Transportation	[63]	1	1
Forecasting	SD	Aircraft manufacturing	[75]	1	1
Inventory management	DES	Generic part manufacturing		–	3
		Automotive	[110]	2	2
		Construction	[101]	1	1
		Chemical products	[83]	1	1
		Recycled parts	–	–	1
		Insurance and education	[32]	1	1
	Monte Carlo simulation	Nuclear and spacecraft	–	–	1
	Other techniques	Retailing	–	–	1
Just-in-time	DES	Generic part manufacturing	–	–	3
	Intelligent simulation	Generic part manufacturing	–	–	1
Process engineering-manufacturing	DES	Ship building	[98]	1	1
		Automotive	[100]	3	3
		Pharmaceuticals	[15]	1	1
		Generic part manufacturing	[33]	2	7
		Electronics	[109]	3	4
		Aluminium gas cylinder	[42]	1	1
	SD	Automotive	[25]	1	1
		Generic part manufacturing	–	–	1
	ABS	Generic part manufacturing	[69]	1	1
	Monte Carlo simulation	Generic part manufacturing	–	–	1
	Petri-nets simulation	Generic part manufacturing	–	–	1
	Virtual simulation	Generic part manufacturing	–	–	1
		Electronics	[22]	1	1
	Intelligent simulation	Generic part manufacturing	–	–	2
		Electronics	–	–	1
	Other hybrid techniques	Generic process industry	–	–	1
	Other techniques	Various manufacturing industries	[38]	1	1
		Beverage	–	–	1
Process engineering-service	DES	Retailing	–	–	1
		Generic service industry	–	–	1
		Printing	[40]	1	1
		Consulting	[52]	1	1
		Distribution	[71]	1	1
		Container terminal	[114]	1	1
		Construction waste handling	[27]	1	1
		Construction	–	–	1
	SD	Logistics	[8]	1	1
		Insurance	[1]	1	1
	Distributed simulation	Electricity generation	[73]	1	1
		Information and communications	[119]	1	1

(*continued on next page*)

Table 4 (*continued*)

Application	Simulation technique	Industry sector	A sample of Class A papers	No. of Class A papers	Total no. of papers (Class A + B + C
Production planning and inventory control	DES	Electronics	[57]	1	1
		Generic part manufacturing	–	–	3
	ABS	Generic part manufacturing	–	–	1
	Distributed simulation	Aluminium production	[143]	1	1
		Heater manufacturing	[58]	1	1
		Automotive	[106]	1	1
	Hybrid approach (DES&SD)	Generic part manufacturing	–	–	2
		Electronics	[104]	1	1
	Other hybrid techniques	Aluminium sheet production	[6]	1	1
Purchasing	DES	Energy	–	–	1
Resource allocation	DES	Transportation	[46]	1	1
		Generic manufacturing	–	–	1
	ABS	Shipping terminals	[39]	1	1
	Monte Carlo simulation	Jet engine repair	–	–	1
	Distributed simulation	Electricity	–	–	1
	Intelligent simulation	Generic manufacturing	–	–	2
	Hybrid approach (DES&SD)	Semi-conductor manufacturing	[105]	1	1
	Other hybrid techniques	Research	[77]	1	1
	Other techniques	Construction	–	–	1
Scheduling	DES	Generic part manufacturing	[121]	1	14
		Semi-conductor manufacturing and electronics	[14]	4	5
		Container terminals	[44]	1	2
		Airline	[138]	1	1
		Re-manufacturing	–	–	1
		Printing	[87]	1	1
	ABS	Generic part manufacturing	–	–	1
	Monte Carlo simulation	Electronics	–	–	1
	Petri-nets simulation	Generic part manufacturing	–	–	1
	Intelligent simulation	Generic part manufacturing	–	–	4
		Computer hardware	[31]	1	1
	Other hybrid techniques	Generic part manufacturing	–	–	1
	Other techniques	Generic part manufacturing	–	–	1
Strategy	DES	Furniture manufacturing	[18]	1	1
	SD	Electronics	[123]	1	1
		Generic part manufacturing	–	–	1
		Consulting	[76]	1	1
		Automotive	[55]	1	2
		Electricity generation	[135]	1	1
		Financial and aircraft manufacturing	[74]	1	1
		Information and communication	[140]	2	2
		News publication	–	–	1
		High-tech	[80]	1	1
		Chain restaurant	[65]	1	1
		Aeroengine manufacturing	[103]	1	1
		National energy management	[88]	1	1
		Construction	[68]	2	3
	ABS	Information and communication	[28]	1	1
		Electricity	[95]	1	1
		Generic part manufacturing	–	–	2
	Simulation gaming	Financial and insurance	[49]	1	1
	Monte Carlo simulation	Generic part manufacturing	–	–	1
		Energy	–	–	1
	Hybrid (SD&DES)	Electronics	[13]	1	1
	Other techniques	Information and communications	–	–	1
Supply chain management	DES	Generic part manufacturing	[20]	3	7
		Chemical products	–	–	1
		Food	[21]	1	1
		Notebook computer	[117]	1	1
		Retailing	[36]	1	1
	SD	Electronics		–	1
		Generic part manufacturing	–	–	2
		Machine tools manufacturing	[4]	1	1
	ABS	Mold manufacturing	–	–	1
		Appliance/electronics/computer	–	–	1
		Computer hardware	[126]	1	1
	Simulation gaming	Chemicals	[133]	1	1
	Petri-nets	Food	[131]	1	1
	Distributed simulation	Automotive	[127]	1	1
		Generic part manufacturing	–	–	1
	Hybrid approach (DES&SD)	Information and communication	[107]	1	1

Table 4 *(continued)*

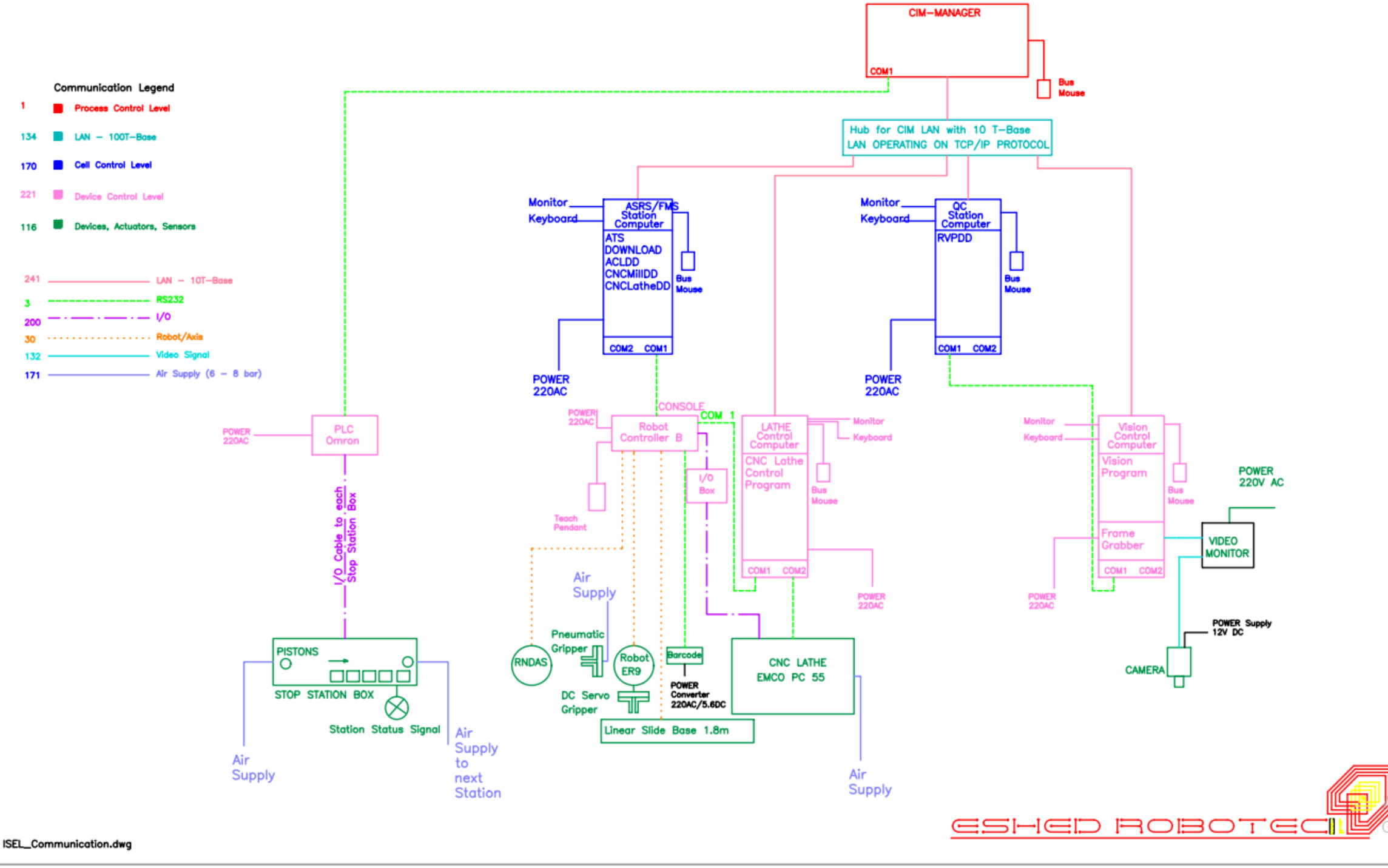
Application	Simulation technique	Industry sector	A sample of Class A papers	No. of Class A papers	Total no. of papers (Class A + B + C)
		Generic part manufacturing	–	–	1
	Other hybrid techniques	Trading	[7]	1	1
		Generic part manufacturing	–	–	2
		Packaging/machine manufacturing/iron metallurgy/apparel manufacturing/dairy	[79]	1	1
Workforce planning	DES	Franchised food	[50]	1	1
		Electronics	[115]	1	1
		Airplane manufacturing	[145]	1	1
		Call centres	–	–	1
	Other hybrid techniques	Steel production	[86]	1	1
Maintenance management	DES	Generic part manufacturing	–	–	1
	Monte Carlo simulation	Generic part manufacturing	–	–	1
	Virtual simulation	Machine building	–	–	1
Knowledge management	DES	Generic part manufacturing	[10]	1	1
		Aircraft manufacturing	–	–	1
		Construction	–	–	1
	SD	Generic part manufacturing	–	–	2
		Pharmaceutical	[137]	1	1
Project management	DES	Aircraft maintenance	–	–	1
		Oil and gas	[19]	1	1
		Chocolate	–	–	1
		Construction	–	–	1
		Software development	[5]	1	1
		Consulting	–	–	1
	SD	Semi-conductor manufacturing	[37]	1	1
		Software development	[108]	2	6
		Generic projects	–	–	1
		Construction	[72]	1	1
	Monte Carlo simulation	Software development	–	–	1
		Construction	–	–	1
	Petri-net simulation	Construction	[112]	1	1
	Intelligent simulation	Construction	–	–	1
	Hybrid approach (DES&SD)	Software development	[81]	1	2
Organizational design	DES	Generic part manufacturing	–	–	1
	SD	Pharmaceutical	[113]	1	1
	ABS	Generic part manufacturing	–	–	3
	Simulation gaming	Information and communications	[92]	1	1
	Other hybrid techniques	Generic manufacturing	–	–	1
	Other techniques	Trading	[41]	1	1
		Pharmaceuticals	[51]	1	1
Management training and education	DES	Education	[139]	1	1
	SD	Education/software development	[99]	1	1
	Simulation gaming	Education	[144]	2	3
		Construction	[91]	1	1
		Clinical instrument manufacturing	[64]	1	1
	Distributed simulation	Education	–	–	1
	Virtual simulation	Education/construction	–	–	1
	Other hybrid techniques	Education	[78]	1	1
	Other techniques	Education	[47]	1	2
		Construction	–	–	1
Financial management	DES	Electronics	[122]	1	1
		New-product-development	–	–	1
	Monte Carlo simulation	Stock markets	[141]	1	1
		Property	[48]	1	1
		Accountancy	–	–	1
Quality management	DES	Software development/education	–	–	1
		Automotive	[30]	1	1
	SD	Computer hardware	[124]	1	1
		Construction	[67]	1	1

Anexo 2

Esquema elétrico da célula Eshed

Communication Layout
ISEL

NOVEMBER 1999



ESHED ROBOTEC

ISEL_Communication.dwg