INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Sistema Robótico Autónomo para ambientes de Terapia de Iodo

Ana Rita Carneiro de Andrade

Relatório Final de Trabalho de Projeto apresentado(a) à:

Escola Superior de Tecnologia e Gestão

Instituto Politécnico de Bragança

para obtenção do grau de Mestre em

Tecnologia Biomédica

Este trabalho foi efetuado sob orientação de:

Professor José Lima

MSc. Maria do Carmo Baptista

Professor Paulo Leitão

Novembro 2015

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Sistema Robótico Autónomo para ambientes de Terapia de Iodo

Ana Rita Carneiro de Andrade

Relatório Final de Trabalho de Projeto apresentado(a) à:

Escola Superior de Tecnologia e Gestão

Instituto Politécnico de Bragança

para obtenção do grau de Mestre em

Tecnologia Biomédica

Este trabalho foi efetuado sob orientação de:

Professor José Lima

MSc. Maria do Carmo Baptista

Professor Paulo Leitão

Este Relatório final do Trabalho de Projeto inclui as críticas e sugestões feitas pelo Júri

Novembro 2015

Agradecimentos

Este trabalho não teria sido possível sem a colaboração e a boa vontade daqueles que agora irei mencionar. A todos o meu sincero agradecimento.

Ao Professor José Lima, pela disponibilidade, interesse e recetividade com que me recebeu, pela prestabilidade com que sempre me ajudou quando surgiam dúvidas. Agradeço pelo estímulo e exigência crescente que me foi impondo à medida que caminhava para a conclusão deste trabalho.

Também queria agradecer ao Professor Paulo Leitão e à Dr. Maria do Carmo pela disponibilidade que sempre demonstraram para esclarecer dúvidas quando estas surgiam assim como a sua preocupação constante para saber se tudo estava a correr bem com o desenvolvimento deste trabalho.

Um grande obrigado aos meus pais e irmão, que sem eles nunca teria conseguido chegar onde me encontro hoje. Obrigada pelo apoio, paciência, sacrifício, pela generosidade e ajuda que sempre demonstraram perante mim. Por fim também gostaria de agradecer aos meus amigos que sempre me ajudaram e apoiaram quando necessitava.

Resumo

Existem diferentes tipos de doenças que necessitam de tratamentos que envolvem radiação ionizante. O caso do cancro da tiroide é um destes exemplos, onde em alguns casos é necessário recorrer a um tratamento com iodo (^{131}I), o qual vai atacar as células cancerígenas.

Este tratamento, envolvendo radiação ionizante, provoca uma constante exposição a este tipo de radiação, a qual pode causar efeitos nefastos para os profissionais de saúde, que lidam com elementos radioativos.

Desta forma, é necessário criar alternativas que previnem a exposição destes profissionais à radiação, minimizando os seus efeitos.

Este trabalho tem como objetivo desenvolver, através de simulação, um robô omnidirecional que se deslocará no ambiente de terapia de iodo para tratamentos de problemas na glândula tiroide, sem a necessidade de presença humana.

O modelo do sistema robótico foi desenvolvido no *Simtwo*, no qual é possível criar um cenário semelhante ao que existe no ambiente de ambulatório, assim como um código com o qual é controlado o robô. A programação do robô omnidirecional envolve a implementação dos modelos cinemático e dinâmico, assim como a manipulação das variáveis que definem as trajetórias a executar.

Abstract

There are different types of diseases that require the use of radiation for the treatments. Thyroid cancer is one of these examples, where in some cases it is necessary to use iodine (^{131}I), which will attack cancer cells.

This treatment involves the use of ionizing radiation, which can bring benefits for patients but can also cause side effects for health professionals. This happens because of the constant exposure to radiation.

Because of this it is necessary to create alternatives to prevent the constant exposure of these professionals. The objective of this work is to develop a prototype, based on simulation, where an omnidirectional robot can move without the interference of anyone to measure the remaining radiation in a room.

This robotic system was developed in a simulator called *Simtwo*, in which we can create a scenario similar to the clinical environment, as well as the code that allow as to control the robot. The creation of this omnidirectional robot involves the implementation of kinematics and dynamic models and the manipulation of different variables that define running paths.

Índice

Índice de imagens

Índice de Tabelas

Capítulo 1- Introdução

Neste capítulo será feita uma breve introdução ao trabalho realizado, assim como ao uso da terapia de iodo para tratamentos de problemas na glândula tiroide. Serão abordados os problemas associados a este tipo de tratamento por radiação ionizada e apresentada a motivação para a utilização de outras técnicas que permitem com que haja mais segurança por parte do pessoal técnico numa sala de tratamento. Por fim, será descrita a estrutura do relatório.

1.1 Motivação e Problema em estudo

Com a descoberta dos raios X em 1895, por Wilhelm Conrad Röntgen, foi possível o desenvolvimento de técnicas de análise que auxiliam no diagnóstico médico. Foi esta descoberta que impulsionou outros cientistas a efetuarem pesquisas na área da radioatividade. No ano de 1896, Henri Becquerel descobriu a radioatividade natural e em 1913, George de Hevesy apresentou o conceito de marcadores radioativos. Este conceito apresentado por George Hevesy ainda se mantém atual e largamente utilizado na Medicina Nuclear.

Com Marie Currie, surgiram as primeiras aplicações terapêuticas de elementos radioativos, uma vez que esta foi capaz de separar o elemento rádio em quantidades suficientes que permitiram a caracterização e o estudo cuidado das suas propriedades [1].

Desde esses tempos tem havido uma grande evolução científica na área da Medicina Nuclear, o que possibilita um diagnóstico precoce, permite o estudo de comportamentos fisiológicos de maneira simples, não invasiva e com baixo risco para o paciente e também é utilizado em determinados tratamentos os quais apresentam uma grande eficácia.

Como já mencionado a Medicina Nuclear é útil para o tratamento de doenças e neste caso em específico, o elemento iodo 131 (^{131}I) é utilizado para o tratamento de patologias relacionadas com a glândula tiroide.

A tiroide é uma glândula que se localiza na zona do pescoço, como se pode verificar na Figura 1. Esta sintetiza, segrega e armazena duas hormonas que são de extrema importância para o bom funcionamento do organismo. Estas hormonas são denominadas por triiodotironina, mais conhecida por T3, e a tiroxina ou T4. A sua principal função é a de permitir um crescimento e desenvolvimento normal, especialmente em crianças

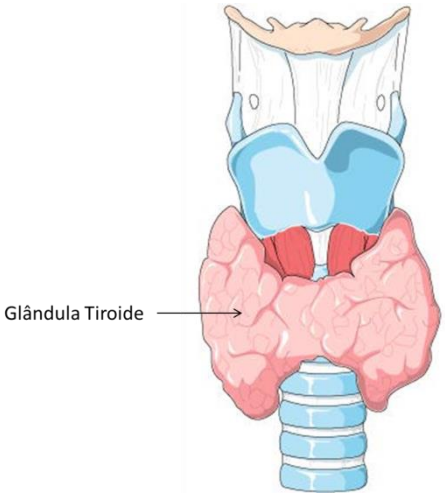


Figura 1- Representação da glândula tiroide.

Existem vários tipos de doenças associados a esta glândula, nomeadamente o hipotiroidismo, hipertiroidismo, doença de Grave, gota, adenoma e cancro na tiroide. Para este trabalho só será necessário abordar o cancro na tiroide, pois só nestes casos é que, por vezes, é necessário efetuar tratamentos de iodo.

O cancro na tiroide é o mais comum e prevalente de todas as doenças endócrinas malignas. Este é tratado, na maioria dos casos, com terapia de iodo, a qual se encontra associada a tratamentos por radiação ionizante.

Este tipo de radiação pode trazer benefícios para a saúde do paciente, porém ser extremamente prejudicial para os profissionais que se encontram em contacto com os pacientes e os radiofármacos.

Desta forma, o uso de iodo radioativo, ^{131}I, para o tratamento deste tipo de cancro é amplamente utilizado nos últimos 50 anos. A produção do ^{131}I pode ser feita através de Telúrio (Te) ou Urânio (U). Quando produzido através do telúrio, o ^{131}I resulta da irradiação de um neutrão para um elemento natural de telúrio. O isótopo ^{130}Te absorve um neutrão do qual resulta o ^{131}Te, o qual a radiação β decai para o ^{131}I num tempo de semi-vida de 25 minutos. Para que seja possível a obtenção do ^{131}I, este tem de ser separado do Telúrio alvo através de uma destilação seca ou molhada e seguido de uma dissolução numa solução alcalina [2].

Como já foi mencionado o iodo radiativo também pode ser obtido através do urânio, utilizando o processo físico denominado de fissão nuclear[1] do ^{235}U (urânio) resultando em fragmentos de fissão, ^{131}I e outro isótopo de iodo. Esta técnica é menos utilizada do que anterior pois esta necessita de equipamentos elaborados, processos de purificação e controlo de desperdícios [3].

O ^{131}I tem como características físicas a solubilidade em água e o facto de ser altamente volátil. O seu ponto de fusão é de 113º e o ponto de ebulição é de 184º [9]. A nível químico o ^{131}I, tem 78 neutrões e 53 protões no núcleo e um tempo de semi-vida de aproximadamente 8 dias. Durante o decaimento do ^{131}I, 90% da radiação advém das partículas β^- e 10% em forma de radiação γ.

Neste caso do tratamento com iodo, a radiação que interage com os tecidos da tiroide é a radiação β^-, que apresenta a capacidade de ionizar a matéria, por sua vez a radiação γ possui fotões que ionizam a matéria de forma indireta o que é prejudicial para indivíduos que se encontrem em contacto com os pacientes [4].

Para evitar o contacto com radiação por parte dos profissionais de saúde, é necessário definir algumas medidas de proteção, tais como o uso de blindagem nas salas, proteção para as diferentes partes do corpo que possam estar expostas à radiação e desenvolver novas formas para se evitar esta exposição.

[1]A fissão nuclear é p processo onde o núcleo de um átomo se divide em dois fragmentos diferentes devido à interação com energia ou ao bombardeamento de neutrões.

1.2 Objetivos

O objetivo deste trabalho consiste em criar um ambiente de simulação onde um robô omnidirecional de três rodas, se possa movimentar numa sala sem necessidade de exposição dos profissionais de saúde à radiação ionizante produzia pelo decaimento do elemento iodo 131 que é utilizado em tratamentos da glândula tiroide. Este protótipo vai percorrer diferentes posições numa sala de tratamento onde com o auxílio de um dosímetro, serão recolhidos os dados referentes à dose de radiação presente na sala, diminuindo assim o tempo de exposição do profissional de saúde.

Para que seja possível a construção da simulação é necessário utilizar o simulador *SimTwo,* o qual foi desenvolvido pelo Prof. Dr. Paulo Costa. O robô em causa terá de ser autónomo, isto é depois de fornecidas posições concretas, este deslocar-se-á pelas salas de tratamento sem ser necessário a presença de profissionais de saúde na sala. Além de ser autónomo, este será omnidirecional, isto irá ter as mesmas características em todas as direções. Estas características serão dadas através dos diferentes modelos utilizados cinemático e dinâmico.

1.3 Estrutura do relatório

Este relatório encontra-se organizado em 6 capítulos.

No Capítulo 1 é elaborada uma pequena introdução ao uso do tratamento de iodo para o cancro da tiroide, os problemas associados, assim como os objetivos e a estrutura do relatório.

No Capítulo 2 são abordados alguns conceitos físicos que permitem a medir a quantidade de radiação presente numa sala de tratamento, assim como os efeitos que este tipo de radiação tem sobre os profissionais de saúde. Além disso também são descritas medidas preventivas para diminuir a exposição dos profissionais de saúde à radiação.

No Capítulo 3 foi realizado um breve estudo do estado da arte sobre a evolução dos sistemas móveis autónomos ao longo dos últimos anos e os diferentes usos para estes sistemas.

No Capítulo 4 encontram- se descritos os modelos que são necessários ter em conta para o funcionamento do robô. Também é descrito o algoritmo que permite com que o robô se localize sozinho, sem que seja necessário saber de ante mão a posição onde se encontra inicialmente.

Nos Capítulos 5, está descrito o simulador utilizado neste trabalho assim como os controladores que são necessários aplicar na simulação. Além disso, também está descrito o modelo de simulação elaborado neste trabalho, nomeadamente os diferentes passos a que lhe estão associados para o funcionamento do robô.

O Capítulo 6 apresenta os resultados obtidos com a simulação, em particular a sua operação. Por fim o Capítulo 7 apresenta as conclusões retiradas deste trabalho.

Capítulo 2- Conceitos Básicos de Radiação e Radioproteção

Neste capítulo serão abordados alguns princípios físicos importantes para a compreensão da importância da implementação de um sistema automático (uso de um robô) para uma menor exposição a radiação por parte dos profissionais de saúde. Os princípios abordados dizem respeito ao campo de radiação, isto é dose absorvidas, doses equivalentes, entre outras, e o impacto que a radiação tem sobre os indivíduos expostos.

2.1 Radiação e Métodos de Medição do Campo de Radiação

Como já foi mencionado no Capítulo 1 existem dois tipos de radiação associados ao tratamento com ^{131}I, são elas a radiação γ e β. A radiação β é caracterizada através da conversão de um protão num neutrão ou vice-versa. Quando se dá o decaimento através da emissão de β^+, um protão no núcleo é convertido num neutrão, seguindo a emissão de uma partícula β^+ (positrão) e um neutrino[2].

O decaimento através da emissão de partículas β^- (eletrão) dá-se pela conversão de um neutrão no núcleo em um protão.com emissão de uma partícula β^- e de um antineutrino[3]. O protão vai permanecer no núcleo enquanto que o eletrão e o antineutrino são emitidos do núcleo.

[2] O neutrino corresponde a uma partícula subatómica sem carga elétrica que resulta do decaimento radioativo de partículas.
[3] Os antineutrinos correspondem às antipartículas dos neutrinos. Este são partículas subatómicas sem carga elétrica produzidas através do decaimento negativo de partículas radiativas.

A radiação γ (gama) é caracterizada por um núcleo instável com excesso de energia e a conversão para um estado mais estável. O excesso de energia apresentado é emitido através dos raios gama, sob a forma de fotões.

Este tipo de raios são extremamente penetrantes, e para que seja possível atenuar ou absorvê-los, é necessário utilizar materiais de elevada densidade [5].

Estes dois tipos de radiação encontram-se presentes no campo de radiação. O campo de radiação refere-se a todas as partículas e as suas trajetórias numa determinada região no espaço ou através de algum limite específico [4]

Uma maneira de ser possível caracterizar o campo de radiação é através da radiometria ou dosimetria.

2.1.1 Radiometria

A radiometria diz respeito ao estudo de transferências de energia radioativa. Existem dois conceitos importantes neste tipo de estudo, são eles o fluxo e o rácio do fluxo.

O fluxo,Φ, é definido pelo quociente do número de partículas, *dN*, e a secção da área de penetração, *dA*. A equação 1 representa o fluxo:

$$\Phi = \frac{dN}{dA} \tag{1}$$

O rácio do fluxo,ϕ, é dado pelo quociente do número de partículas presentes num determinado espaço, por unidade de tempo. A equação 2 representa o rácio do fluxo:

$$\phi = \frac{d\Phi}{dt} \tag{2}$$

2.1.2 Dosimetria

A dosimetria permite determinar a dose de radiação num ponto, ou recebida por um individuo quando exposto a fontes radioativas, nomeadamente o iodo. Existem diferentes maneiras de determinar a dose de radiação, entre elas a dose absorvida, a dose equivalente e a dose efetiva.

A dose absorvida, *D*, é definida pela energia absorvida por unidade de massa em gray (G). Na equação 3 encontra-se representado o cálculo da dose absorvida:

$$D = \frac{dE}{dm} \tag{3}$$

Onde *dE* corresponde à média de energia fornecida à matéria através de radiação ionizante num elemento de volume e *dm* é a massa da matéria no elemento de volume. Este conceito é bastante utilizado, pois a quantidade de radiação presente na massa dos tecidos pode facilmente estar correlacionada com o perigo de contaminação. Porém, a dose absorvida não fornece informação sobre a massa total de tecido que esta exposto e a distribuição da energia absorvida [4].

A dose equivalente, H_T, em Sv (sievert), corresponde à dose absorvida por um tecido ou órgão, ponderada pelo tipo de radiação *R*. Esta é definida pela equação 4:

$$H_T = w_R \, D_{T,R} \tag{4}$$

Onde $D_{T,R}$, diz respeito à média da dose absorvida, por um órgão ou tecido, *T*, e w_R diz respeito ao fator de ponderação de radiação. Para os fotões (γ) e os eletrões (β^-) emitidos, o fator de ponderação de radiação (w_R) é de 1, considerando o tratamento com ^{131}I, o que leva à conclusão de que a dose equivalente é igual à dose absorvida por um tecido ou órgão.

Quando o campo de radiação é composto por diferentes valores de w_R, a dose equivalente é dada pela soma da doses equivalentes individuais [4].

A dose efetiva (E_D), em Sv (sievert), corresponde ao somatório da dose equivalente (H_T) de cada órgão ou tecido e o respetivo fator de ponderação tecidular w_T , esta é dada pela equação 5:

$$E_D = \sum_T w_T \, H_T \tag{5}$$

O cálculo da dose efetiva permite determinar o risco total a que o corpo está sujeito quando se encontra em contacto com a radiação. Na Tabela 1 estão presentes o valor de diferentes fatores de ponderação para outros órgãos e tecidos.

Tabela 1- Fator de ponderação, $\boldsymbol{w_T}$, dos tecidos ou órgãos [6].

w_T	Órgão ou Tecido
0.12	Peito, medula óssea, cólon, pulmões, estômago
0.08	Gónadas
0.12	Tecidos Diversos [4]
0.04	Bexiga, fígado, esófago. Tiroide
0.01	Superfície óssea, cérebro, glândulas salivares, pele

2.2 Efeitos da Exposição a Radiação Ionizante

Todos os tipos de radiação, tanto os que são benéficos para a saúde assim como os que a prejudicam, resultam da capacidade da radiação poder ionizar a matéria dos tecidos por onde atravessa. Este tipo de interações com os tecidos biológicos levam à libertação de energia na matéria viva, podendo causar reações adversas quando estes passam determinados limites.

Quando a exposição a radiação é elevado, ou o individuo em causa encontra-se exposto à radiação num período elevado de tempo, pode apresentar dois tipos de efeitos biológicos sendo eles determinísticos ou estocásticos.

[4] O valor de ω_T para os tecidos diversos aplica-se à média aritmética das doses dos 13 órgãos seguidamente enumerados: tecido suprarrenal, região extratorácica, vesícula biliar, coração, rins, gânglios linfáticos, músculo, mucosa bucal, pâncreas, próstata (para o sexo masculino), intestino delgado, baço, timo, útero/colo do útero (para o sexo feminino).

Os efeitos determinísticos ocorrem quando o número de células mortas ultrapassa a capacidade de renovação das mesmas, isto é, quando o estado de equilíbrio entre produção e morte é perturbado. Porém existe um limite em que os sintomas não se encontram presentes. No caso especifico do tratamento do cancro na tiroide, as células cancerígenas serão destruídas através da radiação emitida pelo iodo não afetando outros órgãos ao ponto em que os efeitos determinísticos podem ser visíveis [2] [4].

Por sua vez os efeitos estocásticos resultam de alterações nas células que perdem a sua capacidade de se dividir, alterando assim de forma aleatória a estrutura do DNA. Estas alterações iniciam uma transformação maligna que pode levar ao aparecimento de cancro. Para este tipo de efeito não existe um limite, logo a probabilidade de aparecimento dos efeitos secundários aumenta com o aumento da dose de radiação a que está exposto [4].

2.3 Radioprotecção

Uma vez conhecidos os efeitos que a radiação tem sobre os seres humanos, é necessário a implementação de medidas que previnem o aparecimento de efeitos secundários. Os principais objetivos destas medidas são diminuir a ocorrência de efeitos determinísticos e limitar a ocorrência de efeitos estocásticos.

Segundo o artigo 5º da Diretiva 2013/59/EURATOM do Conselho da União Europeia, foram estabelecidos requisitos legais e um regime de controlo regulador que refletem um sistema de proteção contra radiações baseado nos princípios de justificação, otimização e de limitação da dose. Estes parâmetros são caracterizados por:

- Justificação: corresponde a qualquer decisão que altere a situação de exposição à radiação, que deve fazer mais bem que mal;
- Otimização: corresponde à probabilidade de ocorrerem exposições, o número de indivíduos expostos e a magnitude das doses individuais, que devem ser mantidos o mais baixo possível. Este princípio é conhecido por conceito de ALARA (*As Low As Reasonably Achivable*);

- Limitação da dose: diz respeito a situações de exposição planeadas, a soma das doses administradas a um individuo que não podem ultrapassar os limites de dose estabelecidos para a exposição profissional.

2.4 Exposição de profissionais à radiação

O Decreto-Lei nº 222/2008, de 17 de novembro, abrange todo o tipo de situações de exposição, por parte dos membros do público às radiações ionizantes de origem artificial, assim como os profissionais expostos e aprendizes/estudantes, sendo estabelecidos uma série de critérios específicos para a proteção dos mesmos.

Quando um profissional se encontra exposto a doses superiores, é necessário monitorizar e vigiar estes trabalhadores. Existem duas categorias que definem os trabalhadores expostos, são elas a categoria A e B.

A categoria A diz respeito aos trabalhadores que se encontrem expostos a doses efetivas superiores a 6 mSv por ano, ou uma dose equivalente superior a três décimas de um dos limites anuais de exposição.

Por sua vez, a categoria B corresponde a todos os trabalhadores que não sejam classificados na categoria A.

Para a proteção contra radiações é necessário tomarem-se algumas medidas relativamente aos locais onde a exposição a radiação ionizante. Daí que são considerados duas zonas, são elas a zona vigiada e controlada.

A zona vigiada corresponde à área que diz respeito a salas de espera e de aquisição de dados.

A zona controlada diz respeito a áreas como a radiofármacia, as zonas de armazenamento de materiais ou lixos radioativos e a sala de administração de radiofármacos [7].

Além destes parâmetros que definem a exposição de um profissional de saúde à radiação, também é necessário ter em conta a dose limite a que este pode estar sujeito dependendo das áreas anatómicas que estão expostas. Pois o limite vai sendo alterado

conforme o coeficiente de absorção dos órgãos expostos. Na Tabela 2 encontram-se representados a dose limite para os trabalhadores exposto segundo a 9º Diretiva 2013/50 EURATOM do Conselho da União Europeia [6].

Tabela 2- Limites de dose por ano para os profissionais segundo 9º Diretiva 2013/59/Euratom do Conselho da União Europeia [6].

Tecido ou órgão	Dose Equivalente por ano
Cristalino	20mSv
Pele	500mSv
Extremidades	500mSv
Dose Efetiva por ano	
Corpo Inteiro	20mSv

2.5 Medidas preventivas contra Radiações Ionizantes

Para todos os profissionais que se encontrem expostos a radiações ionizantes, é necessário minimizar os efeitos desta exposição. Logo é necessário obter-se uma proteção adequada combinando três variáveis: a distância dos profissionais à fonte de radiação, o tempo de exposição e as barreiras de proteção (blindagem).

O aumento de distância é uma forma simples e eficaz de diminuir a exposição, uma vez que quanto maior for a distância à fonte de radiação menor será a intensidade do feixe. Para se minimizar a dose nas mãos e no resto do corpo, são utilizadas pinças para manusear as fontes radioativas.

No que diz respeito ao tempo de exposição, nem sempre é possível diminuir, pois estas tarefas demoram algum tempo a serem realizadas. Daí que a única forma de diminuir o tempo de exposição é ter um plano específico do que é necessário fazer e tentar efetuar esse trabalho o mais rápido possível [8] [9].

As barreiras de proteção (blindagem) dependem do tipo de radiação com que se esta a lidar. Neste caso em específico a radiação ionizante que se encontra presente advém dos

electrões (β^-) e os fotões (γ). A blindagem consiste na sobreposição de materiais distintos entre a radiação e o alvo a proteger.

Na Figura 2 encontram-se representados os diferentes tipos de blindagem utilizados dependendo do tipo de radiação ionizante.

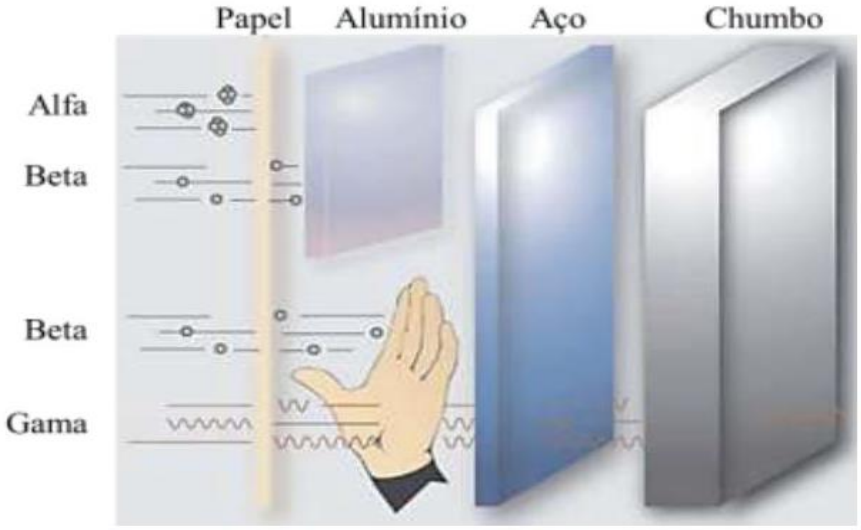


Figura 2- Diferentes tipos de blindagem associados aos diferentes tipos de radiação ionizante [10].

O uso da blindagem proporciona uma maneira mais confiável de limitar a exposição pessoal diminuindo o débito de dose. Em teoria, esta deve ser usada para reduzir os débitos de dose para níveis desejados, porém na prática a quantidade de blindagem utilizada depende do equilíbrio do custo e benefícios esperados [8].

Existem também outras medidas e equipamentos que ajudam na diminuição dos riscos de exposição, tais como aventais de chumbo, luvas, óculos com chumbo e protetores de tiroide. Também é necessário blindar os frascos e seringas que contém os elementos radioativos, pois estes são objetos-alvo.

Além destas medidas também é necessário que os profissionais de saúde tomem certas atitudes referentes à sua proteção individual.

Estes utilizam um dosímetro de corpo inteiro, usados no tronco, para permitir a leitura da dose de radiação. Porém, se existir uma maior exposição nas mãos, é necessário recorrer-se a um dosímetro de extremidades. Assim é possível avaliar os níveis de dose

acumulada por cada profissional, permitindo uma comparação com os limites previstos pela legislação [11].

2.6 Dosímetro AT1121, AT1123

O dosímetro utilizado nas salas de terapia de iodo é o dosímetro AT1121 da ATOMTEX, representado na Figura 3. A principal função do dosímetro é medir a dose equivalente e energia, a curto e longo prazo, da radiação gama e X.

Além da função mencionada atrás também é possível a deteção de fontes de radiação gama, fontes de radiação beta, deteção de fontes de radiação móveis, entre outras. Este tipo de dosímetro guarda automaticamente o valor máximo obtido pela dose durante o tempo de operação, assim como é capaz de armazenar até 999 resultados de medições na memória do aparelho por um longo período de tempo, os quais serão posteriormente transferidos para o computador [12].

Figura 3- Dosímetro AT1121, AT1123 utilizado nas salas de terapia de iodo [12].

Capítulo 3- Revisão Bibliográfica

Neste Capítulo será realizada uma breve revisão sobre o estado da arte relativamente aos métodos de localização utilizados na robótica, assim como os diferentes tipos de robôs existentes. Também serão abordados os diferentes tipos de configurações móveis que existem, nomeadamente a configuração de triciclo, diferencial e omnidirecional.

3.1 Evolução do uso de sistemas Automatizados

A criação de robôs móveis autónomos veio facilitar a realização de diversas tarefas do quotidiano, assim como tarefas relacionadas com a automação industrial, a assistência em cirurgias, diversos procedimentos médicos, entre outros [13]. Porém existe uma grande variedade de sistemas robóticos a serem utilizados o que leva a ser necessário fazer a seleção do tipo de algoritmos e *hardware* a se utilizar nas diferentes situações.

Como já foi mencionado, existem muitos robôs que utilizam diferentes algoritmos e *hardware* para que seja possível se localizarem e identificarem obstáculos em diversos ambientes. Por exemplo, no caso dos algoritmos utilizados para a localização do robô têm-se o *Perfect Match*, o *Simultaneous Localisation and Mapping algorithms* (SLAM), descrito em [14], o *Iteractive Closest Point algorithms*, entre outros.

O algoritmo *Iterative Closet Point* (ICP) é um algoritmo no qual são efetuadas observações consecutivas que permitem a correspondência ponto a ponto e posterior análise da contribuição de cada ponto, obtido pelo laser, para a função custo [13] [15].

O *Perfect Match*, descrito em. [16], permite a determinação da posição inicial do robô de forma rápida, sem que seja necessário se ter conhecimento prévio do espaço onde se encontra ou estados anteriores do robô. O método de auto-localização encontra-se dividido em duas fases. A primeira consiste na determinação da posição inicial do

veículo onde esta é estimada automaticamente no referencial de coordenadas global [13].

A estimativa efetuada para determinar a posição inicial é usada para a seguinte fase do algoritmo que se denomina de *position tracking*. Nesta fase os dados obtidos para determinar a posição do robô já se encontram calculados e encontram-se no referencial de coordenadas global. Logo os dados odométricos do robô e os dados obtidos sobre o ambiente são utilizados para a aplicação do algoritmo de *Perfect Match* [13].

Para além destes algoritmos também são utilizados sensores, lasers e câmaras, a nível do *hardware*, que permitem com que os robôs se localizem e orientem num espaço. Os robôs mencionados abaixo são exemplos do uso desses instrumentos.

Um exemplo de *hardware* utilizado é o robô PR2, o qual se pode ver na Figura 4, sendo este um robô com quatro rodas omnidirecionais, equipado com 2 lasers e uma câmara [17]. Os dois lasers utilizados por este robô são lasers planares LRF (*Laser Range Finders*) com os quais se obtém um mapa do ambiente envolvente [13] [18].

Figura 4- Robô PR2 [19].

Outros robôs utilizados no quotidiano são o Roomba, que se encontra representado na Figura 5, que auxilia nas tarefas domésticas de forma automática

Figura 5- Robô Roomba.

O Rovio, representado na Figura 6, é um robô omnidirecional que supervisiona o interior de casas [13]. Estes dois robôs são exemplos do uso do *Autonomous Guided Vehicles* (AGVs) [15].

Figura 6- Robô Rovio.

Além dos métodos de localização mencionados atrás, existem outros métodos que se encontram divididos em dois grupos sendo estes denominados de localização absoluta e relativa, respetivamente [13].

Na localização relativa são utilizados sistemas odométricos e sistemas de navegação de inércia [13].

A odometria é o método mais comum usado para a localização do robô. Com este método obtém-se uma boa exatidão a curto prazo, é um método barato e permite a obtenção de taxas de amostragem elevadas. Porém este método consiste na integração incremental de informação do movimento do robô ao longo do tempo o que leva a uma acumulação de erros. Apesar destas limitações este método é importante para o sistema de navegação do robô [15].

O sistema de navegação de inércia utiliza giroscópio e acelerómetros para medir as taxas de rotação e aceleração, respetivamente. Este sistema tem como vantagem o facto de não necessitarem de referências externas, porém não é apropriado para se obter uma boa exatidão da posição do robô ao longo de um período de tempo elevado [15].

Na localização absoluta são utilizados compassos magnéticos, sistemas de navegação com pontos ativos, sistemas de posição global (GPS), marcadores de navegação e posicionamento baseado num mapa [15]. Para um maior esclarecimento sobre as técnicas e métodos utilizados é possível ver em. [15].

Para além dos diferentes algoritmos e *hardware* utilizados, existem também diferentes sistemas configurações móveis para o robô. Entre eles a configuração de triciclo, a diferencial e omnidirecional.

A configuração de triciclo, representada na Figura 7, é uma não holomónica, isto significa que este tipo de configuração não permite com que o robô mude de direção de forma instantânea. Este sistema de locomoção apresenta duas rodas convencionais fixas sobre um eixo e uma roda convencional centrada que tem funções de tração e orientação.

Esta configuração tem como vantagem o facto de a cinemática ser relativamente simples, tornando-o adequado para pesquisas em diversas áreas. Porém em algumas situações o centro de gravidade do robô localiza-se nos limites da superfície de equilíbrio definido pelas três rodas, o que pode causar uma perda de tração e um erro quando se tem de determinar a posição do robô [20].

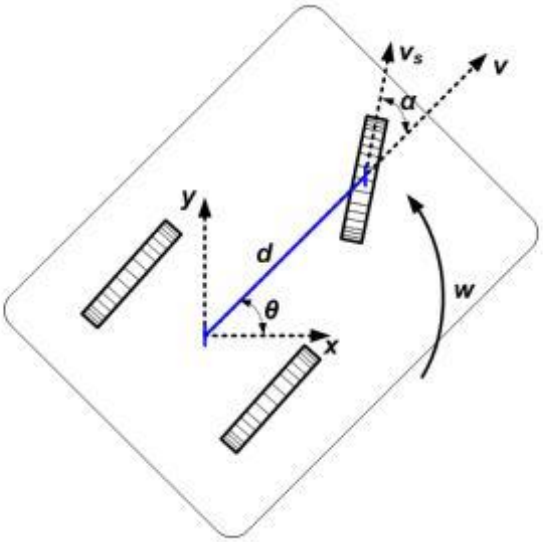


Figura 7- Configuração de Triciclo.

Por sua vez, a configuração diferencial tem como características a existência de duas rodas independentes, as quais se podem visualizar na Figura 8, um terceiro apoio para manter a sua estabilidade e este tipo de robô pode rodar sobre si.

Porém este tipo de configuração não permite efetuar os movimentos de translação segundo o eixo que passa pelos veios do motor e não é holonómico, tal como a configuração de triciclo [21].

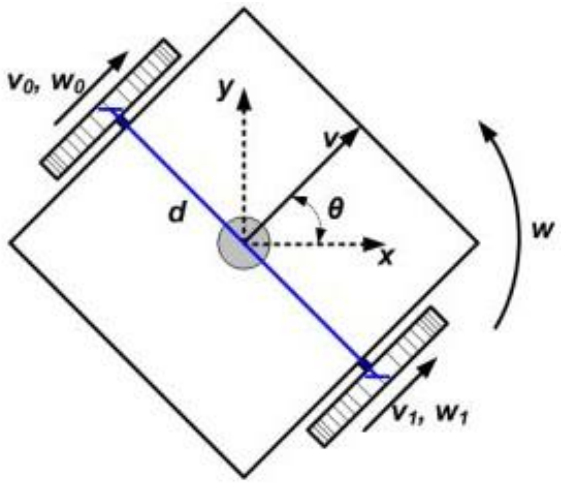


Figura 8- Configuração Diferencial.

No que diz respeito à configuração omnidirecional, esta tem por base o facto do robô se poder movimentar em qualquer direção sem que seja necessário qualquer rotação previamente definida, isto significa que este robô é holonómico.

Este tipo de robô é capaz de realizar com facilidade tarefas em ambientes onde existem obstáculos estáticos e dinâmicos, pois de acordo com a rotação de cada umas das rodas, o robô pode andar em linha reta, girar ou deslocar-se lateralmente sem haver a necessidade de mudar a sua direção.

Estas características apresentam uma grande vantagem em relação às configurações anteriores, uma vez que este tipo de robô pode ser aplicado em salas de tratamento onde existem obstáculos estáticos (macas, poltrona etc.) e dinâmicos nomeadamente a pessoa que recebe o tratamento.

Para se obter um robô omnidirecional basta que este apresente apenas três rodas, como o representado na Figura 9 do Capítulo 4 secção 4.1. Este tipo de configuração é cada vez mais utilizado devido às características que lhes estão associadas.

Capítulo 4- Características de Robôs Omnidirecionais e Algoritmo de localização

Neste capítulo será efetuado uma breve caracterização dos robôs omnidirecionais de 3 rodas, assim como a modelação destes. Além desta caracterização será explicado o algoritmo de localização *Perfect Match*. Na parte da modelação serão descritos os diferentes modelos, tais como o modelo dinâmico e cinemático, os quais são necessários para o funcionamento do sistema. Em relação ao algoritmo, este encontra-se dividido em diversas partes sendo elas a minimização do erro, o *Resilient Back-Propagation* e por fim a implementação da segunda derivada da estimativa do erro.

4.1 Caracterização e Modelação de Robôs Omnidirecionais de 3 Rodas

O modelo de robô utilizado neste trabalho é o omnidirecional de três rodas, equipado com um motor DC e sensores de distância [22]

O uso de robôs omnidirecionais permite o seu movimento em todas as direções, sendo este tipo de movimento uma grande vantagem em relação a outro tipo de modelos. O facto de o robô se movimentar de um local até outro com independência linear e velocidades angulares permite minimizar o tempo de reação, o número de manobras diminui o que leva a um processo mais simples [22].

O robô em estudo, representado na Figura 9, apresenta um robô com três rodas, o sentido do eixo, as forças e velocidades relevantes para o sistema robótico [22] [21].

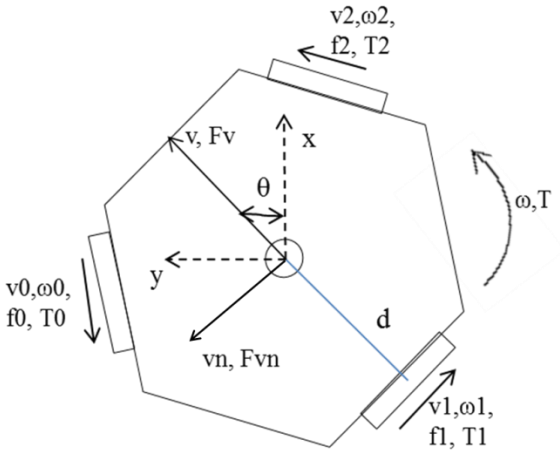


Figura 9- Robô de três rodas.

Na Figura 9 também estão representados o tipo de denotação utilizado ao longo deste trabalho:

- x,y, θ- posição do robô (x, y) e θ o ângulo;
- d- distância entre o ponto central e as rodas;
- $\upsilon_0, \upsilon_1, \upsilon_2$ (m/s) - velocidade linear das rodas;
- $\omega_0, \omega_1, \omega_2$ (rad/s)- velocidade angular das rodas;
- F_0, F_1, F_2 (N)- força de tração das rodas;
- T_0, T_1, T_2 (N.m)- binário da tração das rodas;
- υ, υn (m/s)- velocidade linear do robô;
- ω (rad/s)- velocidade angular do robô;
- Fυ, Fυn (N)- força de tração no robô segundo as direções υ e υn;
- T (N.m)- binário de rotação no robô segundo o seu eixo de rotação.

4.2 Modelos

É essencial aumentar o desempenho dinâmico de robôs para que seja possível obter um maior controlo com o objetivo de aumentar a performance do robô, para isso foram desenvolvidos modelos dinâmicos e cinemáticos [22] [21].

Estes modelos são baseados num sistema linear e não-linear, onde a determinação dos parâmetros tem sido alvo de contínuo estudo [23] [24] [25]. A construção do modelo dinâmico permite o estudo das forças envolvidas. Já o modelo cinemático permite definir as velocidades do robô segundo as posições onde este se encontra.

4.2.1 Modelo Cinemático

Na robótica móvel, a cinemática é aplicada ao movimento dos robôs ignorando as forças e as massas, considerando apenas velocidades e posições. A cinemática representa as relações geométricas que regem o sistema físico [21].

Sabendo a posição do robô (x, y, θ) é possível calcular as velocidades ($\upsilon, \upsilon_n, \omega$) através da derivada dada pelas equações 1, 2, 3.

$$\upsilon x(t) = \frac{dx(t)}{dt} \tag{1}$$

$$\upsilon y(t) = \frac{dy(t)}{dt} \tag{2}$$

$$\omega(t) = \frac{d\omega(t)}{dt} \tag{3}$$

Através das velocidades lineares υ_x e υ_y é possível calcular as velocidades lineares υ e υn pela equação da cinemática (equação 4).

$$\begin{bmatrix} \upsilon(t) \\ \upsilon_n(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \upsilon_x(t) \\ \upsilon_y(t) \\ \omega(t) \end{bmatrix} \tag{4}$$

Utilizando a relação que existe entre a velocidade das rodas e a velocidade do robô, é possível determinar as velocidades lineares das rodas (equação 5) [22].

$$\begin{bmatrix} v_0(t) \\ v_1(t) \\ v_2(t) \end{bmatrix} = \begin{bmatrix} -\sin(\pi/3) & \cos(\pi/3) & d \\ 0 & -1 & d \\ \sin(\pi/3) & \cos(\pi/3) & d \end{bmatrix} . \begin{bmatrix} v(t) \\ vn(t) \\ \omega(t) \end{bmatrix} \quad (5)$$

Aplicando na equação 5 a cinemática inversa, é possível obter as seguintes equações que representam as velocidades linear e angular do robô.

$$v(t) = \frac{\sqrt{3}}{3} . (v_2(t) - v_0(t)) \quad (6)$$

$$v_n(t) = \frac{1}{3} . (v_2(t) + v_0(t) - \frac{2}{3} . v1(t) \quad (7)$$

$$\omega(t) = \frac{1}{3 \cdot d} \cdot (v_0(t) + v_1(t) + v_2(t) \quad (8)$$

4.2.2 Modelo Dinâmico

A dinâmica do robô representa o estudo do movimento relacionado com as formas presentes no sistema físico incluindo a energia e as velocidades [21]. As equações 9, 10 e 11 representam as equações dinâmicas para o sistema de eixos presente na Figura 9.

$$M \cdot \frac{dv(t)}{dt} = \sum Fv(t) - FBv(t) - FCv(t) \quad (9)$$

$$M \cdot \frac{dvn(t)}{dt} = \sum Fvn(t) - FBvn(t) - FCvn(t) \quad (10)$$

$$J \cdot \frac{d\omega(t)}{dt} = \sum T(t) - TB\omega(t) - FC\omega(t) \quad (11)$$

Onde:

- M (Kg)- massa do robô;
- J (Kg.m^2)- momento de inércia do robô;
- $F_{Bυ}$, $F_{Bυn}$ (N)- força de atrito viscoso (dinâmico) segundo υ e υn;
- $T_{Bω}$ (N.m)- binário de atrito viscoso no robô segundo o eixo de rotação;
- $F_{Cυ,}$ $F_{Cυn}$ (N)- força de atrito de Coulomb no robô segundo υ e υn;
- $T_{Cω}$ (N.m)- binário de atrito de Coulomb segundo o eixo de rotação.

As equações 9, 10 e 11 representam o somatório das forças do robô segundo as direções υ, υ_n e o eixo de rotação ω. As forças que se encontram presentes no robô são designadas de atrito viscoso ou dinâmico e atrito de Coulomb ou estático [21].

As forças de atrito viscoso ($F_{B\upsilon}$, $F_{B\upsilon n}$, $T_{B\omega}$) apresentam uma relação linear entre as forças aplicadas e velocidade, enquanto que o atrito de Coulomb ($F_{C\upsilon}$, $F_{C\upsilon n}$, $T_{C\omega}$) apresenta uma amplitude constante [21].

As equações para as forças de atrito viscoso são:

$$FC\upsilon(t) = B\upsilon \cdot \upsilon(t) \quad (12)$$

$$FC\upsilon n(t) = B\upsilon n \cdot \upsilon n(t) \quad (13)$$

$$TB\omega(t) = B\omega \cdot \omega(t) \quad (14)$$

Onde, $B\upsilon$ *e* $B\upsilon n$ são os coeficientes de atrito viscoso para as direções υ *e* υ_n. $B\omega$ (N.m/rad.s) corresponde ao coeficiente de atrito viscoso para ω (este corresponde ao eixo de rotação do robô).

Por sua vez, as forças de atrito de Coulomb são representados por:

$$FC\upsilon(t) = C\upsilon \cdot sign(\upsilon(t)) \quad (15)$$

$$FC\upsilon n(t) = C\upsilon n \cdot sign(\upsilon n(t)) \quad (16)$$

$$TC\omega(t) = C\omega \cdot sign(\omega(t)) \quad (17)$$

Onde $C\upsilon$, $C\upsilon_n$ (N) correspondem ao coeficiente de atrito de Coulomb segundo υ *e* υ_n. $C\omega$ (N.m) corresponde ao coeficiente de atrito de Coulomb segundo o eixo de rotação do robô.

A relação que existe entre as forças de tração e binário de rotação do robô com as forças de tração nas rodas, é descrita por:

$$\sum F\upsilon(t) = (f_2(t) - f_0(t)) \cdot \sin\frac{\pi}{3} \quad (18)$$

$$\sum F\upsilon n(t) = -f_1(t) + (f_2(t) + f_0(t)) \cdot \cos\frac{\pi}{3} \quad (19)$$

$$\sum T(t) = \big(f_0(t) + f_1(t) + f_2(t)\big) \cdot d \quad (20)$$

A força de tração em cada roda é necessário calcular através de um binário de tração nas rodas que, por sua vez, é determinado usando a corrente consumido pelo motor, como é descrito nas equações:

$$f_j(t) = \frac{T_j}{r} \tag{21}$$

$$T_j(t) = l \cdot K_t \cdot i_j(t) \tag{22}$$

Onde,

- l- fator de restrição da caixa;
- r (m)- raio das rodas;
- K_t (N.m/A)- constante de binário dos motores;
- i_j (A)- corrente do motor.

Os modelos mencionados nesta secção, são utilizados para a construção da simulação do robô no *software SimTwo* (descrito no Capítulo 5). Todas estas equações já se encontram implementadas nesse *software*, sem que seja necessário a sua implementação manual.

4.3 Algoritmo de Localização Perfect Match

Esta secção descreve o algoritmo *Perfect Match,* encontra-se presente como referência para futuro trabalho.

Para que o robô seja autónomo, este necessita de se localizar num ambiente dinâmico, isto é possui a capacidade de se mover numa área ilimitada sem ter necessidade de qualquer tipo de preparação posterior [26].

Os sistemas de localização utilizam o algoritmo de *Matching* para que possa ser usado como um instrumento de medida que se encontra conjugado com os dados odométricos do veículo [27].

O algoritmo utilizado para efetuar o *Matching* é baseado no algoritmo computacional *Perfect Match* que será descrito mais à frente, porém para mais informações consultar [16].

Neste algoritmo a posição do veículo é obtida através de pontos 2D retirados do ambiente onde se encontra o robô. Estes pontos são adquiridos através de um laser, onde depois de obtidos os pontos estes são combinados com o mapa do ambiente onde se localiza o robô, que foi previamente calculado. Por sua vez, a posição do robô é obtida através do ajuste da minimização do erro entre os dados adquiridos e o mapa do edifício [27].

O *Perfect Match* assume valores anteriormente obtidos sobre a posição do robô e de seguida efetua os seguintes passos [28]:

- Cálculo do gradiente do erro;
- Processo de otimização utilizando o algoritmo *Resilient Back- Propagation (RPROP);*
- Fusão da estimativa da localização com parâmetros odométricos.

Os primeiros dois passos continuam a decorrer até que atinjam o número máximo de iterações, por sua vez o terceiro passo só acontece quando o valor do RPROP é obtido [28]

4.3.1 Minimização do Erro

O algoritmo de localização utiliza dados provenientes de diversos sensores ou lasers, os quais são utilizados na prática, onde o objetivo principal é fazer com que os dados obtidos coincidem com a informação que se encontra disponibilizada num mapa, assumindo, deste modo, a melhor posição e orientação [16]. Porém neste caso a posição onde o robô se encontra é obtida através do simulador.

Seja (p,ϕ) o par que define a posição $p=(p_x,p_y)$ e a orientação ϕ do robô no sistema de coordenadas global. A lista de linhas e pontos detetados é dada através da posição do robô e correspondendo ao vetor $s_1 \ldots s_n$, como se pode verificar na Figura 10 Esta

encontra-se expressa no referencial do robô, onde é possível verificar a posição, P, no sistema de coordenadas global, sendo dado pela seguinte expressão:

$$p + \begin{bmatrix} \cos \emptyset & -\sin \emptyset \\ \sin \emptyset & \cos \emptyset \end{bmatrix} \tag{34}$$

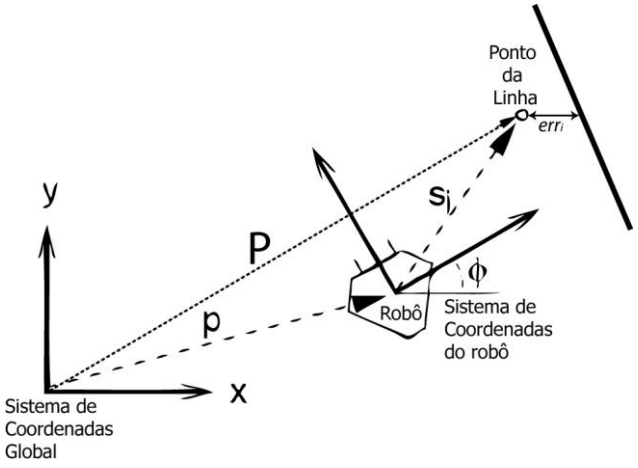


Figura 10- Esboço do ponto e linhas detetados, relativamente ao eixo de coordenadas do robô e ao eixo de coordenadas global [27].

Seja

$$d \cdot (p + \begin{bmatrix} \cos \emptyset & -\sin \emptyset \\ \sin \emptyset & \cos \emptyset \end{bmatrix} \cdot s_i) \tag{35}$$

A distância, uma função contínua e diferenciável, que pode ser obtida através do conhecimento das medidas do espaço onde o robô se irá movimentar, por exemplo a distância entre o robô e uma parede [16].

Para que seja possível estimar a posição e a orientação do robô é necessário calcular a função custo, pois quanto menor for o erro, melhor será a estimativa calculada com os pontos adquiridos pelo laser [29]. A equação utilizada para calcular o menor erro é a equação 36 apresentada abaixo:

$$minimize\ E = \sum_{i=1}^{n} err(d \cdot \left(p + \begin{bmatrix} \cos \emptyset & -\sin \emptyset \\ \sin \emptyset & \cos \emptyset \end{bmatrix} \cdot s_i\right)) \tag{36}$$

Devido ao facto de existir ruido inerente à imperfeição da informação do laser, que pode não ser exata, dependendo do laser utilizado, muitos pontos encontram-se mal detetados, o que vai influenciar e distorcer a estimativa que se está a calcular [29].

Por isso é utilizada uma aproximação da função custo que limita a influência dos pontos que têm demasiado erro. Logo a aproximação da função custo é dada pela equação 37:

$$err(\quad) = 1 - \frac{c^2}{c^2 + (d \cdot (p + \begin{bmatrix} \cos \emptyset & -\sin \emptyset \\ \sin \emptyset & \cos \emptyset \end{bmatrix} \cdot s_i))^2} \tag{37}$$

Esta função fica semelhante a uma função de erro quadrático, a qual se encontra limitada superiormente pela constante c, para erros elevados, pelo que a influência de pontos mal detetados é reduzida, como se pode verificar pela Figura 11.

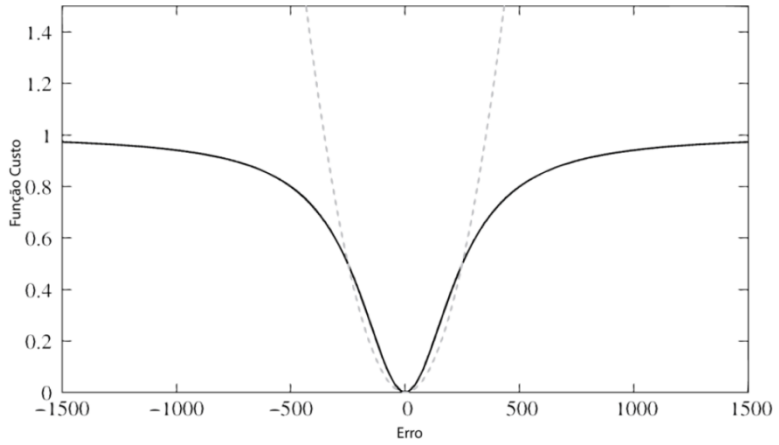


Figura 11-Comparação da função erro quadrático com a função mais robusta apresentada na equação 37 [29].

Devido à não linearidade da função de minimização dada pela equação 37, não é possível calcular a sua solução. Porém com o uso da equação 35, sendo esta diferenciável em praticamente todo o mapa, é possível calcular o gradiente na

globalidade do mapa, sendo assim necessário interpolar apenas uma pequena quantidade de pontos não diferenciáveis que se encontram nos limites do mapa [16]. Para resolver o problema da não linearidade da função de minimização, é possível recorrer ao método do gradiente descendente, isto significa que é utilizado o algoritmo de treino *Resilient Back Propagation* (RPROP), para o qual é necessário o gradiente da função custo [16]. Devido à rápida convergência do algoritmo e à sua elevada robustez, é possível resolver convenientemente a tarefa de minimização, com o uso de algumas iterações, obtendo-se desta forma uma convergência rápida para a melhor estimativa da posição e orientação do robô [29].

4.3.2 Processo de otimização utilizando o algoritmo Resilient Back- Propagation

O cálculo da função custo não estima a localização do robô, mas apenas fornece informação sobre o quão correto é a estimativa da sua localização atual. Para que seja possível calcular a localização do robô, é necessário otimizar a estimativa atual, ou seja, garantir a convergência da localização atual do robô para aquela onde o robô se encontra na realidade. Com base nos dados do laser, a localização atual do robô é aquela em que a função custo possui o erro mais baixo. Para estimar a localização do robô é utilizado um algoritmo RPROP [29].

Este algoritmo utiliza o anterior estado do robô e efetua a estimativa para a nova posição deste, a qual é utilizada na próxima iteração do RPROP [26].

O processo de funcionamento do algoritmo RPROP pode ser descrito da seguinte forma, durante um número limitado de iterações, os próximos passos são efetuados com a intenção de se estimar os valores de $x_{v,}$ y_v, $\theta_{v.:}$

1. Se a derivada atual de $\frac{\partial E}{\partial x_v}(t)$, $\frac{\partial E}{\partial y_v}(t)$ e $\frac{\partial E}{\partial \theta_v}(t)$, dependendo da variável, é diferente de zero, estes são comparados com as derivadas anteriores, $\frac{\partial E}{\partial x_v}(t-1)$, $\frac{\partial E}{\partial y_v}(t-1)$ e $\frac{\partial E}{\partial \theta_v}(t-1)$;

2. Se o produto $\frac{\partial E}{\partial x_v}(t) \cdot \frac{\partial E}{\partial x_v}(t-1)$ estiver a baixo de zero, significa que o algoritmo já passou o mínimo local, então a direção de convergência tem de ser invertida;
3. Se o produto de $\frac{\partial E}{\partial x_v}(t) \cdot \frac{\partial E}{\partial x_v}(t-1)$ for superior a zero, significa que o algoritmo continua a convergir para um mínimo local, e a direção de convergência deve ser mantida com o mesmo valor [26].

Para mais detalhes sobre o algoritmo RPROP, o artigo [26].

4.3.3 Fusão da estimativa da localização com parâmetros odométricos

A abordagem efetuada descreve a posição e orientação do robô que melhor se adapta à informação extraída do laser. Devido à existência de vibrações no robô e aos diversos erros introduzidos pelas medições dos sensores, devido ao aumento da velocidade do robô, a posição estimada é afetada por uma quantidade considerável de ruido e imprecisão [29] [16].

Para reduzir o ruído, é proposto avaliar a dependência temporal das posições estimadas através das informações obtida pelo laser. Uma vez que as posições subsequentes do robô são vizinhas e se encontram ligadas por alguma transição, dependendo da velocidade do robô, é possível utilizar uma média estocástica ponderada, que na realidade é uma aplicação simplificada do Filtro de Kalman (é possível ler mais sobre este filtro em [26] e [27]) [16].

4.3.3.1 Variância da odometria

A variância da estimativa efetuada para determinar a posição do robô, através da odometria, pode ser modelada pelo seu grau de incerteza.

Considerando (r_t, ψ_t), a estimativa da posição e orientação do robô no instante de tempo t e $\sigma^2_{r_x,t}$, $\sigma^2_{r_y,t}$ e $\sigma^2_{\psi,t}$ as respetivas variâncias, estas obtidas pelas equações 38, 39 e

40. Depois do robô se mover a estimativa da posição e as respetivas variâncias são atualizadas [16].

A velocidade υ e a velocidade de rotação ω são dadas pela odometria, isto é através do conhecimento da localização do robô, e a atualização das variâncias tem em conta a imprecisão do movimento:

$$\sigma^2_{\hat{\psi},t} = \sigma^2_{\hat{\psi},t-\tau} + \alpha\left(\hat{\psi}_t - \psi_{t-\tau}\right)^2 \quad (38)$$

$$\sigma^2_{\hat{r}_x,t} = \sigma^2_{r_x,t-\tau} + \alpha\left(\hat{r}_{x,t} - r_{x,t-\tau}\right)^2 \quad (39)$$

$$\sigma^2_{\hat{r}_y,t} = \sigma^2_{r_y,t-\tau} + \alpha\left(\hat{r}_{y,t} - r_{y,t-\tau}\right)^2 \quad (40)$$

Os parâmetros α devem ser maiores que 0, uma vez que estes traduzem a maior ou menor precisão do movimento. Nas equações 39 e 40, são ignorados os movimentos de rotação e translação para que seja possível manter os cálculos estatísticos mais simples [16].

Uma vez que os valores da variação da odometria são calculados relativamente ao deslocamento do robô, é necessário transportar esses valores para o eixo de coordenadas globais. Para isso efetua-se uma rotação das coordenadas, pelo ângulo que o robô apresenta nesse momento [29].

Para cada instante de tempo esta a ser considerado que a odometria do robô apenas possui um erro pequeno, no seu deslocamento, limitado pelo erro máximo admissível, em metros, isto significa que, em cada instante o robô só se desloca em um determinado número de metros [29].

4.3.3.2 Estimativa da variância para o algoritmo de localização

Depois de se processar a informação do laser e calculada a estimativa respetiva a essa informação, (p, ϕ), é possível determinar uma estimativa para a posição mais exata combinando (p, ϕ) e (r, ψ).

No entanto, é necessário calcular o valor das variâncias para (p,ϕ), que modelam a incerteza desta estimativa, que é calculada utilizando a estimativa obtida pelo algoritmo de localização [29].

A estimativa de localização obtida através do laser é influenciada por vários aspetos tais como, a vibração mecânica do robô, o erro de medida do laser, precisão do sistema óptico assim como a estrutura dos pontos medidos pelos sensores. A estimativa calculada para localização pode resultar com alguns parâmetros, porém pode não funcionar para outros, o que significa que a incerteza é diferente para os diferentes parâmetros p_x, p_y e ϕ [29].

Desta forma utiliza-se a função custo para determinar a variância de cada parâmetro acima mencionado. As equações 41,42 e 43 representam as variâncias a calcular.

$$\sigma_{p_{x,t}}^2 = \frac{K_x}{\frac{\partial^2 E}{(\partial_{p_x})^2}} \tag{41}$$

$$\sigma_{p_{y,t}}^2 = \frac{K_y}{\frac{\partial^2 E}{(\partial_{p_y})^2}} \tag{42}$$

$$\sigma_{\phi,t}^2 = \frac{K_\phi}{\frac{\partial^2 E}{(\partial_{p_\phi})^2}} \tag{43}$$

4.3.3.3 Fusão da estimativa da variância com a odometria

A fusão da estimativa é dada pela média pesada, supondo duas distribuições Gaussianas independentes, onde $\sigma_{p_{x,t}}^2$, $\sigma_{p_{y,t}}^2$ e $\sigma_{\phi,x}^2$ correspondem à variância de p_x, p_y e ϕ obtidos no algoritmo de localização (representada nas equações 41, 42 e 43) e $\sigma_{\hat{r}_{x,t}}^2$, $\sigma_{\hat{r}_{y,t}}^2$ e $\sigma_{\hat{\psi},t}^2$ corresponde à variância de $\hat{r}_{x,t}$, $\hat{r}_{y,t}$ e $\hat{\psi}_t$ obtidos através da odometria do robô (representadas nas equações 44, 45, 46, 47, 48 e 49).

$$r_{x,t} = \frac{\sigma_{p_{x,t}}^2 \cdot \hat{r}_t + \sigma_{\hat{r}_x,t}^2 \cdot p_{x,t}}{\sigma_{p_{x,t}}^2 + \sigma_{\hat{r}_x,t}^2} \quad (44)$$

$$\sigma_{r_x,t}^2 = \frac{\sigma_{p_{x,t}}^2 \cdot \sigma_{\hat{r}_x,t}^2}{\sigma_{p_{x,t}}^2 + \sigma_{\hat{r}_x,t}^2} \quad (45)$$

$$r_{y,t} = \frac{\sigma_{p_{y,t}}^2 \cdot \hat{r}_t + \sigma_{\hat{r}_y,t}^2 \cdot p_{y,t}}{\sigma_{p_{y,t}}^2 + \sigma_{\hat{r}_y,t}^2} \quad (46)$$

$$\sigma_{r_y,t}^2 = \frac{\sigma_{p_{y,t}}^2 \cdot \sigma_{\hat{r}_y,t}^2}{\sigma_{p_{y,t}}^2 + \sigma_{\hat{r}_y,t}^2} \quad (47)$$

$$\psi_t = \frac{\sigma_{\phi,t}^2 \cdot \hat{\psi}_t + \sigma_{\psi,t}^2 \cdot \phi_t}{\sigma_{\phi,t}^2 + \sigma_{\psi,t}^2} \quad (48)$$

$$\sigma_{\psi,t}^2 = \frac{\sigma_{\phi,t}^2 \cdot \sigma_{\psi,t}^2}{\sigma_{\phi}^2 + \sigma_{\psi,t}^2} \quad (49)$$

A filtragem efetuada pelo uso das estimativas ajuda tanto no aumento da robustez como no aumento da precisão. Desta forma as medidas erradas não levam o robô a perder-se uma vez que o filtro não permite que o robô salte para uma localização diferente, o que poderia acontecer quando só se utiliza o algoritmo de localização [29].

Capítulo 5- Introdução ao simulador e Criação da Simulação

Neste capítulo é apresentado o simulador utilizado, o tipo de controladores utilizado e a criação do ambiente e programação do robô. É explicado as diferentes janelas encontradas no simulador SimTwo, assim como as funções que cada uma tem dentro deste. Também são abordados os diferentes tipos de controladores que são utilizados na maioria dos sistemas implementados nos robôs, assim como a suas funções e apresentada de forma esquemática a maneira como funcionam dentro do sistema.

Por fim, será explicado todos os passos necessários para a criação da simulação. De entre estes temos a construção do ambiente onde se dá a simulação, o código com o qual é possível o funcionamento do robô, assim como o procedimento de controlo do mesmo.

5.1 Introdução ao SimTwo

A plataforma de simulação SimTwo [30]foi desenvolvida pelo Prof. Dr. Paulo Costa, onde é possível efetuar a simulação realista, especialmente a nível da dinâmica e de modelos de diversos tipos de robôs, tais como:

- Robôs móveis com diferentes configurações:
    - Diferenciais;
    - Com rodas omnidirecionais.
- Manipuladores;
- Quadrúpedes;
- Humanoides;

- Qualquer tipo de robô terrestre que possa ser descrito com uma mistura de articulações rotativas e rodas classicamente omnidirecionais;
- Veículos "mais leves que o ar" com ou sem hélices para propulsão [30].

Este simulador também permite a visualização 3D em tempo real da simulação, o que se torna útil para uma melhor supervisão do projeto. Como já foi mencionado este simulador já apresenta os diferentes modelos incorporados o que facilita a criação do robô uma vez que não é necessário implementar as equações referentes aos modelos dinâmico e cinemático, utilizados neste trabalho.

Além se encontrar incorporado estes modelos, o simulador também possibilita a obtenção de sinais de referência para esses controladores através de um controlador de alto nível implementado pelo utilizador. Este controlador pode utilizar uma linguagem *script* editável no próprio SimTwo ou usar um programa remoto via rede ou porta série [30].

O simulador SimTwo recorre a varias bibliotecas *Open Source*, tais como:

- *GLScene*- permite a visualização 3D;
- *ODE*- corresponde ao motor de simulação de corpos rígidos;
- *Pascal Script*- permite implementar um sistema de controlo programável;
- *SynEdit*- implementa o editor de *scripts*;
- *OmniXml*- facilita o recurso a ficheiros de configuração em formato XML;
- *RxLib*- corresponde a um conjunto variado de componentes [30].

5.2 Descrição Geral do Simulador

Este simulador é constituído por diferentes janelas, tais como a janela de visualização 3D do simulador, a de configuração, a do editor de código, a janela dos gráficos (*chart*) entre outras.

Na Figura 12 encontra-se representado a janela que permite a visualização em 3D do robô. Esta janela é utilizada para a supervisão da simulação, sendo os parâmetros da visualização definidos na janela de configuração, mais especificamente no separador *Graphics* representado na Figura 13 [30].

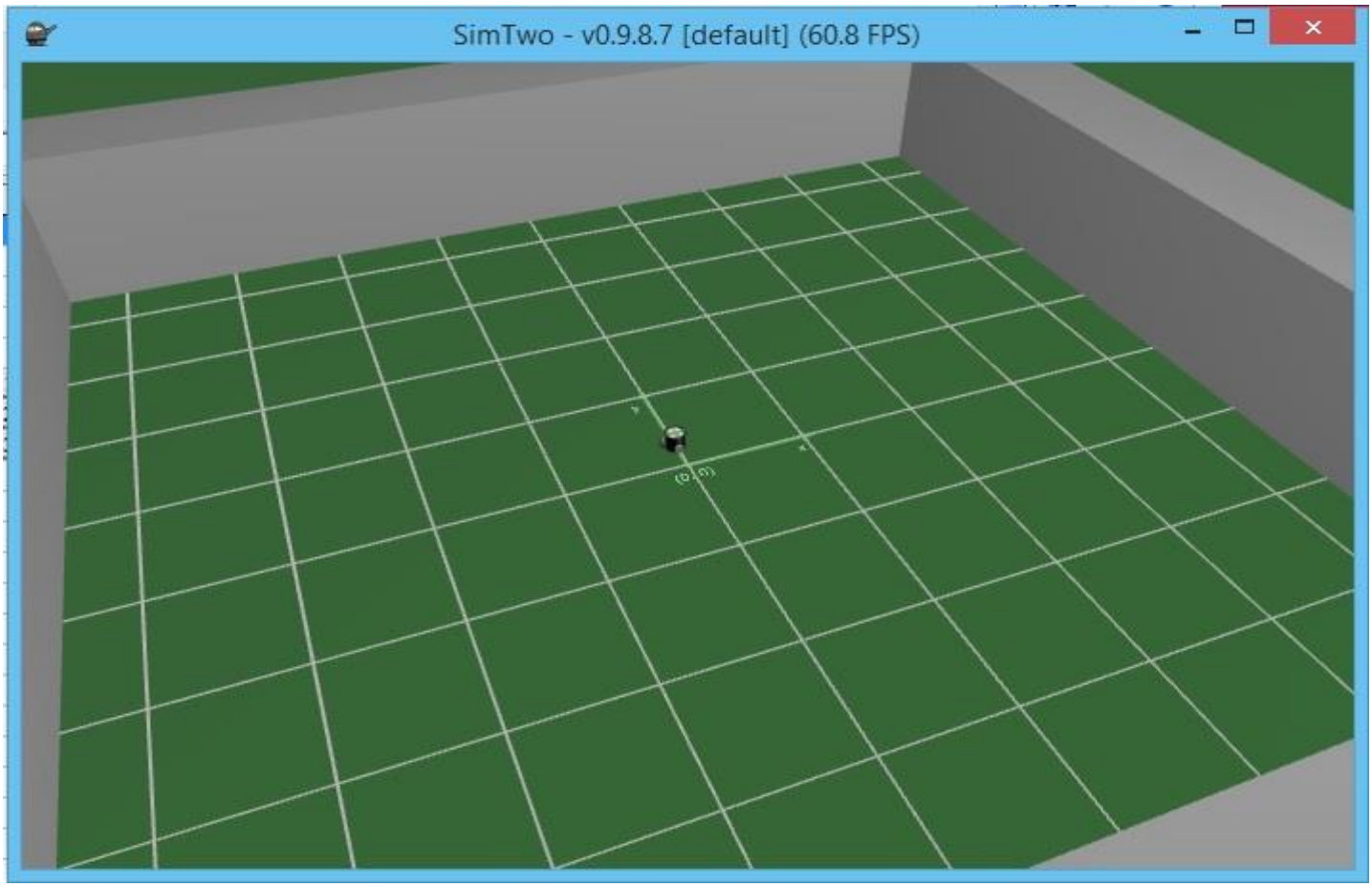


Figura 12- Janela de visualização 3D do SimTwo.

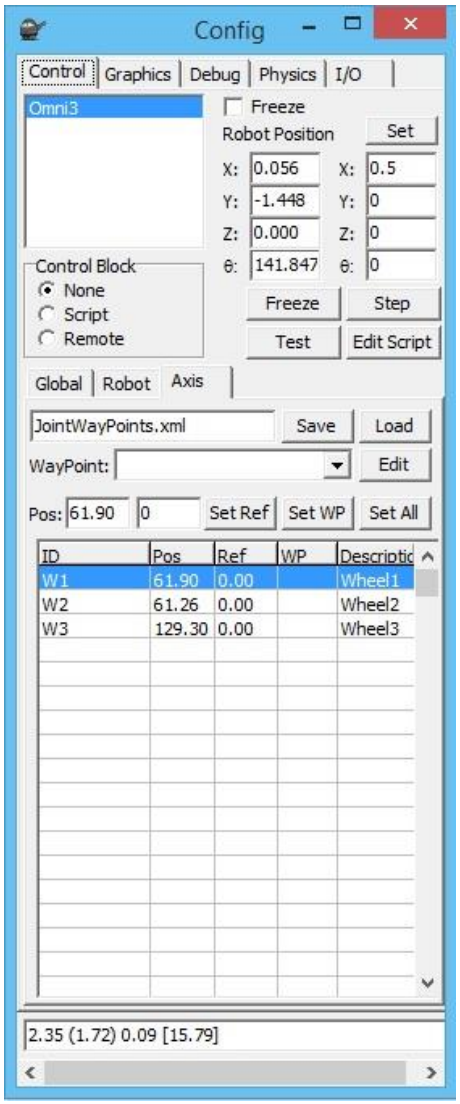


Figura 13- Janela dos parâmetros de configuração do SimTwo.

A nível do controlo do robô, o simulador possui uma solução completamente implementada, que corresponde a um interpretador de linguagem Pascal. Este encontra-se incluído no simulador com recurso à biblioteca *open source Pascal Script* onde é possível implementar o controlo dos robôs unicamente em *Pascal Script* ou utilizando o código como uma camada de interface com um controlador externo [30].

Os parâmetros de controlo do robô são configurados num dos separadores encontrados na janela de configuração (Figura 13) denominado de *Control*, assim como os interfaces referentes aos parâmetros de entrada e saída são configurados no separador I/O.

Este simulador também permite a visualização do comportamento das variáveis ao longo do tempo sendo possível a sua visualização na janela *Chart* (Figura 14).

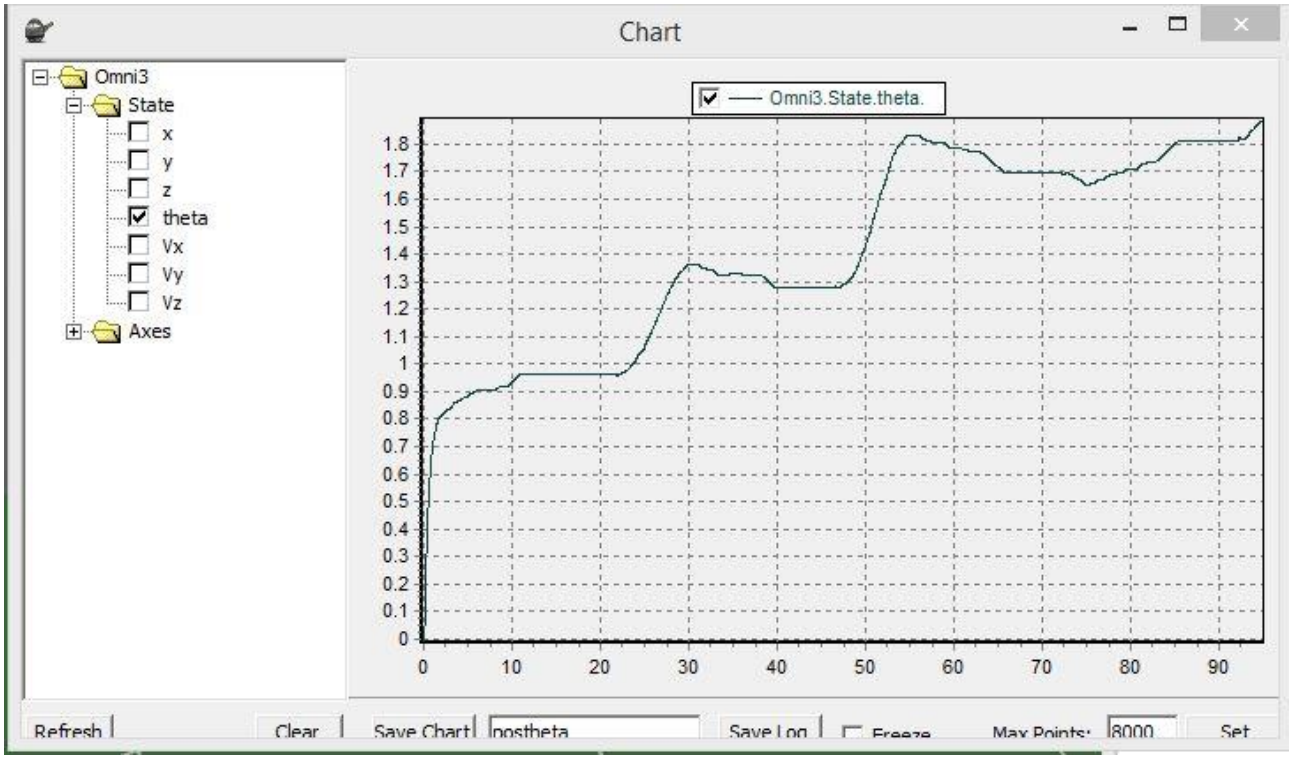


Figura 14- Janela referente à representação dos gráficos.

Além das janelas já mencionadas acima, o simulador também apresenta uma janela correspondente ao *Editor* (Figura 15), onde é possível efetuar diversas operações utilizando operações previamente definidas em código de *Pascal Script*.

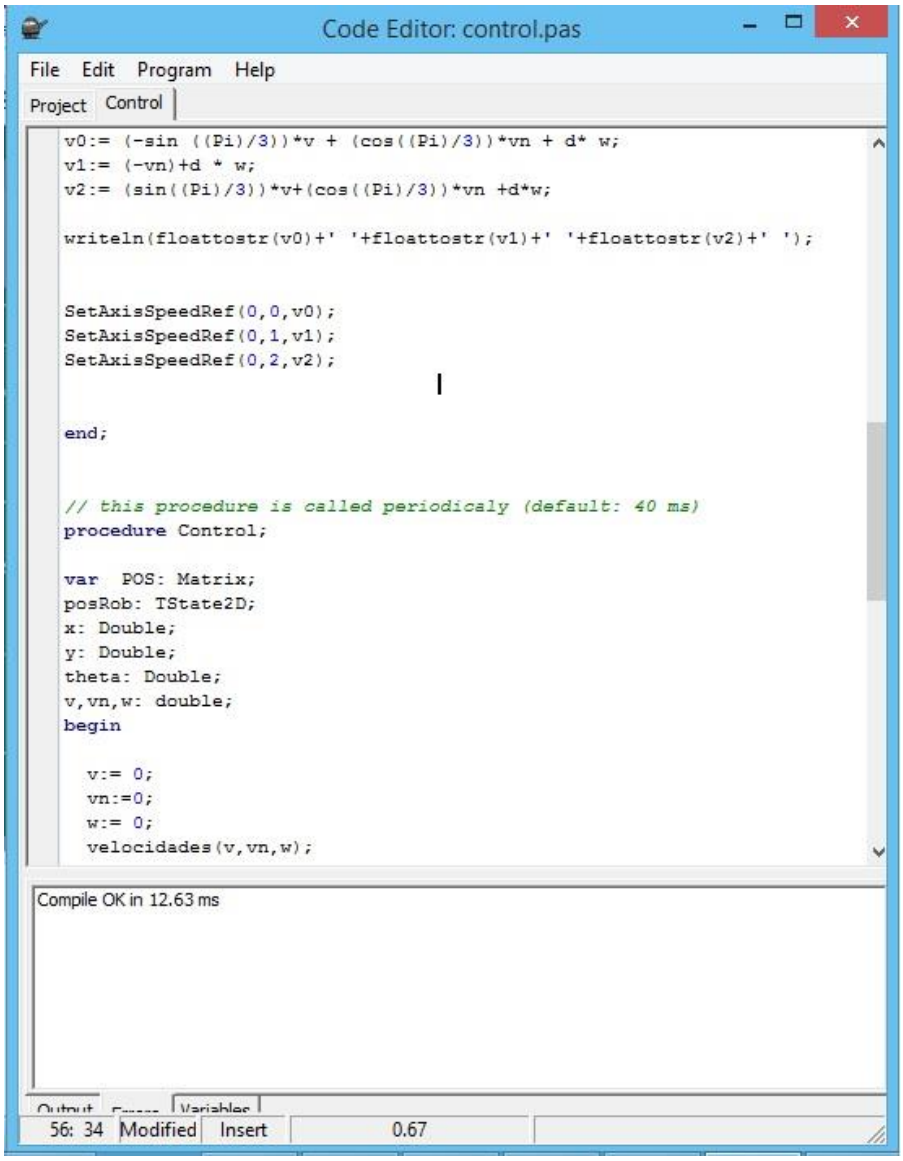


Figura 15- Janela referente ao editor de código.

5.3 Criação do Cenário

A criação de todo o Cenário no qual se encontra representado o ambiente onde se encontra o robô é feita através de ficheiros *Extensible Markup Language* ou *XML*. Este formato permite definir estruturas hierárquicas de informação, neste caso em concreto estruturas que tomam a forma de objetos colocados num mundo virtual e agrupados hierarquicamente num esquema de secções e subsecções [30].

5.3.1 Mundo Envolvente

No topo da hierarquia encontra-se a secção *scene*, contida num ficheiro *scene.xml*. Nesta secção encontram-se todas as secções utilizadas na definição do "mundo envolvente", como é denominado no ambiente virtual onde ocorre a simulação.

Dentro desta secção existem subsecções, que se dividem os elementos do "mundo envolvente" em classes. Estas secções encontram-se representadas na Tabela 3.

Tabela 3- Secções permitidas na secção *scene*.

Secções	Descrição
Robot	Robôs construídos a partir de objetos básicos.
Obstacles	Estruturas inamovíveis.
Things	Corpos livres que podem ser movidos.
Track	Permite desenhar uma pista ou outro tipo de marcadores no chão.

A secção correspondente ao robô representa as entidades controláveis na simulação. As secções *things* e *obstacles* correspondem a objetos não controláveis da simulação sendo efetuada a distinção entre aqueles que podem ser movidos, *things*, e aqueles que são inamovíveis, *obstacles* [30].

Toda esta informação pode ser incluída em apenas um ficheiro *scene.xml*, porém é mais fácil dividi-la em diferentes ficheiros. A semântica utilizada no simulador permite a definição perante o atributo *file*. Abaixo encontra-se representado um exemplo referente à criação do mundo envolvente:

```
<?xml version="1.0" ?>
<scene>
   <robot>
      <! -- Um Robô -- >
  </robot>
   <obstacles file='obstacles.xml'/>
  </scene>
```

Os ficheiros de *things* e de *obstacles* seguem uma estrutura semelhante.

5.3.2 Objetos

As estruturas virtuais são definidas recorrendo a alguns objetos básicos. Todos os objetos possuem propriedades básicas que permitem com que a plataforma de simulação do SimTwo construir estruturas compatíveis com a biblioteca de simulação de corpos rígidos ODE. Estes objetos básicos são utilizados nos ficheiros das *things, obstacles* e na construção do próprio robô.

Na Tabela 4 encontram-se alguns objetos básicos utilizados no simulador. Alguns destes objetos são meramente estruturais, como por exemplo o *cuboid*, porém existem outros, como por exemplo o sensor, que representam estruturas mais complexas.

Tabela 4- Objetos básicos utilizados no simulador SimTwo.

Tipo	Descrição
Cuboid	Trata-se de um paralelepípedo
Cylinder	Trata-se de um cilindro
Sphere	Trata-se de uma esfera
Belt	Trata-se de uma esteira rolante
Sensor	Trata-se de um sensor

Na Tabela 5 encontram-se descritas as propriedades necessárias para definir os diversos objetos.

Tabela 5- Propriedades básicas dos objetos do SimTwo.

Propriedade	Componentes	Descrição
ID	value	String identificadora do objeto.
Size	x, y, z	Tamanho em x, y, z em metros.
Pos	x, y, z	Posição do centro de massa em x, y, z em m.
Rot_Deg	x, y, z	Orientação para cada eixo x, y, z em graus.
Color_rgb	r, g, b	Cor com base no sistema rgb, onde a intensidade de cada componente R, G, B em 8 bits.
Mass	value	Massa do objeto em Kg.

5.3.3 Instâncias do Robô

Os robôs são as entidades controláveis no simulador, onde são permitidas diversas instâncias destas entidades, sendo que a cada uma corresponde uma secção do robô dentro da secção *scene* [30].

Cada instância de um robô possui determinadas propriedades apresentada na Tabela 6. A maioria destas propriedades é partilhada com objetos básicos. Porém existe uma propriedade específica denominada *body* a qual corresponde à construção do corpo do robô, a qual necessita de existir num ficheiro externo.

Tabela 6- Propriedades básicas dos robôs no SimTwo.

Propriedade	Componentes	Descrição
ID	Nome	*String* identificadora do objeto.
Pos	x, y, z	Posição em x, y, z em m.
Rot_Deg	x, y, z	Orientação sobre o eixo x, y, z em graus.
Body	File	Caminho do ficheiro que contém a descrição do corpo do robô.

Um exemplo do código utilizado para a construção do corpo do robô no ficheiro scene.xml encontra-se representado em baixo.

```
<robot>
    <ID name='Omni3'/>
    <pos x='0' y='0' z='0'/>
    <rot_deg z='90'/>
    <body file='Omni3.xml'/>
  </robot>
```

5.3.4 Construção do Robô

Para a construção do robô são utilizados vários objetos básicos, por exemplo *solids*, *shells* entre outros. Estes objetos são escolhidos conforme as características que se pretendem dar ao robô, isto é se o robô for omnidirecional os objetos básicos serão diferentes daqueles utilizados caso este tenha *joints*. Na Tabela 7 encontram-se representados as diferentes secções que o robô pode ter.

Tabela 7- Secções permitidas dentro da secção do robô.

Secção	Descrição
Solids	Sólidos com massa
Shells	Superfícies sem massa, usadas apenas para colisões
Articulations	Articulações
Wheels	Rodas usadas para a locomoção
Sensors	Sensores

A secção das articulações não será mencionada, por não ser relevante para o trabalho.

5.3.5 Sólidos e *Shells*

As secções *solids* e *shells* permitem definir os vários corpos rígidos que constituem o robô. Por isso são utilizados alguns objetos básicos, mencionados em 5.2.2, nomeadamente o *cuboid* e *cylinder*.

Estes objetos possuem distribuição de massa e forma, pelo que são utilizados tanto na simulação dinâmica como na simulação de colisões. Porém os objetos utilizados nas *shells* possuem somente forma, pelo que apenas participam na simulação de colisões.

Para que seja possível transformar uma *shell* em sólido é necessário atribuir uma massa que se encontre distribuída uniformemente.

5.3.6 Rodas

As rodas correspondem ao sistema de atuação do robô. Estas permitem simplificar a construção de um sistema locomotor de rodas ao criar os corpos necessários para o efeito. São definidas características comuns e posteriormente são criadas subsecções para cada elemento, contendo as suas definições individuais. Para a construção das rodas são atribuídas características gerais representadas no *default* e de seguida é criada uma subsecção na qual são identificadas as diferentes rodas. O código apresentado abaixo representa um exemplo utilizado, para uma só roda:

```
<wheels>
    <default>
      <omni/>
      <tyre mass='0.8' radius='0.04' width='0.03' centerdist='0.09'/>
      <surface mu='1' mu2='0.001'/>
      <axis angle='0'/>
      <motor ri='0.3' ki='2.4e-2' vmax='12' imax='4' active='1'/>
      <gear ratio='12'/>
      <friction bv='1e-5' fc='1e-3' coulomblimit='1e-2'/>
      <encoder ppr='1000' mean='0' stdev='0'/>
      <controller mode='pidspeed' kp='0.2' ki='0' kd='0.01' kf='0.05'
active='1' period='10'/>
      <color_rgb r='128' g='0' b='128'/>
    </default>
    <wheel>
      <axis angle='-60'/>
    </wheel>
</wheels>
```

Cada roda pode estar associada a um sistema de controlo, que supõe um motor elétrico de corrente contínua com uma caixa redutora, um controlador e um *encoder* ótico permitidos pelo SimTwo que se encontram referidos na Tabela 8 [30].

Tabela 8- Componentes da cadeia de atuação do SimTwo.

Componente	Descrição
Motor	Parâmetros relativos ao motor
Gear	Parâmetros relativos à caixa redutora
Friction	Parâmetros relativos à fricção na cadeia de atuação
Encoder	Parâmetros relativos ao *encoder*
Controller	Parâmetros relativos ao controlador
Axis	Parâmetros ao eixo da roda
Omni	Indica qua a roda é omnidirecional
Tyre	Parâmetros relativos às rodas

5.4 Controlo

O controlo de alto nível é efetuado através de um programa intermédio em Pascal, interpretado com ajuda da biblioteca *Lazarus Script*. Este tipo de linguagem, associado com algumas interfaces de entrada e saída, permite uma grande flexibilidade na implementação do controlo.

Este comando permite o controlo dos vários motores, porém é mais vantajoso utilizar os controladores de velocidade locais referidos na secção 5.3.6. Assim sendo, em vez de ser responsável por gerar tensões a aplicar nos motores, o programa *Lazarus Script* pode limitar-se a definir a referencia desejada da velocidade ou posição.

5.4.1 Programa de Controlo

O código é criado com o recurso a um editor de texto que se encontra incluído no simulador. Este código é guardado num formato especial em ficheiros com a extensão ulf.

Neste simulador é necessário definir dois procedimentos, são eles o *Initialize* e o *Control*. O *Initialize* é chamado quando o programa inicia e o *Control* é chamado a cada 40 ms. A estrutura do código será a seguinte:

```
procedure Control;
begin
{Ciclo de Controlo }
end;
procedure Initialize;
begin
{Executado quando o programa se inicia}
end;
```

O procedimento *Control* é utilizado para se implementar um ciclo de controlo discreto com um período de 40 ms. Durante este ciclo é possível recolher informação do estado do robô, bem como aplicar determinadas características como velocidade e a posição.

Além destas funcionalidades, o utilizador também pode definir outras funções e procedimentos, e quando for necessário controlar mais de que um robô.

5.4.2 Interação com a Simulação

Como já foi mencionado é possível controlar mais de um robô na mesma simulação, porém é necessário que cada robô tenha um número identificativo, iniciado em zero. Cada robô possui diversos eixos controladores, ou *axis*, e vários corpos, ou *solid*, onde cada um também é identificado por um número que depois é utilizado quando no editor é necessário utilizar, por exemplo, as rodas do robô.

A interação com a simulação é efetuada através de alguns procedimentos e funções, as quais levam em conta a hierarquia da informação. As funções que são aplicadas no robô possuem como primeiro argumento o número identificativo do robô. O número do ID do robô pode ser obtido através da seguinte função:

```
function GetRobot Index (R: integer ; ID : string ) : integer;
```

Os procedimentos que permitem obter obter informações sobre a posição e velocidade do robô possuem os seguintes protótipos:

```
function GetRobotPos2D(R: LongInt): TState2D;
 function GetRobotTheta(R: LongInt): Double;
 function GetRobotVel2D(R: LongInt): TState2D;
```

```
function GetRobotVx(R: LongInt): Double;
function GetRobotVy(R: LongInt): Double;
function GetRobotW(R: LongInt): Double;
function GetRobotX(R: LongInt): Double;
function GetRobotY(R: LongInt): Double;
```

No caso das funções aplicadas no robô, como por exemplo a aplicação de velocidade nas rodas ou a posição para a qual se quer que o robô se dirija, são utilizadas as seguintes funções, onde *i* corresponde ao eixo e *R* ao robô:

```
procedure SetAxisPosRef(R: LongInt; i: LongInt; aPos: Double);
 procedure SetAxisSpeedRef(R: LongInt; i: LongInt; aSpeed: Double);
procedure SetAxisStateRef(R: LongInt; i: LongInt; aState: TAxisState);
```

Caso seja necessário obter-se informações sobre o estado de um determinado eixo do robô, são usadas as seguintes funções:

```
function GetAxisOdo(R: LongInt; i: LongInt): LongInt;
 function GetAxisPos(R: LongInt; i: LongInt): Double;
 function GetAxisPosDeg(R: LongInt; i: LongInt): Double;
 function GetAxisPosRef(R: LongInt; i: LongInt): Double;
 function GetAxisPosRefDeg(R: LongInt; i: LongInt): Double;
 function GetAxisSpeed(R: LongInt; i: LongInt): Double;
 function GetAxisSpeedDeg(R: LongInt; i: LongInt): Double;
 function GetAxisSpeedRef(R: LongInt; i: LongInt): Double;
 function GetAxisSpeedRefDeg(R: LongInt; i: LongInt): Double;
```

Além destas funções utilizadas também existem outras que se encontram na documentação do simulador ou no sistema de ajuda do mesmo. Tanto as funções apresentadas como as restantes são de fácil entendimento, uma vez que o próprio nome da função determina o uso que se lhe pode dar.

5.5 Controladores PID

Os controladores do tipo Proporcional, Integral e Derivativo (PID), apresentam diversas funções tais como: fornece feedback sobre o estado do sistema, possibilita a eliminação de desvios estacionários através da ação de um integral e possibilita a antecipação de ações do futuro através da derivada, uma vez que a derivada permite definir o comportamento de uma função ao longo do tempo [31].

Este tipo de controladores é utilizado para diversos problemas de controlo, em particular quando os processos dinâmicos estão a começar e os requisitos para o desempenho são modestos [31].

Entre 90 a 95% dos problemas de controlo são solucionados com a utilização correta dos controladores PID. A utilização em grande escala deste tipo de controladores, deve-se ao facto de estes serem fáceis de implementar, de baixo custo e versáteis com a capacidade de alterar os comportamentos transitórios e de regime permanente [32].

De acordo com Aströn [31], a principal razão para a baixa performance dos processos automatizados deve-se a problemas em válvulas, sensores e a incorreta parametrização dos controladores PID utilizados nos diversos processos.

Como já foi mencionado o controlador PID envolve 3 ações quase intuitivas que se encontram descritas na Tabela 9.

Tabela 9- Ações que representam o controlador PID [31].

Proporcional (P)	Correção proporcional ao erro	A correção a ser aplicada ao processo deve crescer na proporção que cresce o erro entre o valor real e o desejado.
Integral (I)	Correção proporcional ao produto do erro com tempo	Erros pequenos mas que existem há muito tempo requerem correção mais intensa.
Derivada (D)	Correção proporcional à taxa de variação do erro	Se o erro está a variar muito rápido, esta taxa de variação deve ser reduzida para evitar oscilações.

Um controlador PID apresenta a seguinte estrutura representada na Figura 16, onde é considerado o sistema de controlo em malha-fechada.

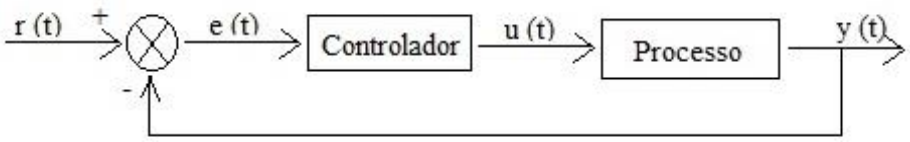


Figura 16- Diagrama de blocos de um sistema de controlo em malha-fechada.

O controlador representado na Figura 16 tem como base o sinal da diferença existente entre o sinal de referência r (t) e o sinal de saída y (t), gera na saída um sinal de controlo u (t) que seja de corrigir e se possível anular a diferença mencionada.

Neste caso em específico, a lei de controlo descrita no diagrama é composta por três termos representada pela equação 1:

$$u\,(t) = u_p(t) + u_i(t) + u_d(t) \qquad (1)$$

Cada termo apresentado no lado direito da equação 1, estão associados a cada um dos tipos de ações do controlador. No diagrama da Figura 16 a parte que representa o controlador PID pode ser descrito em três blocos distintos, os quais estão representado na Figura 17, onde o sinal e (t) (erro) é utilizado como entrada.

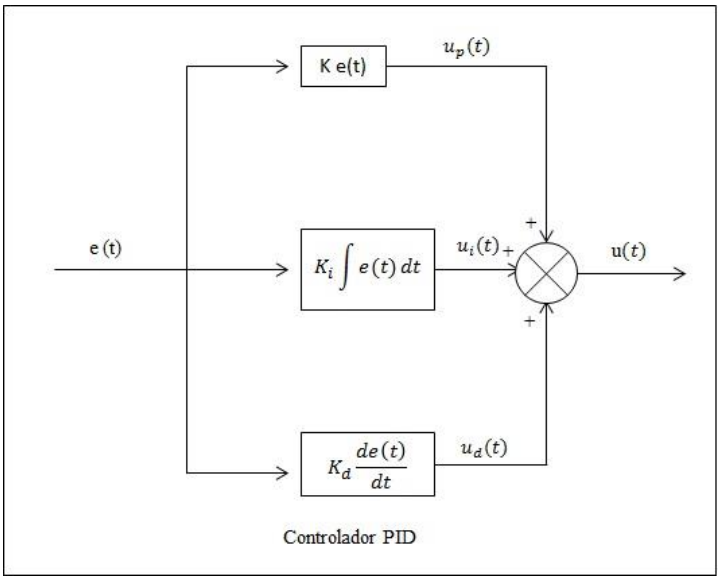


Figura 17- Diagrama de blocos de um controlador do tipo PID.

O primeiro bloco constituído pela constante K, o qual pertence ao controlador PID, é responsável pela ação proporcional do controlador. O sinal de saída deste bloco é dado pela equação 2:

$$u_p(t) = K\, e(t) \tag{2}$$

Da mesma forma é possível escrever os sinais de saída relativos aos blocos integral e derivativo, representado pelas equações 3 e 4:

$$u_i(t) = k_i \int e(t)\, dt \tag{3}$$

$$u_d(t) = K_d \frac{de(t)}{dt} \tag{4}$$

O efeito de cada uma destas ações e as suas implicações no comportamento dinâmico de um sistema de controlo ocorre em sequência.

5.6 Criação do ambiente- Cenário

Nas secções atrás explicam a simulação utilizada e o controlador PID usado para a criação do ambiente de simulação e o controlo desta, respetivamente. Nesta secção será descrito todo o processo que foi desenvolvido para a criação da simulação.

Para a criação da simulação foi necessário criar um ambiente onde se encontra o robô, fazendo com que esta seja o mais próximo possível da realidade. Para isso foi necessário ter a noção da sala de exames assim como os acessórios que lá existem, como é possível visualizar na Figura 18.

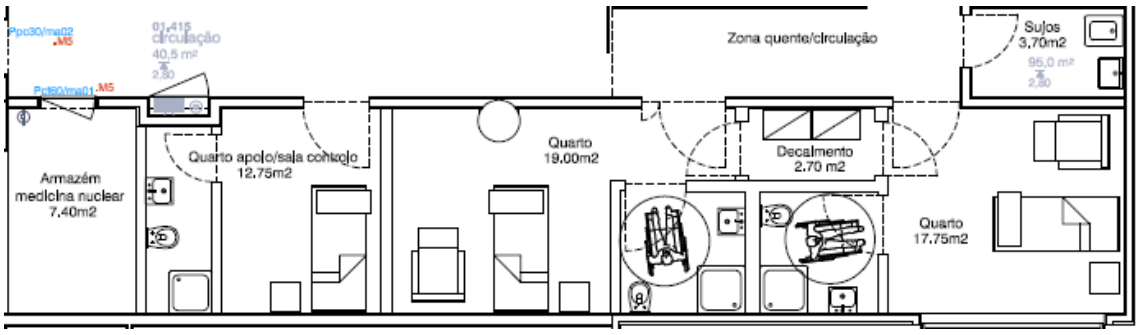


Figura 18- Salas de tratamento com I^{131}.

Na Figura 19 encontra-se um esquema da sala, onde se encontram as paredes da sala, para que seja possível delimitar o espaço onde o robô se pode movimentar. Também é necessário colocar a maca para que se possa ser o mais fiel possível com a realidade.

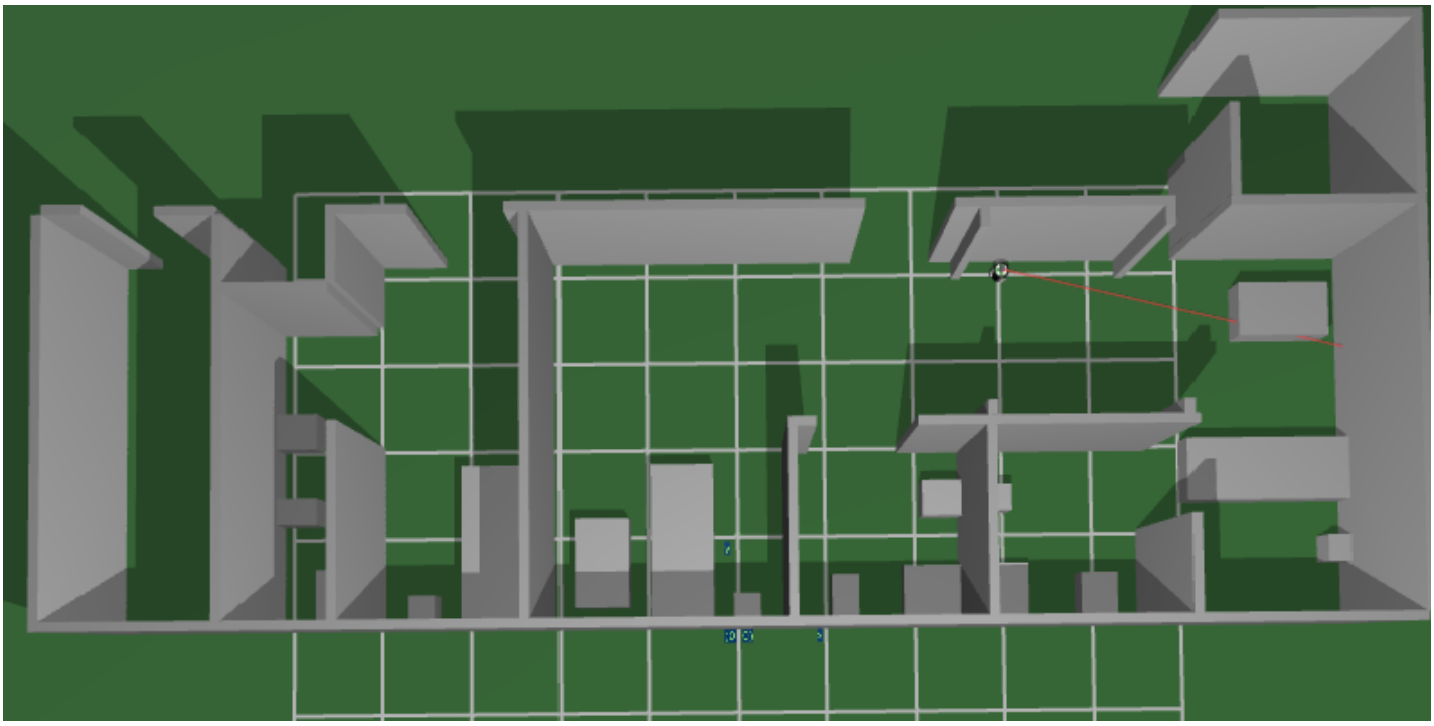


Figura 19- Ambiente da simulação.

Como foi explicado no Capítulo 5, secção 5.3, o cenário da simulação é desenvolvido por etapas, sendo a primeira a criação de um ficheiro *xml* normalmente denominado de *scene* onde é colocado o código que corresponde à identificação do robô criado no ficheiro "Omni3". Além da linha de comando correspondente à identificação do robô também está presente a identificação dos obstáculos (obstacles file='obstacles.xml') presentes na simulação. O código que diz respeito a esta descrição encontra-se representado no Anexo A.

Neste código encontram-se representado a identificação do nome do ficheiro xml (ID name='Omni3') onde está construído a estrutura do robô, assim como a posição onde o robô se encontra o robô quando a simulação é iniciada (pos x='3' y='4' z='0' ; rot_deg z='0').

No código abaixo encontra-se as linhas correspondentes à criação do robô assim como o laser que será utilizado para que posteriormente este se possa localizar utilizando o algoritmo descrito no Capítulo 5. Para que seja mais fácil de compreender o este código será dividido em partes.

```
<?xml version="1.0" ?>
<robot>
  <kind value='omni3'/>
```

Esta parte do código corresponde à identificação do robô a qual é utilizada no ficheiro *scene*. No Anexo A encontra-se a parte do código que corresponde à criação do corpo do robô. Para a construção deste é necessário utilizar o objeto básico *cylinder*, atribuir-lhe a massa pretendida, assim como o tamanho que este vai ter a posição em x, y e a rotação do mesmo. Além destes parâmetros também é definido a cor que este irá através do sistema rgb.

O código apresentado no Anexo A representa a construção das 3 rodas omnidirecionais necessárias para o funcionamento do robô. Numa primeira parte é atribuída uma massa às rodas, raio, largura e a distância ao centro. De seguida é atribuído as constantes do modelo do motor, onde é colocado um valor para a resistência, a constante do motor que relaciona linearmente o binário com a corrente, a constante de Coulomb e a velocidade de rotação, a tensão máxima e a corrente máxima de alimentação.

Também é necessário colocar os valores utilizados para o modelo dinâmico, neste caso o coeficiente de atrito de Coulomb e o coeficiente de atrito viscoso (pode-se encontrar mais informação no Capítulo 4). Além destes coeficientes é necessário escolher o tipo de controlo, neste caso o *pidspeed* (onde é implementado um controlador PID que tenta controlar a velocidade e onde os ganhos k_p, k_i e k_d devem ser definidos). Os ganhos utilizados foram obtidos previamente de uma simulação na qual é usado um robô com as mesmas características.

A segunda parte corresponde à colocação das três rodas com um ângulo de 120º entre elas. Na Figura 20 é possível visualizar o resultado obtido após a aplicação do código referente à construção do robô omnidirecional.

Figura 20- Robô omnidirecional.

De seguida foram criados os lasers que mais tarde serão utilizados para a construção, futura, do algoritmo *Perfect Match*. Numa primeira parte é necessário criar a estruturas que vão dar lugar ao laser, para isso é necessário atribuir qual o objeto básico a utilizar, neste caso foi o *cylinder,* e fazer a sua identificação através do ID.

Depois de feitos estes dois passos é atribuído o tamanho e posição que o laser iria ter. Os valores atribuídos foram escolhidos para que os feixes conseguissem abranger todo o espaço envolvente. É possível verificar o código utilizado no Anexo A.

Na segunda parte deste código está representado as características do feixe, tais como o tamanho do feixe, a largura inicial e final deste e a posição e rotação, o qual se pode verificar na Figura 21. Além destes parâmetros, também é necessário colocar quantos graus o laser vai girar, o ganho que vai ter e o ruído.

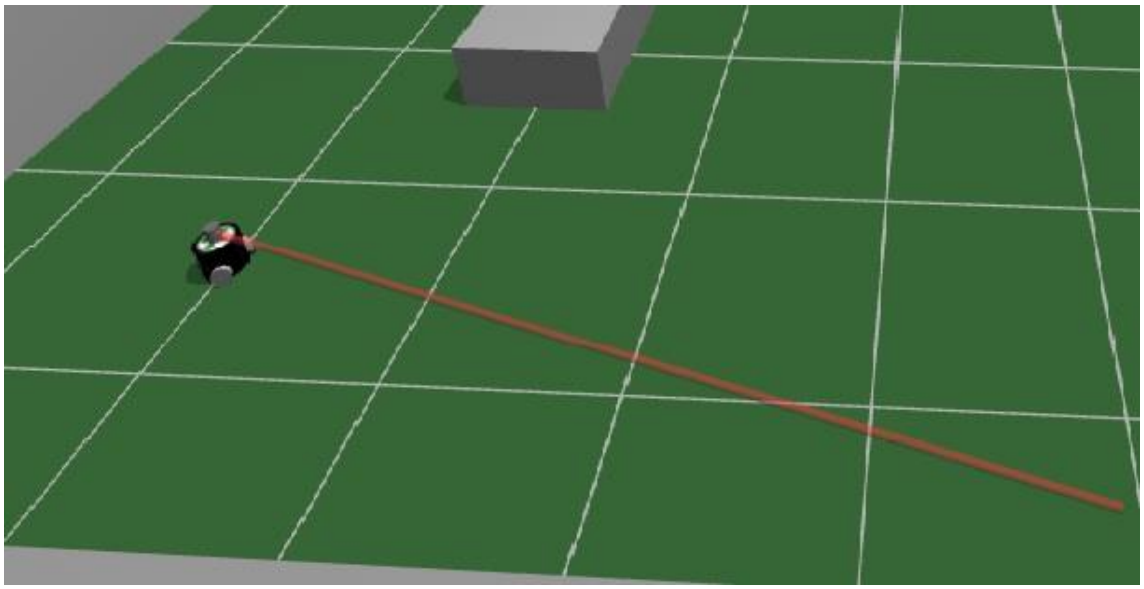


Figura 21- Representação do laser.

Por fim foram construídas as paredes que correspondem ao tamanho das salas assim como a maca e restantes acessórios que são colocados nas salas. No Anexo A só se encontra representado parte do código pois o restante é semelhante.

5.7 Controlo da simulação- Control

Nesta parte da simulação é criado um código com o qual é possível controlar o robô. Como é explicado neste capítulo, o Control é dividido em duas partes o "procedure control" e o "procedure initialize", onde no primeiro é desenvolvido um ciclo o qual é executado a cada 40 ms e o segundo é invocado sempre que o simulador é executado.

Para que seja possível compreender o que é necessário considerar para o algoritmo de controlo do robô, foi desenvolvido o diagrama da Figura 22. Este diagrama em blocos representa as velocidades lineares do robô, que são as velocidades iniciais segundo *vx*,*vy* *e* ω, representadas por vi. Após a entrada no ciclo, estas velocidades sofrem uma filtração através dos controladores PID, de seguida é necessário determinar os novos valores para as velocidades (v, v_n *e* ω) utilizando as equações mencionadas no Capítulo 4, as quais são calculadas no sistema do algoritmo. Por fim são enviadas para as rodas as velocidades lineares v_0, v_1 e v_2, representadas no diagrama por vf.

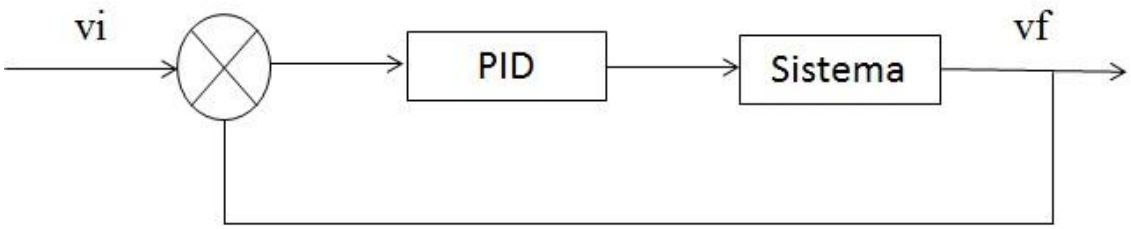


Figura 22- Diagrama em blocos do controlador do robô.

No caso do "procedure initialize", apresentado abaixo, a simulação vai ser iniciada com a posição onde o robô se encontra utilizando a função pré definida *SetRobotPos* na qual é necessário atribuir coordenadas para x, y e z assim como um ângulo de θ. De seguida são definidas três variáveis que serão utilizadas na parte do "*procedure control*", as quais serão explicadas mais à frente.

```
procedure Initialize;
begin
  SetRobotPos(0,-3,-2.5, 0,0);
  apontlinha := 7;
  wait_state := false;
  wait_time:= 2;
end;
```

Além destes passos previamente criados pelo simulador, também foi necessário desenvolver uma função Velocidades, onde é possível atribuir diferentes velocidades às rodas fazendo com que o robô se movimente de modo controlada.

Nesta função Velocidades, apresentada abaixo, encontram-se representadas como variáveis de entrada *v e* v_n, que correspondem lineares em x e y, respetivamente, e ω que corresponde à velocidade angular.

```
function Velocidades( v, vn, w : Double):double;
```

Para que seja mais simples de se compreender a função criada, esta será apresentada de forma esquemática na Figura 23. Após a obtenção da posição inicial, é necessário definir as posições para onde o robô se deve dirigir, sendo estas definidas pelo utilizador, as variáveis utilizadas são xt, yt e thetat.

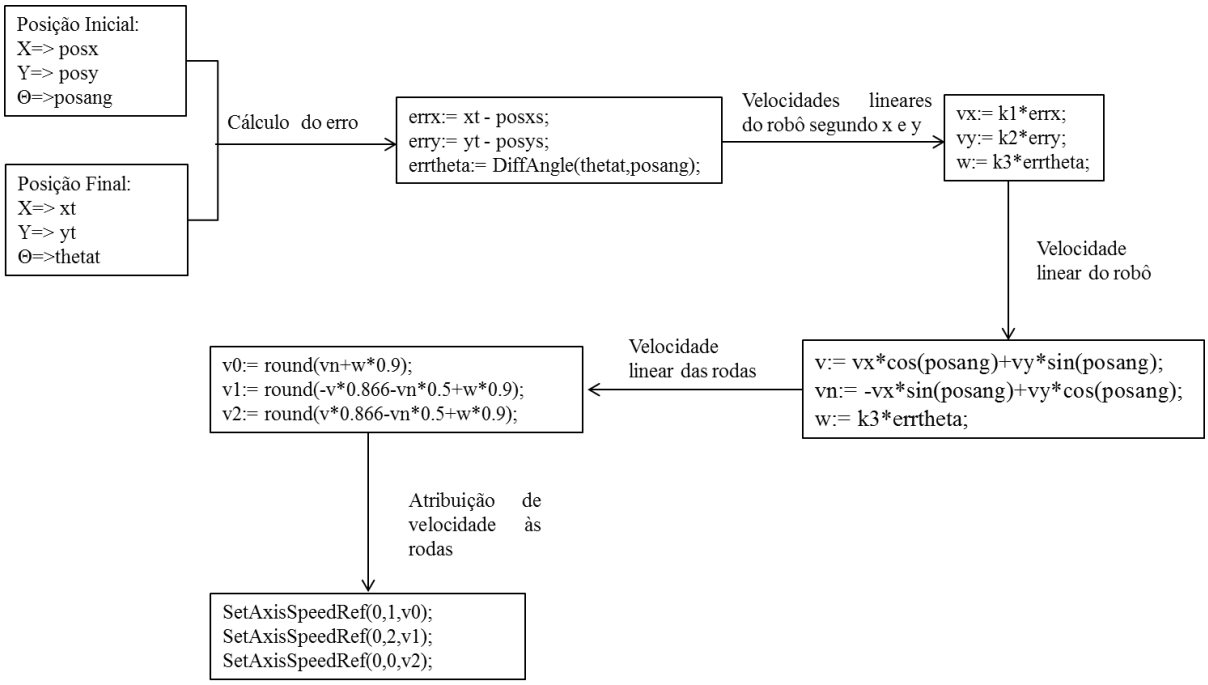


Figura 23- Diagrama da aplicação das velocidades nas rodas.

Uma vez obtidas todas estas informações é necessário calcular o erro para cada parâmetro *x, y e θ*. O erro irá permitir ajustar o quanto o robô deve se movimentar para chegar à posição pretendida. De seguida o erro é multiplicado por uma constante, neste caso definida por k1, k2 e k3, obtendo-se assim a velocidade linear segundo os eixos x e y. Os valores de k1, k2 e k3 são obtidos por tentativa em erro, pois este valor só diz respeito à rapidez com que o robô se vai movimentar. Neste caso os valores que melhor se adaptam foram k1=20, k2=20 e k3=10.

Este cálculo diz respeito à utilização de um controlador de PID. Porém neste caso em específico, só foi necessário a utilização de um controlador de proporcionalidade uma vez que o erro obtido era muito pequeno, obtendo-se assim ótimos resultados. Daí que não foi necessário implementar o controlador de integração e de derivação.

Nos gráficos representados pelas Figura 24, Figura 25 e Figura 26, é possível verificar como o robô se comporta quando este se desloca de uma posição para a outra, sendo estas previamente definidas pelo operador.

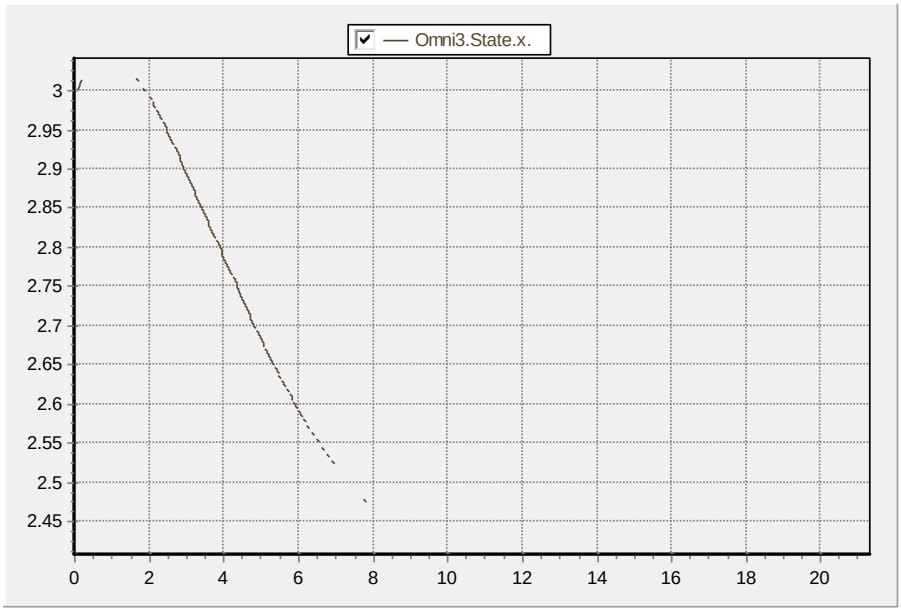


Figura 24-Comportamento do robô desde a posição inicial até à posição seguinte segundo x, ao longo do tempo.

Na Figura 24 encontra-se representado o comportamento do robô tem desde a posição 3 até à posição x=2.5. É possível verificar que o erro que existe é baixo.

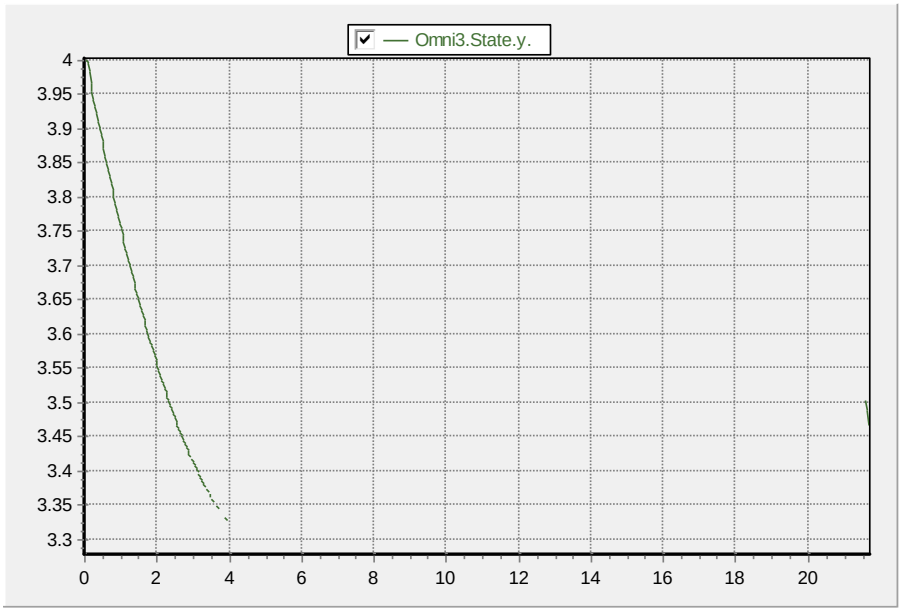


Figura 25- Comportamento do robô desde a posição inicial até à posição seguinte segundo y, ao longo do tempo.

O comportamento segundo y, representado na Figura 25, vai de encontro com o comportamento segundo x. Isto é, o robô vai da posição inicial 4 até à seguinte (x=3.5) sem necessitar de grandes desvios e de forma rápida.

Na Figura 26, encontra-se representado o comportamento do robô segundo o ângulo θ. Este gráfico é um pouco diferente dos representados atrás, uma vez que o valor do ângulo é obtido de forma automática através da linha de código, representada abaixo:

```
angaut:= arccos(posxs/(sqrt((posxs*posxs)+(posys*posys))));
```

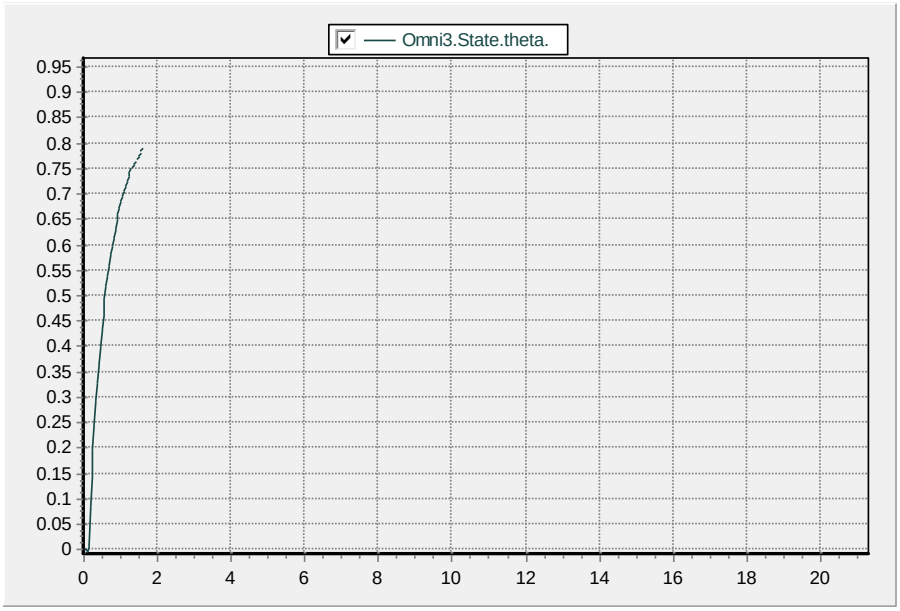


Figura 26-Comportamento do robô segundo θ, ao longo do tempo.

Porém neste caso, o valor do θ também poderia ser colocado manualmente utilizando a linha de código referida abaixo.

```
thetat:=GetRCValue(apontlinha,4);
```

De seguida é necessário calcular as velocidades lineares do robô (v, vn e w) através da equação da cinemática, abordada no Capítulo 4 pela equação 4. Após determinada estas velocidades é necessário calcular as velocidades lineares das rodas, as quais vão ser aplicadas nas rodas, através da função do simulador (*SetAxisSpeedRef).*

Além de todos estes parâmetros, também foi necessário colocar na função duas equações, apresentadas abaixo, que permitem ver a distância que o robô tem de percorrer para ir de um ponto ao outro. Estas equações correspondem à distância euclidiana e à diferença entre o ângulo inicial e o final.

```
dist:= sqrt(((xt-posxs)*(xt-posxs))+((yt-posys)*(yt-posys)));
ang:= abs(thetat-posang);
```

Na parte que diz respeito ao "*procedure control*" foi criado um ciclo que irá ser inicializado a cada 40ms da simulação. Numa primeira parte é utilizado um *procedure* definido no simulador como "SetRCValue". Com este procedimento é possível visualizar diferentes parâmetros numa folha de cálculo, denominada *sheets* no simulador. Como é possível verificar abaixo, para aplicar este procedimento é necessário definir a linha, a coluna e o parâmetro que se pretende visualizar na *sheet.*

```
SetRCValue(7, 6 ,format('%.4g',[posxs]));
SetRCValue(8, 6 ,format('%.4g',[posys]));
```

De seguida foi criado um ciclo no qual o robô vai avançando de posição em posição sem que seja necessário estar sempre a alterar a mesma na folha de cálculo. Neste ciclo são utilizadas três variáveis definidas no "procedure initialize" denominadas de apontlinha, wait_time e wait_state.

A variável "apontlinha" indica em qual linha é iniciada a leitura de valores na *sheet* e vai auxiliar na leitura das posições através de um incremento. A variável "wait_state" é do tipo boleana, o que significa que esta função vai assumir valores *true* ou *false.* Por fim a variável "wait_time" vai corresponder ao tempo que o robô vai ficar parado na mesma posição. Abaixo encontra-se representado o código ao quando está atribuída a descrição.

```
if ((dist < 0.15) and not (wait_state)) then begin
 wait_state := true;
 wait_time := 10;
end;

if wait_state then wait_time := wait_time - 0.04;
```

```
if ((wait_time < 0) and (wait_state)) then begin
 inc(apontlinha);
 wait_state := false;
end;
```

O ciclo acima inicia-se quando a distância entre duas posições diferentes é inferior a 0.15 e o "wait_state" é verdade. Se se verificar estas duas condições o robô irá parar com um "wait_time" de 10 s (definido pelo utilizador), uma vez que o "wait_state" é verdade.

Quando o robô se encontra na posição pretendida e o "wait_state" é falso, vai ser efetuado um decréscimo no "wait_time" de 0.04 em 0.04s. Quando o "wait_time" é inferior a zero e o "wait_state" falso, irá ser efetuado um incremento no apontlinha, isto é o simulador irá ler a posição da linha seguinte.

Por fim é necessário delimitar o número de linhas que serão lidas pelo ciclo, neste caso foi escolhido até à linha 20, e a linha onde irá ser lida a primeira posição a linha 7.

Depois de percorridas todas as posições definidas, o robô irá voltar para a posição inicial através da função SetRobotPos.

Também é necessário colocar a função já definida pelo simulador como GetRCvalue que permite ir buscar os valores definidos na *sheet* para as diferentes posições que o robô tem de percorrer.

```
if apontlinha = 20 then SetRobotPos(0,3,4, 0,0);;
SetRCValue(14, 6, inttostr(apontlinha));
SetRCValue(15,6, floattostr(wait_time));

xt:= GetRCValue(apontlinha,2);
yt:= GetRCValue(apontlinha,3);
```

Capítulo 6- Resultados Obtidos

Neste capítulo serão apresentados os resultados obtidos neste trabalho, isto é será apresentado o modo como o algoritmo funciona na simulação, assim como a forma como a velocidade linear varia dependendo de cada roda.

6.1 Funcionamento e Comportamento da Simulação

Para que seja possível demonstrar o modo como o algoritmo funciona é necessário determinar algumas posições onde o robô se possa movimentar. Este algoritmo só necessita das posições segundo x e y. Para que seja de fácil entendimento é escolhida uma zona específica onde o robô se irá movimentar, esta encontra-se representado na Figura 27.

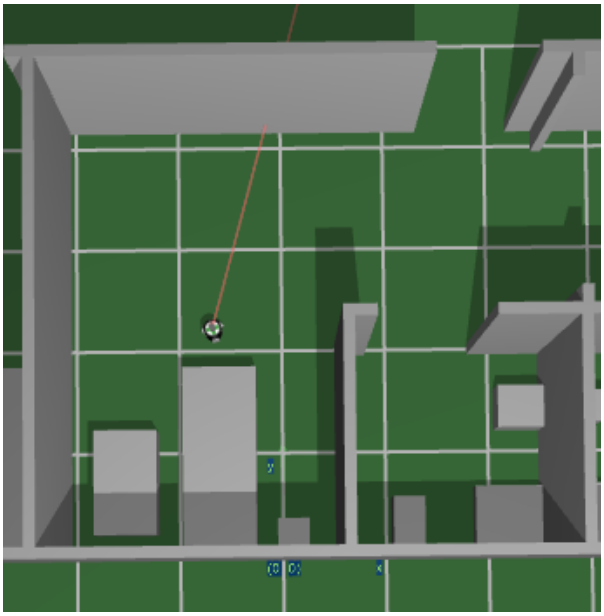


Figura 27- Zona onde o robô se irá movimentar.

Na Tabela 10 encontram-se representadas as posições onde o robô irá se movimentar segundo x e y.

Tabela 10- Posições segundo x e y.

X	Y
2.5	3.5
1	3.5
-0.5	3
-0.5	2.5
-1.2	2.3
-1.2	3
-1	3.5
0.8	3.5
1	2.6
1	1.5
1.3	1.5
1.3	2
2	3

Após a atribuição dos valores é colocado o simulador a correr. O ciclo começa na posição x=3 e y=4, dada como posição inicial no começo da simulação. A posição para onde o robô se irá deslocar será a x=2.5 e y=3.5.

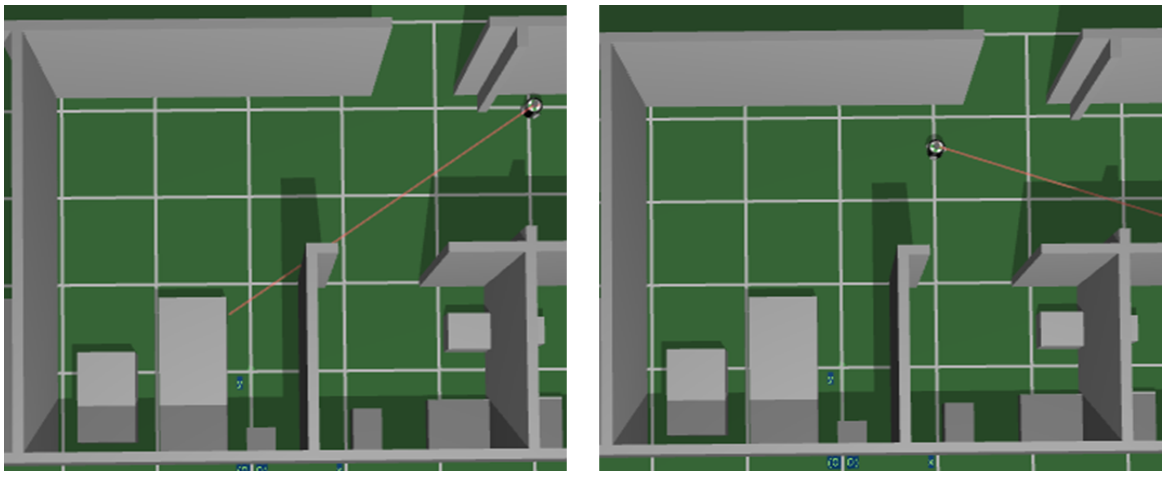


Figura 28- A) representa a posição inicial do robô e b) a posição para onde se dirige.

Como é possível verificar na Figura 28 a) o robô encontra-se na posição inicial que foi mencionada atrás, já na Figura 28 b) é possível verificar a posição para onde o robô se deslocou. Esta figura representa um exemplo de como o robô se desloca ao longo da simulação e até percorrer todas as posições mencionada na tabela acima.

Como foi explicado no Capítulo 5 secção 5.7, o robô para em cada posição durante 10 segundos, quando a distância euclidiana é inferior a 0.15. Estes valores encontram-se apresentados na folha de cálculo *sheet*, representada na Figura 29. Nesta figura é possível verificar à esquerda as diferentes posições segundo x e y mencionadas atrás, e à direita encontra-se apresentada a posição real onde o robô se encontra em x e y e o ângulo que o robô apresenta na altura do deslocamento. É de notar o ângulo é obtido automaticamente sem ser necessário recorrer à colocação manual deste.

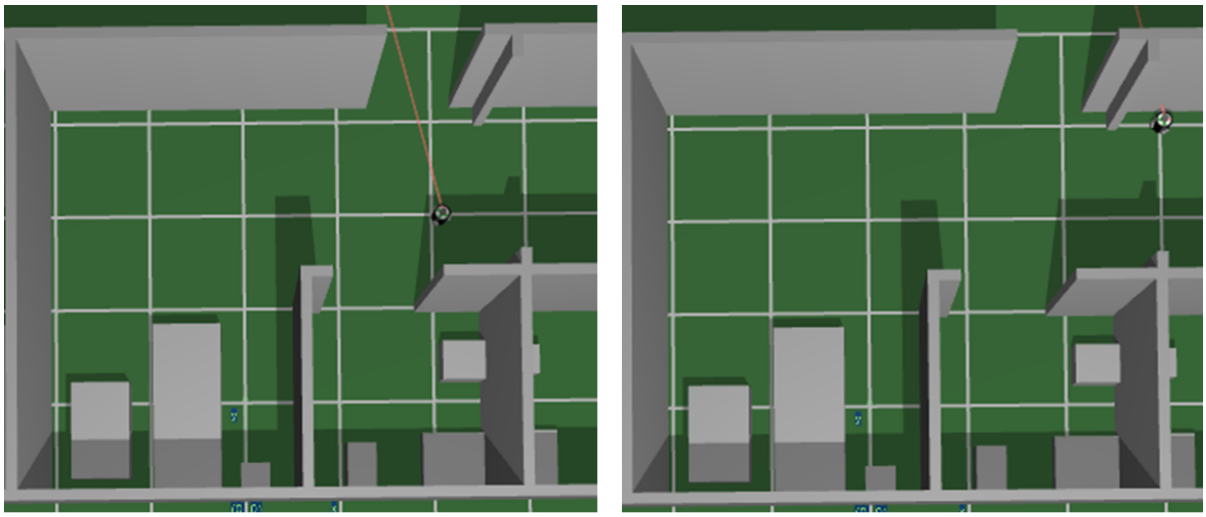
Além da posição real do robô, também encontra-se representada a distância euclidiana, isto é a distância que vai desde a posição inicial à final, e a diferença que existe entre os ângulos. Também é possível verificar o ponto que está a ser lido na simulação, isto é, apresenta a linha que está a ser lida no caso da figura abaixo é a linha 8. Por fim, encontra-se representado o tempo que o robô está parado na mesma posição, efetuando-se um contagem decrescente.

	x	y
Pos 1	2,5	3,5
Pos 2	1	3,5
Pos 3	-0,5	3
Pos 4	-0,5	2,5
Pos 5	-1,2	2,3
Pos 6	-1,2	3
Pos 7	-1	3,5
Pos 8	0,8	3,5
Pos 9	1	2,6
Pos 10	1	1,5
Pos 11	1,3	1,5
Pos 12	1,3	2
Pos 13	2	3

X	1,011
Y	3,583
Ângulo	1,324
Distância	0,0832
Dif. Ângulo	0,02841
Ponto Atual	8
Tempo	8,92

Figura 29- Folha de cálculo *sheet.*

Na Figura 30 a) encontra-se representada a última posição colocada na folha de cálculo onde x=2 e y=3, e na Figura 30 b) é possível verificar que o robô voltou à posição inicial onde a simulação começou.



a) b)

Figura 30- A) representa a posição 13 e b) a posição inicial.

Para que seja possível verificar o comportamento das velocidades lineares do robô, durante a simulação, foram avaliados os as velocidades vx e vy. Os gráficos apresentados abaixo foram retirados do simulador SimTwo.

Na Figura 31 encontra-se representado o comportamento da velocidade vx, desde a posição inicial x=3 e y=4 até à posição x=2.5 e y=3.5.

Figura 31- Gráfico referente ao comportamento da velocidade vx em função do tempo.

Como é possível verificar no gráfico da figura acima, a velocidade linear vx, varia quando este se encontra em movimento, isto é o robô está a encaminhar-se para a posição definida. O intervalo onde se pode verificar a maior variação das velocidades é entre os 2 e 12 s (estes valores dizem respeito ao tempo decorrido na simulação).

Quando se chega aos 12 s verifica-se que a velocidade se mantém mais constante, até que nos 14 s esta não se altera, isto significa que o robô encontra-se na posição que foi definida pelo utilizador. A partir dos 14 s o valor da velocidade é nulo pois significa que o robô se encontra parado naquela posição. As irregularidades que se verificam na linha que representa a velocidade vx dizem respeito as vibrações às quais o robô está sujeito.

O comportamento do robô mantém-se mais ou menos constante ao longo de toda a simulação havendo ajustes referentes as diferentes posições onde este se desloca.

Na Figura 32 encontra-se representado o comportamento do robô em vy, desde de x=3 e y=4 até x=2.5 e y=3.5.

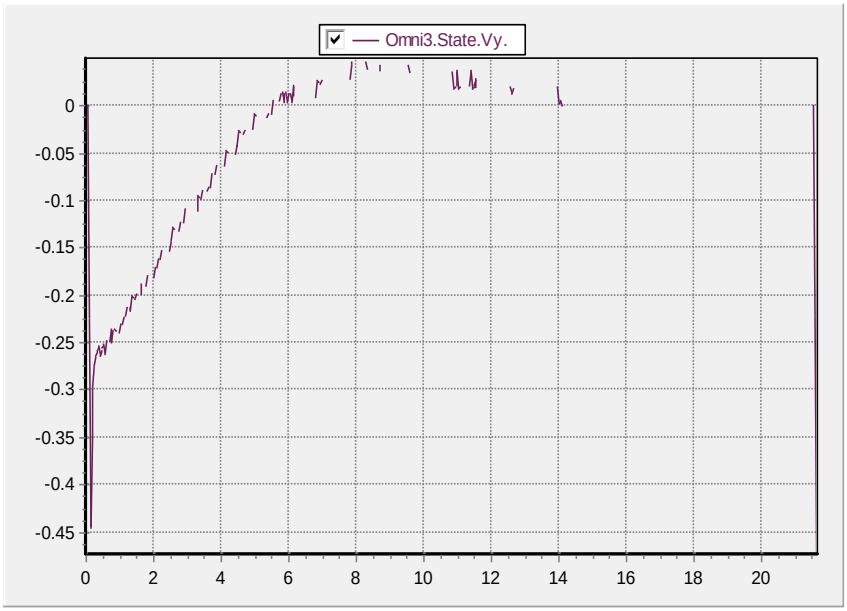


Figura 32- Gráfico referente ao comportamento do robô segundo vy em relação ao tempo.

No gráfico acima é possível verificar que desde 0 até mais ou menos 9 s a velocidade do robô encontra-se a sofrer ajustes, até que a partir dos 11 s a velocidade do robô mantém-se mais ou menos constante até que a partir dos 14 s esta não se altera.

Tal como já foi explicado acima, a velocidade mantém-se constante pois o robô chegou à posição previamente definida e encontra-se parado. Ao longo da simulação o comportamento da velocidade vy não varia muito. Porém á que ter em conta que existem pequenos ajustes dependendo da posição os o robô se encontra.

Em ambos os casos entre o tempo decorrido entre 9 e 14 s o robô encontra-se constante sofrendo uma pequena alteração aos 14 s. Isto quer dizer que, como foi explicado no Capítulo 5, o tempo começa a contar quando a distância euclidiana entre os dois pontos é inferior a 0.1, daí haver essa pequena alteração no gráfico.

Como foi mencionado no Capítulo 5, o ângulo é ajustado automaticamente conforme a posição para onde o robô tem de se deslocar. Na Figura 33 encontra-se representado o comportamento do ângulo θ desde x=3, y=4 até à posição 3 (ver Tabela 10).

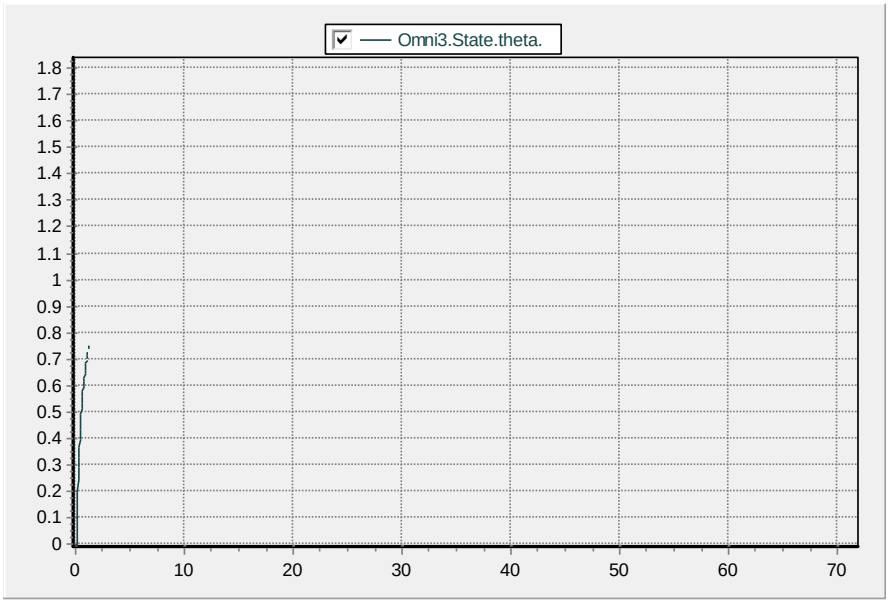


Figura 33- Comportamento do ângulo θ em relação ao tempo.

Como é possível verificar o ajuste que é feito no ângulo é pequeno. Os instantes que são importantes de ter em atenção são de 0 a 10, 20 a 40, e 50 a 65 s. Estes dizem respeito aos ajustes que são feitos conforme a posição para onde o robô tem de se dirigir, logo vai se alterando. Nas zonas do gráfico onde este se encontra constante, significa que ângulo já foi ajustado.

6.2 Constituição do Hardware do Robô

Neste trabalho prático também foi iniciado a construção do *hardware* do robô. Este robô é constituído por uma placa de Arduíno, dois motores EMG30 e três rodas omnidirecionais. Para que seja possível a melhor compreensão dos passos que são necessários realizar, é colocado no Anexo B o esquema do robô.

Além destes componentes o robô também tem um *shield* do Arduíno, representada na Figura 34, onde são colocados os diferentes componentes responsáveis pela alimentação e gestão da carga da bateria.

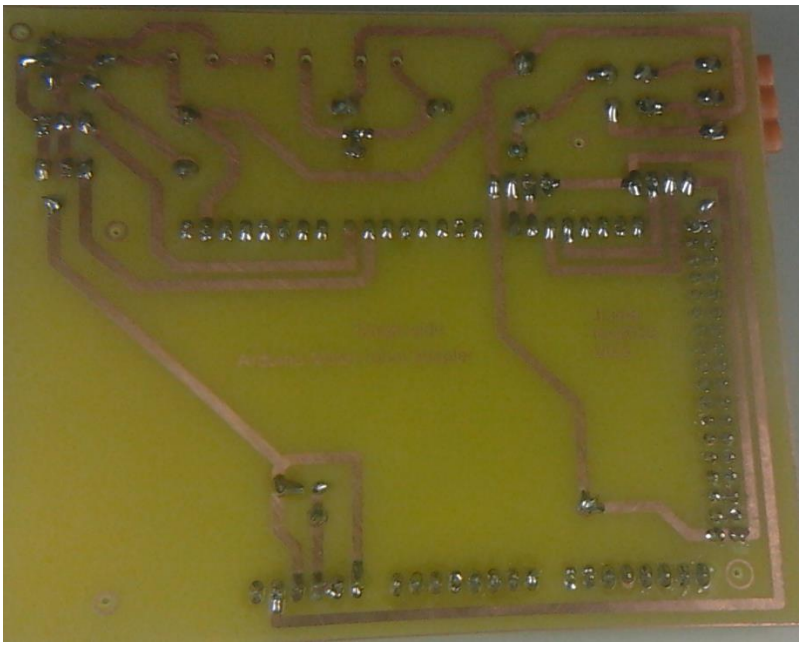


Figura 34- Placa adaptadora do Arduíno.

Porém para que seja possível colocar os diferentes componentes no local correto é necessário seguir um plano, referente à Figura 35, onde indica onde cada componente deve ser colocado.

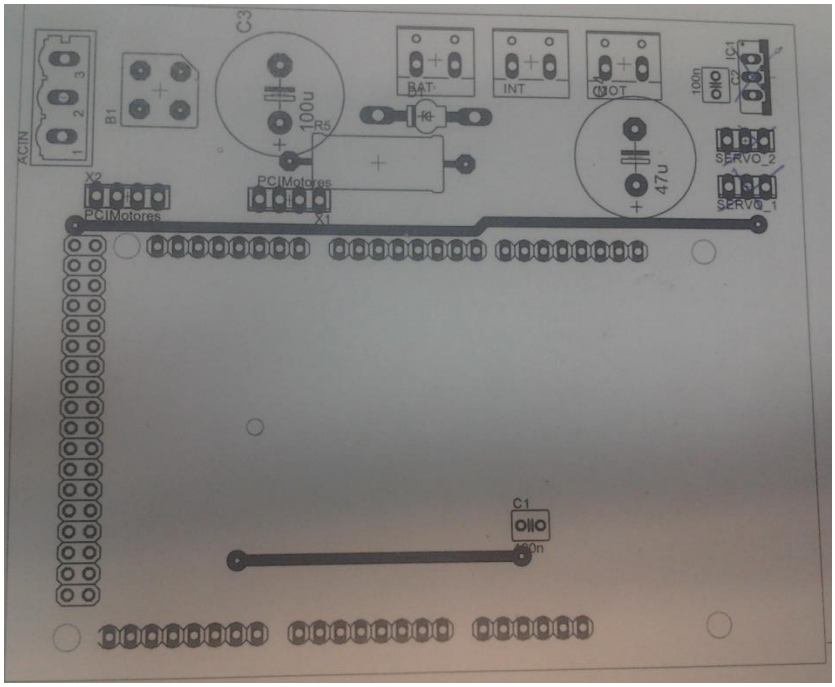


Figura 35- Esquema para a correta colocação dos componentes de hardware.

Na Figura 36, encontra-se representado o esquema e a placa adaptadora já com os componentes soldados, para que seja possível realizar a comparação entre estes.

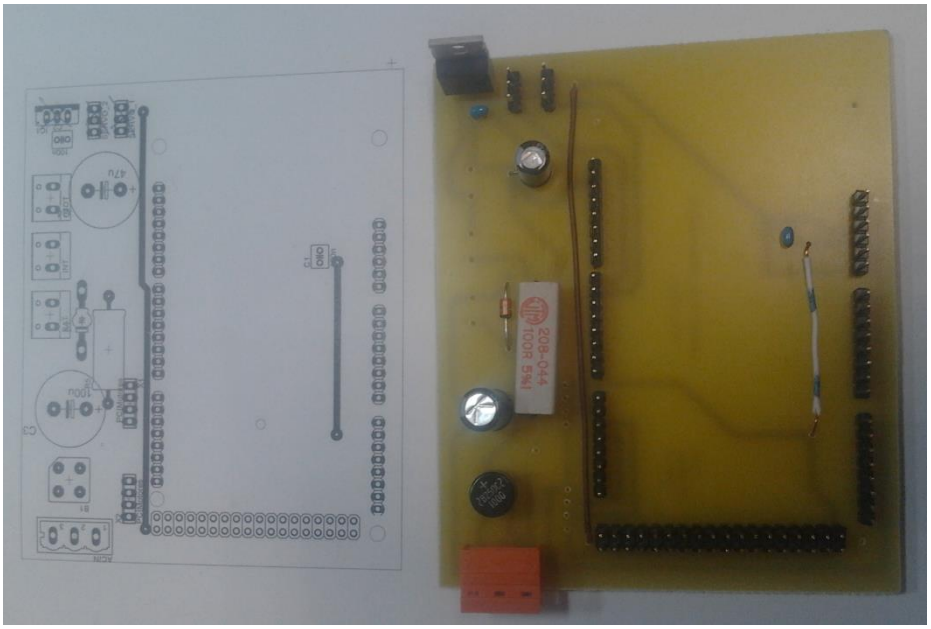


Figura 36- Comparação entre o esquema e a placa adaptadora.

De seguida é necessário colocar o Arduíno, o qual se pode verificar pela Figura 37, no *shield*.

Figura 37- Placa de Arduíno [34].

De seguida foi necessário efetuar a ligação do Arduíno às placas que irão ser lidadas aos motores EMG30, os quais se podem verificar na Figura 38.

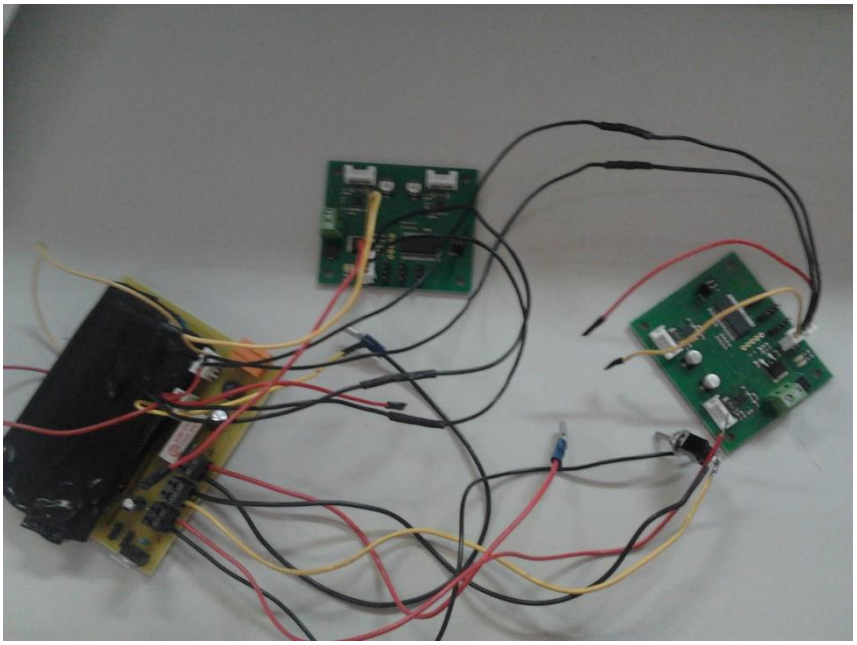


Figura 38- Placas (MD25) referentes ao motor e placa do Arduíno ligadas.

Estes motores, representados na Figura 39, de 12 V, encontram-se totalmente equipados com codificadores[5] e uma caixa de redução 30:1. Este é ideal para aplicações robóticas pequenas ou médias e inclui um condensador de supressão de ruido padrão entre os enrolamentos do motor.

Este tipo de motor tem como especificações:

- Tensão: 12 V;
- Binário: 1.5 Kg/cm;
- Velocidade: 170 rpm;
- Corrente: 530 mA;
- Velocidade em vazio: 216 rpm;
- Corrente sem carga: 150 mA;
- *Stall current*: 2.5 A.

[5] Os codificadores ou *encoders* medem o quanto o motor se desloca para que o controlador md25 saiba como atuar.

Figura 39- Motor EMG30 [35].

Capítulo 7- Conclusão

Neste capítulo serão abordadas as conclusões retiradas ao longo do desenvolvimento do trabalho.

O uso de iodo (^{131}I) é utilizado para o tratamento de patologias relacionadas com a glândula tiroide, nomeadamente o cancro da tiroide. Este tipo de tratamento envolve radiação ionizante, a qual pode causar problemas de saúde aos profissionais que estejam em constante contacto com a radiação.

Para que seja possível diminuir o período de exposição por parte destes profissionais é necessário criar novas alternativas. Por isso, foi desenvolvido um *software* que permite o controlo de um robô omnidirecional sem que seja necessário a presença de um profissional de saúde na sala de tratamento para detetar a radiação presente na sala.

Numa primeira fase foi necessário criar o ambiente de simulação, no *software* do *SimTwo* isto é criar o espaço onde o robô se desloca na realidade, neste caso nas diversas salas de tratamento. Após a criação deste espaço foi construído um robô omnidirecional. Para a obtenção deste robô é necessário o uso de diversos modelos, entre estes os modelos cinemático, dinâmico, do motor. Estes permitem a simulação do comportamento real do robô.

Após a criação do ambiente de simulação, foi criado o *software* que permite com que o robô se desloque de forma autónoma, com uma determinada velocidade e numa determinada direção.

Após terminada esta fase foi criado um sistema que permite o robô deslocar-se para diferentes posições definidas pelo operador sem que este esteja sempre a alterá-las. Além de serem colocadas diferentes posições, também foi necessário que o robô se encontre parado em algumas posições durante a simulação.

Depois de criado este *software*, foi mostrado como funciona a simulação para um exemplo específico. Nesse exemplo é possível verificar que o robô se desloca de forma autónoma pela sala de tratamento. O que se pode concluir que o software criado funciona de forma correta.

Além desta demonstração, também foram analisados os gráficos referentes à velocidade do robô ao longo da simulação. Estes permitiram concluir que o comportamento das velocidades é praticamente constante ao longo do tempo. Além de permitirem analisar o comportamento das velocidades também se consegue ver o comportamento da simulação, isto é quando o robô para numa numa determinada posição a velocidade deste tende a manter-se constante para aquele determinado período de tempo definido.

Por fim também foi iniciado o processo de construção do *hardware*, que diz respeito ao protótipo final do robô.

Para trabalhos futuros seria interessante implementar-se o algoritmo *Perfect Match* (descrito no Capítulo 4 secção 4.3) e a construção final do protótipo.

Bibliografia

[1] Nobel Lectures, "Nobel Prize," Physics 1901-1921, Elsevier Publishing Company, 1967. [Online]. Available: http://www.nobelprize.org/nobel_prizes/physics/laureates/1903/marie-curie-bio.html. [Acedido em 2 Abril 2015].

[2] D. M. Guimarães, "Reclassification of a shielding design project for facilities administering 131I for thyroid carcinoma therapy," Porto, 2014.

[3] V. R. Preedy, G. N. Burrow e R. R. Watson, Comprehensive handbook of iodine: nutritional, biochemical, pathological and therapeutic aspects., 2009.

[4] J. Pisco, Imagiologia Básica- Texto e Atlas, 2º ed., Lisboa: Lidel, 2009.

[5] R. Zimmermann, "Nuclear Medicine: Radioactivity for Diagnosis and Therapy," *EDP Sciences,* 2006.

[6] E. Union, "Official Journal of the European Union," vol. L 013, 2014.

[7] "Decreto Lei nº 222/2008," *Diário da República,* pp. 8003-8004, 17 Novembro 2008.

[8] J. Turner, Atoms, Radiation and Radiation Protection, 3º ed., Oak Ridge:Wiley-VCH, 2007.

[9] M. Machado, V. Menezes, C. Queiroz, D. Silva , L. Sampaio e A. Almeida, "Revisão: radioproteção aplicada à Medicina Nuclear," *Revista Brasileira de Física Médica,* vol. 4, pp. 47-52, 2011.

[10] I.A.E.A., *Radiation, People and Environment,* Austria: IAEA, 2004.

[11] C. Lopes, *Aquisição de dados de leituras de dose para integraçao a sistema de partição de produtos radioativos*, Bragança, 2014.

[12] ATOMTEX, "ATOMTEX, Instruments and Technologies for Nuclear Measurements and Radiation Monitoring," [Online]. Available: http://www.atomtex.com/en/products/portable-dosimeters/at1121-at1123-x-ray-and-gamma-radiation-dosimeters.

[13] M. Pinto, H. Soobreira , A. P. Moreira, H. Mendonça e A. Matos, "Self-Localisation of indoor mobile robots using muti-hypotheses and a matching algorithm," *International Journal of Advances in Engineering & Technology,* vol. 5, n.º 1, pp. 1-12, 2012.

[14] N. Karlsson , E. D. Bernardo, J. Ostrowski, L. Gonçalves, P. Pirjanian e M. E. Munich, "The vSLAM Algorithm for Robust Localization and Mapping," em *Int. Conf. Robotics and Automation,* Pasadena, California, USA, 2005.

[15] J. Borensteín, H. Everett, L. Feng e D. Wehe, "Mobile Robot Psitioning: Sensors and Techniques," *Robotics Systems,* pp. 231-249, 1997.

[16] M. Lauer, S. Lange e M. Riedmiller, "Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-Localization," *RoboCup symposium,* pp. 142-53, Julho 2005.

[17] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey e K. Konolige, "The Office Marathon: Robust Navigation in an Indoor Office Environment".

[18] L. Iocchi, S. Pellegrini e G. D. Tipaldi , "Building multi-level planar maps integrating stereo vision and IMU sensors," Italy.

[19] "Willow Garage," [Online]. Available: https://www.willowgarage.com/pages/contact. [Acedido em 5 Setembro 2015].

[20] C. Coutinho, "Robótica Móvel - Sistema de Condução Autónoma," Lisboa, 2014.

[21] H. Oliveira, "Análise do Desempenho e da Dinâmica de Robôs Omnidireccionais

de Três e Quatro Rodas," Porto, 2007.

[22] J. Gonçalves, J. Lima, P. Oliveira e P. Costa , *Sensor and actuator modeling of s realistic wheeled mobile robot simulator*, Porto, 2008.

[23] A. P. M. a. P. J. C. A. S. Conceição, "Model identification of a four wheeled omni-directional mobile robot," em *Controlo*, Instituto Superior Técnico, Lisboa, 2006.

[24] G. Campion, G. Bastin e B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic," em *IEEE Transactions on Robotics and Automation*, 1996.

[25] H. G. a. P. Young, "Time-domain approaches to continuous-time model identification of dynamical systems from sampled data," em *American Control Conference*, Boston, Massachusetts, 2004.

[26] M. Pinto, A. P. Moreira, A. Matos e H. Sobreira, "Novel 3D Matching Self-Localization Algorithm," *International Journal of Advances in Engineering & Technology*, vol. 5, n.º 1, pp. 1-12, Novembro 2012.

[27] H. Sobreira, M. Pinto, A. P. Moreira, P. G. Costa e J. Lima, "Robust Robot Localization based on the Perfect Match Algorithm," Porto e Bragança.

[28] M. Pinto, H. Soobreira , A. P. Moreira, H. Mendonça e A. Matos, "Self-Localisation of indoor moboli robots using muti-hypotheses and a matching algorithm," *International Journal of Advances in Engineering & Technology,* vol. 5, n.º 1, pp. 1-12, 2012.

[29] *Algoritmo de localização*, Bragança.

[30] P. Costa, "SimTwo," [Online]. Available: http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo.

[31] K. J. Astrom e T. Hagglund, PID Controllers, 2º ed., 1995.

[32] L. F. A. Pereira e J. F. Haffner, *Controladores do tipo Proporcional, Integral e Diferencial*, Pontifícia Universidade Católica do Rio Grande do Sul.

[33] Novus, “Controle PID Básico,” 2003.

[34] Arduido, “Arduido,” [Online]. Available: http://www.arduino.cc/. [Acedido em 20 Abril 2015].

[35] “EMG30, mounting bracket and wheel specification,” [Online]. Available: http://www.robot-electronics.co.uk/htm/emg30.htm. [Acedido em 20 Abril 2015].

Anexo A: Código referente à construção do ambiente de simulação

Neste anexo encontram-se representados o código utilizado para a criação do ambiente de simulação.

O código abaixo diz respeito à identificação do robô assim como dos obstáculos.

```
<?xml version="1.0" ?>
<scene>
   <robot>
    <ID name='Omni3'/>
    <pos x='3' y='4' z='0'/>
    <rot_deg z='0'/>
    <body file='Omni3.xml'/>
  </robot>
   <obstacles file='obstacles.xml'/>
   </robot>
 </scene>
```

Abaixo encontra-se a parte do código que corresponde à criação do corpo do robô.

```
<solids>
    <cylinder>
      <ID value='1'/>
      <mass value='1.8'/>
      <size x='0.09' y='0' z='0.135'/>
      <pos x='0' y='0' z='0.08'/>
      <rot_deg x='0' y='0' z='0'/>
      <color_rgb r='128' g='0' b='0'/>
      <texture name='MatBallTriangle' scale='6'/>
    </cylinder>
      </solids>
```

Código referente à criação dos lasers. Onde numa primeira parte estão representados a constituição física do laser e numa segunda parte as características deste.

```
<shells>
    <cylinder>
      <ID value='LaserRanger'/>
      <size x='0.03' z='0.06'/>
      <pos x='0' y='0' z='0.1'/>
      <rot_deg x='0' y='0' z='0'/>
      <color_rgb r='64' g='64' b='64'/>
     </cylinder>
    </shells>

    <sensors>
    <ranger2d>
      <ID value='ranger2d'/>
      <beam length='4' initial_width='0.015' final_width='0.015'/>
      <period value = '1'/>
      <pos x='0' y='0' z='0.1'/>
      <rot_deg x='0' y='0' z='0'/>
      <tag value='00'/>
      <beam angle='360' rays='360'/>
      <noise stdev='0.00' stdev_p='0' offset='0' gain='1'/>
      <color_rgb r='255' g='0' b='0'/>
    </ranger2d>
</sensors>
```

Abaixo encontra-se representado o código referente à construção do robô omnidirecional, nomeadamente a construção das rodas e características do motor.

```
<wheels>
    <default>
      <omni/>
      <tyre mass='0.8' radius='0.04' width='0.03' centerdist='0.09'/>
      <surface mu='1' mu2='0.001'/>
      <axis angle='0'/>
      <motor ri='0.3' ki='2.4e-2' vmax='12' imax='4' active='1'/>
      <gear ratio='12'/>
      <friction bv='1e-5' fc='1e-3' coulomblimit='1e-2'/>
      <encoder ppr='1000' mean='0' stdev='0'/>
      <controller mode='pidspeed' kp='0.2' ki='0' kd='0.01' kf='0.05'
active='1' period='10'/>
      <color_rgb r='128' g='0' b='128'/>
    </default>

    <wheel>
      <axis angle='-60'/>
    </wheel>
    <wheel>
      <axis angle='60'/>
    </wheel>
    <wheel>
      <axis angle='180'/>
    </wheel>
  </wheels>
</robot>
```

Por fim encontra-se representado parte do código que diz respeito à criação das estruturas referentes às salas de tratamento.

```
<obstacles>
  <cuboid>
    <imovable/>
    <size x='9' y='1' z='3'/>
    <pos x='0' y='5' z='0'/>
    <rot_deg x='0' y='0' z='0'/>
    <color_rgb r='128' g='128' b='128'/>
  </cuboid>
…

</obstacles>
```

Anexo B: Esquema do Robô

Neste anexo encontra-se representado o esquema referente ao robô.

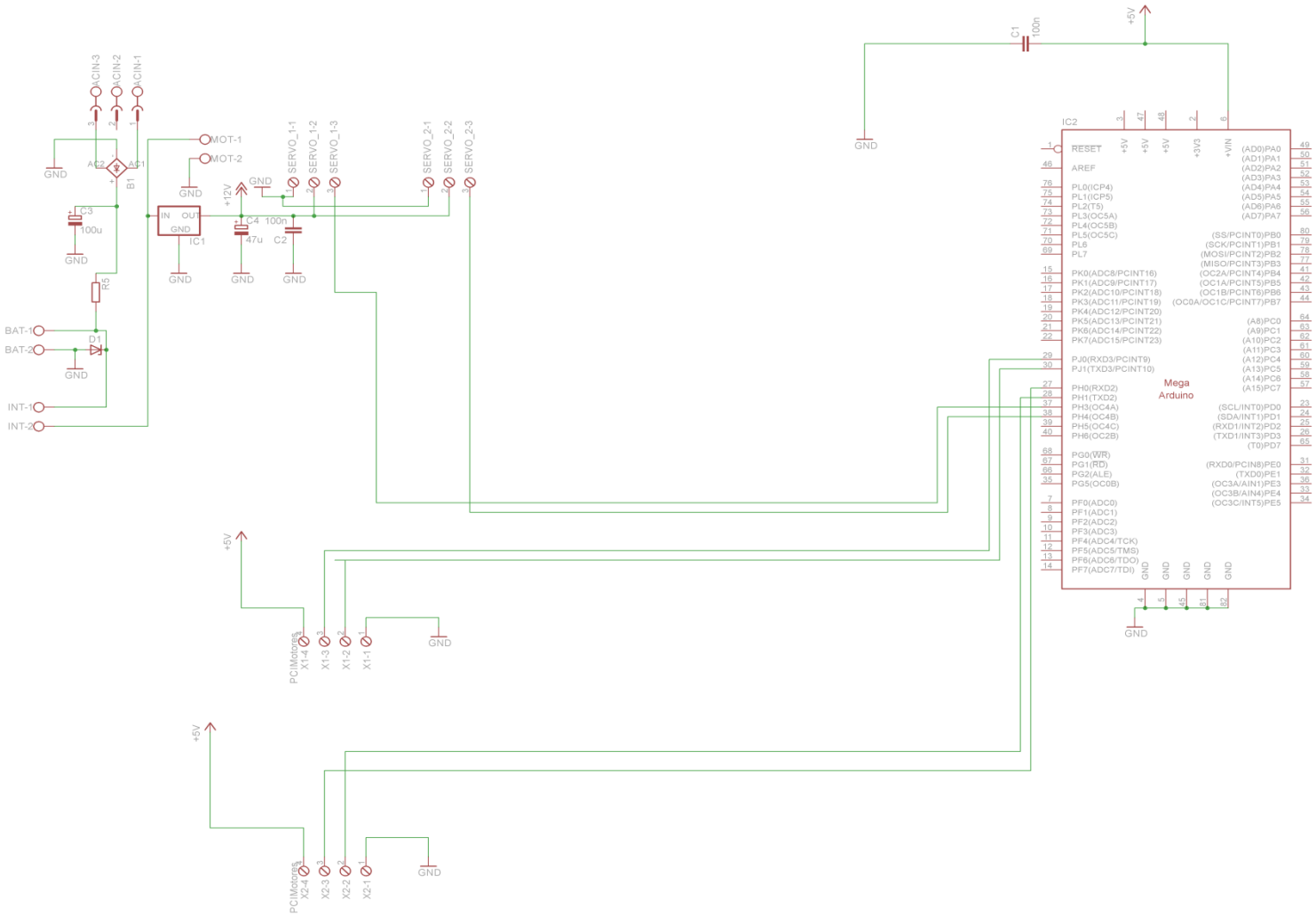


Figura 40- Esquema do robô.