Faculdade de Engenharia da Universidade do Porto


# Indústria 4.0: Colaboradores robóticos em cenários de logística

**Lucas Santos Damas**

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: António Pedro Rodrigues Aguiar

21 janeiro de 2019

# Resumo

Nas últimas décadas tem-se assistido a uma crescente intensificação em investimento, investigação e desenvolvimento na área da robótica com particular relevo em aplicações na indústria. Esta dissertação insere-se no contexto da indústria 4.0 e tem como objetivo principal o estudo, desenvolvimento, implementação em simulação e análise de resultados de um exemplo de uma fábrica autónoma que utiliza robôs móveis em cenários de logística.

A primeira parte da tese apresenta uma breve descrição da motivação do problema no âmbito do tema, objetivos e tópicos relativos ao planeamento de caminhos e algoritmos de movimento de robôs. De seguida, é apresentado de uma forma geral a arquitetura do sistema de rede de robôs móveis para aplicações de serviços de logística em fábricas, sendo exposto a metodologia e ferramentas utilizadas, a arquitetura concetual e a arquitetura funcional, para um caso específico em que são considerados 2 robôs móveis num ambiente de fábrica. Neste cenário, os robôs que são totalmente autónomos, têm como função realizar as tarefas pedidas que são enviadas através de um painel de controlo. A fábrica é composta por 2 sítios de descargas onde será entregue o produto final; 2 sítios de carga onde os robôs vão carregar as matérias primas (neste projeto garrafas de plástico) e 3 subestações de manufatura, onde as 3 enchem as garrafas com sabores diferentes, mas com custos diferentes e aleatórios dentro de uma gama aceitável para o fabrico de garrafas de água. De acordo com o número de pedidos feitos no painel de controlo do sistema, os robôs terão de transportar a matéria prima da zona de carga até à zona de descarga, passando pelas subestações de manufatura. As trajetórias para cada local pretendido, são pré-definidas e otimizadas. Também é imposto que os robôs não possam de modo algum ter colisões. Para o efeito, é considerado que cada robô tenha como sensor um lidar e que esteja continuamente a correr um algoritmo para evitar colisões (entre obstáculos na fábrica e outros robôs). No exemplo, que foi implementado em simulação, existem também custos de produção e de transporte associados, bem como simulação de possíveis avarias nos vários itens da fábrica (Carga, Subestação, Descarga), que são geradas de acordo com uma determinada função de fiabilidade probabilística.

A última parte da dissertação é dedicado aos testes e demonstrações de resultados em simulação para vários casos. Também se apontam perspetivas futuras no âmbito do trabalho.

# Abstract

In the last few decades there has been a growing intensification in investment, research and development in the area of robotics with particular emphasis on applications in industry. This dissertation is in the scope of the industry 4.0 and has as main objective the study, development, implementation in simulation and results analysis of an example of an autonomous factory that uses mobile robots in logistics scenarios.

The first part of the thesis presents a brief description of the motivation of the problem within the theme, objectives and topics related to robot path planning and motion control algorithms. Next, the architecture of the mobile robot network system for applications of logistics services in factories is presented, being exposed the methodology and tools used, the conceptual architecture and the functional architecture for a specific case in which 2 mobile robots are considered in a factory environment. In this scenario, the robots that are fully autonomous have to perform the requested tasks that are sent through a control panel. The factory consists of 2 discharging sites where the final product will be delivered, 2 loading sites where the robots will load the raw materials (in this project plastic bottles) and 3 manufacturing substations where the 3 fill the bottles with different flavors, but with different and random costs within an acceptable range for the manufacture of water bottles. According to the number of orders placed on the control panel of the system, the robots will have to transport the feedstock from the loading zone to the unloading zone through the manufacturing substations. The trajectories for each target location are pre-defined and optimized. Also, it is imposed that the robots cannot in any way have collisions. In order to avoid collisions (between obstacles in the factory and other robots), it is considered that each robot has a Lidar sensor and is continuously running collision avoidance algorithm. In the example, which was implemented in simulation, there are also associated production and transport costs, as well as simulation of possible faults in the various items of the plant (Load, Substation, Discharge), which are generated according to a certain probabilistic reliability function.

The last part of the dissertation is devoted to testing and demonstration of results in simulation for several cases. Future perspectives are also described in the scope of the work.

# Agradecimentos

Este projeto representa o fim de uma etapa da minha vida, a de estudante. Representa uma parte da minha vida que vou deixar para trás, mas que será sempre relembrada com saudade e nostalgia. Uma etapa longa, com altos e baixos, cheia de incertezas que, no fim, culminou numa sensação de objetivo cumprido. A conclusão deste percurso não seria possível sem o apoio da minha família, amigos e professores.

Em primeiro lugar, agradeço ao meu orientador professor António Aguiar pela pronta disponibilidade em me ajudar, na sabedoria transmitida durante a elaboração da dissertação e paciência que teve. Um obrigado aos professores que tive durante todas as fases de estudante, porque eles contribuíram, cada um à sua maneira, para ser quem sou hoje.

Em segundo lugar, não podia de deixar de agradecer aos amigos que conheci durante os meus anos de infância e adolescência com quem ainda convivo e converso inúmeras vezes. São um grupo de pessoas que me apoiaram e tornaram muitas vezes a dificuldade numa mera especulação. Tenho a certeza que este grupo ficará para a vida; a estes, junto os amigos que fiz nestes últimos 5 anos de formação em engenharia, com quem desenvolvi boas amizades e sempre pude contar quando precisava de ajuda a nível académico e pessoal. Agradeço às minhas amizades recentes, que em pouco tempo mostraram muito. Segundo o autor "Lilian Tonet", "As pessoas entram em nossa vida por acaso, mas não é por acaso que elas permanecem."

Por fim, agradeço às pessoas que mais contribuíram para a minha vida, que sempre me indicaram o caminho que consideraram correto, partilharam os valores que hoje fazem de mim uma pessoa melhor, apoiaram nos bons e maus momentos, tornaram-me na pessoa e no homem que sou hoje, com muito orgulho. Aos meus pais, irmã, avós, tios e primos, o meu enorme e sincero obrigado por tudo o que fizeram por mim. Vocês são e serão sempre o melhor da minha vida. Nunca conseguirei agradecer, nem retribuir o suficiente, o privilégio de pertencer a um núcleo familiar tão integro. Obrigado, obrigado, obrigado…

Lucas Santos Damas

“A person who never made a mistake never tried anything new.”

Albert Einstein

# Conteúdo

**Resumo** ..... **iii**

**Abstract** ..... **v**

**Agradecimentos** ..... **vii**

**Conteúdo** ..... **xi**

**Lista de figuras** ..... **xiii**

**Lista de tabelas** ..... **xv**

**Abreviaturas e Símbolos** ..... **xvii**

**Capítulo 1** ..... **1**

Introdução ..... 1
1.1 - Contexto ..... 1
1.2 - Objetivos ..... 2
1.3 - Estrutura do documento ..... 3

**Capítulo 2** ..... **5**

Revisão Bibliográfica ..... 5
2.1- Indústria 4.0 ..... 5
2.2- A automação na indústria ..... 7
2.3- Planeamento de caminhos para robôs móveis ..... 9
2.4- Controlador para seguir pontos predefinidos ..... 15

**Capítulo 3** ..... **19**

Arquitetura do sistema de rede de robôs móveis ..... 19
3.1- Metodologias e ferramentas utilizadas ..... 20
3.2- Arquitetura concetual ..... 21
3.3- Cinemática dos robôs ..... 29

..... **29**
3.4- Arquitetura funcional ..... 30
3.5- Diagrama detalhado dos sistemas principais (robôs e fábrica) ..... 33
3.6- Otimização de custos de trajetória e custos de produto ..... 36
3.7- Controlador para desviar de obstáculos criados no mapa ..... 40
3.8- Controlador para desviar de um robô do outro ..... 42
3.9- Avarias ..... 43

3.10-Base Dados ..... 45
3.11-Interface utilizador fábrica ..... 48

**Capítulo 4 ..... 51**
Demonstração de resultados ..... 51
4.1- Tempos de produção ..... 52
4.2- Teste de avarias ..... 55
4.3- Teste do controlador para desviar obstáculos ..... 59

**Capítulo 5 ..... 61**
Conclusão e trabalho futuro ..... 61
5.1- Conclusão ..... 61
5.2- Trabalho futuro ..... 62

**Referências ..... 63**

# Lista de figuras

**Figura 2.1.1** - Revoluções industriais. Fonte: [industry4.0_2019].....5

**Figura 2.2.1** - Uma fábrica da Amazon. Fonte: [Shead2019].....8

**Figura 2.3.1** - Diagrama de Voronoi. Fonte: [Aurenhammer1991]. ..... 10

**Figura 2.3.2** - Visibility graph. Fonte: [VisibilityGraphs2019]..... 10

**Figura 2.3.3** - (a) Decomposição em células trapezoides (b) gráfico adjacente que corresponde aos caminhos entre células. Fonte: [Abbadi2015]. ..... 11

**Figura 2.3.4** - (a) Decomposição de células aproximadas com o método de aproximação de voxel (b) método de aproximação quad-tree. Fonte: [Abbadi2015]. ..... 12

**Figura 2.3.5** - Exemplo de um campo potencial. Fonte: [PotentialFields2013]. ..... 12

**Figura 2.3.6** - Algoritmo PRM (4000 nós e 5m). ..... 14

**Figura 2.3.7** - Algoritmo PRM (2000 nós e 5m). ..... 14

**Figura 2.3.8** - Algoritmo PRM (100 nós e 5m). ..... 14

**Figura 2.3.9** - Algoritmo PRM (100 nós e 1m). ..... 14

**Figura 2.4.1** - Caso real do controlador PurePursuit. Fonte: [PPC2019]. ..... 16

**Figura 2.4.2** - Caso real do controlador PurePursuit com small Look Ahead. Fonte: [PPC2019] ..... 16

**Figura 2.4.3** - Caso real do controlador PurePursuit com Large Look Ahead. Fonte: [PPC2019]17

**Figura 3.2.1** - Diagrama concetual do SORS..... 21

**Figura 3.2.2** - Dependência de serviços robóticos. ..... 23

**Figura 3.2.3** - Grupo de serviço de acionamento de dispositivos. ..... 24

**Figura 3.2.4** - Grupo do serviço de Conhecimento..... 25

**Figura 3.2.5** - Diagrama dos serviços de tarefas..... 26

**Figura 3.3.1** - Robô diferencial. Fonte: [MRS2018] ..... 29

**Figura 3.4.1**- Diagrama funcional ..... 32

**Figura 3.5.1** - Fluxograma de baixo nível..... 35

**Figura 3.6.1** - Diagrama de árvore (AOC)..... 38

**Figura 3.6.2** - Diagrama de todas as distâncias da fábrica utilizadas no algoritmo..... 39

**Figura 3.7.1** - Esquema representativo do controlador evitar obstáculos ..... 40

**Figura 3.9.1** - Painel de controlo da fábrica ..... 44

**Figura 3.9.2** - Função densidade de probabilidade de uma variável aleatória com distribuição exponencial. R(t) é a função de fiabilidade. ..... 45

**Figura 3.11.1 -** Interface controlo de ordens da fábrica, a) tipo de produto a ser produzido, b) quantidade do produto a ser produzido, c) descrição completa do estado dos pedidos, d) botão atualizar a tabela, e) inserção do pedido na base de dados. ..... 48

**Figura 3.11.2 -** Interface da visualização do uso de cada máquina da fábrica, a) descrição completa do tipo, número de peças e tempo de utilização de cada máquina, b) botão atualizar a tabela. ..... 49

**Figura 3.11.3 -** Interface da visualização da descarga da fábrica, a) descrição completa do tipo e número de descargas, b) botão atualizar a tabela. ..... 49

**Figura 4.1** - Mapa utilizado no simulador ..... 51

**Figura 4.2** - Fábrica em funcionamento ..... 52

**Figura 4.1.1** - Colisão inesperada ..... 53

**Figura 4.1.2** - Fábrica sem avarias ..... 53

**Figura 4.1.3** - Fábrica com avaria em 1 carga ..... 53

**Figura 4.1.4** - Fábrica com avarias em 2 subestações ..... 54

**Figura 4.1.5** - Fábrica com avaria em 1 descarga ..... 54

**Figura 4.1.6** - Fábrica com um único robô ..... 54

**Figura 4.2.1** - Histograma com o $\wedge = 0.00001$ ..... 56

**Figura 4.2.2** - Histograma com o $\wedge = 0.0001$ ..... 56

**Figura 4.2.3** - Histograma com o $\wedge = 0.01$ ..... 56

**Figura 4.2.4** - Tempo necessário até ocorrer a 1ª avaria. ..... 57

**Figura 4.2.5** - Números de pedidos até ocorrer avaria. ..... 57

**Figura 4.2.6** - Tempo necessário para a fábrica parar. ..... 58

**Figura 4.2.7** - Pedidos até a fábrica deixar de funcionar. ..... 58

**Figura 4.3.1** - Teste 1 do controlador evitar obstáculos ..... 59

**Figura 4.3.2** - Cálculos do teste 1 ..... 59

**Figura 4.3.3** - Teste 2 do controlador evitar obstáculos ..... 59

**Figura 4.3.4** - Cálculos do teste 2 ..... 59

**Figura 4.3.5** - Teste 3 do controlador evitar obstáculos ..... 60

**Figura 4.3.6** - Cálculos do teste 3 ..... 60

# Lista de tabelas

**Tabela 3.10.1** - Tabelas da base de dados com o nome de cada tabela a negrito. ..... 46

# Abreviaturas e Símbolos

**Lista de abreviaturas**

| | |
|---|---|
| AOC | Algoritmo de otimização de custos |
| FEUP | Faculdade de Engenharia da Universidade do Porto |
| GUI | *Graphical user interface* |
| MES | *Manufacturing execution systems* |
| PRM | *Probabilistic roadmap* |
| SORS | Sistema Robótico Orientado a Serviços |

# Capítulo 1

# Introdução

Este capítulo tem como objetivo introduzir o tema "Indústria 4.0: Colaboradores robóticos em cenários de logística". Na secção 1.1 apresenta-se o contexto do problema, em seguida é apresentado os objetivos do projeto na secção 1.2 e, por último, é descrito a estrutura do documento científico em questão.

## 1.1 - Contexto

A utilização de sistemas robóticos tem estado cada vez mais presente em qualquer indústria ou serviço que tenha o objetivo de aumentar a eficácia e rapidez dos seus processos. É crescente e imprescindível o uso de robôs para processos mecânicos e repetitivos. A presença de robôs num ambiente fabril tem inúmeras vantagens tais como a rapidez, eficiência e no custo que acarreta. Nem tudo são vantagens, porque essa implementação requer inúmeros requisitos para que tudo funcione a cem por cento.

A segurança tanto a nível de software e hardware são partes fulcrais num ambiente fabril, portanto não poderemos deixar de parte este ponto tão importante. Quanto melhor for a segurança mais confiável será o equipamento e algoritmos.

Quanto ao software existem requisitos que têm de ser cumpridos, tais como a segurança de dados, comandos fiáveis no entanto não poderá existir nenhum "bug" que afete o bom funcionamento da fábrica e previsibilidade dos robôs ou máquinas.

O hardware terá que suportar o software implementado e sensores que ajudarão se algum imprevisto acontecer e consequentemente contornar o problema, ou seja, conseguir que o sistema seja o mais previsível possível.

Os pontos atrás mencionados tornam-se ainda mais relevantes no contexto da indústria 4.0 que é responsável pela fusão do mundo físico com o digital, deste modo os processos de fabrico mudam drasticamente. A aplicação da internet industrial e da robótica trazem inovações capazes de aumentar a produtividade e estimular os negócios.

As indústrias estão mais competitivas entre si, por isso, têm de se superar dia após dia tanto a nível de produto como a nível de produção. A possibilidade de ter uma fábrica inteligente facilita tanto a monitorização como a produção de todo o ciclo de um determinado produto. A incorporação de itens inteligentes permitirá obter dados de produção, melhoramentos entre integração do *Manufacturing execution systems* (MES) com a fábrica propriamente dita, flexibilidade na produção, implementação de novas configurações ou reformular a produção com mais rapidez, redução de falhas e tempo de processamento, possibilidade de baixar os custos de produção e diminuição de desperdícios.

O estudo da junção da indústria 4.0 com colaboradores robóticos em cenário de logística levará ao aprimorar do conhecimento de uma realidade que não é indiferente nos dias de hoje, e, a possibilidade de contribuir para uma revolução industrial com consequentes vantagens para os consumidores, produtores e até para o meio ambiente.

## 1.2 - Objetivos

Esta dissertação insere-se no contexto da indústria 4.0 e tem como objetivo principal o estudo, desenvolvimento, implementação em simulação e análise de resultados de um exemplo de uma fábrica totalmente autónoma.

Mais especificamente, pretende-se em primeiro lugar desenvolver uma arquitetura do sistema de rede de robôs móveis para aplicações de serviços de logística em fábricas, descrevendo os vários requisitos e metodologias associadas que incluem os algoritmos de planeamento de trajetórias, de controlo de movimento dos robôs, de otimização de custos, etc.

Em segundo lugar, pretende-se desenvolver um simulador em Matlab que serve para demonstrar e avaliar os algoritmos utilizados. Um outro objetivo do projeto foi a elaboração de uma interface gráfica que facilita consideravelmente a operação do simulador.

No primeiro ponto, começa-se por desenvolver a arquitetura concetual para um caso geral e depois para um caso mais específico, a arquitetura funcional. O caso específico consiste em dois robôs móveis num ambiente de fábrica que são totalmente autónomos e têm como função realizar as tarefas pedidas que são enviadas através de um painel de controlo. A fabrica é composta por

dois sítios de descargas onde será entregue o produto final, dois sítios de carga onde os robôs vão carregar as matérias primas (neste projeto garrafas de plástico) e três subestações de transformação, onde as três enchem as garrafas com sabores diferentes. Estas têm custos diferentes e aleatórios dentro de uma gama aceitável para o fabrico de garrafas de água. De acordo com o número de pedidos feitos no painel de controlo do sistema, os robôs terão de transportar a matéria prima da zona de carga até à zona de descarga passando pelas subestações de transformação. As trajetórias para cada local pretendido, terão que ser otimizadas de acordo com um funcional de custo. Também é imposto que os robôs não possam de modo algum ter colisões, sendo capacitados para isso com um lidar.

No segundo ponto, pretende-se avaliar em simulação os custos de produção (custos de produção nas máquinas e de transporte associados), bem como os tempos de execução dos produtos e o comportamento da fábrica perante avarias nos vários itens da mesma (Carga, Subestação, Descarga), que são geradas de acordo com uma determinada função de fiabilidade probabilística.

## 1.3 - Estrutura do documento

A estrutura do documento da dissertação está dividida em 5 capítulos.

O capítulo 2 apresenta uma breve descrição da motivação no âmbito dos objetivos da indústria 4.0, e nas adaptações à automação industrial. Também descreve alguns algoritmos utilizados para desenvolver o projeto, tais como o planeamento de caminhos e controlador para seguir pontos pré-definidos.

O capítulo 3 contém uma descrição detalhada das várias fases de desenvolvimento do projeto do sistema robótico orientado a serviços (fábrica autónoma) de modo a cumprir os objetivos delineados.

Em particular, apresenta-se a metodologia com a descrição das ferramentas e

algoritmos utilizados, modelo da cinemática dos robôs, a descrição geral da arquitetura proposta, o algoritmo de otimização de custos, o projeto do controlador para desviar dos obstáculos e a descrição da interface e da Base de dados.

No capítulo 4 são apresentados os diversos testes realizados até ao fim do projeto.

Em suma, o capítulo 5 apresenta propostas para trabalho futuro e conclusões gerais do projeto desenvolvido.

# Introdução

# Capítulo 2

# Revisão Bibliográfica

## 2.1- Indústria 4.0

O mundo está em constante mudança. A sociedade que nos rodeia está cada vez mais exigente e competitiva. A indústria, uma parte fulcral da economia, não poderia ficar para trás. Ao longo do tempo, foram surgindo inúmeras revoluções industriais **[Lu2017]** (ver figura 2.1.1):

- **1ª revolução** (aparecimento de tecnologias mecânicas como ferrovias e a vapor);
- **2ª revolução** (inovação da eletricidade, linhas de montagem e difusão da produção);
- **3ª revolução** (advém da informatização, avanço eletrónico e transformação através de robôs).

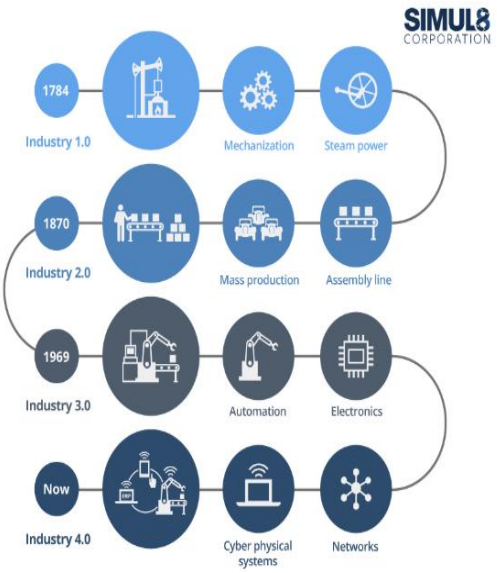



Figura 2.1.1 - Revoluções industriais.
Fonte: **[industry4.0_2019]**

Para todos os que pensavam que a indústria estava estagnada, que o futuro tecnológico era longínquo e ainda que a indústria não conseguia suprimir as exigências de uma sociedade cada vez mais dependente da economia, estavam ingenuamente enganados. Antes uma ideia, hoje, uma realidade até que surgiu a indústria 4.0, uma nova revolução industrial. Atualmente, a Alemanha é o país do mundo onde existe mais implementação e desenvolvimento da indústria 4.0, consequentemente, a maior parte dos artigos científicos, desenvolvidos nesta área, advêm deste mesmo país.

A indústria 4.0 tem como objetivos principais [Marr2019], [Rüßmann2015]: a capacidade de operação em tempo real; localização e monitorização remota em tempo real de todos os processos através de vários sensores; descentralização (a possibilidade da sincronização e comunicação entre sensores inviabiliza a utilização de único sistema de controlo); orientação a serviços (utilização de sensores que enviam constantemente dados para uma nuvem, que serão posteriormente analisados estatisticamente, um exemplo é o serviço de refeições a pacientes de um hospital com base nos dados recebidos sobre as necessidades do paciente em questão) e modularidade (utilização de módulos que são controlados localmente sem hierarquia, o que permite consequentemente uma maior flexibilidade e adaptação às exigências do mercado).

Os principais pilares desta nova revolução assentam:

- ➔ Na internet das coisas (consiste na interligação em rede de objetos físicos por meio de dispositivos eletrónicos embarcados que permitem a recolha e troca de dados);

- ➔ Na análise de um grande número de dados (estruturas de dados muito extensas e complexas que utilizam novas abordagens para a recolha, análise e gerenciamento de informação);

- ➔ Na segurança dos dados recolhidos e da sua transmissão.

Em termos práticos, a possibilidade de ter as máquinas conectadas possibilita obter um enorme volume de dados que podem informar a manutenção, desempenho e outros problemas, bem como analisar esses dados para identificar padrões que seriam impossíveis para um ser humano num período razoável. Com a capacidade que proporciona a indústria 4.0 é possível otimizar a logística e as cadeias de suprimentos. Por exemplo, se existir algum atraso na chegada das matérias-primas de um certo produto, um sistema conectado poderá ajustar-se proactivamente a essa realidade e

modificar as prioridades de fabricação. Outro exemplo é a possibilidade de um robô mudar de uma subestação se esta estiver com defeito.

A realidade de que só as grandes empresas é que poderiam ter robôs também está a mudar, a robótica está cada vez mais acessível e disponível para empresas de todos os tamanhos. Deste modo, as empresas poderão ser mais competitivas. A utilização de robôs na indústria oferece um suporte rápido, seguro, reduz os custos e permite uma melhor utilização do espaço aos fabricantes.

Esta nova realidade está em constante e rápida evolução, rapidamente, poderemos ter uma visão completa das capacidades que poderá proporcionar a indústria 4.0. As empresas estão cada vez mais a adaptar-se a esta tecnologia e a compreenderem o seu potencial.

## 2.2- A automação na indústria

A indústria 4.0 proporcionou que a automação entrasse numa nova era. A automação é um dos setores da indústria mais automatizados, no entanto, à medida que a era da indústria 4.0 se inicia, uma nova forma de pensar se forma. O setor fabril procura dia após dia eficiência para as suas linhas de produção. Nos últimos anos, a convergência de várias novas tecnologias permitiu novos tipos de sistemas automatizados. Neste setor está a haver um grande desenvolvimento em robôs colaborativos, ou seja, sistemas robóticos projetados para trabalhar com segurança ao lado dos humanos ou sem eles, como se pode observar a figura 2.2.1, que representa uma fábrica da empresa Amazon.

Estes sistemas robóticos são capacitados cada vez mais pelos avanços da visão das máquinas, poder de computação, sensoriamento e pelo uso pesado de novos e mais poderosos algoritmos/softwares. Hoje em dia, nas empresas mais sofisticadas, tudo que há para automatizar está automatizado. O que foi deixado foram algumas situações em que os operários são necessários nas linhas de produção, por razões de segurança. No entanto, os robôs estão a ficar cada vez mais "inteligentes", e eles podem trabalhar de maneira semelhante, mas mais eficientes em relação ao modo como os humanos trabalham.

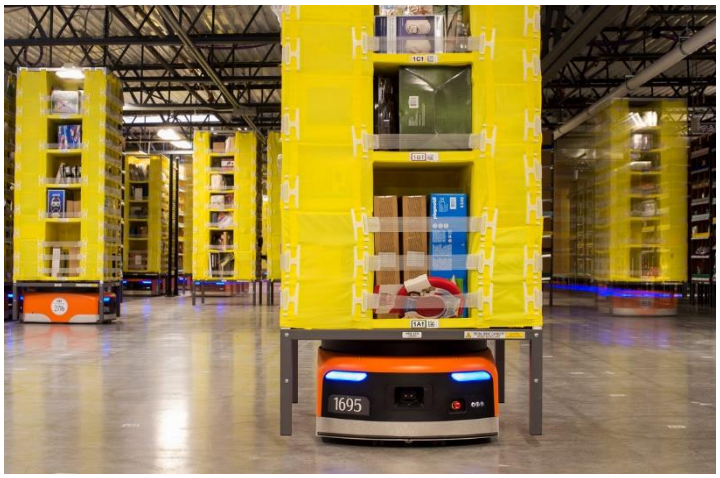



Figura 2.2.1-Uma fábrica da Amazon. Fonte: [Shead2019]

A indústria 4.0 adaptada à automação industrial apresenta vantagens e essas são descritas a seguir [IR2019]:

- **Custo-efetividade** consiste na redução dos custos de produção, eliminando os custos internos para compensar os salários.

- **A garantia de qualidade** é esperada com o uso de máquinas em produção. Os robôs industriais serão capazes de garantir consistência com a produção em massa de produtos manufaturados.

- **A eficiência de produção otimizada** significa que um operador poderá definir padrões de quantidade e qualidade que serão atendidos pelos robôs. As quotas de produção não serão prejudicadas pela baixa concentração, tempo de pausa e acidentes com funcionários, entre outras coisas. Poderá ser estipulada uma velocidade de produção e as previsões de produção de um certo número de produto serão mais fidedignas.

- **Limitação do trabalho humano em ambientes perigosos** significa que existem empregos industriais que costumam colocar os trabalhadores em maior risco físico em comparação com os outros setores. A diminuição do risco apresentado aos funcionários no trabalho é atraente para os executivos, deste modo preserva a reputação da empresa e minimiza possíveis responsabilidades legais.

## 2.3- Planeamento de caminhos para robôs móveis

Para o projeto do sistema de rede de robôs móveis para aplicações de serviços de logística em fábricas é necessário a utilização de um algoritmo de planeamento de caminhos que tenha em consideração os vários requisitos associados. Deste modo, é necessário perceber quais os tipos de algoritmos que existem e de acordo com as suas caraterísticas então selecionar o que melhor se adapta ao projeto.

Nesta parte, são explicados alguns algoritmos de planeamento de trajetórias e de forma mais pormenorizada aquele que é utilizado.

**Algoritmos Bug**

Os algoritmos Bug surgiram para resolver problemas em que o mapa é desconhecido, isto é, o robô tem de chegar à meta, mas não tem informação nenhuma sobre o mapa que está inserido. Este algoritmo utiliza os sensores para descobrir o que existe no caminho e evitar colisões. Neste não se constroem mapas. O comportamento deste é muito primitivo e simples, uma vez encontrado um obstáculo ao longo do caminho, na linha reta do robô em direção ao objetivo, este seguirá um caminho ao redor das fronteiras do obstáculo, a uma certa distância segura, a fim de encontrar o ponto com menor distância para o objetivo predefinido.

Este algoritmo não é utilizado para o nosso cenário de estudo, porque o mapa é conhecido e também o caminho percorrido pelos robôs seria maior (quando comparado com outros algoritmos), porque este teria de contornar obstáculos em vez se dirigir logo para a posição final.

**Algoritmo Roadmap**

Neste algoritmo existem nós e ligações que podem ter um significado físico, ou seja pode ser a trajetória que o robô poderá seguir. Os nós correspondem a uma localização específica e as ligações são os caminhos a percorrer entre esses nós.

O planeamento de trajetórias resume-se a um problema de pesquisa em grafo. A pesquisa pode ser executada pelas técnicas usais de pesquisa em grafo. A dificuldade desta abordagem é a construção do roadmap. Em seguida será explicado algumas das técnicas usadas para a construção do mesmo.

**Diagrama Voronoi (DV)**

Esta técnica [Aurenhammer1991] consiste em criar um conjunto de pontos equidistantes de dois ou mais obstáculos. O mapa é divido em várias regiões que só existe um obstáculo. Qualquer ponto na região está mais perto do obstáculo dessa região do que de qualquer outro obstáculo. Este método cria caminhos que estão muito afastados dos obstáculos.

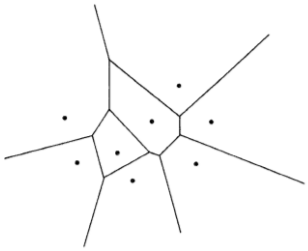


Figura 2.3.1- Diagrama de Voronoi. Fonte: [Aurenhammer1991]

**Gráfico de visibilidade (VG)**

Esta técnica [Samir2014] é usada na maior parte das vezes em espaços de duas dimensões (2D) em que os nós são vértices dos obstáculos e as ligações entre nós só existem se entre os vértices não existem obstáculos.

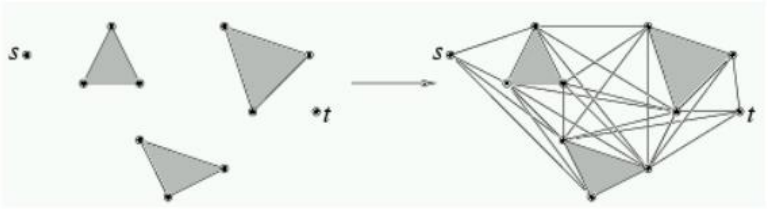



Figura 2.3.2- Visibility graph. Fonte: [VisibilityGraphs2019]

**Decomposição em células**

Os algoritmos de decomposição em células [Abbadi2015] são soluções antigas aplicáveis para o planeamento de caminhos. A ideia deste método é encontrar regiões livres de obstáculos e construir um gráfico de adjacência para eles. A ideia de dividir o espaço em secções é apresentada em muitas pesquisas. Em geral, existem duas categorias de algoritmos de decomposição de células. Existe o método exato de decomposição celular e de aproximação.

A primeira categoria usa algoritmos geométricos para determinar explicitamente os obstáculos e constrói as células. A união de todas as células geradas é exatamente igual ao espaço livre. No entanto, muitas vezes não é uma tarefa fácil encontrar células livres exatas, especialmente em grandes dimensões, o que leva para a segunda categoria, que utiliza técnicas de aproximação para dividir os espaços em quad-tree, octree division, voxel grid, etc. Numa aplicação de planeamento de movimento, este algoritmo é utilizado dividindo-se o espaço de trabalho em regiões menores chamadas células. Em seguida, é criado um gráfico de conectividade de acordo com as relações de adjacência entre as células livres. Os nós do gráfico representam, as células, enquanto as arestas do grafo representa as relações de adjacência entre as células. Desta conectividade do gráfico, pode ser encontrado um caminho contínuo seguindo as células livres adjacentes.

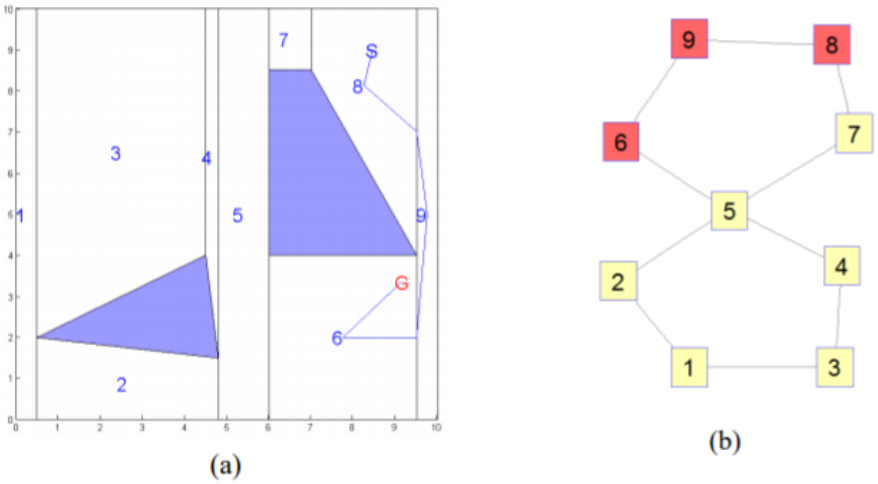



Figura 2.3.3- (a) Decomposição em células trapezoides (b) gráfico adjacente que corresponde aos caminhos entre células. Fonte: [Abbadi2015]

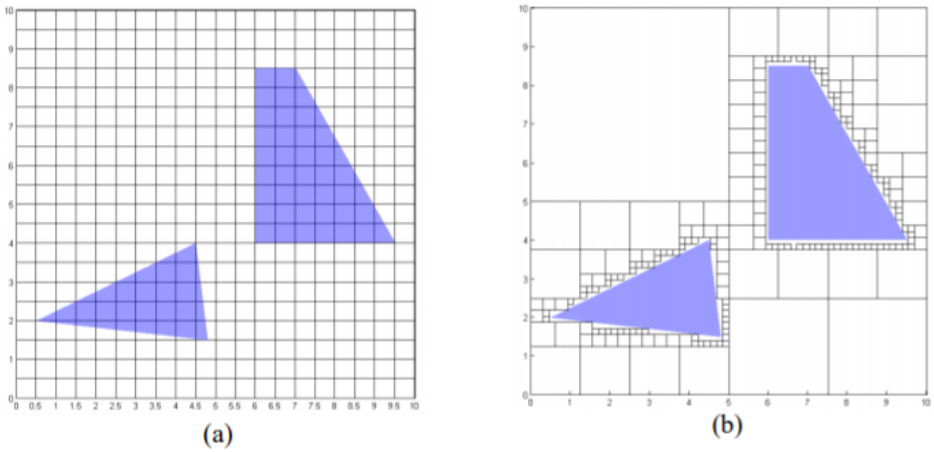



Figura 2.3.4- (a) Decomposição de células aproximadas com o método de aproximação de voxel (b) método de aproximação quad-tree. Fonte: [Abbadi2015]

**Campos potencial**

O algoritmo de campo potencial [Samir2014] consiste em prever obstáculos como uma sub-tarefa do planeamento do caminho do robô, abordagem deliberativa. O algoritmo de campo potencial assume que o robô é dirigido por forças virtuais que o atraiam para o objetivo, ou rejeitam os obstáculos. O real caminho é determinado pelo resultado dessas forças virtuais.

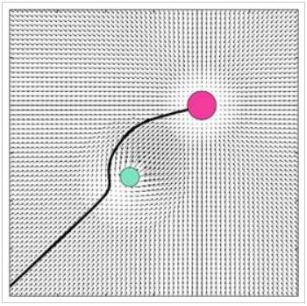


Figura 2.3.5- Exemplo de um campo potencial [PotentialFields2013]

**Probabilistic roadmap (PRM)**

Neste projeto foi utilizado este algoritmo, porque oferece tempo de resposta rápido, cria inúmeras trajetórias proporcionando trajetos muitos eficientes e tem a capacidade de criar trajetórias em ambientes muito complexos, ou seja, com muitos obstáculos. Em seguida será explicado minuciosamente em que consiste. Este planeamento de rotas (PRM) [PRM2019] é importantíssimo para o projeto, porque a sua utilização determina um trajeto no mapa entre o ponto inicial e final pré-definidos.

Este algoritmo cria uma rede de caminhos possíveis tendo como entrada um mapa do ambiente. A função Matlab robotics.PRM da "Robotic Systems Toolbox" implementa o algoritmo de planeamento probabilístico PRM. O algoritmo cria uma rede de caminhos possíveis num dado mapa baseado em espaços ocupados e livres. Em particular, cria nós aleatoriamente espalhados pelo mapa que depois criam entre si conexões com base em parâmetros do algoritmo PRM. Os parâmetros do algoritmo PRM são o número de nós que serão distribuídos aleatoriamente pelo espaço livre no mapa e a distância de conexão máxima entre nós.

Para a utilização deste algoritmo é necessário a criação de um mapa monocromático com extensão *bmp* para o algoritmo *probabilistic roadmap* (PRM) consiga identificar com exatidão o que é o espaço livre e espaço ocupado. No momento em que o mapa está a ser desenhado é importante ter cuidado com o desenho das linhas que delimitam os espaços, porque se estas não estiverem completamente preenchidas o algoritmo criará trajetórias entre os espaços livres e os ocupados.

Para ajustar o número de nós é necessário a propriedade NumNodes que especifica o número de pontos, ou nós, colocados no mapa. O aumento do número de nós tem a vantagem de aumentar a possibilidade de escolha do caminho, como mostra as figuras 2.3.6 e 2.3.7, fornecendo caminhos mais viáveis, mas implicando um aumento de memória e número de iterações e consequentemente, aumenta o tempo de execução do programa. É importante salientar que para uma boa execução do algoritmo temos de ter em conta a dimensão do mapa, ou seja quanto maior for o mapa maior terá de ser o número de nós para conseguir cobrir toda a área livre. Devido ao posicionamento aleatório dos nós, é preciso tentar ter o número suficientemente grande para cobrir as zonas todas.

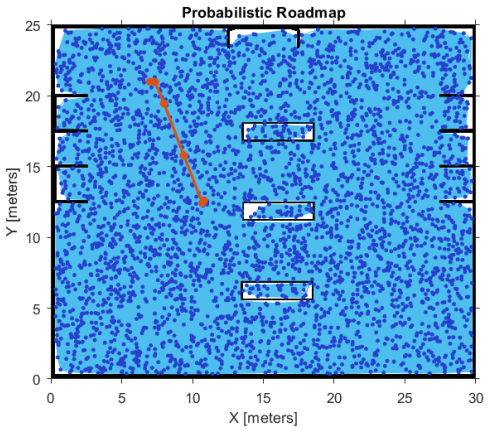



Figura 2.3.6 –Algoritmo PRM (4000 nós e 5m)

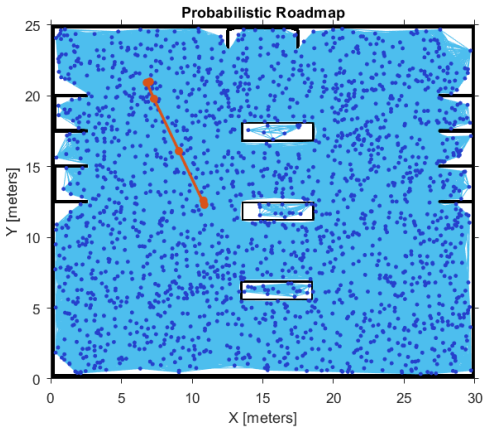



Figura 2.3.7 – Algoritmo PRM (2000 nós e 5m)

O outro parâmetro deste algoritmo é a distância máxima de conexão entre nós. Para usar este parâmetro é necessário usar a propriedade ConnectionDistance que será a máxima distância das conexões. Cada nó é conectado a todos os nós dentro dessa distância de conexão que não possuem obstáculos entre eles. As figuras 2.3.8 e 2.3.9 ilustram ao diminuir esta propriedade, irá diminuir a distância de conexão entre nós e consequentemente diminuir o número de conexões, reduzir o tempo de computação e simplificar o mapa. No entanto, uma distância reduzida limita o número de caminhos disponíveis para encontrar um caminho livre de obstáculos. Quanto maior for complexidade do mapa, ou seja, ter mais obstáculos, menor deverá ser a distância de conexão e maior número de nós para aumentar a probabilidade de encontrar uma solução, consequentemente o algoritmo conseguirá ter alternativas de trajetórias, podendo, deste modo, contornar facilmente os obstáculos. Para mapas mais simples, deverá ter uma distância de conexão mais alta com pequeno número de nós para aumentar a eficiência do algoritmo.

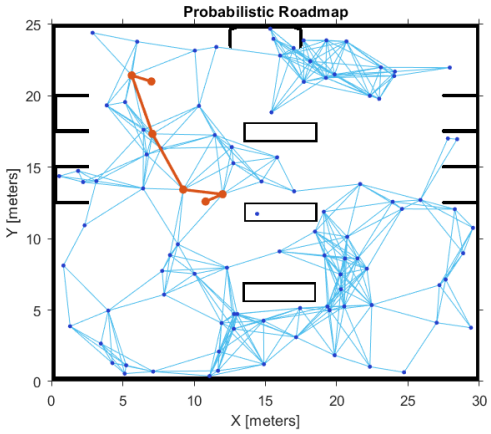



Figura 2.3.8 –Algoritmo PRM (100 nós e 5m)

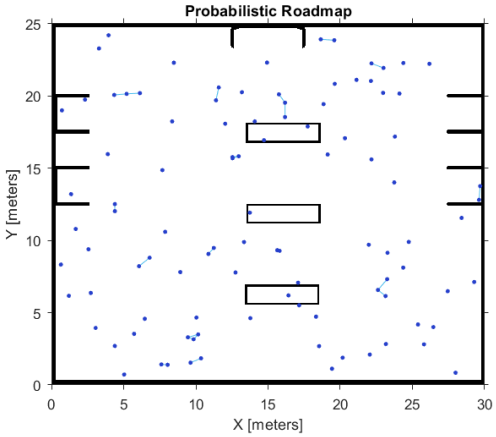



Figura 2.3.9 –Algoritmo PRM (100 nós e 1m)

## 2.4- Controlador para seguir pontos predefinidos

Para o bom funcionamento do projeto da dissertação foi necessário a utilização de um algoritmo que conseguisse seguir pontos de trajetórias pré-definidas criadas pelo algoritmo de planeamento de trajetórias, ou seja que conseguisse suprimir a exigência de seguir pontos aleatórios conectados com outros pontos. Dado os requisitos, o controlador que proporciona maior fiabilidade ao projeto é o controlador PurePursuit. Apesar de ser fiável também facilita por estar programado em matlab.

O PurePursuit é um algoritmo que segue rotas [PPC2019]. Ele calcula a velocidade angular que move o robô da sua posição atual para alcançar algum ponto mais perto do caminho. A velocidade linear é constante e é definida em qualquer instante. O algoritmo move o ponto de look-ahead (consiste no ponto que o robô segue na trajetória) no caminho baseado na posição atual do robô até ao último ponto do caminho. Isto pode ser entendido como um robô a perseguir constantemente um ponto na frente dele. A propriedade LookAheadDistance decide até que ponto o ponto de lookahead é colocado (salientar que este ponto está na trajetória pré-definida pelo algoritmo planeamento de trajetória).

Neste trabalho, utilizou-se a função "robotics.PurePursuit" que está contida na biblioteca Robotics System Toolbox e que implementa este tipo de controlador. Para criar um controlador PurePursuit é necessário especificar uma lista de pontos (waypoints) com coordenadas [x,y] do planeamento de trajetórias, as velocidades angulares e lineares máximas. Estas especificações da velocidade são determinadas com base nas especificações do robô. Depois definidas as entradas (posições dos *waypoints* e a posição do robô com a sua orientação angular [x, y theta]) é possível calcular os comandos de velocidade linear e angular para o robô. Uma propriedade também muito importante é a LookAheadDistance. Esta propriedade consiste na distância da posição do robô ao caminho pré-definido que o robô deverá procurar a partir da posição atual para calcular o comando de velocidade angular. A figura 2.4.1 mostra que o caminho pré estipulado a cinzento tracejado não corresponde à distância real percorrida pelo robô. Como se pode ver, este não segue o caminho, mas irá sempre procura-lo. Deste modo, diminui a distância a percorrer.

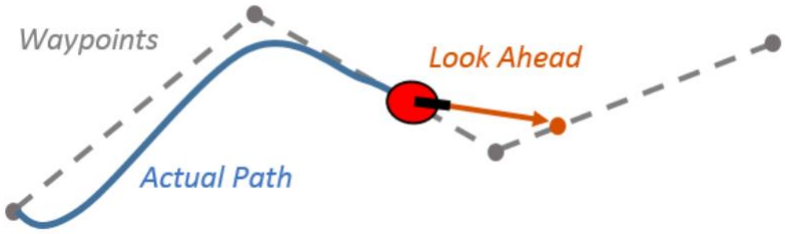



Figura 2.4.1- Caso real do controlador PurePursuit. Fonte: [PPC2019]

A LookAheadDistance é uma propriedade deveras importante, porque define o caminho que o robô deve procurar a partir da localização atual. Este parâmetro altera a maneira como o robô rastreia o caminho. Este algoritmo tem dois objetivos que são recuperar o caminho ou manter o trajeto que está a efetuar. Para recuperar rapidamente o caminho entre waypoints, um pequeno LookAheadDistance fará com que o robô se mova rapidamente em direção ao caminho. No entanto, como pode ser visto na figura 2.4.2, o robô ultrapassa e oscila ao longo caminho desejado. Para reduzir as oscilações ao longo deste é aconselhável aumentar LookAheadDistance, como pode ser visto na figura 2.4.3. O aumento tem a desvantagem de haver curvaturas maiores perto dos cantos.

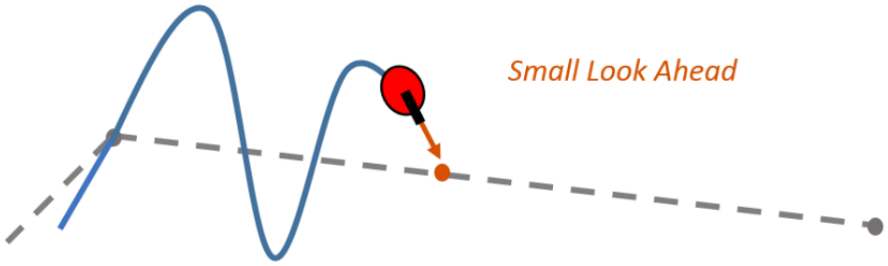



Figura 2.4.2- Caso real do controlador PurePursuit com small Look Ahead. Fonte: [PPC2019]

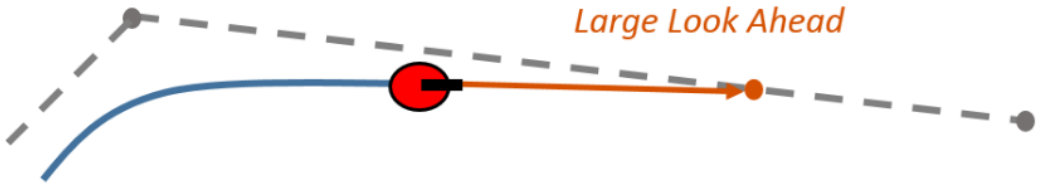



Figura 2.4.3- Caso real do controlador PurePursuit com Large Look Ahead. Fonte: [PPC2019]

A propriedade LookAheadDistance deve ser ajustada consoante o mapa e o sistema de robôs. As velocidades lineares e angulares também afetarão a resposta do algoritmo. Devem ser reguladas para melhorar a eficiência do algoritmo em questão.

É importante salientar que apesar deste controlador ser fiável para a exigência do projeto, este tem algumas desvantagens. Algumas limitações deste algoritmo são:

- ➔ O controlador pode não seguir exatamente os caminhos diretos entre os waypoints. Os parâmetros devem ser ajustados para otimizar o desempenho e convergir para o caminho ao longo do tempo.

- ➔ Este PurePursuit algoritmo não estabiliza o robô em um ponto. No projeto deverá haver um limite de distância para o ponto final. Isto deve ser aplicado para parar o robô próximo ao objetivo desejado.

# Capítulo 3

# Arquitetura do sistema de rede de robôs móveis

Este capítulo é dedicado ao projeto e arquitetura do sistema de rede de robôs móveis para aplicações de serviços de logística em fábricas utilizando múltiplos robots móveis, ou seja é desenvolvido um Sistema Robótico Orientado a Serviços (SORS). O capítulo encontra-se dividido em 11 secções: metodologia, com a descrição das ferramentas e algoritmos utilizados (secção 3.1), arquitetura concetual do trabalho onde descreve a interação entre conceitos (secção 3.2), modelo da cinemática dos robôs, isto é, descrição física do movimento dos robôs (secção 3.3), arquitetura funcional onde apresenta uma visão geral de todo o sistema implementado (secção 3.4), diagrama detalhado dos sistemas principais onde descreve de uma forma mais pormenorizada e esquematicamente os vários sistemas relacionados com os robôs e a fábrica (secção 3.5), apresentação do algoritmo de otimização de custo do robô (secção 3.6), projeto do controlador para desviar dos obstáculos presentes no mapa (secção 3.7), controlador para desviar de qualquer robô próximo (secção 3.8), avarias de equipamentos e robôs (secção 3.9), descrição da interface com a explicação das suas funções e finalidades (secção 3.10) e por fim descrição da Base de dados (secção 3.11).

Todos os sistemas descritos desempenham um papel fundamental no projeto para a simulação de uma fábrica totalmente autónoma.

## 3.1- Metodologias e ferramentas utilizadas

A metodologia utilizada consistiu em seguir o método científico estruturado de acordo com as seguintes atividades:

- ➔ Estudo do estado da arte (reportado no capítulo anterior);
- ➔ Especificação dos requisitos mínimos e funcionalidades gerais para uma classe particular de fábricas autónomas;
- ➔ Projeto da arquitetura do sistema geral e dos vários componentes que a integram;
- ➔ Implementação e validação das soluções obtidas através de simulação.

Notar que principalmente os pontos 3 e 4 seguiram uma lógica iterativa de realimentação sistemática de forma a melhor os resultados obtidos.

Neste projeto é utilizado o algoritmo Visualizer2D [MRS2018] que está na caixa de ferramentas ("Mobile Robotics Simulation Toolbox") obtida como addon do matlab. Este algoritmo, apesar de muito incompleto, foi o ponto de partida do projeto em questão, porque possibilitou a utilização de uma parte gráfica que neste momento ajuda na visualização do movimento dos robôs e proporciona ter um simulador que mostre com uma taxa de amostragem definida previamente com a função "waitfor". A taxa de amostragem pré-definida é de 0.1s. Para definir trajetos dos vários robôs utilizou-se um algoritmo de planeamento de trajetórias [PRM2019] explicado no capítulo 2 secção 2.3. Neste projeto também foi utilizado um algoritmo ("PUREPURSUIT") [PPC2019] que faz com que os robôs sigam trajetórias pré-definidas com o algoritmo de planeamento de trajetórias. Será explicado mais pormenorizadamente no capítulo 3 secção 3.7.

Todos estes algoritmos são programados em matlab, por isso para a elaboração do simulador do projeto foi imprescindível a ferramenta de programação matlab. Esta proporcionou não só simular uma fábrica, como também fazer ligações a um servidor MYSQL assim como usar uma interface que simulasse avarias. Neste trabalho, foi o programa Microsoft Visual Studio que proporcionou a elaboração de uma interface onde o operador poderá fazer pedidos a servidor MySQL, visualizar o estado do pedido e visualizar a estatística da fábrica. O programa Paint possibilitou o desenho de um mapa monocromático, para ser colocado no projeto. Através dos programas PowerPoint e draw.io foram feitos diagramas e imagens para demonstrar de uma forma

mais gráfica e intuitiva partes do projeto. Em termos de software foram utilizados unicamente os programas anteriormente referidos. Em relação ao hardware, um computador. Deverá ter no mínimo qualquer intel ou AMD processador x84-64 com quatro núcleos lógicos, SSD com 22 GB de disco, 8 GB de RAM e deverá ter uma placa gráfica de apoio OpenGL 3.3 com 1GB. Todos estes requisitos são recomendados pela versão matlab 2018a [SRR2018].

## 3.2- Arquitetura concetual

A figura 3.2.1 ilustra o diagrama concetual que foi desenvolvido baseado no trabalho em [Oliveira2015] para um sistema robótico orientado a serviços (SORS), estando representado de uma informal de alto nível. Esta arquitetura descreve interações entre conceitos. Este tipo de arquitetura é detalhada e usa linguagem não técnica que pode ser facilmente entendida por todas as partes interessadas.

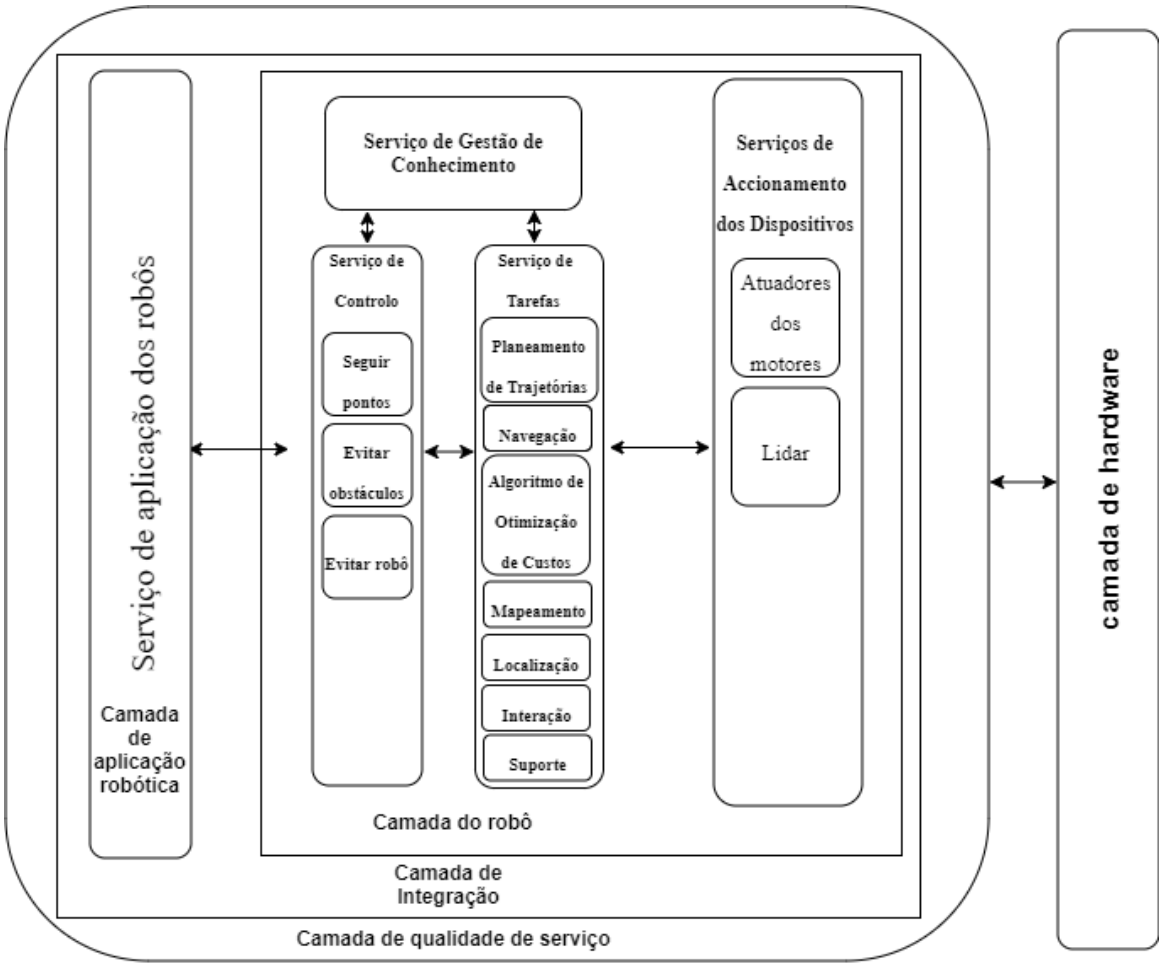



Figura 3.2.1- Diagrama concetual do SORS

Este diagrama é composto por 4 camadas para o bom funcionamento do SORS [Oliveira2015]:

**Camada do robô:** contém elementos abrangidos por sistemas robóticos. É composto por quatro grupos de serviços:
- serviços de acionamento dos dispositivos, que coordenam e monitoram sensores, atuadores e recursos do robô;
- serviços de gestão de conhecimento, que são responsáveis pela gestão e pelo processamento de informações úteis para o sistema robótico, como as características do meio e os objetos dentro dele;
- serviços de tarefas, que englobam as principais tarefas atribuídas aos robôs, incluído mapeamento, localização e navegação;
- serviço de controlo, que coordena tarefas executadas por um agente robótico para executar mais atividades complexas, por exemplo transporte de objetos e seguir pontos pré-definidos.

**Camada da aplicação do robô:** coordena um ou mais robôs para executar uma aplicação robótica, isto é, a missão robótica. Esta camada é responsável pelas estratégias que podem ser usadas para gerir as atividades robóticas necessárias para atingir os objetivos da aplicação e é responsável por identificar os serviços da camada do robô mais adequados para executar cada tarefa, tendo em conta o estado atual e caraterísticas.

**Camada de integração:** intermeia, roteia e transporta solicitações entre consumidores de serviços e prestadores de serviços. Esta camada é essencial, visto que permite a integração e composição de serviços de todos os níveis dentro do sistema robótico. Esta fornece meios para a descoberta de serviços e suporta a comunicação indireta entre os elementos do SORS.

**Camada de qualidade de serviço:** supervisiona os serviços contidos nas outras três camadas anteriores explicadas para verificar a sua conformidade com os requisitos de qualidade SORS, ou seja, observa e sinaliza quando um requisito de qualidade não é cumprido por um serviço da SORS. Essa garante que os sistemas robóticos abranjam níveis adequados de confiabilidade, disponibilidade e segurança. É fundamental, pois os serviços para o SORS podem ser desenvolvidos por várias instituições ou empresas e apresentam características diferentes. Os sistemas robóticos são frequentemente criados para contextos críticos de segurança e sua qualidade geral depende da qualidade dos serviços de que são compostos.

Depois da explicação dada sobre as camadas será explicada a taxonomia dos serviços que compõe o diagrama concetual. Existem vários serviços, mas nem todos tem a mesma complexidade, como podemos ver na figura 3.2.2.

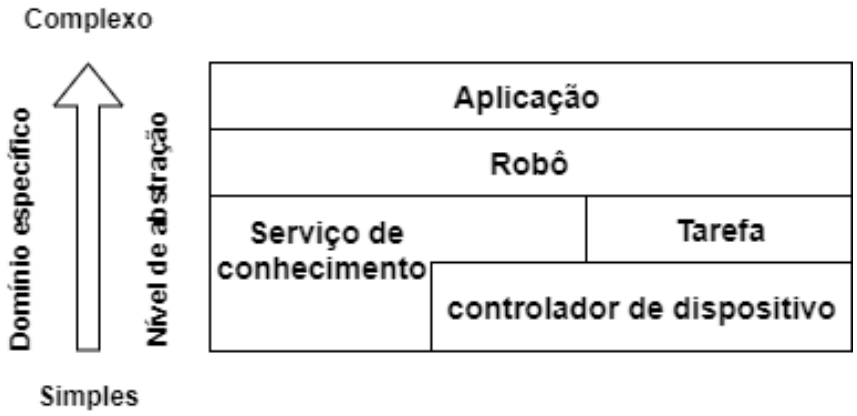



Figura 3.2.2- Dependência de serviços robóticos. Fonte: [Oliveira2015]

Os serviços de camadas inferiores [Oliveira2015] são entidades mais precisas e fornecem funcionalidades que podem ser usadas em diferentes domínios de aplicação. As superiores coordenam o serviço em camadas de nível inferior para executar a atividade, portanto, mais dependentes do domínio da aplicação.

- **Serviços de controlo dos dispositivos**

Este grupo inclui serviços que encapsulam drivers de hardware, atuando como uma camada de abstração de hardware. Estes tipos de serviços são usados como soluções para fornecer uma melhor integração entre recursos. Existem dois de serviços neste grupo: atuadores e sensores. Estes dispositivos podem ser de vários tipos como se pode ver na figura 3.2.2

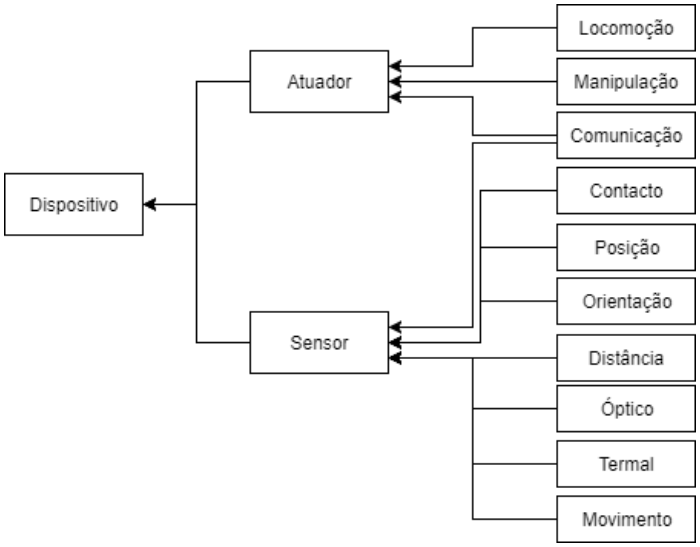



Figura 3.2.3- Grupo de serviço de acionamento de dispositivos. Fonte: [Oliveira2015]

**Locomoção:** Fornecer mobilidade ao robô. Inclui software para controlar as rodas (por exemplo);

**Manipulação:** Permite que o robô manipule ou mantenha objetos do mapa que está inserido. Utilização de drivers para controlar armas e garras (por exemplo) são considerados serviços de dispositivos de manipulação;

**Comunicação:** Este serviço desempenha um papel de atuador e sensor. Esta é normalmente bidirecional e o sistema robótico pode usar este tipo de serviço para interagir com o mapa onde está inserido. Drivers para rede e radiofrequência são alguns serviços de comunicação;

**Contacto:** Controla sensores que detetam se o robô está em contato com objetos no mapa. Drivers para barreiras e para-choque são alguns de serviços de contato;

**Posição:** Obtém informação da localização do robô no mapa. Driver do dispositivo GPS é exemplo bem conhecido deste serviço;

**Orientação:** Fornece informação sobre a orientação do robô no mapa. O excitador da bússola é um exemplo;

**Distância:** Mede a distância entre o robô e os objetos ou paredes no mapa. São exemplos os Drivers para lidares e sonares;

**Ótico:** Converte imagens do mapa em informações digitais que podem ser processadas. Como por exemplo Drivers para câmeras;

**Termal:** Usado para medir a temperatura de objetos dentro do mapa que está inserido. Drivers para câmeras térmicas é um exemplo.

**Movimento:** mede a distância percorrida pelo robô. O driver codificar é exemplo deste serviço.

- **Serviço de conhecimento**

Este grupo compreende serviços que coletam, interpretam, armazenam e compartilham informações necessárias para a realização de tarefas e controlo do robô como um todo. Isto permite que os sistemas robóticos captem as características do mapa e os objetos nele. Os serviços de conhecimento usam não apenas dados de sensores, mas também informações semânticas de uma ampla gama de fontes, como ontologia (assimilação e codificação do conhecimento) e máquinas.

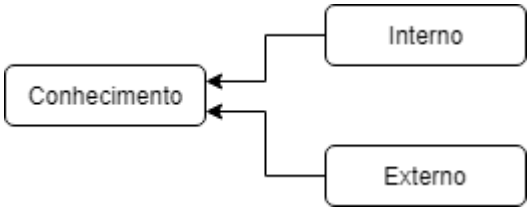



Figura 3.2.4- Grupo do serviço de Conhecimento Fonte: [Oliveira2015]

**Interno:** reúne, armazena e compartilha informações obtidas dentro dos limites do mapa onde está inserido. Essas informações podem ser obtidas através de sensores e bases de dados e hospedadas num servidor back-end. A informação adquirida por sensores é produzida geralmente por serviços de tarefas e, em seguida, armazenada num serviço de conhecimento interno para ser compartilhado entre robôs. Exemplo de um conhecimento tipo interno é um serviço hospedado num servidor back-end que fornece informações sobre como interagir com um determinado objeto no mapa.

**Externo:** Obtém e interpreta informações de fontes externas ao mapa. Através deste tipo de serviço, um robô habilitado para aceder à internet pode pesquisar, acessar e obter informações de

conteúdo legível na internet. Fontes externas de conhecimento permitem ao sistema robótico captar procedimentos, conceitos semânticos e relacionamentos que não foram considerados durante o tempo de projeto. Um exemplo é um robô autónomo aprender a fazer uma receita através de informação obtida na internet.

- **Serviço de tarefas**

Este grupo engloba serviço que executam as tarefas normalmente fornecidas por sistemas robóticos. Os serviços de tarefas permitem que os robôs executem atividades simples. Um exemplo é um robô se mover de uma posição para outra, isso pode ser implementado de acordo com diferentes comportamentos, isto é, move-se para uma determinada posição, seguindo uma parede ou utilizando trajetórias pré-definidas. O grupo pode ser dividido em vários tipos como mostra a figura 3.2.5.

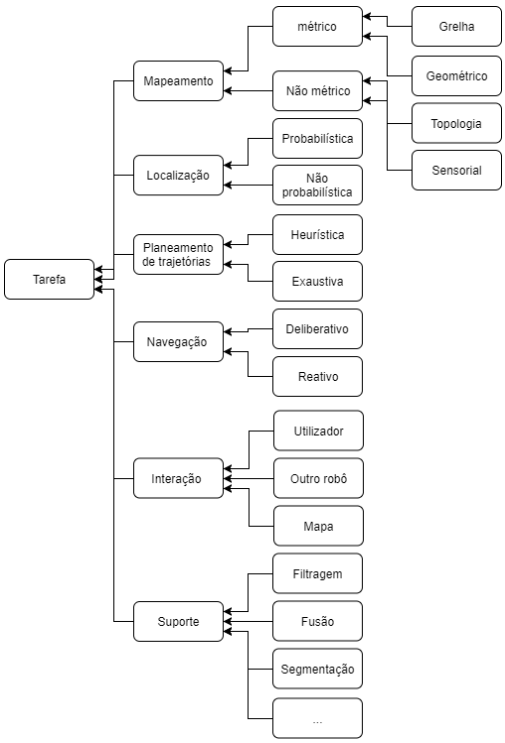



Figura 3.2.5 – Diagrama dos serviços de tarefas. Fonte: [Oliveira2015]

**Mapeamento:** Utiliza algoritmos que estimam e criam representações do mapa do ambiente do robô. Há dois tipos de serviços de mapeamento como mostra a figura 3.2.5.

- ➔ Mapas Métricos, são representações 2D/3D que usam coordenadas geométricas ou em grelha para descrever a localização de objetos dentro de um mapa.
- ➔ Mapas não métricos, são representação lógicas que pode ser topológico como gráfico de adjacência (representa um mapa finito) ou sensorial como sequências de imagens

**Localização:** estima a posição do robô. Como a figura 3.2.5 existem dois tipos de serviços de localização:

- ➔ Probabilístico, tem como base teórica as técnicas estatísticas, a título de exemplo existe o filtro de Kalman que é um algoritmo conhecido que pode ser considerado como serviço de localização;
- ➔ Determinístico que se baseia unicamente em dados que recebe, exemplo o GPS;

**Planeamento de trajetórias:** define os trajetos bons ou ótimos entre duas ou mais posições no mapa. Este tipo de serviços geralmente é suportado por um sistema de mapeamento com base em duas estratégias que são as pesquisas heurísticas e exaustivas.

**Navegação:** este serviço é utilizado pelo robô para navegar pelo mapa e a classificação está associado à arquitetura de controlo que implementa. Este serviço pode ser classificado em dois tipos:

- ➔ Deliberativo (executa o caminho de acordo com um plano predefinido)
- ➔ Reativo (controla o robô com base nos seus dados sensoriais)

**Interação:** Permite que os robôs trabalhem em conjunto com o mapa e com outros robôs ou até mesmo interagir com um utilizador. Um serviço deste tipo suporta a troca de dados e procedimentos de invocação entre os sistemas robóticos e os outros sistemas. Por exemplo, este tipo de serviço pode solicitar a um sistema ligar o equipamento de refrigeração.

**Suporte:** Este tipo de serviço fornece funcionalidades de uso geral que suportam o desenvolvimento de sistemas. Estas funcionalidades podem ser de vários tipos: filtragem de dados, fusão de dados, cálculos matemáticos, segmentação de imagens e outros. Como serviços de suporte não estão disponíveis unicamente para robótica. Existem inúmeros serviços que pertencem a este grupo que são difíceis de determinar.

- **Serviço do robô**

Este serviço coordena outros serviços situados em camadas inferiores ou menos abstratas como é possível ver na figura 3.2.2. Permite a utilização de um robô e facilita a coordenação de sistemas multi-robóticos. Conforme ilustrado na figura 3.3.6, há dois tipos de robôs: não móveis e móveis.

**Não-móvel:** permite funcionalidades a robô sem capacidade de mobilidade. Um exemplo é um sistema robótico que controla um robô industrial projetado, ou seja, um manipulador de objetos.

**Móvel:** permite a locomoção e manipulação de objetos. Devido aos diferentes tipos de robôs móveis, bem como as representações distintas de posição e orientação.

- **Serviço da aplicação**

Este grupo inclui serviços que gerem robôs para executar atividades mais complexas. Esta camada do sistema adquire conhecimento através de cada serviço do robô, processa e, em seguida, solicita um conjunto de tarefas que satisfazem uma determinada atividade. Isto permite que quem faz a aplicação se concentre unicamente nos detalhes da implementação de um robô. Existem três tipos de serviços da aplicação, que são:

➔ **Aplicação de um robô**

Descreve, coordena e monitoriza atividades robóticas de alto nível. Por exemplo, um robô aspirador.

➔ **Aplicação de vários robôs**

Descreve, coordena e monitoriza vários robôs para executar uma determinada aplicação. Este tipo de serviço aloca tarefas de acordo com as características e disponibilidade específicas de cada robô. Um exemplo é a utilização de vários manipuladores numa linha de fábrica.

➔ **Aplicação em enxame de robôs simples**

Este tipo aplicação coordena e monitoriza uma grande quantidade de robôs simples para executar aplicações cooperativas. Neste tipo de aplicação, todos os robôs têm o mesmo serviço e fornecem as mesmas funcionalidades. Um exemplo deste tipo de aplicação está nas missões de resgate de

desastres. Enxames de robôs de tamanhos diferentes podem ser enviados para locais onde os resgatadores não podem chegar com segurança, para detetar a presença de vida através de sensores infravermelhos.

## 3.3- Cinemática dos robôs

Neste projeto, adotou-se dois robôs com rodas diferenciais como mostra a figura 3.3.1.

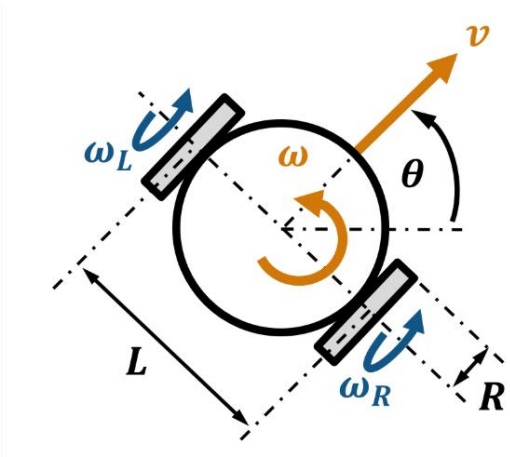



Figura 3.3.1- Robô diferencial. Fonte: [MRS2018]

Este tipo de veículos têm duas rodas independentes que podem ser usadas para controlar a velocidade linear e a angular. Para controlar o robô em questão, só se atua na velocidade de rotação das rodas $\omega_L$ e $\omega_R$ em rad/s (entrada do sistema). As saídas do sistema são a velocidade linear v em m/s e a velocidade angular $\omega$ do robô em rad/s.

Na cinemática direta, através das velocidades das rodas obtêm-se a velocidade linear e angular. Na cinemática inversa é ao contrário, ou seja, através da velocidade linear e angular consegue-se achar a velocidade angular das rodas.

As equações da cinemática direta [MRS2018] são as seguintes:

$$v = \frac{R}{2}(\omega_R + \omega_L)$$

$$\omega = \frac{R}{L}(\omega_R - \omega_L)$$

Através da cinemática inversa [MRS2018] é possível achar os valores das velocidades angulares $\omega_L$ e $\omega_R$ das rodas de acordo com o seguinte.

$$\omega_{\mathrm{L}} = \frac{1}{\mathrm{R}}\left(v - \frac{\omega}{2}\right)$$

$$\omega_{\mathrm{R}} = \frac{1}{\mathrm{R}}\left(v + \frac{\omega}{2}\right)$$

## 3.4- Arquitetura funcional

Nesta secção apresenta-se a arquitetura funcional que foi implementada em simulação para um caso específico de uma fábrica autónoma composta por um sistema de rede de robots móveis em operações logísticas. O diagrama na figura 3.4.1 apresenta uma visão geral de todo o sistema, identificando os principais componentes que foram desenvolvidos para o produto e suas interfaces. Este tipo de diagrama usa termos não técnicos a moderadamente técnicos que devem ser compreensíveis para os administradores do sistema. As finalidades deste tipo de diagramas são:

- ➔ Utilização em projetos preliminares

Em fases preliminares do desenvolvimento de um pojeto, a necessidade de dimensionar o projeto e facilitar a identificação das partes do projeto que podem ser arriscadas ou demoradas.

- ➔ Visão completa do projeto

Ao longo do desenvolvimento do projeto existe a necessidade cada vez maior de fazer uma descrição gráfica e intuitiva dos vários subsistemas e componentes do sistema.

Depois de explicar as vantagens de fazer um diagrama funcional, é importante evidenciar que o projeto está feito de forma a que se for em acrescentados robôs à fábrica não haja problemas. Este está feito de modo a que cada parte seja independente e cada uma com a devida função. As funções que cada robô tem, dependem de vários fatores:

- ➔ Disponibilidade das várias zonas do mapa;
- ➔ Carga necessária para o devido trajeto;
- ➔ Produto a ser processado (custo varia consoante a fábrica) e distâncias de trajetos;
- ➔ Obstáculos detetados no mapa (trajeto diferente);
- ➔ Número de pedidos feitos à fábrica.

Em segundo lugar, os robôs e as máquinas têm funções pré-definidas anteriormente, no entanto algo pode não funcionar como é suposto, por isso o projeto em questão não só teve em conta os processos e tarefas necessárias para o sucesso no transporte do produto como também em pormenores que não é normal acontecerem, mas na realidade acontecem. Mais precisamente, tomou-se em consideração que o sistema de carga, as subestações (zona de enchimento das garrafas de água), o sistema de descarga e os robôs de transporte podem avariar. Se isto acontecer, tudo tem de se adaptar até a um certo limite. No seguimento da explicação, no projeto temos duas cargas e se estas duas deixarem de funcionar tudo para, porque não existe matéria- prima a chegar do exterior, mas se só uma funcionar, então todas as cargas de matéria-prima dos robôs, dependendo da disponibilidade e prioridade dos robôs e da máquina de carga, serão feitas. Nas fábricas, nas zonas de descarga os robôs funcionam da mesma forma, quando todas as partes de uma secção estiverem avariadas tudo para, porque o produto final terá obrigatoriamente que passar por todas as fases, ou seja o carregamento de matéria-prima, transformação da mesma e descarga. Como na confeção do produto nada pode interferir no sistema, este tem de se adaptar aos recursos que tem.

Todas estas tarefas só são cumpridas devidamente se projeto estiver devidamente estruturado. Essa estruturação passa por saber dividir o projeto em partes como se vê na figura 3.4.1.

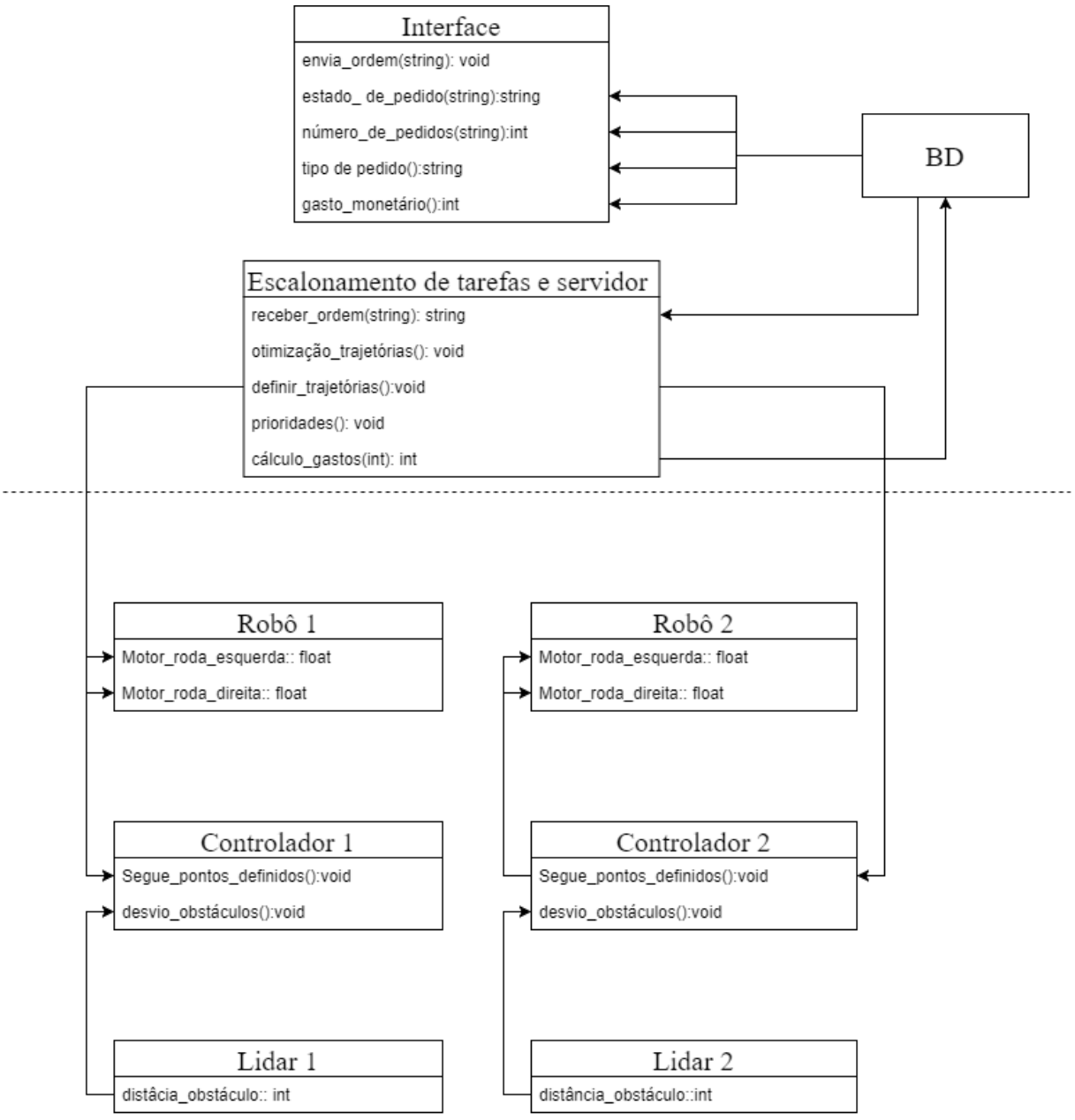



Figura 3.4.1- Diagrama funcional

**Como se vê no diagrama funcional existem vários blocos:**

- **Interface:** responsável por mostrar o estado do pedido feito e fazer encomendas à fábrica. Esta tem só ligação à base dados;

- **Escalonamento de tarefas e servidor:** responsável por receber a ordem da base dados (enviada pela interface) e enviar tarefas a cada robô existente na fábrica;

- **Robô:** recebe comandos do controlador e consequentemente ativa os motores das rodas;

- **Controlador:** responsável pelo robô seguir o trajeto definido pelo escalonamento de tarefas e por se desviar de obstáculos;

- **Lidar:** mede a distância do robô a um obstáculo e envia para o controlador.

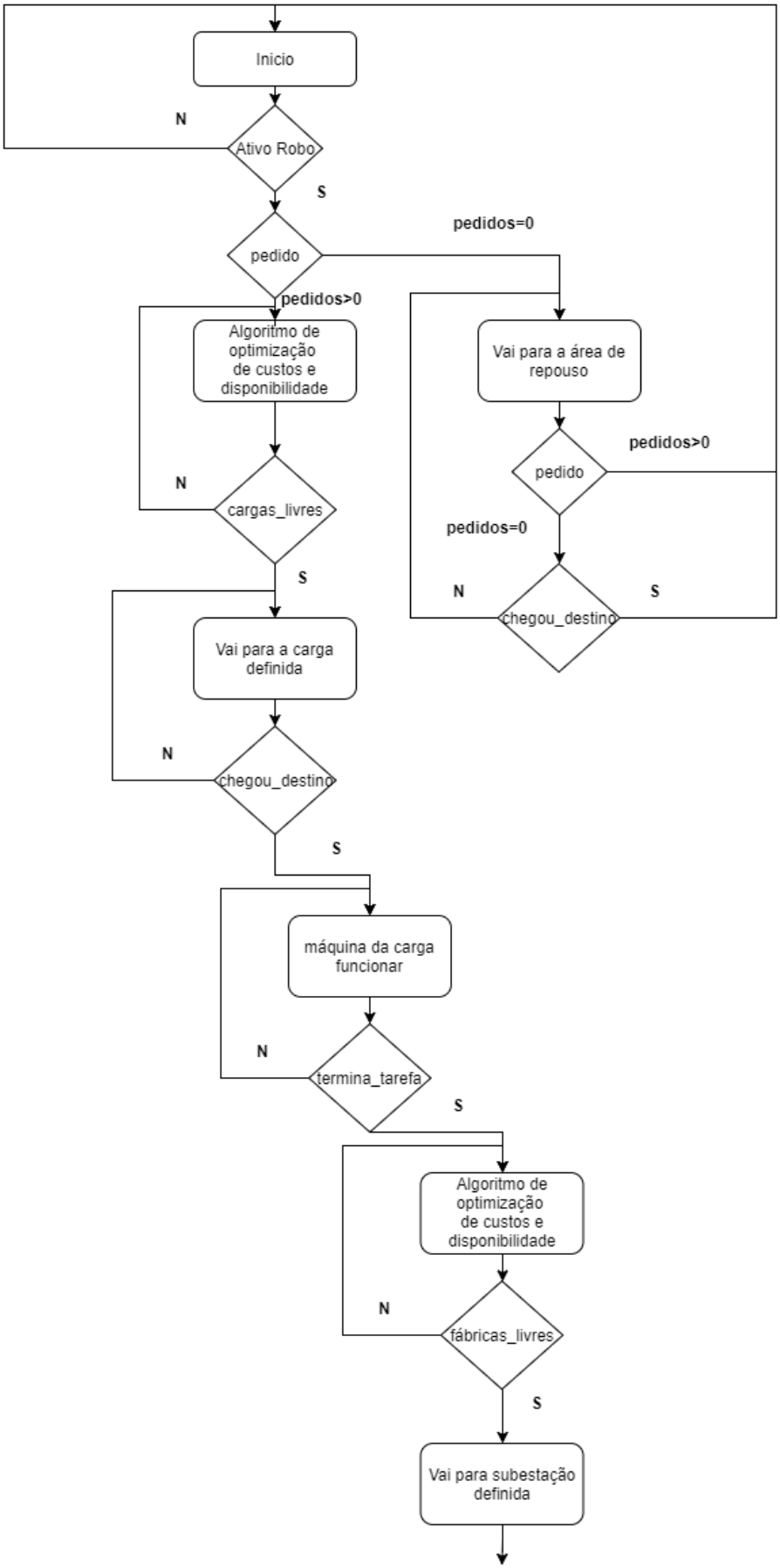
## 3.5- Diagrama detalhado dos sistemas principais (robôs e fábrica)

Nesta secção descreve-se de uma forma mais pormenorizada e esquematizada os vários sistemas relacionados com os robôs e a fábrica.

A figura 3.5.1 apresenta o fluxograma que foi implementado em Matlab para uma fábrica sem imprevistos, ou seja, como se não houvesse avarias, obstáculos e falta de energia. Referir que todas as posições dos robôs e das máquinas são atualizadas num espaço de tempo de 0.1s. Os números de pedidos referidos no diagrama a seguir representados são alterados por uma interface. A área de repouso (local limitado espacialmente no mapa do robô) consiste num local da fábrica que cada robô possui para quando não tiver nenhum pedido na base de dados ou a bateria dos robôs estiver a acabar. O algoritmo de otimização de custos e disponibilidade é explicado na secção 3.6.

O fluxograma na figura 3.5.1 só é ativado, quando algum robô está ativo e existe um pedido. Se não existir um pedido o robô dirige-se para a área de repouso. Se houver pedidos, este irá passar obrigatoriamente por um local de carga, subestação e local de descarga. Em cada local terá uma máquina que carrega e descarrega matéria prima ou produto final dependendo da finalidade do mesmo. Qualquer robô terá de proceder ao transporte de um local de carga, para uma subestação, onde será transformada e posteriormente carrega o produto final na subestação e leva para um local de descarga.

**Arquitetura do sistema de rede de robôs móveis**



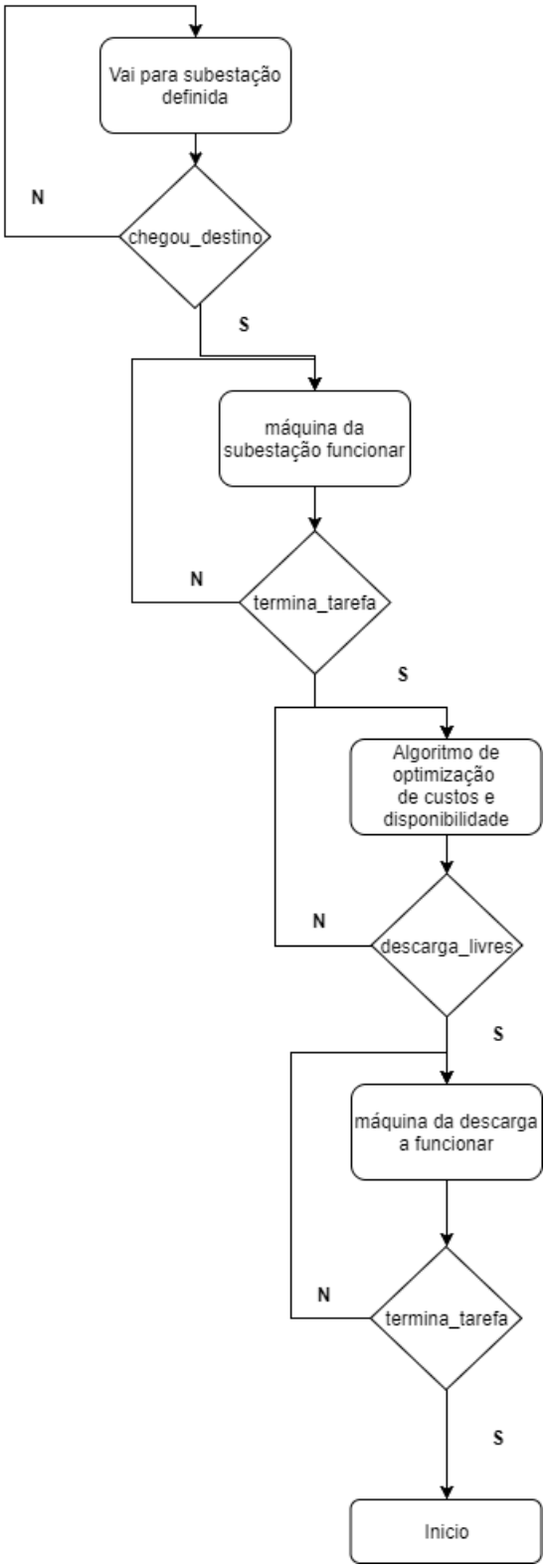



Figura 3.5.1- Fluxograma de baixo nível.

## 3.6- Otimização de custos de trajetória e custos de produto

**Algoritmo de otimização de custo (AOC)**

Hoje, num mercado com empresas cada vez mais competitivas é preciso ter ferramentas ou conhecimentos que possibilitem a produção mais rápida e mais barata de um de um certo item. Com isto é possível obter mais lucro. Neste projeto é utilizado um algoritmo que ajuda na diminuição do custo de um certo produto. Com este algoritmo foi possível fazer todas as combinações de trajetos e custo de produção de fábrica diminuindo o valor a pagar pelo produto.

Inicialmente será explicado o algoritmo de uma forma generalizada e posteriormente no caso particular (como se utilizou no projeto). Os fatores que influenciam o custo final são:

- ➔ Distância da posição do robô à carga;
- ➔ Distância da carga à subestação;
- ➔ Custo do produto em cada subestação (variável sempre que é executado o projeto);
- ➔ Distância da subestação à descarga;
- ➔ Custo associado a cada posto da fábrica (manutenção, energia gasta de cada máquina e no caso das subestações o valor associado à manufatura de um produto);

Este algoritmo irá calcular o mínimo do custo teórico que um produto custará desde a chegada da matéria-prima na zona de carga até o produto final chegar à zona de descarga. Estes cálculos têm em consideração a disponibilidade dos postos, porque estes poderão estar a ser utilizados por outros robôs ou poderão estar avariados. A fórmula geral matemática para o custo mínimo é a seguinte:

$$custo = \min \sum_{Ci} \sum_{Si} \sum_{Di} \sum_{Ri} [dist(Ri, Ci) \times T + cost(Ci) + dist(Ci, Si) \times T + cost(Si) + dist(Si, Di) \times T + cost(Di)]$$

Variáveis utilizadas:

- ➔ dist(a,b) representa a distância entre os postos de trabalho a e b;
- ➔ C representa a posto de Carga
- ➔ S representa subestação de manufatura
- ➔ D representa o posto de descarga
- ➔ R representa o Robô
- ➔ i identifica o número da carga, subestação, descarga e robô
- ➔ T custo associado à distância em €/m
- ➔ cost(a) é o custo associado ao posto a da fábrica

O diagrama de árvore da figura 3.6.1. explica o algoritmo que foi utilizado para determinar o custo mínimo. O algoritmo consiste em percorrer o diagrama de árvore e calcular o menor valor possível em cada nó. Em cada nó da árvore verifica-se se cada posto ou robô da fábrica está livre ou avariado. Se o posto ou robô não tiver condições para efetuar a tarefa desejada então esse nó é retirado do algoritmo de otimização de custo (AOC). Notar que com este algoritmo é sempre possível adicionar um número (finito) de postos e robôs ao sistema.

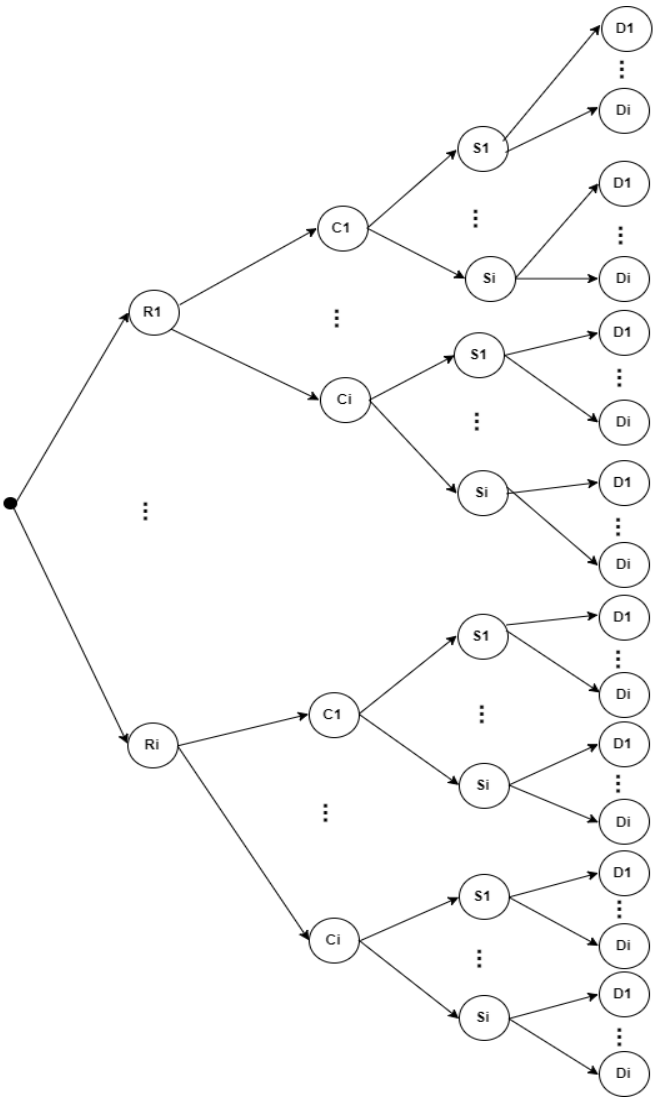



Figura 3.6.1- Diagrama de árvore (AOC)

Depois de explicado o AOC de uma forma geral é necessário colocar o problema de uma forma particular. O projeto desenvolvido tem 7 postos como mostra a figura 3.6.2 e 2 robôs que irão transportar matéria-prima e produto final. Depois do mapa feito, foi necessário calcular todas as distâncias necessárias para calcular o custo de transporte. Para isso foram usados as seguintes fórmulas em que $(x_2, y_2)$ e $(x_1, y_1)$ são as coordenadas da posição final e inicial respetivamente do trajeto do robô:

$$dx = |x_2 - x_1|$$
$$dy = |y_2 - y_1|$$
$$d = \sqrt{dx^2 + dy^2}$$

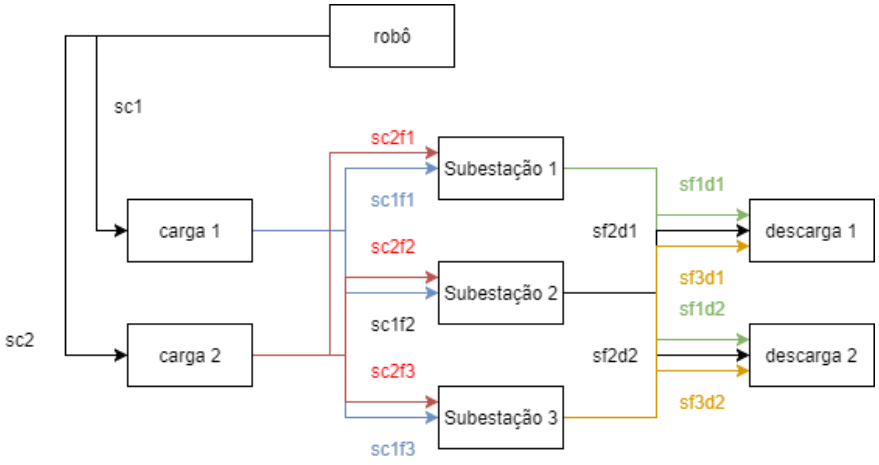



Figura 3.6.2- Diagrama de todas as distâncias da fábrica utilizadas no algoritmo

Depois de calcular as distâncias é necessário multiplicar por uma constante T que é o custo por metro percorrido pelo robô. No projeto o valor desta constante é 0.1€/m. Sabendo todas as combinações de todos os valores do custo do transporte é necessário acrescentar a este valor o custo referente a cada posto da fábrica. Depois de calculados todas as combinações será verificado em cada posto da fábrica qual será o melhor trajeto para que o custo a pagar seja menor. Os trajetos a escolher podem ser influenciados por avarias e por ocupações (os postos podem estar ocupados por outros robôs).

## 3.7- Controlador para desviar de obstáculos criados no mapa

Esta secção descreve o algoritmo responsável por evitar obstáculos da própria logística da fábrica ou outros colocados lá posteriormente.

A figura 3.7.1 apresenta a estrutura do controlador para evitar obstáculos. Considera-se que os robôs incluem como sensor um lidar com um número finito de n lasers. Na implementação foi considerado n=9. Os 9 lasers têm a mesma distância entre si, abrangem os 180º da frente robô e um alcance de 5m. Com isto é possível saber se existe algum objeto numa grande superfície. Todos estes parâmetros são definidos previamente no código, mas naturalmente se estivéssemos a utilizar um lidar físico teríamos de adaptar às limitações do hardware.

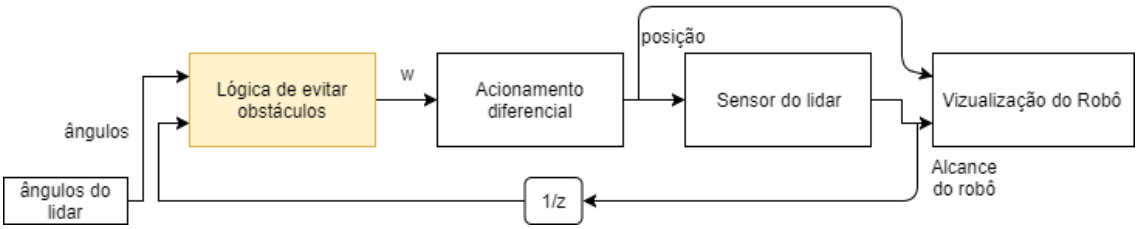



Figura 3.7.1- Esquema representativo do controlador evitar obstáculos

Para explicar este controlador é necessário dividir em partes ou blocos de modo a facilitar a compreensão e a explicação.

No primeiro bloco (“ângulos do lidar”), está contido um vetor com 9 valores que indicam a posição relativa angular dos 9 lasers do lidar (podiam ter mais lasers) e esses valores estão compreendidos entre $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ equidistantes. Considera-se que n é um número ímpar de lasers, de modo a que um fique em 0 rad e os outros lasers criem simetria do lado esquerdo e direito do robô. Para n=9, o vetor em radianos utilizado é:

$$ScanAngles = \begin{bmatrix} -1.570796326794897;\ -1.178097245096172;\ -0.785398163397448;\ -0.392699081698724;\ 0; \\ 0.392699081698724;\ 0.785398163397448;\ 1.178097245096172;\ 1.570796326794897 \end{bmatrix}$$

O segundo bloco ("lógica de evitar obstáculos"), recebe o vetor $ScanAngles$, e do vetor alcance do robô (consiste num vetor que em cada índice representa a distância de cada 1 dos lasers do robô ao obstáculo). Depois de receber os vetores, é verificado se os índices do vetor alcance ($range$) do robô não são nulos. Entretanto é criado um vetor booleano cujo valor dos seus índices dependem dos do vetor range. Os valores deste vetor são 1 se o vetor range for não nulo e 0 se for nulo (só são nulos se não detetarem nenhum obstáculo), no índice correspondente. Com a ajuda do vetor booleano foram criados 2 novos vetores, 1 vetor (angulosvalidos) onde só tinha os valores do $ScanAngles$ cujo o índice correspondia a 1 no índice booleano e o outro vetor ($rangesvalidos$) é calculado da mesma forma, mas com os valores do $range$. Os valores do vetor $range$ são atualizados a cada 0.1s.

Exemplo:

$$range = \begin{bmatrix} 3.973228983156362 \\ 4.759166620524910 \\ 4.919792286454625 \\ 4.554040932013504 \\ 6.066275230497077 \\ NaN \\ NaN \\ NaN \\ 9.9130441517109931 \end{bmatrix} \rightarrow \ vetor_{booleano} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

A seguir estão representados os vetores angulosvalidos e $rangesvalidos$ resultado do $vetor_{booleano}$ com $ScanAngles$ e o $range$ respetivamente. Os vetores $ScanAngles$ e $range$ ficaram unicamente com os valores cujo o índices corresponde ao valor não nulo do $vetor_{booleano}$.

$$\text{angulosvalidos} = \begin{bmatrix} -1.570796326794897 \\ -1.178097245096172 \\ -0.785398163397448 \\ -0.392699081698724 \\ 0 \\ 1.570796326794897 \end{bmatrix} \qquad rangesvalidos = \begin{bmatrix} 3.973228983156362 \\ 4.759166620524910 \\ 4.919792286454625 \\ 4.554040932013504 \\ 6.066275230497077 \\ 9.9130441517109931 \end{bmatrix}$$

A cada 0.1s (tempo de amostragem utilizado neste projeto) são calculados os vetores angulosvalidos e $rangesvalidos$. Depois de calculados os vetores fazemos a verificação se algum dos índices do vetor $rangesvalidos$ é menor do que um valor limite de 0.2 (valor estipulado neste

projeto), se for menor ativam o controlador de evitar obstáculos, caso contrário continuam no controlador anterior para seguir pontos de via. Ativado o controlador é calculado a média do vetor resultado da operação "angulosvalidos./ $rangesvalidos$" (cada elemento do resultado é a divisão do índice correspondente dos dois vetores da operação feita). Ao calcular a média sabemos se o robô tem o obstáculo à sua direita ou esquerda. Se tiver obstáculos mais à direita então a velocidade das rodas são 12.5 rad/s e -5 rad/s para a roda esquerda e direita respetivamente. Se tiver obstáculos mais à esquerda do robô, então os valores das velocidades angulares anteriormente referidas trocam (estas velocidades angulares são enviadas para o bloco acionamento diferencial).

Como neste projeto é utilizado também o controlador de seguir pontos e para evitar a comutação contínua entre o controlador de evitar obstáculos é importante referir que foi implementado uma estratégia de histerese que consiste na desativação do controlador de evitar obstáculos, quando o robô estiver a uma distância de 0.3, ou seja, foi criado um intervalo de segurança entre a ativação do controlador e a sua desativação. Com isto foi evitada a ativação contínua de controladores, ou seja, em termos simples impedir o robô de virar para a esquerda e para a direita continuamente se encontrar um obstáculo.

O bloco sensor do lidar tem de receber obrigatoriamente a posição do robô de modo a atualizar a posição referência, ou seja, atualizar a posição do robô face ao obstáculo.

O último bloco (visualização do robô), serve unicamente para receber os parâmetros da posição e do alcance do robô a qualquer objeto. Estes parâmetros são importantes, porque com eles é possível ter representação visual e é possível fazer uma simulação.

## 3.8- Controlador para desviar de um robô do outro

Como anteriormente foi referido, para o projeto do simulador foi utilizado uma biblioteca (Visualizer2D [10]) para ser possível a visualização dos robôs a interagirem com a fábrica. Nesta biblioteca inicialmente só se conseguia visualizar um robô e ter um mapa monocromático exemplo do matlab. Como a finalidade do projeto era ter uma fábrica com mais que um robô, foi necessário alterar o código fonte da biblioteca de modo a poder ter mais que um o robô e ser possível adicionar máquinas visíveis em vários pontos da fábrica. Os lidares utilizados neste projeto só identificam objetos a preto, pré-definidos na imagem monocromática adicionada à biblioteca. Como os robôs e as máquinas não fazem parte desta imagem monocromática (mapa da fábrica), os lidares utilizados não conseguem identificar o outro robô, deste modo, foi necessário criar uma alternativa que conseguisse simular um robô a evitar a colisão com outro. Por isso, foi criado um controlador com a única finalidade de evitar a colisão dos vários robôs. Nesta secção é importante frisar que não era necessário criar este controlador num caso real, porque o lidar conseguia identificar qualquer objeto, independentemente da cor, portanto o controlador explicado na secção 3.6 conseguiria

evitar qualquer robô. Como neste caso, a única maneira de demonstrar o projeto era através de um simulador, para aproximar o caso á realidade, foi criado este controlador.

Passando ao controlador propriamente dito. Em cada ciclo 0.1s é possível saber a posição em x, y e o ɵ (ângulo do sentido do robô) de cada robô. Com isto é possível criar um controlador que consiste na seguinte lógica.

Em primeiro lugar, calcular a distância entre robôs:

$$D = \sqrt{\left(y_{r_1} - y_{r2}\right)^2 + \left(x_{r_1} - x_{r2}\right)^2}$$

Em segundo lugar, verificar se o a distância é menor que 0.2. Se não for, só é verificado na iteração a seguir, mas se for então terá de ser verificado o sentido do robô. Para esta verificação é criado uma condição com duas inequações com um operador AND que são:

$$angulo\ r_2 + \pi \leq angulo\ r_1 + \frac{\pi}{6}$$
$$angulo\ r_2 + \pi \geq angulo\ r_1 - \frac{\pi}{6}$$

Depois de verificada a condição e se esta for verdade, então a velocidade das rodas muda para 12.5 rad/s e -5 rad/s para a roda esquerda e direita respetivamente. O outro robô que se encontra nas mesmas condições terá uma velocidade -5 rad/s e 12.5 rad/s para a roda direita e esquerda respetivamente, de modo a que os dois robôs rodem em sentidos opostos. Estas velocidades só se mantem se as condições anteriores se mantiverem.

## 3.9- Avarias

Esta secção do trabalho tenta aproximar o simulador à realidade. Num ambiente fabril real é normal existirem avarias de equipamentos. Para captar no simulador esta realidade, foi adicionado a cada máquina uma probabilidade de existência de avarias e também um painel de controlo, representado na figura 3.9.1, que coloca qualquer máquina inoperacional (necessário para manutenções e substituições).


Figura 3.9.1- Painel de controlo da fábrica

Para além, da possibilidade de forçar uma avaria, o simulador está preparado com um módulo que simula avarias que ocorrem de acordo com uma probabilidade que depende do número de utilizações da máquina. Essa simulação de avaria consiste no seguinte procedimento.

Para cada máquina (Carga, Subestação, Descarga) é atribuído uma variável aleatória T, definido como tempo de vida, que varia de acordo com uma distribuição de probabilidade imposta pelo utilizador. Associa-se também a cada máquina a função de fiabilidade R(t) que denota a probabilidade de a máquina não sofrer qualquer avaria no intervalo de tempo [0, t]. Matematicamente, R(t) = P(T>t)

Nesta implementação, foi utilizada a distribuição de probabilidade exponencial, isto é, T ~exp(lambda), onde lambda é um parâmetro positivo (ver figura 3.9.2). Neste caso, $R(t) = e^{-\Lambda * t}$. Notar que quanto maior for lambda, maior é a probabilidade de ocorrer uma avaria mais cedo.

A simulação consiste em cada vez que existe uma utilização da máquina, calcular p=R(t), em que t corresponde ao número de ocorrências e correr uma rotina que produz um número aleatório x uniforme no intervalo [0, 1]. Se x for menor que p, não existe avaria, caso contrário, existe uma avaria. Para um determinado número de utilizações (neste caso 100) foi forçado p=1, o que significa que durante as primeiras 100 utilizações a máquina não tem avarias.

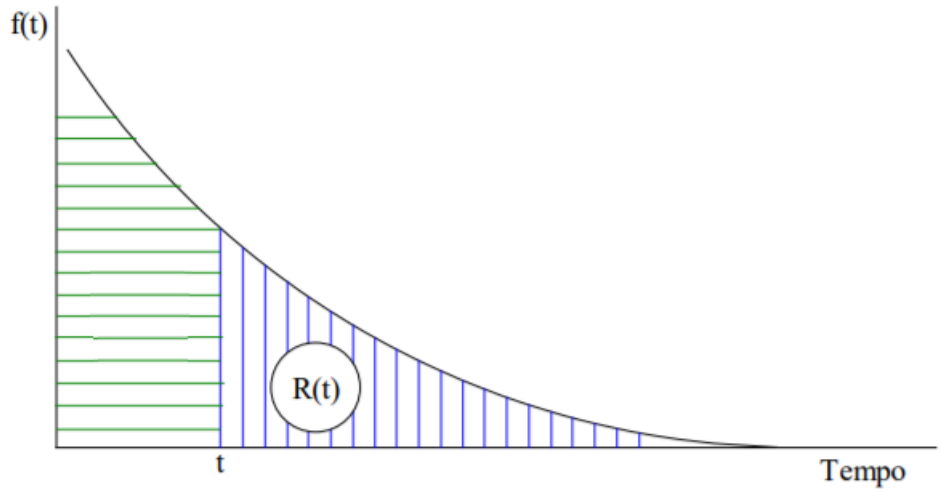



Figura 3.9.2- Função densidade de probabilidade de uma variável aleatória com distribuição exponencial. R(t) é a função de fiabilidade.

## 3.10-Base Dados

Para armazenar as informações sobre o projeto, é desenvolvido um banco de dados. Este banco de dados (BD) usa uma estrutura baseada em SQL que está armazenado no servidor da FEUP para MYSQL.

Neste projeto a BD é importante para guardar informação referente à fábrica e para guardar o estado do pedido descriminado. Deste modo, é possível aceder e colocar informação através de uma interface gráfica, através desta é possível pedir um tipo de produto e a sua quantidade. Depois disso a fábrica acede à base dados e verifica se existe algum pedido, se sim produz. A fábrica, na produção de um certo pedido, envia informação necessária para o operador compreender o estado da fábrica e do pedido. A tabela 3.10.1 contem os vários itens que são guardados, na tabela pedidos, que inclui o tipo de produto a produzir, a quantidade a produzir, o tempo do início do pedido (feito pelo o operador), o tempo em que a fábrica começou a produzir efetivamente o pedido e o fim da execução do pedido (os tempos guardados estão no formato (hh:mm:ss), o número de peças que estão em processamento, o número de peças que já chegaram à descarga de um determinado pedido, as peças que não foram processadas e o custo da produção total. Na tabela máquina, são guardados o tipo de máquina, o tempo de uso, total de peças produzidas e o tipo de peças produzidas. Na tabela descarga são guardados o número total de peças e o tipo de peças produzidas.

| **pedidos** | pedido | peça | quantidade | Inicio_pedido | Inicio_execucao | fim_execucao | pecas_em_processamento |
|---|---|---|---|---|---|---|---|
| pecas_processadas | pecas_pendentes | custo | | | | | |

| **maquina** | maquina | tempo_utilizaçao | total_pecas_produzidas | p1 | p2 | p3 |
|---|---|---|---|---|---|---|

| **descarga** | total_pecas | p1 | p2 | p3 |
|---|---|---|---|---|

Tabela 3.10.1- Tabelas da base de dados com o nome de cada tabela a negrito

Depois de implemendas as tabelas da base dados foi necessário criar uma classe separada da interface e da fábrica. Nesta classe está implementado várias operações tais como:

- Pesquisa;
- Inserção;
- Remoção;
- atualização.

Estas operações possibilitam a manipulação das tabelas da BD. A seguir é apresentado as funções utilizadas no projeto para manipular a base dados e uma breve explicação das mesmas.

## Funções de pesquisa

- ➔ Verifica_n_ordem(pedido, tabela)
  - Verifica se já existe o pedido na base dados
- ➔ Verifica_maquinas(maquina)
  - Verifica se já há máquina na base dados
- ➔ Verifica_descarga(descarga)
  - Verifica se já existe a zona de descarga na base dados
- ➔ Com_pedidos
  - Obtém o número de encomendas da tabela pedidos
- ➔ Lista_pedidos(pedidos)
  - Obtém o pedido todo descriminado
- ➔ Pedidos_maquina(maquina)
  - Obtém a informação de toda a máquina

- ➔ Pedido_descarga(descarga)
  - Obtém a informação da zona de descarga

## Funções de inserção

- ➔ Insere_BD (tabela, pedido, peca, quantidade)
  - Insere na tabela "pedidos" o pedido com tipo de peça e a sua quantidade

## Funções de atualização

- ➔ Atualiza_inicio_execucao (pedido)
  - Atualiza os tempos de início de execução do pedido na BD

- ➔ Atualiza_fim_exucucao (pedido)
  - Atualiza os tempos do fim de execução do pedido na BD

- ➔ Incrementa_pecas_em_processamento (pedido, tabela)
  - Incrementa na BD as peças que estão em processamento no pedido da tabela

- ➔ Decrementa_pecas_em_processamento (pedido, tabela)
  - Decrementa na BD as peças que estão em processamento no pedido da tabela

- ➔ Incrementa_pecas_processadas(pedido, tabela)
  - Incrementa na BD as peças que já foram processadas no pedido da tabela

- ➔ Atualiza_maquina (maquina, coluna_a_atualizar)
  - Incrementa na coluna_a_atualizar +1

- ➔ Reinicia_valores()
  - Reinicia valores a 0 nas máquinas

- ➔ Reinicia_valores_descarga()
  - Reinicia valores a 0 nas zonas de descarga

- ➔ Atualiza_tempo_operacao(maquina,tempo)
  - Incrementa o tempo de operação na máquina correspondente

- ➔ Atualiza_descarga (descarga, coluna_a_atualizar)
  - Incrementa na coluna_a_atualizar +1

## Função de remover

- ➔ Eliminacao_pedidos (pedido, tabela)
  - Elimina pedido da tabela

## 3.11-Interface utilizador fábrica

Neste projeto foi feito uma interface gráfica que tem as seguintes vantagens:

- ➔ Ser possível o operador pedir um pedido indicando a quantidade e o tipo de produto a ser produzido de uma forma intuitiva e de fácil controlo;
- ➔ Ser possível saber os dados referentes à fábrica de uma forma descriminada, rápida e intuitiva;
- ➔ Fornecer um ambiente visual mais agradável para trabalhar;
- ➔ Permite o operador controlar, manipular várias coisas ao mesmo tempo.

A GUI utilizada foi programada em c#. Esta linguagem de programação oferece melhores ferramentas para desenvolver uma melhor GUI, tanto a nível de controlo, como visual.
O resultado visual desta interface está nas figuras 3.11.1, 3.11.2 e 3.11.3.

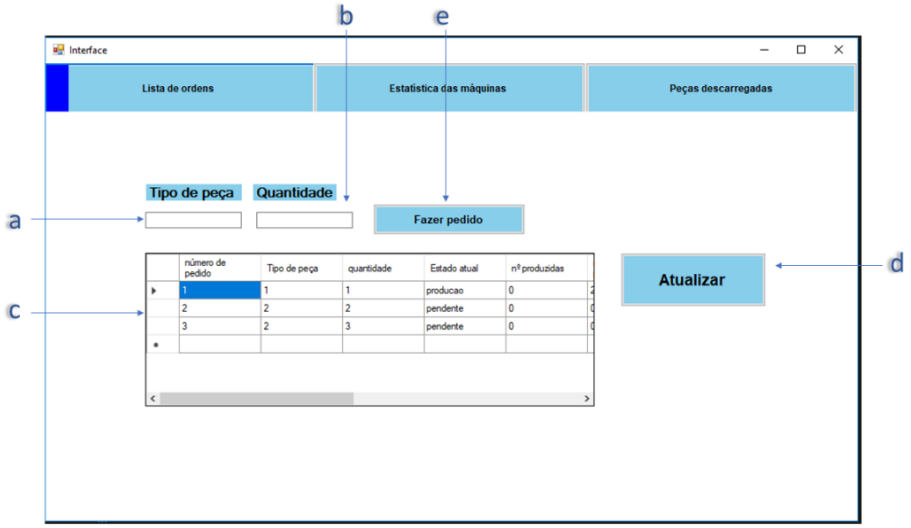



Figura 3.11.1- Interface controlo de ordens da fábrica, a) tipo de produto a ser produzido, b) quantidade do produto a ser produzido, c) descrição completa do estado dos pedidos, d) botão atualizar a tabela, e) inserção do pedido na base de dados

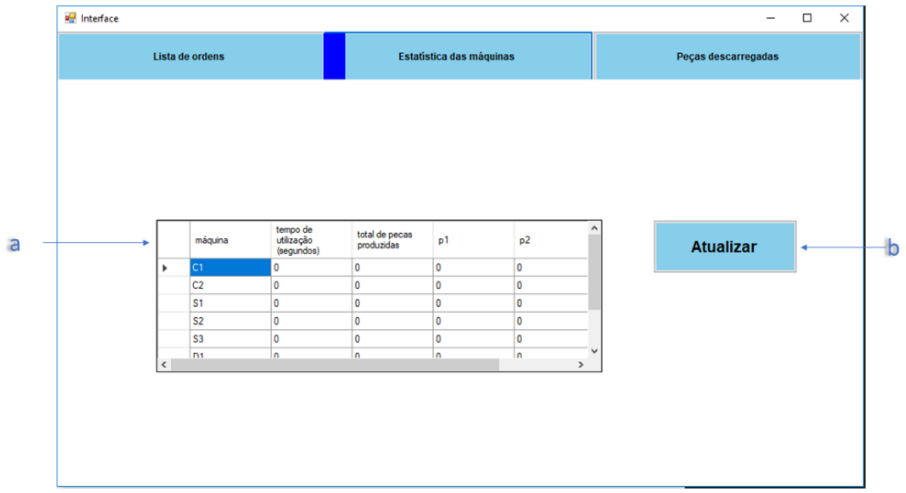



Figura 3.11.2- Interface da visualização do uso de cada máquina da fábrica, a) descrição completa do tipo, número de peças e tempo de utilização de cada máquina, b) botão atualizar a tabela

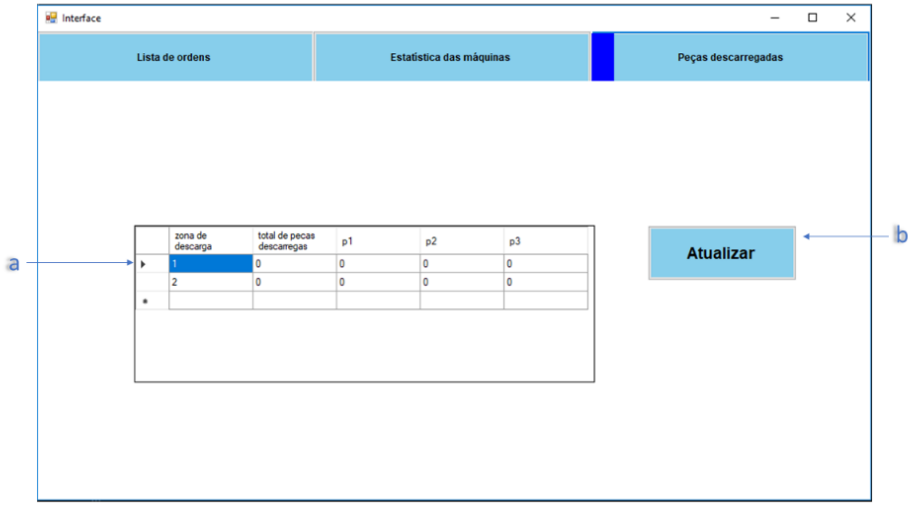



Figura 3.11.3- Interface da visualização da descarga da fábrica, a) descrição completa do tipo e número de descargas, b) botão atualizar a tabela

# Capítulo 4

# Demonstração de resultados

Nesta parte são apresentados testes e demonstrações do projeto e também serão apresentadas algumas explicações e conclusões dos mesmos.

Para apresentar e demonstrar os resultados é criado um simulador em matlab que é apresentado na figura 4.1 e figura 4.2. Possibilitando ter a visualização gráfica dos processos dos robôs e das máquinas implementadas num ambiente fabril. Este capítulo é dividido em várias secções, mostrando os testes de cada parte do projeto e as suas conclusões.

Na secção 4.1 será demostrado os tempos de produção da fábrica com tudo operacional e com avarias de algumas máquinas.

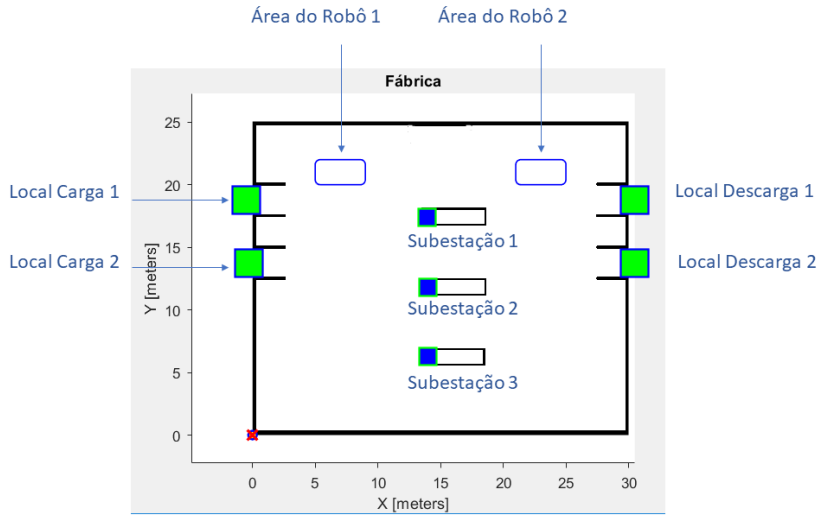



Figura 4.1- Mapa utilizado no simulador

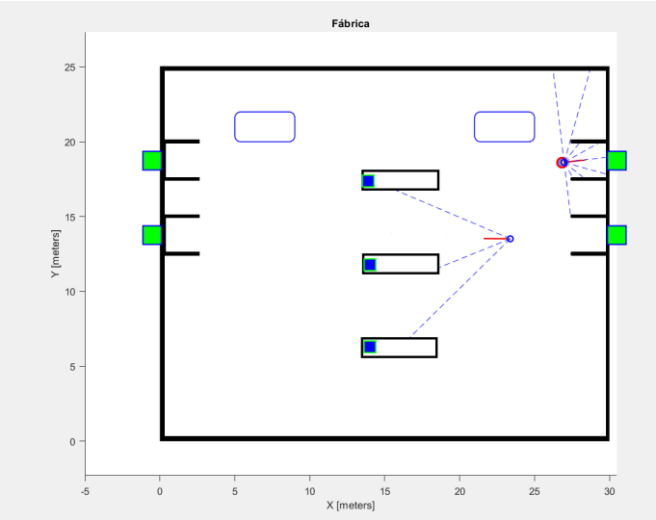



Figura 4.2- Fábrica em funcionamento

## 4.1- Tempos de produção

Nesta secção é demonstrado o tempo de execução (em simulação) de um pedido com e sem avarias das máquinas. O tempo de execução de um pedido depende de várias variáveis que são:

- ➔ A velocidade angular máxima e velocidade linear de um robô;
- ➔ Avarias das máquinas da fábrica;
- ➔ Número de robôs que trabalham numa fábrica.

A velocidade angular máxima e velocidade linear dos robôs que permitem confiabiliade (cerca de 700 testes não existiu qualquer colisão) do sistema é $1.5\ \mathrm{rads/s}$ e $0.75\ m/s$ respetivamente. Foram testadas velocidades superiores, mas obtinha-se erros devido à implementação do controlador de desviar obstáculos (explicado na secção 3.7) que não tinha tempo de resposta em alguns casos.

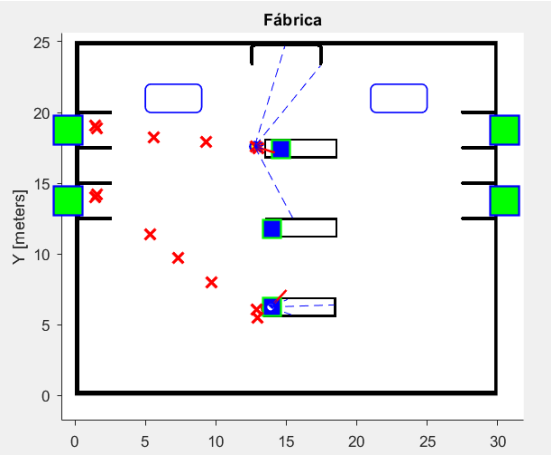



Figura 4.1.1- Colisão inesperada

As velocidades máximas utilizadas são $1.5\ \mathrm{rads/s}$ e $0.75\ m/s$. No entanto, verificou-se que se as velocidades diminuissem para $1.5\ \mathrm{rads/s}$ e $0.5\ m/s$, então para uma encomenda de 2 peças o tempo seria 34 segundos mais lento, ou seja haveria uma quebra na produção de 26%.

Depois foi testado como influencia as avarias das máquinas no tempo de produção de pedidos. Deste modo, foi testado quanto tempo demora a produzir 1, 2, 3, 4 e 5 peças com e sem avarias. As avarias forçadas que se utilizou para fazer testes de tempo de produção foram: numa máquina da carga, em duas máquinas das subestações e numa máquina da descarga. É necessário evidenciar que não é testada a fábrica com avarias nas duas máquinas de carga, nas duas máquinas de descarga ou nas três máquinas das subestações, porque, deste modo, a fábrica parava de produzir. Para os testes são utilizados 10 amostras para cada número de peças pedidas em cada situação e depois é calculada a média das 10 amostras. Nos gráficos das figuras 4.1.2 e 4.1.3 são apresentados os tempos de produção para cada situação. Os tempos que cada robô demora entre cada posto sofrem pequenas variações devido à função planeamento de trajetórias que define uma nova trajetória todas as vezes que é chamada.

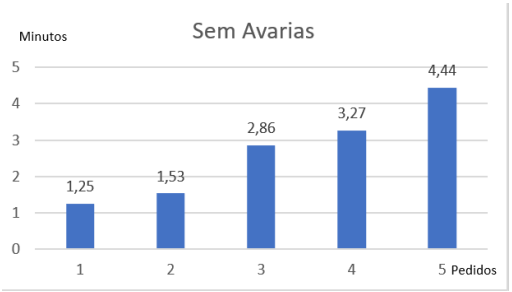



Figura 4.1.2- Fábrica sem avarias

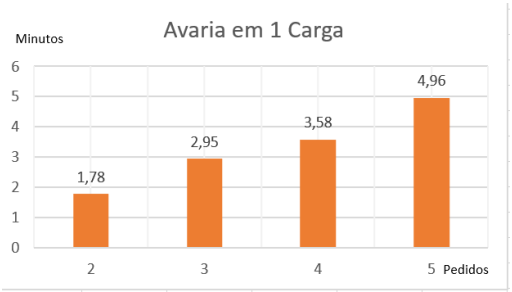



Figura 4.1.3- Fábrica com avaria em 1 carga

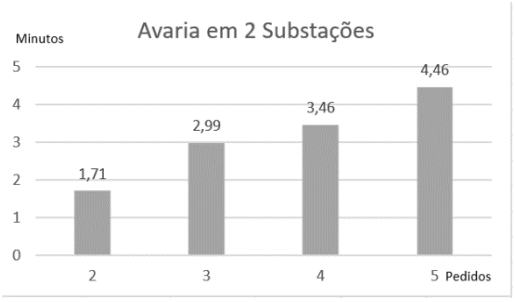



Figura 4.1.4- Fábrica com avarias em 2 subestações

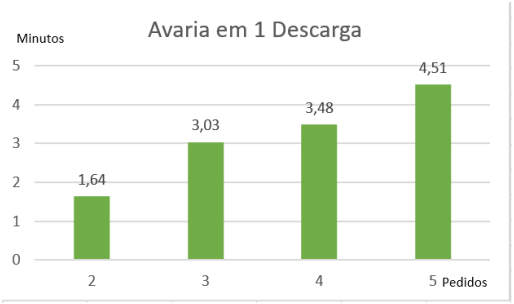



Figura 4.1.5- Fábrica com avaria em 1 descarga

Perante os vários testes que foram efetuados é possível concluir que as avarias pouco influenciam no tempo de produção dos pedidos nesta fábrica. A máxima diferença de tempo é 31 segundos (10.5%) na encomenda de 5 pedidos, ver as figuras 4.1.2 e 4.1.3. As diferenças de tempo são consequências das esperas entre robôs. Se as máquinas referentes a cada posto da fábrica demorassem mais tempo a fazer a sua tarefa, o tempo de produção era maior, porque os robôs ficavam mais tempo à espera que a máquina ficasse livre. Deste modo, as avarias iriam influenciar muito nos tempos de produção.

Por último, nesta secção avalia-se a influência do número de robôs que operam na fábrica. Para comparação é utilizado um único robô para fazer os pedidos das encomendas. Deste modo, foi testado quanto tempo demoraria a produzir 2, 3, 4 e 5 peças como se pode ver na figura 4.1.5. Sabendo os tempos comparou-se com os tempos dos mesmos pedidos, mas com dois robôs como se pode ver na figura 4.1.2.

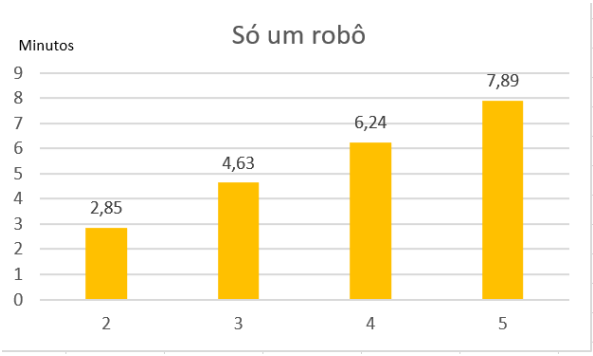



Figura 4.1.6- Fábrica com um único robô

Deste teste, pode-se concluir que a utilização de 2 robôs em vez 1 para 2 pedidos, por exemplo torna a fábrica 1 minuto e 19 segundos mais rápido, ou seja 186% mais rápido. Portanto aumentar o número de robôs torna a fábrica mais eficiente.

Estes tempos são referentes ao simulador, mas é possivel estimar qual seria o tempo real, ou seja, saber quanto tempo um robô demora a fazer o trajeto se estivesse a operar numa fábrica real. Para isso, teria-se de fazer o seguinte procedimento: multiplicar o tempo (em segundos) medido no simulador com o tempo que o programa demora a fazer uma iteração, pré-definido no projeto com 0.1s. O resultado desta multiplicação dá o número de interações necessárias para cumprir o tempo medido no simulador. Sabendo o número total iterações no simulador e se considerarmos uma iteração demora 1 segundo num robô real, então é possível saber o tempo necessário para cumprir uma tarefa numa fábrica real.

Por exemplo:

Tempo para cumprir uma tarefa (em simulador) = 171 segundos (2.85 minutos)
Tempo o programa demora a fazer uma iteração = 0.1 segundos
Tempo que um robô real demora para fazer uma iteração= 1 segundo

$$Nºitera\c{c}\tilde{o}es = \frac{171}{0.1} = 1710\ itera\c{c}\tilde{o}es$$

$$T_{rob\hat{o}_{real}} = 1710 * 1 = 1710\ s\ (28.5\ minutos)$$

## 4.2- Teste de avarias

Nesta secção, é demonstrado como varia o número de avarias com o número de utilizações de uma máquina. O valor de ∧ referente à distribuição de probabilidade exponencial é uma constante positiva, que neste projeto foi definida com valor de 0.00001. Nas figuras 4.2.1 e 4.2.2 podemos perceber como o valor ∧ interfere no aparecimento da falha, ou seja com um ∧ maior as avarias aparecem mais cedo do que com um ∧ menor. Podemos verificar que não existe avaria nas primeiras 100 utilizações, porque foi forçado p a ser igual a 1 (ver secção 3.9). Apartir das 100 utilizações a probabilidade de avaria aumenta com o número de ocorrênicas. A explicação do algorimo está na secção 3.9.

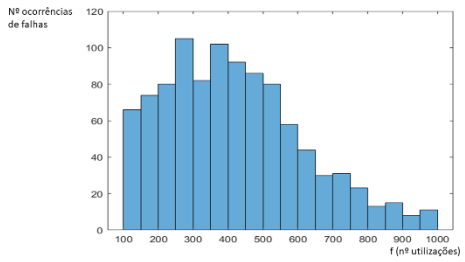



Figura 4.2.1- Histograma com o $\wedge = 0.00001$

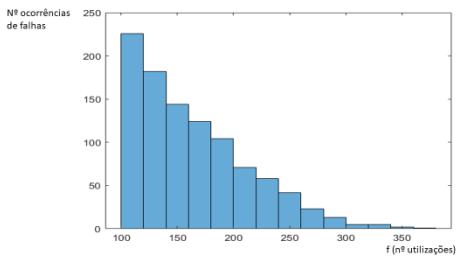



Figura 4.2.2- Histograma com o $\wedge = 0.0001$

Para ser testado o algoritmo em tempos razoáveis foi necessário aumentar $\wedge$ para 0.01, de modo a que as avarias das máquinas da fábrica ocorressem mais cedo, ou seja, o número das avarias ocorressem com menos utilizações. Como as avarias aparecem com um menor número de utilizações é necessária uma adaptação, ou seja, só é forçado unicamente a primeira utilização a não existir avaria. A seguir a figura 4.2.3 mostra como varia o número de utilizações das máquinas com a ocorrência de avarias para este caso particular onde são feitos testes.

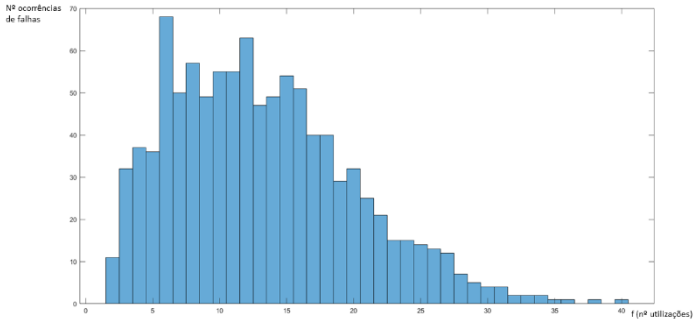



Figura 4.2.3- Histograma com o $\wedge = 0.01$

Depois da adaptação do algoritmo ao projeto é necessário saber alguns parâmetros importantes no bom funcionamento da fábrica e obter informação necessário para prever alguns imprevistos. Quanto maior for o número de amostras retiradas da simulação mais fidedigna fica a previsão. Em seguida apresentam-se os testes realizados para obter os tempos de produção perante eventuais falhas que foram simuladas de acordo com o que foi descrito na secção 3.9 utilizando a distribuição de probabilidade exponencial para a fiabilidade das máquinas. Para cada parâmetro foram feitos 20 testes. Os tempos utilizados são virtuais (do simulador), mas são proporcionais ao tempo real (ver secção 4.1). Estes indicadores são:

➔ Tempo necessário para ocorrer a primeira avaria como mostra a figura 4.2.4. Com estes dados é possível prever, quando uma máquina avaria e deste modo preparar a sua substituição.

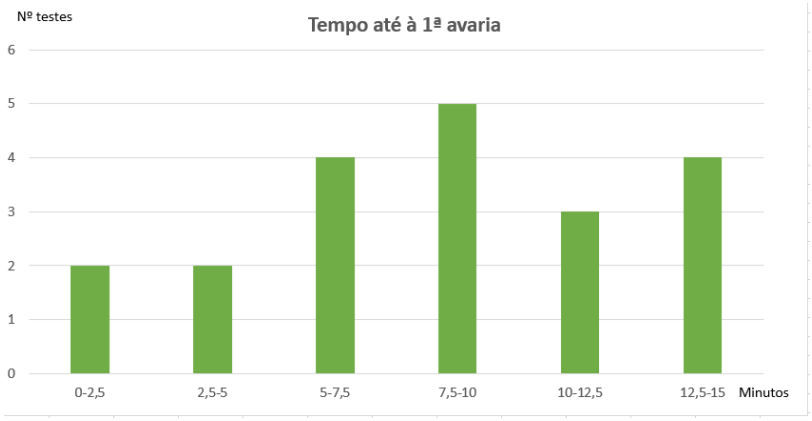



Figura 4.2.4- Tempo necessário até ocorrer a 1ª avaria.

➔ Número de pedidos feitos até à ocorrência da primeira avaria como mostra a figura 4.2.5. Com estes dados é possível saber o custo necessário acrescentar ao valor final do produto para compensar a manutenção das máquinas.

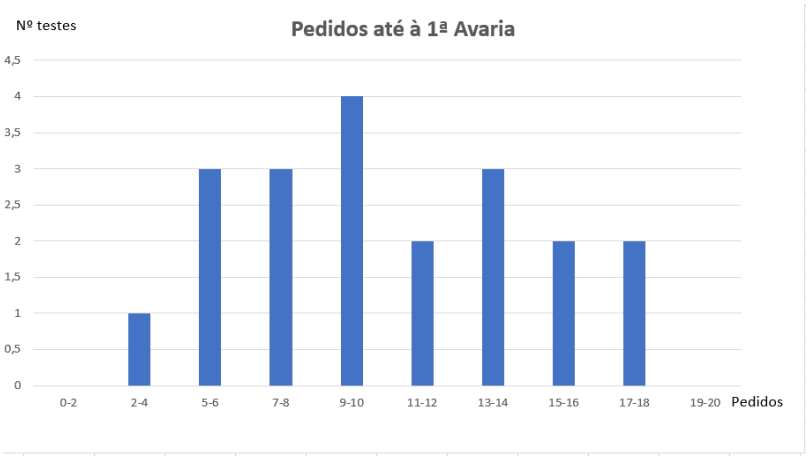



Figura 4.2.5- Números de pedidos até ocorrer a 1ª avaria.

➔ Tempo necessário para as máquinas todas de uma zona da fábrica avariarem (cargas, subestações ou descargas) como mostra a figura 4.2.6. Com isto é possível prever, quando a fábrica deixa de funcionar e preparar soluções.

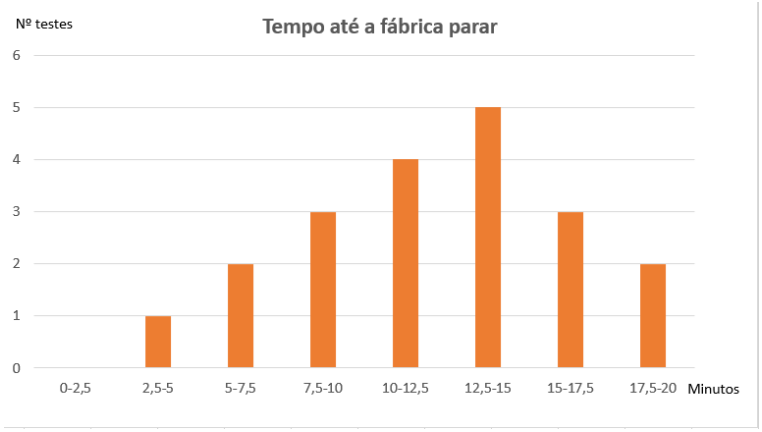



Figura 4.2.6- Tempo necessário para a fábrica parar.

➔ Número de pedidos feitos até a fábrica parar de funcionar como mostra a figura 4.2.7. Com isto é possível prever o número de pedidos que serão feitos até haver manutenção obrigatória.

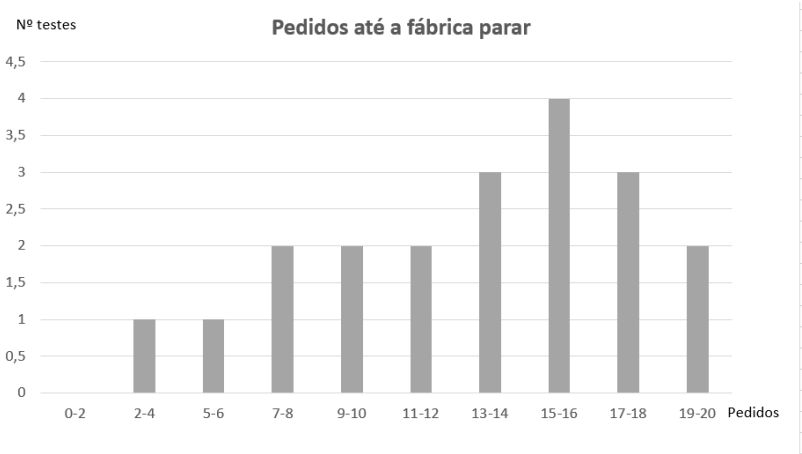



Figura 4.2.7 – Pedidos até a fábrica deixar de funcionar.

## 4.3- Teste do controlador para desviar obstáculos

Nesta secção é demonstrado o teste do controlador e explicação de resultados. Para a demonstração do mesmo é utilizado um caso prático do projeto.

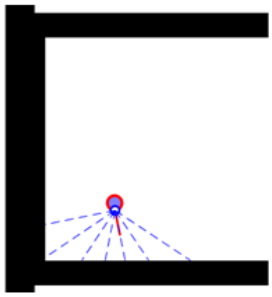


Figura 4.3.1- Teste 1 do controlador evitar obstáculos

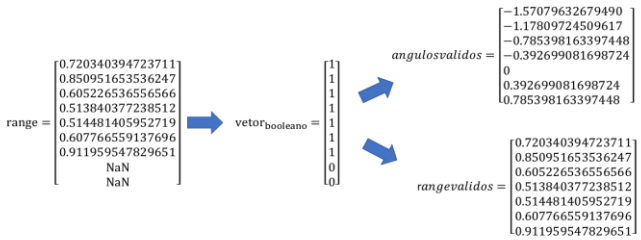



Figura 4.3.2- Cálculos do teste 1

O controlador evitar obstáculos é ativado, quando o robô está a uma distância menor do que 0.2 m do obstáculo. Como o lidar obtêm valores superiores como se pode ver no vetor range da figura 4.2.2 não ativa.

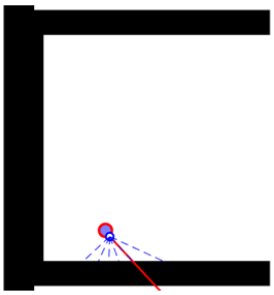


Figura 4.3.3- Teste 2 do controlador evitar obstáculos

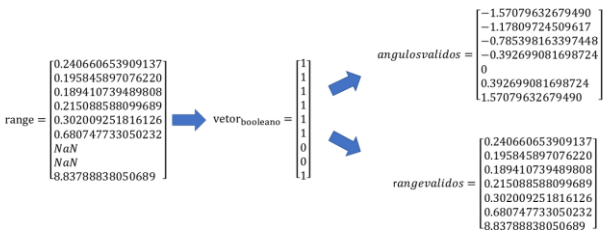



Figura 4.3.4- Cálculos do teste 2

No teste 2 o controlador é ativado, porque o robô está a menos de 0.2m do obstáculo, como se pode ver no vetor range da figura 4.2.4. Depois de ativar, faz o cálculo da divisão de cada elemento correspondente entre os vetores angulosvalidos e rangevalidos. Depois de fazer a divisão é calculado a média de todos os valores do vetor. O valor obtido da média é 2.537 negativo. Se esse

valor for negativo o robô roda para a esquerda e se for positivo roda para a direita. Logo o robô roda para a esquerda (a velocidade das rodas são -5 rad/s e 12.5 rad/s para a roda esquerda e direita respetivamente) e evita o obstáculo.

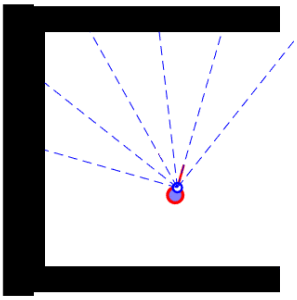


Figura 4.3.5- Teste 3 do controlador evitar obstáculos

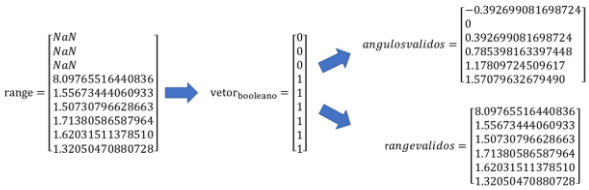



Figura 4.3.6- Cálculos do teste 3

No teste 3 o controlador evitar obstáculos está desativado, porque o robô está a mais de 0.3 m do obstáculo em todos os ângulos medidos pelo lidar, como se pode ver range da figura 4.2.6. Depois de ativado o controlador em 0.2 m, existe uma margem de 0.1m entre a desativação do controlador evitar obstáculos e ativação de controlador seguir pontos da trajetória que permite não haver inúmeras comutações em pouco espaço de tempo e possibilitar o robô seguir o seu trajeto pretendido.

# Capítulo 5

# Conclusão e trabalho futuro

## 5.1- Conclusão

A indústria 4.0 mudou a forma de pensar e de agir das empresas. A principal consequência da mudança é a competitividade. Se uma empresa não se superar e não tiver uma mentalidade inovadora está sujeita à certeza de ser ultrapassada pelas demais. A mudança muitas vezes é difícil, dá trabalho e tem custos, mas a médio e longo prazos compensa. A capacidade de superação e de mudança são dos parâmetros mais importantes de uma empresa em crescimento.

Esta dissertação aponta para uma possibilidade de uma realidade no futuro (que pode ser muito breve) em que os produtos são fabricados em fábricas totalmente autónomas.

Em particular, esta dissertação focou-se no estudo, desenvolvimento, implementação em simulação e análise de resultados de um exemplo de uma fábrica autónoma que utiliza robôs moveis em cenários de logística.

O simulador desenvolvido incorpora vários módulos, sendo de destacar os algoritmos de planeamento de trajetórias e de controlo de movimento dos robôs, o algoritmo de otimização e decisão que coloca prioridades de tarefas e seleciona as trajetórias de modo a minimizar o custo de fabrico, o módulo de avarias que permite avaliar o efeito destas na performance geral da fábrica, o módulo de segurança de informação, guardando informação da fábrica numa base dados MYSQL e módulo de interface com o utilizador que facilita a utilização do simulador.

Apesar de tudo o que foi proposto funcionar com êxito, existem, no entanto, situações que podem ser melhoradas, nomeadamente na validação de alguns parâmetros que podem ter uma influência significativa no comportamento geral da fábrica e que se forem alterados cegamente podem gerar erro ou paragem da fábrica. Por exemplo, no controlador de evitar obstáculos, é necessário ter cuidado em não ultrapassar a velocidade angular e a linear dos robôs $1.5\,\mathrm{rads/s}$ e $0.75\,m/s$, respetivamente, caso contrário pode haver colisões. Na parte gráfica do simulador, existem problemas de redimensionamento e resolução da imagem com grandes ampliações da

fábrica. Em relação ao módulo das avarias é preciso ter em consideração que se todas as máquinas de uma zona (ex: cargas, subestações ou descargas) avariarem a fábrica não tem capacidade de produzir e o simulador para.

Foi possível demonstrar o funcionamento de uma fábrica completa como era previsto no início do projeto e fazer inúmeros testes e tirar conclusões tanto num ambiente de simulador, como prever resultados num ambiente real.

## 5.2- Trabalho futuro

É seguro afirmar que os objetivos propostos no início do projeto da dissertação foram atingidos já que foi implementado com sucesso o simulador e criado uma fábrica com todos os requisitos pré estipulados. Contudo, é sempre possível acrescentar alguns requisitos no simulador que num ambiente real obrigatoriamente seriam implementados tais como:

- ➔ Algoritmo de localização com o propósito de saber em cada instante a posição real dos dois robôs;
- ➔ Comunicação entre robôs e fábrica (por exemplo posição do robô e início e fim de processos das máquinas).
- ➔ Simulação de baterias dos robôs
- ➔ Capacidade intuitiva e simples de mudar o layout da fábrica

# Referências

**[Abbadi2015]** Abbadi, A., & Přenosil, V. (2015). Safe path planning using cell decomposition approximation. Distance Learning, Simulation and Communication, 8: p.pp. 2-3.
**[Aurenhammer1991]** Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure: p.pp. 55-59.
**[Barbosa2013]** Barbosa, F. M. (2013). Introdução á fiabilidade de sistemas elétricos de energia. Faculdade de Engenharia Universidade do Porto: p 23.
**[Bauer2013]** Klaus Bauer, Dr. Bernhard Diegner, Johannes Diemer, Wolfgang Dorst, Dr. Stefan Ferber, Rainer Glatz, Ariane Hellinger, Dr. Werner Herfs, Marion Horstmann, Dr. Thomas Kaufmann, Dr. Constanze Kurz, Dr. Ulrich Löwen, Veronika Stumpf (2013) Recommendations for implementing the strategic initiative INDUSTRIE 4.0: p.pp. 13-24.
**[industry4.0_2019]** Planning for Industry 4.0 with Simulation. Acedido em: 13, janeiro, 2019, em: https://www.simul8.com/manufacturing/implementing-industry-4-0-with-simulation
**[IR2019]** The Advantages and Disadvantages of Industrial Robots. Acedido em 13 janeiro 2019, em: https://www.applerubber.com/blog/the-advantages-and-disadvantages-of-industrial-robots/
**[Liao2017]** Liao, Y., Deschamps, F., Loures, E. D. F. R., & Ramos, L. F. P. (2017). Past, present and future of Industry 4.0-a systematic literature review and research agenda proposal. International journal of production research, 55(12), 3609-3629.
**[Lu2017]** Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. Journal of Industrial Information Integration, 6, 1-10.
**[Marr2019]** Bernard Marr, what is Industry 4.0? Here's A Super Easy Explanation For Anyone. Acedido em 13 janeiro 2019, em: https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/
**[MRS2018]** Mobile Robotics Simulation. Acedido em: 10, agosto, 2018, em: https://www.mathworks.com/matlabcentral/fileexchange/66586-mobile-robotics-simulation-toolbox
**[Oliveira2015]** de Oliveira, L. B. R. (2015). Conception architecturale des systèmes robotiques orientée services (Doctoral dissertation, Université de Bretagne Sud): p.pp. 55-62.
**[PotentialFields2013]** Local Navigation using Potential Fields(2013) Acedido em 13 janeiro 2019, em: https://randomaccessmaths.wordpress.com/2013/10/27/potential-field-nav/
**[PPC2019]** Pure Pursuit Controller. Acedido em 13 janeiro 2019, em: https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html
**[PRM2019]** Probabilistic Roadmaps. Acedido em 13 janeiro 2019, em: https://www.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html

**[Rüßmann2015]** Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., & Harnisch, M. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. Boston Consulting Group, 9.: p.pp. 3-7.

**[Samir2014]** Samir, N. (2014). Collision Avoidance and Simple Path Planning for Autonomous Robotic Exploration.
Disponível em: http://www.diva-portal.org/smash/get/diva2:1017998/FULLTEXT02

**[Shead2019]** Sam Shead, Amazon now has 45,000 robots in its warehouses. Acedido em 13 janeiro 2019, em: https://www.businessinsider.com/amazons-robot-army-has-grown-by-50-2017-1

**[SRR2018]** System Requirements - Release 2018a. Acedido em 13 janeiro 2019, em: https://www.mathworks.com/content/dam/mathworks/mathworks-dot com/support/sysreq/files/SystemRequirements-Release2018a_Windows.pdf

**[VisibilityGraphs2019]** Shortest Paths and Visibility Graphs Acedido em 13 janeiro 2019, em: https://www.cse.wustl.edu/~pless/546/lectures/l22.html