Implementação e Teste de Soluções de Navegação de Robôs Móveis com Base no Sistema NXT/EV3 da LEGO®

Nuno Jorge Ferreira Pinto

Dissertação de Mestrado

Orientadores na FEUP: Prof. Manuel Romano Barbosa

Prof. António Mendes Lopes

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Mecânica

Setembro 2016

” ... Aprender nunca esgota a mente.”

Leonardo da Vinci

Resumo

A robótica móvel é uma área que tem vindo a ser desenvolvida ao longo dos anos, sendo atualmente indispensável em ambientes industriais e estando implementada em grande número de fábricas e armazéns que necessitam de transporte de materiais. Os tradicionais tapetes transportadores deixam de ser tão competitivos devido às vantagens que os robôs móveis apresentam, tais como a possibilidade de se poder alterar o trajeto do robô, recorrendo apenas à programação do mesmo, sem a necessidade de alterar o percurso físico como seria o caso dos tapetes transportadores. Atualmente os robôs móveis são praticamente autónomos, necessitando de pouca intervenção humana nas suas tarefas.

Um dos aspetos mais desafiantes na área da robótica móvel é a navegação e localização do robô no ambiente de trabalho, razão pela qual surgiu esta dissertação. A locomoção de veículos diferenciais assim como a implementação de várias trajetórias e a posterior reconstrução das mesmas com recurso a odometria têm particular ênfase neste trabalho. A implementação das soluções propostas foi feita através das plataformas de baixo custo da Lego® Mindstorms® NXT e EV3, com recurso ao *software* Matlab/Simulink. Foram implementadas duas estratégias de controlo das plataformas, uma com controlo em posição, na qual o movimento dos veículos era limitado pelo número de graus que cada roda devia rodar, e outra com controlo em velocidade, na qual o movimento era limitado por um perfil de velocidade ao longo do tempo total de deslocamento. Para ambas as estratégias, foram implementados perfis de deslocamento em parábola, semicírculo e linha reta. Posteriormente foram analisados os resultados obtidos.

Implementation and Test of Mobile Robot Navigation Solutions Based on Lego® Mindstorms® NXT and EV3 system

Abstract

The mobile robotics field has been under development over the years, currently indispensable in industrial environments, being implemented in a large number of factories and warehouses that requires transport of materials. Traditional conveyors aren't so competitive due to mobile robot's advantages such as the possibility of changing the robot path, using only the programming of the same, without the need of changing the physical route as it would be necessary in the conveyor's case. Currently, mobile robots are almost fully autonomous, requiring little human intervention in their tasks.

One of the most challenging aspects in the mobile robot's field is the navigation and localization of the robot in his workplace and that's the main reason of this dissertation.

Differential vehicles motion as well as the implementation of different trajectories and the subsequent reconstruction of the same using odometry, have particular emphasis in this work.

The implementation of proposed solutions was made through Lego® Mindstorms® NXT and EV3, using Matlab/Simulink software and in the end, the results were analyzed.

Agradecimentos

Aos meus orientadores Prof. Manuel Romano Barbosa e Prof. António Mendes Lopes, por todo o apoio, paciência, disponibilidade e pelas valiosas contribuições ao longo desta dissertação.

Aos meus colegas e amigos, pela companhia, ajuda e motivação em todas as fases deste projeto. Ao Sr. Joaquim pela companhia e ajuda.

Aos meus pais pelo carinho, apoio, ajuda incondicional e confiança depositada em mim ao longo de toda a minha vida académica.

Obrigado.

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

1 Introdução

No século XX, com a necessidade de aumentar a produtividade e melhorar a qualidade dos produtos, surgiu a ideia de se construírem robôs. Desta forma, determinadas tarefas poderiam ser realizadas com maior velocidade, precisão e segurança. Os primeiros robôs industriais começaram a ser utilizados na década de 1960 e rapidamente se percebeu que seriam o futuro da indústria. Com a pesquisa e desenvolvimento realizados entre 1960 e 1975, e com o aparecimento do microprocessador, os robôs industriais tornaram-se mais versáteis e eficientes, provando serem de enorme valia na produção industrial [1].

Com o desenvolvimento tecnológico dos últimos anos, a qualidade de vida dos humanos sofreu um significativo aumento. A utilização de veículos autónomos tem vindo a ser cada vez mais utilizada para auxiliar o homem ou mesmo substituí-lo em diversas tarefas, tais como transporte de mercadorias, limpeza e manutenção de certos ambientes, segurança, exploração de ambientes perigosos ou nocivos ao homem, em tarefas repetitivas ou de precisão [2].

Existem dois grandes grupos no que diz respeito ao tipo de robôs: os robôs manipuladores e os robôs moveis. Os manipuladores robóticos são utilizados maioritariamente no sector industrial. Na classe dos robôs móveis, inserem-se vários tipos de robôs com a capacidade de se moverem no espaço tridimensional (robôs aéreos e aquáticos) ou no plano (robôs terrestres).

Um dos principais motivos de nos últimos anos se notar um maior aumento de aplicações envolvendo robôs móveis, deve-se à capacidade de estes se moverem livremente no espaço em que estão inseridos, limitados apenas por eventuais obstáculos.

Para que um sistema robótico seja eficiente, é preciso torná-lo suficientemente autónomo na execução das suas tarefas. Robôs móveis devem apresentar um comportamento autónomo, recorrendo ao uso de vários sensores para operarem sem falhas, reagirem a mudanças de ambiente, serem robustos, flexíveis e adaptáveis ao ambiente em que estão inseridos.

Roland Siegwart e Illah R. Nourbakhsh (2004) dividem o processo de navegação em quatro níveis hierárquicos, conforme mostra a Figura 1:

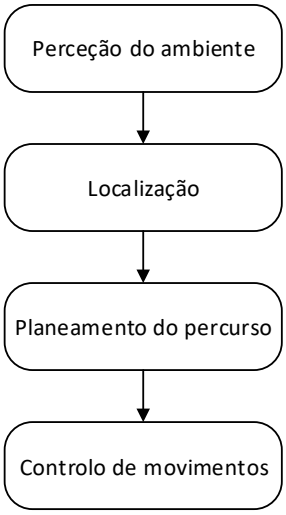

Figura 1 – Arquitetura de navegação de um robô móvel

- Perceção do ambiente: inicialmente, o robô recolhe informações dos sensores com os quais está equipado;
- Localização: são determinadas, a posição e orientação do robô no espaço, com base no modelo obtido na perceção;
- Planeamento do percurso: o controlador do robô calcula qual a curva geométrica a realizar para atingir a localização final, evitando possíveis obstáculos e cumprindo as restrições temporais impostas;
- Controlo de movimentos: são calculadas as velocidades que devem ser atribuídas aos motores, por forma a garantir que o robô cumpra a trajetória calculada no intervalo de tempo especificado.

Dos quatro níveis hierárquicos que compõem o processo de navegação de um robô móvel, este trabalho será mais focado na localização do veículo no seu meio ambiente.

1.1 Enquadramento e Motivação

O presente trabalho enquadra-se na área da movimentação autónoma de um robô/veículo móvel com rodas de baixo custo.

Nesta área, são feitos estudos e desenvolvimentos que possam ser testados e posteriormente aplicados em veículos, de forma a estes serem dotados de capacidades de planeamento e seguimento de trajetórias para que alcancem, não só os objetivos propostos, como também melhorem a segurança em ambientes fabris.

A navegação é das capacidades mais desafiantes para os robôs móveis. É necessário que um robô tenha a capacidade de se deslocar de forma eficaz e segura, entre o ponto de origem e o de destino. Para que tenha sucesso, é necessário que a perceção (interpretação dos dados que os sensores transmitem ao robô), localização (determinação da posição do robô em relação ao meio ambiente), planeamento de trajetórias (o robô deve decidir como agir e atingir os objetivos) e controlo de movimento (atribuição dos valores a enviar para os motores) estejam em perfeita sintonia.

Estando a área da robótica em constante evolução, apresentando cada vez mais novas potencialidades, o desenvolvimento de um trabalho focalizado nesta temática torna-se motivador.

1.2 Objetivos do Projeto

Atualmente, em ambiente industrial, é recorrente a utilização de veículos que se movimentam de forma autónoma. Com o presente trabalho pretende-se estudar, analisar e implementar diferentes soluções de navegação de veículos guiados autonomamente (AGVs) através dos sistemas de baixo custo NXT/EV3 da LEGO®. Para esse efeito é utilizado o *software* Matlab/Simulink como forma de desenvolvimento de programas para implementar nos veículos, sendo uma plataforma de programação com mais capacidades em relação ao *software* da LEGO.

Pretende-se desenvolver uma solução para localização dos veículos em questão com base em odometria.

Após o desenvolvimento da solução para a localização dos veículos, serão realizados testes a partir da deslocação dos mesmos, ao longo de diferentes trajetórias. Assim, será possível analisar a importância e os problemas da odometria, fazer uma comparação entre os dois sistemas utilizados (NXT e EV3) e analisar as capacidades do *software* utilizado (Matlab/Simulink).

1.3 Estrutura do Relatório

Este documento encontra-se dividido em seis capítulos.

No presente capítulo, **Capítulo 1 – Introdução**, é feita uma introdução ao trabalho e são apresentadas as motivações, enquadramento do trabalho e objetivos.

Capitulo 2 – Estado da Arte – É apresentada uma evolução do estado da arte no que diz respeito às principais áreas científicas envolvidas neste trabalho.

Capitulo 3 – Modelo Cinemático de um Veículo Diferencial – Faz-se a descrição dos conceitos teóricos necessários para o desenvolvimento e implementação de um sistema de condução autónoma.

Capitulo 4 – Descrição das Plataformas Utilizadas – É feita uma apresentação das plataformas da Lego Mindstorms NXT/EV3 e do *software* Matlab/Simulink.

Capitulo 5 – Desenvolvimento, Testes e Resultados – São apresentados os algoritmos implementados para a obtenção das trajetórias e os resultados obtidos.

Capitulo 6 – Conclusões e Perspetivas de Trabalho Futuro – São retiradas as conclusões obtidas ao longo do trabalho realizado.

2 Estado da Arte

Neste capítulo é apresentado o estudo realizado sobre os desenvolvimentos ocorridos no universo da robótica móvel no que diz respeito às principais áreas científicas relacionadas com este trabalho. É feita uma breve introdução da evolução da robótica, dando particular ênfase aos robôs móveis e às suas caraterísticas principais, assim como à localização de veículos autónomos com base em odometria.

Estando inserida no campo tecnológico que agrupa computadores, robôs e automação, a robótica é um termo que já não é desconhecido, devido ao elevado uso de sistemas cada vez mais autónomos, nomeadamente no ramo industrial, mas também em ambientes mais familiares.

A origem da palavra robô ("Robota") data de 1921 e foi usada pelo escritor checo Karel Capek na obra de ficção cientifica R.U.R. (Rossum's Universal Robots). Em checo, "Robota" significa trabalho forçado ou servidão [2].

Existem várias opiniões do que é um robô. Por exemplo, Joseph Engelberger, pioneiro em robôs industriais, disse uma vez: "Eu não consigo definir um robô, mas consigo reconhecer um quando o vir" [3].

De acordo com a *Encyclopaedia Britannica,* um robô é qualquer dispositivo operado automaticamente que substituí o esforço humano [4].

Já a *Robotics Industries Association* diz que um robô pode ser considerado como um dispositivo mecânico articulado reprogramável, que de forma autónoma e recorrendo à sua capacidade de processamento, consegue obter informação do meio envolvente utilizando sensores e tomar decisões sobre as suas ações com base nessa informação e com informação previamente estipulada, sendo ainda capaz de manipular objetos utilizando atuadores [5].

O primeiro robô industrial (Unimate) foi produzido em 1961 (ano em que a primeira patente de robôs foi aprovada) pela empresa Unimation, fundada por George Devol e Joseph Engelberger [6].

Apesar de a indústria dos robôs manipuladores ser um sucesso e compreender um mercado de grandes dimensões, estes apresentam uma grande desvantagem: a falta de mobilidade [7].

Devido a essa limitação, apareceu a robótica móvel. Enquanto que maioritariamente os manipuladores robóticos destinam-se à indústria de produção, os robôs móveis têm vindo a ser utilizados em diversos ambientes, sejam eles industriais, urbanos, familiares ou mesmo ambientes hostis, como por exemplo a exploração espacial.

2.1 Veículos Automatizados do Tipo AGV

No ramo industrial, o termo normalmente utilizado para um robô móvel é AGV (automated guided vehicle), i.e., veículo guiado automaticamente.

Os AGVs têm vindo a ser das soluções mais utilizadas no transporte de material em fábricas de produção ou em armazéns, permitindo às empresas que optam por estes sistemas, um aumento da produtividade devido ao melhor aproveitamento do tempo, espaço, controlo da produção e expedição de produtos, pois a necessidade da interação de um operador com a linha de produção é reduzida.

Com a natural evolução da tecnologia, mais concretamente na área da movimentação autónoma, a utilização de AGVs tem vindo a aumentar, assim como o seu campo de aplicação (Figura 2).

Figura 2 – Linha de AGVs no porto de Hamburgo [8]

O primeiro AGV, desenvolvido pela Barrett Electronics Corporation, apareceu no mercado na década de 1950 e não passava de um camião reboque que seguia um fio condutor inserido no pavimento de uma fábrica. A localização do veículo recorria a odometria e calibração através de marcas localizadas ao longo do percurso [9].

Devido ao facto de este tipo de AGV necessitar de uma instalação de percursos físicos pré-definidos (fio condutor inserido no pavimento), torna-se limitado em relação aos movimentos que pode efetuar, levando a cabo o desenvolvimento de novos veículos com o objetivo de ultrapassar estas limitações.

Entre 1966 e 1972, foi desenvolvido no centro de inteligência artificial do instituto de pesquisa de Stanford, o robô Shakey (Figura 3), o primeiro a recorrer a um computador para controlar os seus movimentos. Recorrendo a uma câmara para perceção de visão, sensores ultrassónicos e de colisão táteis, navegava em ambientes altamente estruturados, como por exemplo escritórios.

Figura 3 – Robô Shakey [10]

O robô navegava pelas salas, emitindo e recebendo sinais de rádio da unidade de processamento externa. Enquanto que as informações sensoriais eram recolhidas a bordo do robô, o processamento de dados e posteriores ações de comando eram feitos exteriormente [10].

Devido ao êxito deste projeto, a pesquisa nesta área tornou-se constante como seria de esperar. Na década de 1970, a NASA desenvolveu o Lunar Rover, projetado para exploração espacial. Estava equipado com uma câmara de visão e diversos sensores, sendo capaz de se locomover em terrenos hostis [11].

Com o natural aumento das capacidades de processamento de dados dos computadores, o desenvolvimento desta área da robótica tornou-se mais aliciante. Os robôs começaram a apresentar cada vez mais potencialidades, tornando-se progressivamente capazes de encontrar soluções para os seus próprios problemas, permitindo-lhes circular em ambientes sem supervisão externa.

Atualmente os robôs do tipo AGV têm a capacidade de se movimentarem através de percursos virtuais recorrendo a sensores para medição da distância dos veículos a posições conhecidas no ambiente de trabalho, sistemas de visão digital e unidades de navegação inercial, proporcionando, com a junção de um mapa do ambiente no qual está inserido, a capacidade de conhecer a cada instante, a localização do veículo.

Apesar de os AGVs serem cada vez mais autónomos, em ambientes fabris são utlizados vários veículos, sendo necessário utilizar sistemas de controlo e gestão de frota (Figura 4).

Estes sistemas são responsáveis por:

- Gestão das ordens para os veículos;
- Controlo de tráfego;
- Interface com outros equipamentos;
- Definição dos *layouts* e percursos.

O computador central analisa e gere a frota de veículos, sincronizando-os com a linha de produção/transporte permitindo um fluxo de veículos ininterrupto e em caso de alerta, emite uma notificação ao utilizador, como por exemplo um veículo que demore mais tempo do que o esperado numa determinada zona, podendo estar encravado.

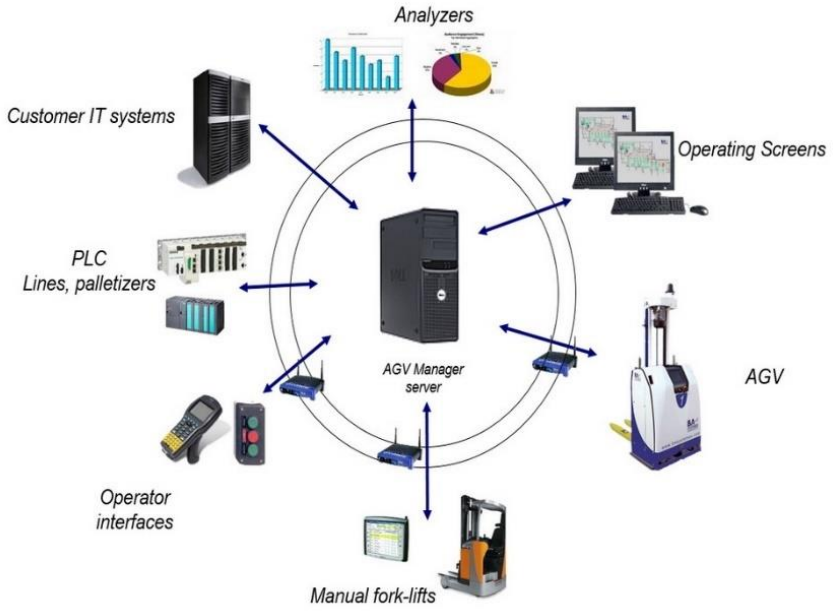

Figura 4 – Sistema de gestão de AGVs [12]

Fabricantes de AGVs, como por exemplo a *Egemin Automation* [13] ou a *Swisslog* [14] disponibilizam *software* próprio para esse controlo.

Na Figura 5, é possível observar os principais componentes que equipam os AGVs, neste caso do fabricante *KLAS*, tais como: um modem para transmissão de dados por rádio frequência, um dispositivo para controlo manual no caso de ser necessário, controlador do veículo, sistema de guiamento, sistema de locomoção e bateria. Os veículos são ainda dotados de botões de emergência e por vezes um LCD para visualização de dados.

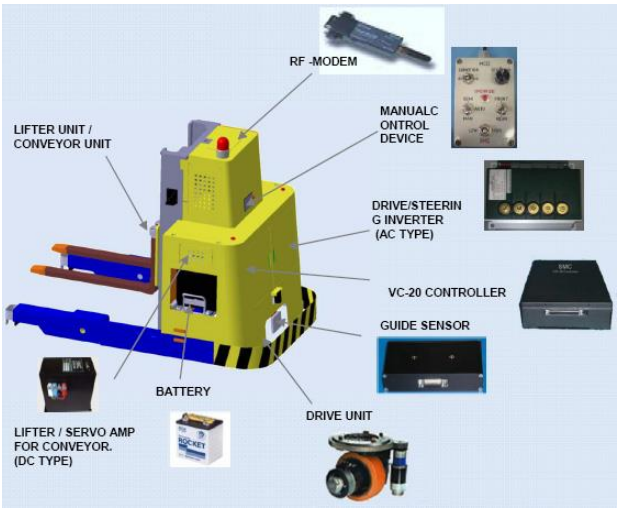

Figura 5 – Exemplo de componentes de um AGV [15]

Como os AGVs mais atuais estão bastante desenvolvidos, é necessário fazer uma abordagem mais geral ao mundo da robótica móvel.

2.2 Robôs Móveis

Tal como já foi referido no capítulo 1, as principais tarefas de um robô móvel, ilustrados na Figura 6, são [7]:

- Perceção do meio que o rodeia;
- Localização do robô em relação ao ambiente em que está inserido;
- Planeamento de trajetórias para chegar ao destino;
- Locomoção.

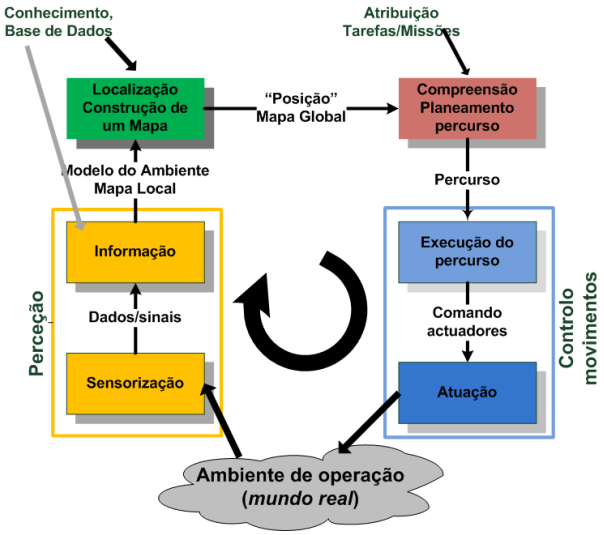

Figura 6 – Configuração de um robô móvel, adaptado de Roland Siegwart & Illah Nourbakhsh

Sistemas de Locomoção

O meio de locomoção de um robô é selecionado dependendo do ambiente no qual este está inserido, de forma a realizar as tarefas às quais é proposto da forma mais eficiente.

No processo de definição de qual o melhor método de locomoção, vários fatores são tidos em conta:

- Estabilidade: número de pontos de contato, centro de gravidade, inclinação do terreno;
- Caraterísticas dos pontos de contato: dimensões do ponto, ângulo de contato, fricção;
- Tipo de ambiente: água, ar, terra;
- Eficiência energética;
- Manutenção.

No diagrama em árvore apresentado na Figura 7, é possível observar os tipos de locomoção de acordo com o ambiente:

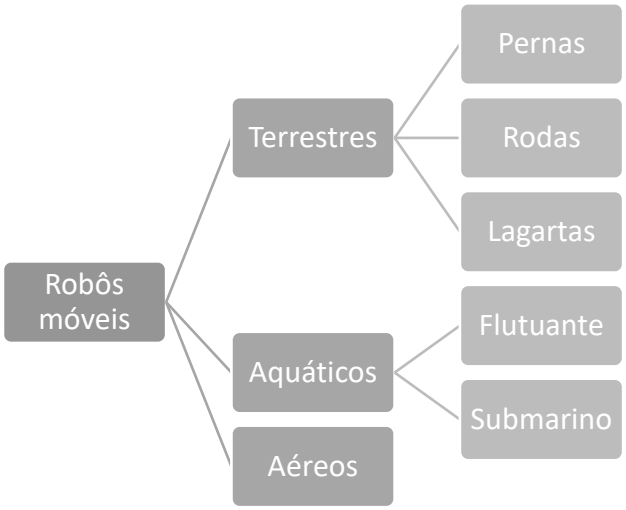

Figura 7 – Tipos de robôs móveis

A locomoção de veículos terrestres dá-se normalmente com recurso a rodas ou a um pequeno número de pernas articuladas.

O movimento por meio de rodas apresenta um rácio velocidade/potência superior aos outros sistemas, dependendo do terreno (tire on soft ground), além de ser a que utiliza um sistema mecânico mais simples no que diz respeito à sua implementação. No entanto, em terrenos irregulares, apresenta algumas dificuldades (figura 8).

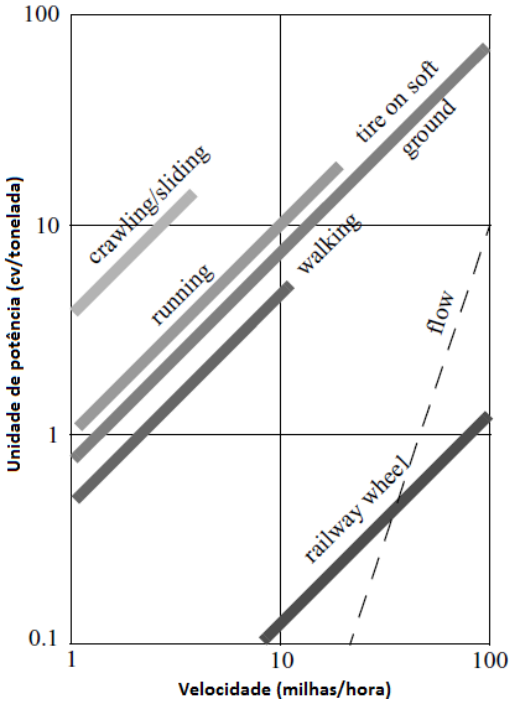

Figura 8 – Comparação dos vários mecanismos de locomoção, tendo em conta o terreno e a potência vs. velocidade [7]

O modelo cinemático de um robô móvel é determinado, não só pelo seu sistema de tração, mas também pela configuração, tipo e número de rodas utilizadas.

Tipo de Rodas

A mobilidade de robôs móveis com rodas é resultado da junção de dois aspetos fundamentais, o tipo de rodas com as quais um robô está equipado e a sua configuração na plataforma.

As rodas mais utilizadas na robótica móvel podem ser observadas na Figura 9 e são basicamente de dois tipos, convencional (a e b) e omnidirecional (c e d). A roda *standard* (a), tem o seu eixo fixo na estrutura do robô e está normalmente associada ao sistema de tração do robô.

A roda orientável não centrada (roda livre) ou roda castor (b), é uma roda orientável que se encontra descentrada do seu ponto de fixação na plataforma, ou seja, a rotação do plano da roda ocorre em redor de um eixo vertical que não passa pelo centro da roda. É utilizada para estabilizar a estrutura mecânica do robô.

A roda omnidirecional ou roda "*swedish"* (c) tem a particularidade de permitir tração na direção perpendicular ao eixo do motor, permitindo deslizar na direção do seu eixo.

Durante a deslocação de um robô equipado com estas rodas, podem ser combinados movimentos de translação e rotação, permitindo-lhe chegar ao ponto estipulado com o ângulo desejado.

A roda esférica (d), sendo a única verdadeiramente omnidirecional, permite movimento em qualquer direção.

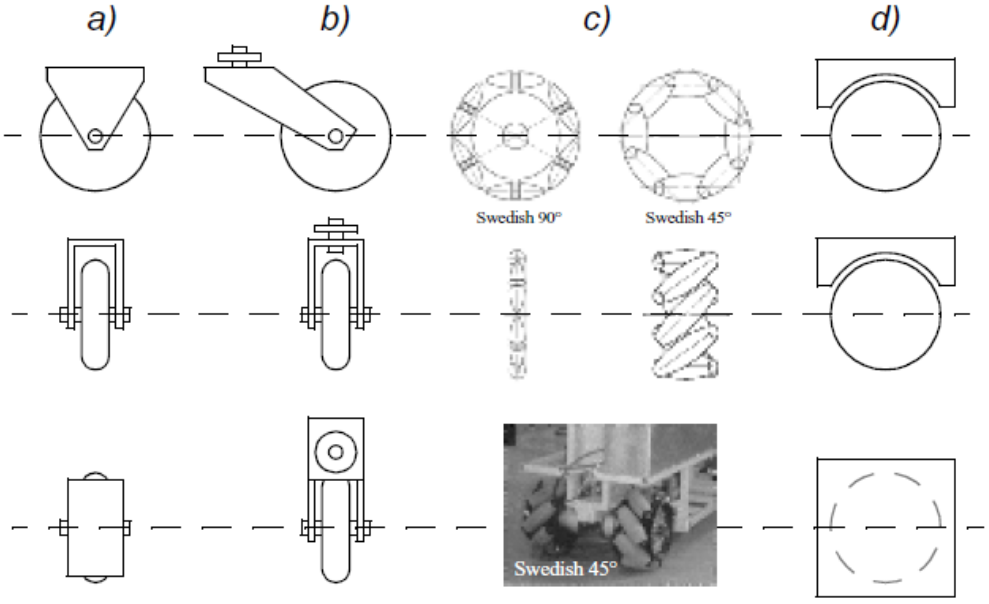

Figura 9 – Tipos de rodas básicos utilizados na robótica móvel [15]

A roda *standard* comparativamente com a roda castor, absorve forças laterais que são transmitidas à plataforma.

As rodas omnidirecionais são as que apresentam mais graus de liberdade, relativamente à direção que o robô pode tomar.

Número e Configuração das Rodas

Conforme já foi referido, a manobrabilidade de um determinado robô móvel está associada à sua capacidade de se orientar/deslocar e resulta da conjugação não só do tipo de rodas, mas também do número e disposição no veículo.

Em robôs com locomoção por rodas, destacam-se os diferentes pontos de apoio (rodas) da plataforma ao pavimento. As configurações de três apoios têm como vantagem, maior adaptabilidade quando o terreno é irregular. Por exemplo, um veículo com quatro pontos de apoio, é automaticamente mais robusto e apresenta maior capacidade de carga em comparação com a configuração anterior, mas em contrapartida, poderá, em caso de alguma irregularidade no terreno e se não forem suficientemente sofisticados (inexistência de um sistema de compensação amortecedor ou forma de distribuição de tração diferencial), ficar com um ou mais pontos de apoio suspensos no ar, levando a que o robô possa entrar numa situação de paragem.

De seguida apresentam-se algumas configurações típicas de robôs móveis:

- **Configuração Omnidirecional**

Robôs do tipo omnidirecional apresentam uma manobrabilidade total no plano **xy**, ou seja, têm a capacidade de se moverem em qualquer direção sem a necessidade de reorientação da plataforma. A utilização de rodas "*swedish"* é praticamente obrigatória pois como já foi referido, permitem que o veículo deslize na direção perpendicular ao eixo de tração. No entanto, o controlo torna-se mais complexo comparativamente por exemplo, a um veículo diferencial, devido ao facto de ser necessário controlar um maior número de rodas, ver Figura 10.

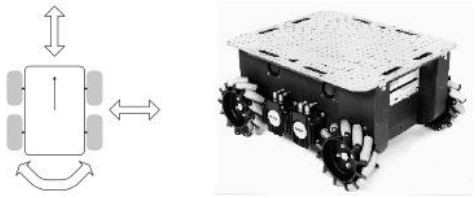

Figura 10 – Robô omnidirecional Uranus e respetiva manobrabilidade [7]

- **Configuração *Synchro Drive***

Uma configuração particular de robôs, popular em ambientes *indoor*, é a chamada *Synchro drive*, a qual apresenta uma montagem de três rodas *standard* direcionais ligadas por duas correias dentadas. Uma das correias está acoplada a um motor que transmite potência para as 3 rodas, enquanto a outra está conectada a um motor que é responsável pela direção do veículo. Apesar de o robô ter a capacidade de se mover em qualquer direção, não se consegue controlar a orientação da plataforma, ver Figura 11.

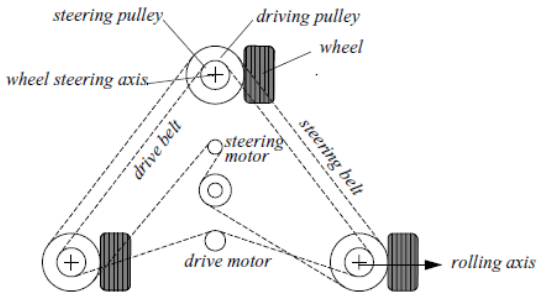

Figura 11 – Configuração Synchro drive [7]

- **Configuração Diferencial**

Este tipo de configuração é, provavelmente, o mecanismo mais simples para a locomoção e controlo de um veículo. Não permite movimento perpendicular ao eixo das rodas, sendo necessário realizar um movimento de rotação para orientar a plataforma para o ponto de destino. A tração é feita com recurso a dois motores associados a cada uma das rodas motoras, montadas segundo o mesmo eixo, e são auxiliados por uma (ou mais) roda livre, a qual fornece estabilidade à estrutura. Sendo de construção e controlo simples, os robôs diferenciais permitem raios de curvatura equivalentes à metade do comprimento dos seus entre eixos. Aplicando a mesma velocidade aos dois motores o robô desloca-se em linha reta, se a velocidade for inversa, gira sobre o centro do entre eixo e se uma das rodas estiver parada e a outra com velocidade, o robô executa uma circunferência de raio igual ao seu entre eixo (Figura 12).

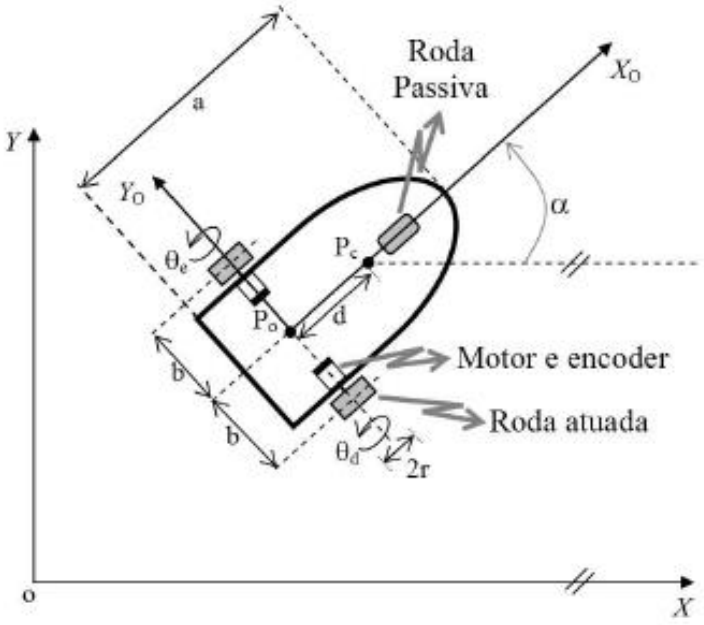

Figura 12 – Configuração diferencial

2.3 Localização

Na robótica móvel, a localização é um dos principais problemas e tem vindo a ser objeto de estudo nas últimas décadas, existindo atualmente várias soluções, sendo uma delas, a localização por odometria. Este processo, aliado a marcadores externos (magnetes) e sensores inerciais (giroscópios, acelerómetros), permite obter bons resultados.

O conceito de localização por odometria foi descrito por Vitruvius no século I ac. Por volta do ano 1500, Leonardo da Vinci construiu um artefacto que usava pedras para calcular a distância percorrida (Figura 13 e 14) [2].

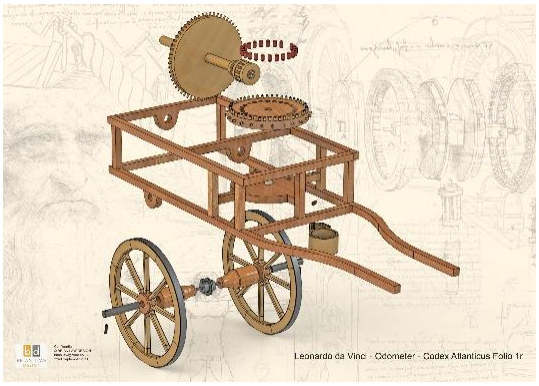

Figura 13 – Modelo do odómetro [16]

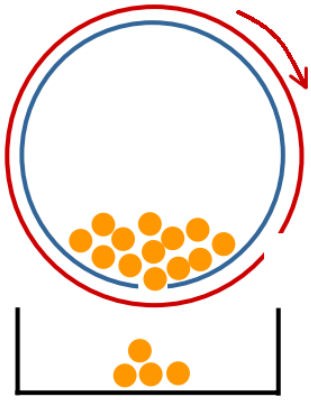

Figura 14 – Exemplo de funcionamento do odómetro

A rotação de uma das rodas origina a rotação de um tambor em torno de um contentor. O tambor e o contentor têm um orifício com as mesmas dimensões, pelo que, após um certo número de rotações das rodas, os dois orifícios coincidem, levando à queda de uma pedra, armazenada numa caixa. Ao fim de um intervalo de tempo, o número de pedras presentes na caixa, permite estimar a distância percorrida, como é possível observar na Figura 14 [2].

A odometria consiste na determinação da posição e orientação de um robô, através da integração dos deslocamentos incrementais das suas rodas, medidos em relação a um referencial fixo, ao longo do tempo. Na Figura 15, é exemplificado este método, considerando um robô que se move em linha reta ao longo do tempo.

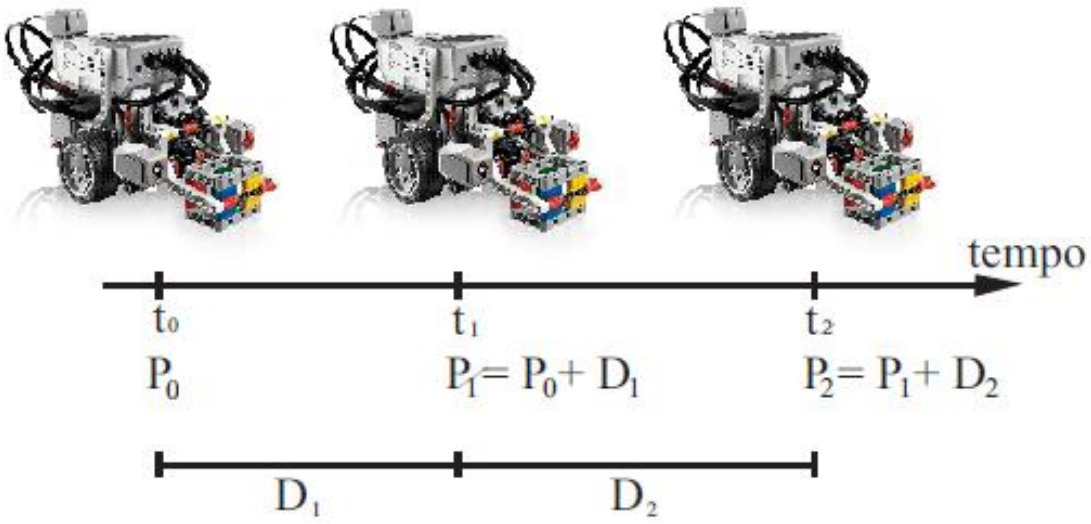

Figura 15 – Deslocamento de um robô medido por odometria

Percebe-se que a posição atual do robô pode ser estimada com base no acúmulo das distâncias percorridas por ele em cada incremento, em relação à sua posição inicial.

Para obter as distâncias percorridas por ambas as rodas, é necessário a utilização de sensores que meçam as rotações das mesmas. Usualmente, em aplicações envolvendo robôs móveis, são usados encoders óticos.

A técnica de odometria é passível de erros, permitindo fazer apenas uma estimativa da localização do robô num dado instante. A acumulação de erros de orientação causa erros na posição, os quais aumentam com a distância percorrida pelo robô. Se num dado instante ocorrer um erro, esse erro vai comprometer as medições dos instantes seguintes.

São consideradas duas categorias de erros, erros sistemáticos e não sistemáticos. Os erros sistemáticos são provenientes de algumas diferenças relativamente ao modelo cinemático do veículo, como por exemplo, diferentes diâmetros das rodas, desalinhamento das rodas, etc. Em relação aos erros não sistemáticos, são causados por situações inesperadas tais como imperfeições no solo, escorregamento de rodas (pavimento escorregadio, aceleração exagerada, etc.) e fenómenos similares [17].

Os erros de odometria, são representados por uma região elíptica em torno da posição atual do robô conforme ilustra a Figura 16. Essa elipse vai aumentando consoante a distância percorrida [17].

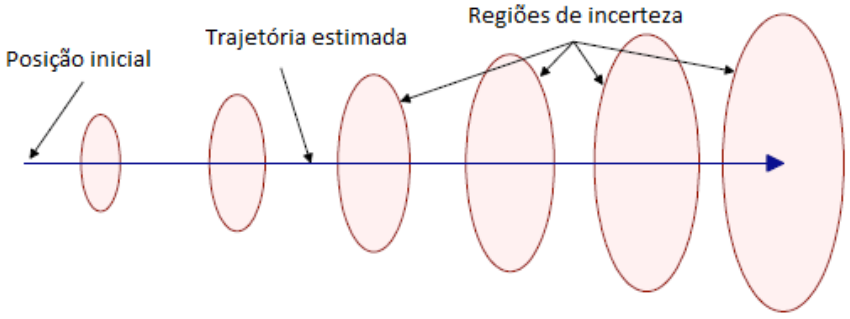

Figura 16 – Regiões de incerteza ao longo da trajetória de um robô adaptado de Borenstein, J., Everett, H.R., Feng, L. 1996

A correção dos erros é possível, recorrendo a outro tipo de sensores que permitam fazer um "reset" periódico aos erros odométricos.

Usualmente, os robôs móveis são equipados com vários tipos de sensores, por forma a recolherem informação do ambiente em que estão inseridos e reagirem a eventuais imprevistos na navegação. Os sensores dividem-se em dois eixos funcionais: propriocetivos (internos)/exterocetivos (externos) e passivos/ativos [7].

Propriocetivos/exterocetivos, refere-se à origem das informações sensoriais obtidas. Propriocetivos medem valores internos do robô, como por exemplo a velocidade do motor ou a carga da bateria. Já os sensores exterocetivos obtêm informações do exterior tais como distâncias, intensidade de luz ou som.

Em relação a serem passivos/ativos, depende da forma como a recolha de informação é efetuada. Se o sensor receber apenas informação do exterior, classifica-se de passivo. Já se o sensor necessitar de emitir energia para o ambiente, sendo a resposta deste a essa energia a informação que recebe, classifica-se de ativo.

Sensores de orientação, como por exemplo o giroscópio ou o acelerómetro, são propriocetivos, enquanto que a bússola é exterocetiva. Normalmente determinam a orientação e inclinação do robô para, em conjunto com a informação da aceleração/velocidade, possibilitarem a determinação da posição atual.

Por forma a detetar obstáculos, é corrente a utilização de sensores de medição de distâncias, como sensores de ultrassons ou laser.

2.4 Navegação

A navegação de um robô móvel consiste na capacidade que este apresenta em se movimentar num determinado ambiente, planeando qual a melhor trajetória a adotar e detetando/evitando obstáculos ou qualquer imprevisto que possa surgir ao longo da sua trajetória.

Tal como já foi referido no capítulo 1, é necessário que a perceção, localização, planeamento do percurso e controlo de movimento, estejam em perfeita sintonia.

Conhecendo o ponto de partida e de chegada, o robô tem de ter a capacidade de identificar qual a melhor trajetória a realizar de modo a alcançar os resultados desejados.

Quanto à deteção de obstáculos, o robô tem de analisar em tempo real os valores obtidos dos sensores assim como ir atualizando e modelando as trajetórias durante a sua execução.

3 Modelo Cinemático de um Veículo Diferencial

Com o objetivo de conhecer as capacidades e limitações de um veículo diferencial, de modo a determinar a velocidade que deve ser aplicada aos atuadores de cada roda e garantir o movimento da plataforma móvel desejado, é necessário conhecer o seu modelo cinemático, de modo a projetar e desenvolver robôs móveis para determinadas tarefas e criar programas de controlo eficazes.

O estudo do movimento, velocidade e aceleração de robôs tem sido analisado ao longo dos anos, principalmente de robôs manipuladores. Nos anos mais recentes, a comunidade robótica atingiu um vasto conhecimento da cinemática e da dinâmica de robôs manipuladores.

A cinemática de robôs móveis apresenta alguns aspetos em comum com robôs manipuladores. O conhecimento do espaço de trabalho de um manipulador robótico é crucial pois define a gama de posições que podem ser alcançadas pela sua extremidade em relação ao seu ponto de fixação no meio ambiente. Na robótica móvel, é igualmente importante porque vai definir a gama de "poses" (posição e orientação do veículo) possíveis que o robô pode atingir no ambiente de trabalho.

A grande diferença entre ambos é a necessidade de se conhecer, em qualquer instante de tempo, a posição atual de um robô móvel, enquanto que no manipulador, como se encontra fixo no seu espaço de trabalho, apresenta sempre um referencial absoluto.

Acrescentando o facto de existirem fatores externos possíveis de influenciar o movimento do robô, tais como por exemplo, o escorregamento das rodas em relação ao pavimento, é passível de notar que, estimar de forma precisa a posição de um robô é extremamente desafiante.

3.1 Caraterísticas de um Veículo Diferencial

Os veículos diferenciais utilizados ao longo deste trabalho apresentam uma configuração diferencial de três apoios, duas rodas motrizes independentes e uma roda livre (NXT) ou esférica (EV3). Ambos os veículos foram considerados como corpos rígidos, ignorando graus de flexibilidade adicionais devido aos eixos das rodas, juntas das rodas de direção e articulações das rodas de apoio. Desta forma, são considerados apenas três graus de liberdade, movimento segundo **xy** e θ, correspondente ao ângulo de rotação segundo o eixo ortogonal ao plano **xy**.

Inicialmente, com o objetivo de especificar a posição do veículo no plano, é estabelecida uma relação entre o referencial global $\mathbf{X_I Y_I}$, correspondente ao plano no qual o veículo se vai deslocar e o referencial local do veículo $\mathbf{X_R Y_R}$. O ponto de referência do veículo considerado foi o ponto P, origem do referencial local na plataforma móvel, com orientação do eixo $\mathbf{X_R}$ (Figura 17). A posição deste ponto no referencial global é especificada pelas coordenadas *x*, y e a diferença angular entre ambos os referenciais é dada por θ. Desta forma, a "pose" do veículo pode ser descrita por um vetor com três componentes no referencial de inércia (fixo):

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

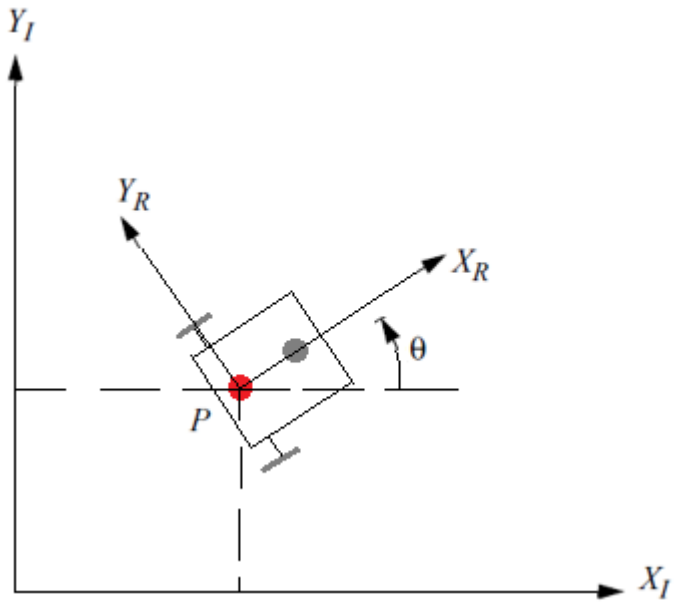

Figura 17 – Referenciais global (fixo) e local (móvel) implementados

Movimentos simples do robô, como por exemplo, deslocamento linear (segundo $\mathbf{X_R}$) ou angular (segundo θ) são obtidos respetivamente impondo velocidades iguais em ambas as rodas, fazendo com que o veículo se desloque na direção de X_R, ou, velocidades simétricas, originando um movimento de rotação do robô θ_R em torno do ponto P.

Para implementar trajetórias mais complexas é necessário ter em conta as restrições cinemáticas dos veículos diferenciais.

3.2 Modelo de um Veículo Diferencial

Em ordem a fazer as transformações necessárias para conhecer o movimento do robô no referencial global, é necessário recorrer à matriz de rotação ortogonal R(θ):

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Desta forma, é possível determinar a velocidade do veículo no referencial local:

$$\dot{\xi}_R = R(\theta) \times \dot{\xi}_I \quad (2)$$

Onde, $\dot{\xi}_R = [\dot{x}_R \quad \dot{y}_R \quad \dot{\theta}_R]^T$ é a velocidade expressa no referencial local e $\dot{\xi}_I = [\dot{x}_I \quad \dot{y}_I \quad \dot{\theta}_I]^T$ é a velocidade expressa no referencial global.

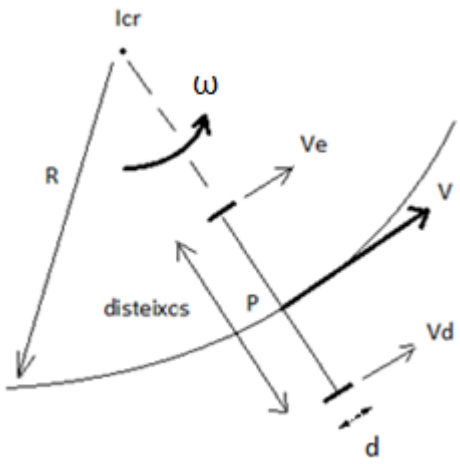

Figura 18 – Exemplo do movimento de um robô diferencial com as respetivas variáveis

Analisando a Figura 18, sendo o ponto P o centro do entre eixo da plataforma, a distância entre eixos (disteixos), os diâmetros de cada roda (d_e e d_d), a velocidade angular e linear da plataforma (ω e V), as velocidades lineares de cada roda (v_e e v_d), as velocidades angulares ($\dot{\phi}_e$ e $\dot{\phi}_d$) e a distância do ponto P ao centro de rotação (R), é possível obter o modelo cinemático do robô:

$$V = \omega . R \tag{3}$$

$$v_e = \omega . \left(R - \frac{disteixos}{2} \right) \tag{4}$$

$$v_d = \omega . \left(R + \frac{disteixos}{2} \right) \tag{5}$$

$$v_e = \dot{\phi}_e . \frac{d_e}{2} \tag{6}$$

$$v_d = \dot{\phi}_d . \frac{d_d}{2} \tag{7}$$

Das equações 4 e 5, resulta outra equação que permite determinar R e ω:

$$R = \frac{disteixos}{2} . \left(\frac{Ve + Vd}{Vd - Ve} \right) \tag{8}$$

$$\omega = \frac{v_d - v_e}{disteixos} \tag{9}$$

Com as equações 3, 8 e 9, é determinada a velocidade linear da plataforma V:

$$V = \frac{v_e + v_d}{2} \tag{10}$$

Substituindo v_e e v_d pelas equações 6 e 7, surge a equação:

$$V = \frac{\dot{\phi}_e . \frac{d_e}{2} + \dot{\phi}_d . \frac{d_d}{2}}{2} = \frac{\dot{\phi}_e . r_e + \dot{\phi}_d . r_d}{2} \tag{11}$$

$$\omega = \frac{\dot{\phi}_d . \frac{d_d}{2} + \dot{\phi}_e . \frac{d_e}{2}}{disteixos} = \frac{\dot{\phi}_d . r_d - \dot{\phi}_e . r_e}{disteixos} \tag{12}$$

em que V = $\dot{x}_R$ e ω = $\dot{\theta}_R$.

Juntando as expressões 11 e 12 num sistema de equações, é possível determinar a velocidade de rotação de cada uma das rodas:

$$\begin{cases} \dot{\phi}_e = \frac{\dot{x}_R}{r_e} - \frac{d * \dot{\theta}_R}{2 * r_e} \\ \dot{\phi}_d = \frac{\dot{x}_R}{r_d} + \frac{d * \dot{\theta}_R}{2 * r_d} \end{cases} \tag{13}$$

O modelo foi obtido, assumindo algumas simplificações:

- o veículo é considerado um corpo rígido;
- não existe escorregamento entre as rodas e o pavimento;
- o contacto entre as rodas e o pavimento é reduzido a um ponto;
- o plano das rodas, em qualquer instante, é sempre vertical ao plano de movimento.

4 Descrição das Plataformas Utilizadas

No decorrer deste trabalho utilizaram-se dois robôs da Lego®, o Mindstorms® NXT e o EV3, ambos com configuração diferencial, capazes de realizarem as tarefas e ensaios necessários para a validação dos programas a desenvolver. Neste capítulo são apresentados os sistemas experimentais utilizados, assim como as funcionalidades do *software* Matlab/Simulink®, utilizado para realizar a programação dos controladores.

4.1 Apresentação dos Sistemas Experimentais

Os veículos móveis utilizados neste trabalho foram os sistemas Lego® Mindstorms® NXT e EV3 ambos montados com configuração diferencial.

O kit NXT (Figura 19) vem equipado com um *smart brick* (controlador), com suporte para três saídas (atuadores) e quatro entradas, utilizadas para conectar os mais diversos sensores. O NXT (Figura 19) possui três motores com encoders óticos incorporados e com resolução de 1 grau, quatro sensores, sendo um de cor, um ultrassónico, um de contato e um de som, assim como peças para a montagem dos protótipos. Quanto ao *software*, vem acompanhado com o Mindstorms NXT-G, o qual é bastante intuitivo pois trata-se de um *software* de programação através de linguagem gráfica.

As especificações técnicas do controlador NXT são:

- Microcontrolador ARM7 de 32-bit;
- 256Kbytes FLASH, 512 Byte RAM;
- Microprocessador de 8-bit;
- 4Kbytes FLASH, 512 Byte RAM;
- Comunicação com computador através de USB 2.0 e *Bluetooth*.

Figura 19 – Lego Mindstorms NXT

O kit EV3 (Figura 20) vem equipado de um controlador com quatro entradas para atuadores e quatro saídas para sensores. Dois motores grandes e um médio, sensores de cor, contato, infravermelhos e peças para a construção dos modelos são enviados juntamente com o controlador.

As especificações técnicas do controlador EV3 são:

- Sistema operativo *Linux*;
- Controlador ARM9 de 300 MHz;
- Memória FLASH de 16 MB;
- Memória RAM de 64 MB;
- Resolução do ecrã de 178x128/Preto e Branco;
- Comunicação com computador via USB 2.0, USB 1.1, *Bluetooth* e *Wi-Fi*;
- Capacidade para cartão microSD, aumentando a memória até 32 GB.

Figura 20 – Lego Mindstorms EV3

Como seria de esperar, a evolução do NXT para o EV3 trouxe algumas melhorias.

O novo sistema apresenta maior capacidade de processamento e memória, permitindo-lhe executar e armazenar programas mais complexos.

A introdução da possibilidade de comunicação do robô com o computador através de *Wi-Fi* traduz-se numa vantagem pois a anterior comunicação via *Bluetooth* revelava-se bastante morosa, tornando o envio de programas e a obtenção de dados mais prático.

Quanto ao acabamento e qualidade de construção, o EV3 encontra-se num nível superior em relação ao NXT, sendo um veículo mais robusto e com menos folgas.

4.2 Linguagens de Programação para Sistemas Lego NXT/EV3

A implementação de um sistema computacional, como o necessário para a condução autónoma, necessita que sejam desenvolvidos algoritmos capazes de desempenhar as tarefas pretendidas do robô. A codificação na qual é feita a descrição da estrutura, métodos e objetivos, é realizada com recurso a uma linguagem de programação.

Atualmente, existem diversas linguagens de programação dos controladores. A escolha da linguagem a ser utilizada num determinado programa depende de diversos fatores, como a complexidade do algoritmo a ser implementado, as interfaces e as bibliotecas necessárias para o programa, ou a velocidade de execução pretendida.

A implementação de algoritmos é possível através de diversas linguagens:

- **Linguagens de Baixo Nível**

  As linguagens de baixo nível são aquelas que mais se aproximam do processador da máquina, ou seja, que compreendem as caraterísticas da arquitetura do computador, utilizando apenas instruções do processador. São consideradas linguagens simples, mas difíceis de utilizar, devido ao elevado número de detalhes técnicos que o programador deve conhecer.

- **Linguagens de Alto Nível**

  São consideradas linguagens de alto nível todas aquelas que se distanciam do código nativo da máquina e se aproximam à linguagem humana, tornando a programação mais simples de ler e escrever.
  Desta forma, o programador não necessita de conhecer caraterísticas do hardware em questão, como instruções e registos para desenvolver uma determinada aplicação.
  Os programas escritos nestas linguagens são convertidos para a linguagem da máquina através de um compilador ou interpretador. Linguagens como C, C++, Java, Python ou Matlab, são exemplos de linguagens de alto nível.

No caso mais concreto da programação dos controladores da Lego®, existem alguns programas tipicamente utilizados, apresentados na Tabela 1.

Tabela 1 – *Softwares* de programação para Lego® Mindstorms® NXT/EV3

Nome	Equipamento		Tipo de linguagem
	NXT	EV3	
NXT-G	Sim	Não	Gráfica
EV3-G	Sim	Sim	Gráfica
LabView	Sim	Sim	Gráfica
RobotC	Sim	Sim	C
leJOS	Sim	Sim	Java
Matlab/Simulink	Sim	Sim	Gráfica/C/Matlab Code

4.3 Matlab/Simulink

O Matlab® é um *software* de alta performance desenvolvido pela Mathworks® que integra análise numérica, cálculo com matrizes, implementação de algoritmos, processamento de sinais e construção de gráficos num ambiente interativo, entre outros.

O *software* Simulink® é uma extensão do Matlab, no qual se constroem programas em linguagem gráfica para simulação e análise de sistemas dinâmicos. Existe uma biblioteca de blocos com inúmeras funções, sejam eles para realizar operações de cálculo, operações lógicas, visualização de resultados, etc. A versão do *software* utilizada no decorrer deste trabalho foi a R2014b.

Para controlar os sistemas da Lego, é necessário instalar dois "*support package*", um para o NXT e outro para o EV3, nos quais se encontram os blocos destinados ao controlo dos robôs, conforme é possível observar na Figura 21.

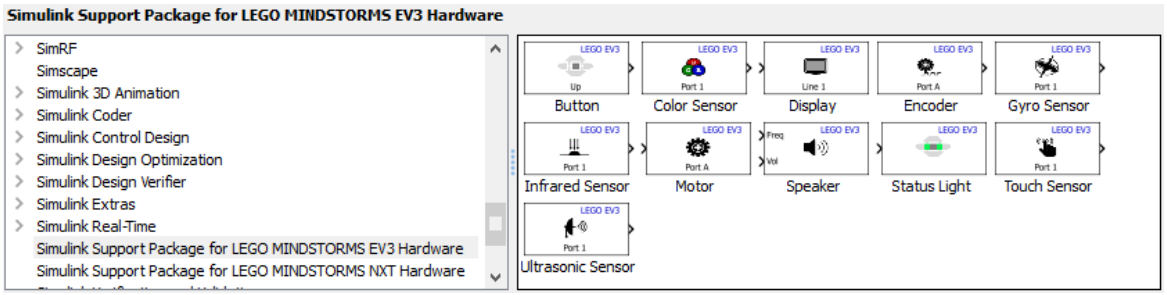

Figura 21 – *Support package* para sistemas Lego

As capacidades do ambiente do Simulink podem ser ampliadas através do uso de *s-functions*. As *s-functions* são blocos do Simulink que, seguindo algumas regras, permitem implementar algoritmos de controlo escritos em código Matlab, C, C++ ou Fortran, interagindo com outros blocos no ambiente do programa.

Como um dos objetivos é a utilização da linguagem C, utilizaram-se as C *MEX S-functions*. O desenvolvimento destes blocos é possível de várias formas:

- *Handwritten C MEX S-functions*, permitem uma maior flexibilidade de programação, mas exigem o conhecimento da API (interface de programação de aplicações) das *s-functions* e do TLC (*Target Language Compiler*).
- *S-function builder,* é uma interface gráfica destinada a facilitar o desenvolvimento de *s-functions.* Desta forma, o utilizador não necessita de ter conhecimento da API destas funções. Permite gerar ficheiros TLC automaticamente.
- *The Legacy Code Tool,* é um conjunto de comandos Matlab que ajudam na criação de *s-functions* para incorporar código C ou C++. Tal como o *S-function builder,* também podem ser gerados ficheiros TLC, mas limita ainda mais o acesso aos métodos da API das *s-functions*.

S-function builder

Selecionado o bloco *S-function builder*, é necessário inserir, além do código, alguma informação essencial para que a *s-function* funcione da forma desejada. Fazendo um duplo clique no bloco (Figura 22), abre-se uma janela onde se inserem os dados para criar a *s-function* (Figura 23).

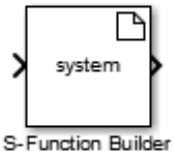

Figura 22 – Bloco de uma *s-function*

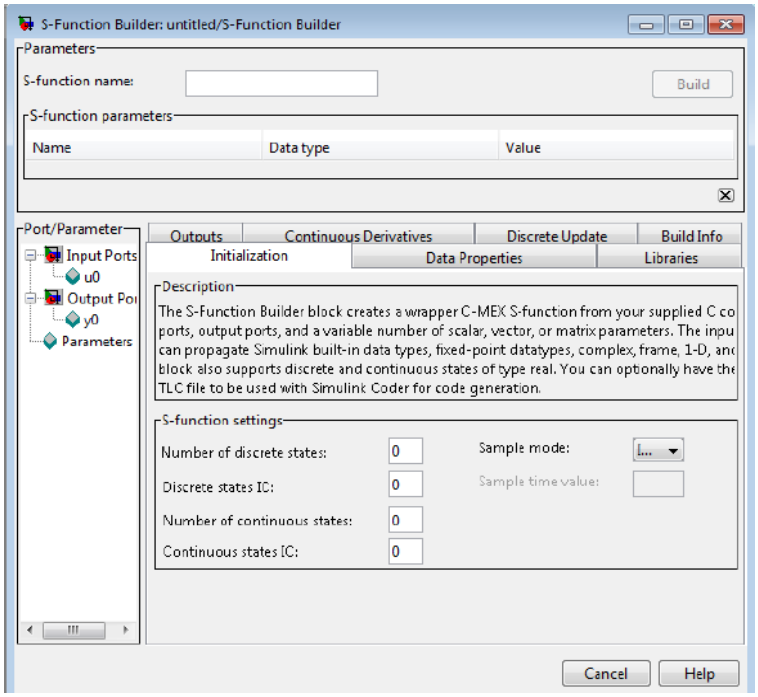

Figura 23 – Janela de uma *s-function builder*

Initialization

No separador "*Initialization*", é escolhido o modo como é feita a amostragem (*Inherited*, contínua ou discreta). Escolhendo o modo discreto, a *s-function* atualiza as suas saídas com um tempo de amostragem especificado pelo utilizador.

Data Properties

Separador que permite adicionar portas e parâmetros à *s-function*, assim como, personalizar o tipo de dados (números inteiros, *double*, *float*, etc.), que cada porta recebe e envia.

Libraries

A biblioteca permite especificar a localização de arquivos de código externo que sejam necessários, no caso de serem utilizados no código do programa, escrito noutros separadores (*outputs*, *continuous derivatives* ou *discrete update*).

No caso de serem necessárias variáveis globais (variáveis que veem o seu valor atualizado em cada iteração), devem ser declaradas neste separador, na janela "*Includes*:" (Figura 24).

Outputs

O código C do programa a realizar é inserido neste separador.

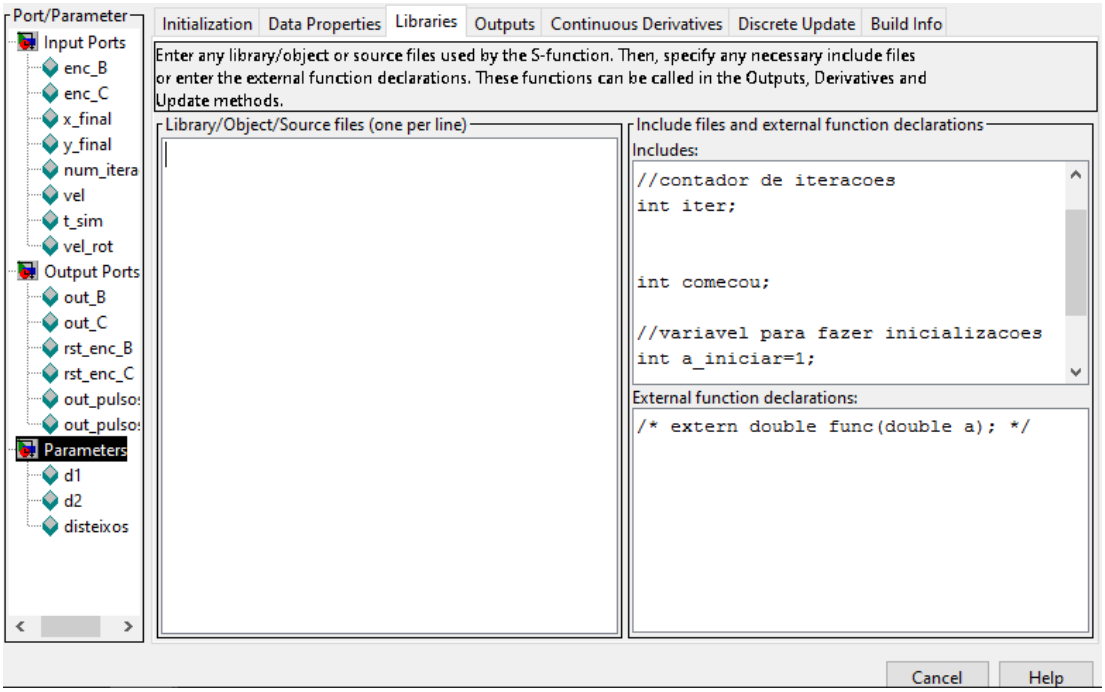

Figura 24 – Separador "Libraries"

4.4 Comunicação Lego-Simulink

O Simulink permite comunicação entre o *software* e o *hardware* em tempo real, permitindo desta forma, a aquisição de dados enquanto o programa corre.

Para poder utilizar os modelos em tempo real, a ligação do controlador ao computador não pode ser realizada por cabo USB, pois desta forma apenas é possível enviar o programa e executá-lo diretamente no controlador.

No caso do Lego NXT, a comunicação foi feita por *Bluetooth*, através de um *Bluetooth dongle* suportado pelo NXT. Concluído o emparelhamento entre os dois dispositivos, é possível executar os programas diretamente a partir do ambiente do Simulink.

Em relação ao Lego EV3, a comunicação foi realizada por *WI-FI*, sendo necessário um *wireless dongle* compatível e um router, de modo a estabelecer uma ligação.

O modo externo deverá ser selecionado no ambiente do *Simulink*, especificando qual o tempo total de simulação, se limitado ou infinito, e o modo como é executada a simulação, se é discreta ou não, e qual o tempo de amostragem.

5 Desenvolvimento, Testes e Resultados

Decidir qual o melhor caminho a seguir com o objetivo de realizar uma determinada tarefa, baseando-se num critério de otimização como por exemplo, qual o caminho mais curto, mais rápido ou o mais eficiente em termos de energia, pertence a um problema da robótica chamado planeamento de caminhos (*path planning).*

Uma das capacidades que um robô móvel deve ter é a de se movimentar num espaço de trabalho específico, evitando obstáculos. Para que tal aconteça, deverá ser capaz de calcular diferentes trajetórias com base na sua configuração inicial e final assim como no seu modelo cinemático, e escolher qual será a mais indicada.

Com o intuito de implementar diferentes trajetórias, foram adotadas duas estratégias com base em odometria. A primeira abordagem executa dois movimentos distintos, um de rotação da plataforma, orientando-a para o ponto de destino e outro de translação, movendo-a até esse ponto. O controlo de ambos os movimentos tem por base o cálculo dos valores dos ângulos de rotação que cada motor deve rodar, ou seja, é feito um controlo em posição.

Como a primeira estratégia apresentava tempos de execução elevados e descontinuidade no movimento, foi pensada uma segunda abordagem que eliminasse esses inconvenientes. O controlo do movimento passa a ser em velocidade, fornecendo os valores de velocidade indicados para cumprir uma determinada trajetória, com base na sua posição. Desta forma, o robô passa a executar um movimento contínuo até atingir o destino.

Para cada estratégia, foram implementadas trajetórias em linha reta, circular e em parábola. Na estratégia com controlo em velocidade, foram ainda implementados dois perfis de velocidade, sinusoidal e trapezoidal, com o objetivo de verificar qual a solução mais eficaz.

5.1 Implementação de uma Trajetória com Controlo em Posição

A primeira abordagem feita no que diz respeito ao planeamento de trajetórias utiliza dois movimentos distintos para chegar ao ponto de destino. A rotação e a translação da plataforma dão-se em diferentes momentos, sendo que, o primeiro movimento (rotação) tem como objetivo direcionar o robô para o ponto seguinte. A translação dá-se imediatamente no instante a seguir, impulsionando o robô até ao ponto de destino.

Como o Simulink não permite selecionar o deslocamento angular de cada motor nem a velocidade a aplicar a cada uma das rodas, são enviados para o controlador do robô os valores da potência (*power*) inseridos nos *inputs* dos blocos dos motores, mas com a rotação limitada, especificando-se através do código desenvolvido, qual o deslocamento angular que cada motor deve realizar nas diferentes iterações. No caso da rotação da plataforma em torno do seu eixo, as potências das duas rodas têm valores opostos. No que diz respeito à translação, as potências têm o mesmo valor, originando um movimento retilíneo.

Após efetuados alguns testes do movimento do robô, tais como deslocamentos controlados em linha reta, circunferência e rotação da plataforma, implementou-se um deslocamento em parábola no qual o movimento é dividido em vários pontos equidistantes segundo o eixo **x** (xd) ao longo da trajetória (Figura 25).

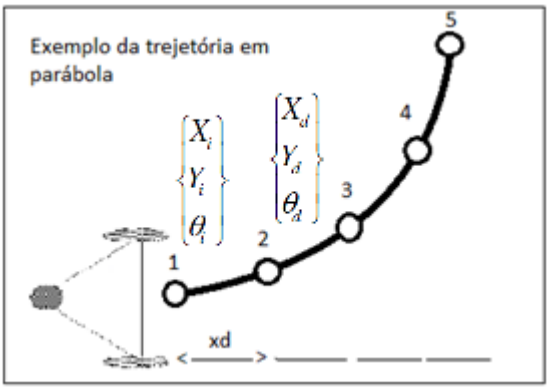

Figura 25 – Exemplo da trajetória percorrida (controlo em posição)

A implementação do programa segue o fluxograma da Figura 26. São necessários alguns dados iniciais para o cálculo da trajetória, tais como:

- Diâmetro das rodas (d_1, d_2);
- Distância entre eixos (disteixos);
- Número de pontos nos quais o movimento se divide (n);
- Configuração inicial do robô (x_i, y_i, θ_i);
- Configuração final do robô (x_f, y_f);
- Velocidade de rotação (Vel_r);
- Velocidade de translação (Vel_l).

O primeiro parâmetro calculado pelo programa com base na configuração inicial e final do veículo é o parâmetro *k* da parábola a ser realizada.

O cálculo para o valor, em graus, que cada roda deve rodar segue o modelo cinemático de um veículo diferencial, descrito no capítulo 3.

A posição inicial vai sendo atualizada conforme a iteração que está a ser executada. Na primeira iteração a posição inicial corresponde ao ponto de origem do robô (*xi*, *yi*, *θi*) e é calculado o primeiro ponto para o qual o veículo se deve deslocar com a configuração (*xd*, *yd*, *θd*). Na iteração seguinte, o ponto de destino da iteração anterior passa a ser a nova posição inicial, repetindo-se este processo até ser atingida a posição final desejada.

A atribuição da potência apresenta três fases distintas. Isso deve-se ao facto de que se forem atribuídas potências muito elevadas, o controlador do robô não consegue efetuar a ordem de paragem dos motores a tempo de cumprir o limite de graus que estes devem rodar, originando trajetórias completamente distintas das desejadas. Após algumas experiências foi determinado o valor da percentagem máxima de potência a atribuir aquando da rotação da plataforma, a qual deve rondar os 30%. Quanto à translação, poderá ser inserida a percentagem de potência desejada pois o programa está feito de forma a limitar a potência para 50%, quando for atingido 80% do deslocamento em cada iteração.

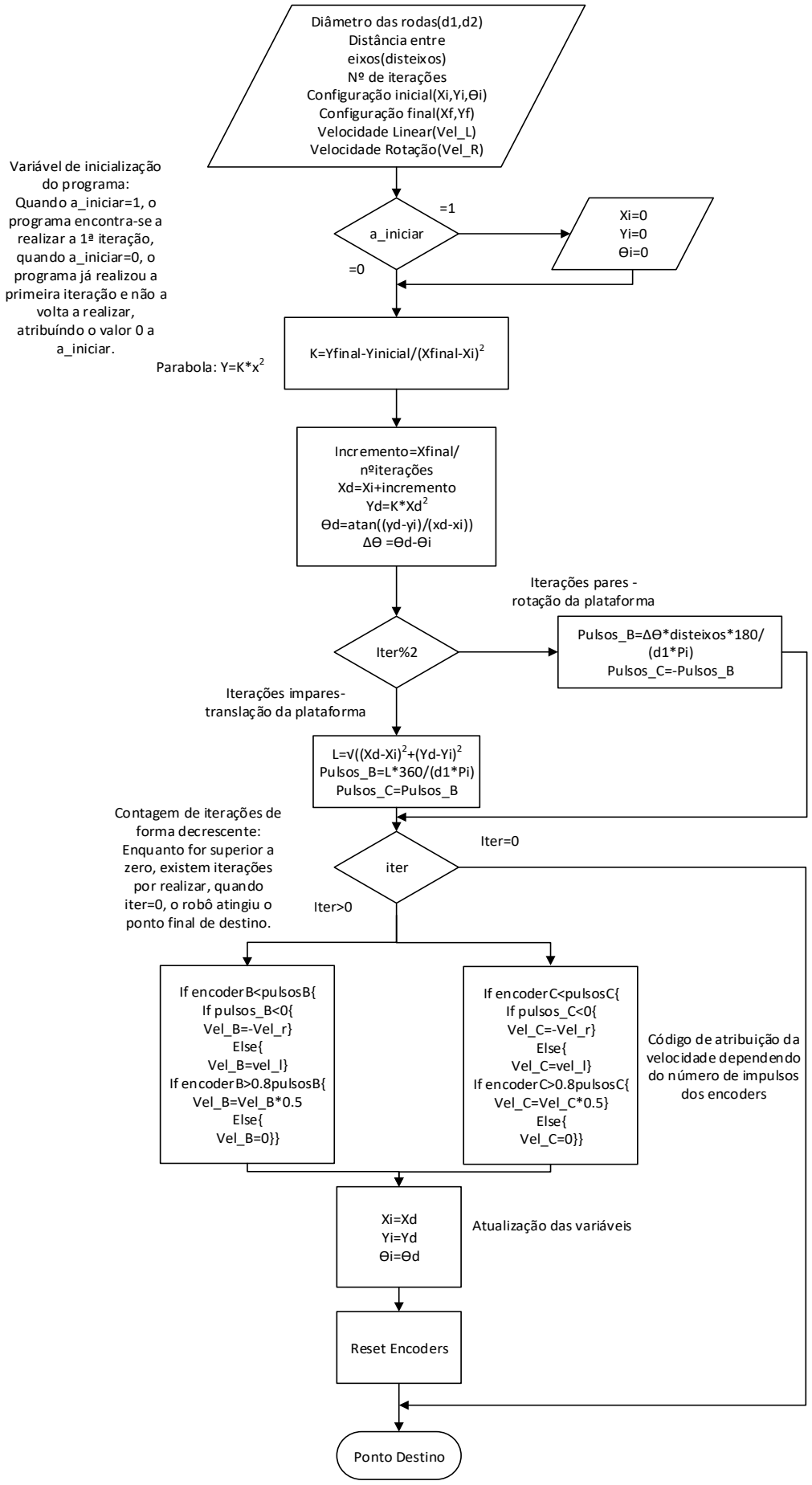

Figura 26 – Fluxograma do movimento do veículo segundo uma parábola com controlo em posição

Analisando o fluxograma da Figura 26, após serem inseridos os dados iniciais, a primeira ação do programa é verificar se está a realizar a primeira iteração (variável "a_iniciar"). No caso de ser verdadeiro, são atribuídas às variáveis (xi, yi, θi) valor nulo (configuração inicial). Após essa ordem, a variável "a_iniciar" passa a zero e os valores de (xi, yi, θi) são atualizados no final de cada iteração.

Outra variável de decisão implementa foi a "iter%2". Neste caso, se a iteração que está a ser realizada for par, é realizada a rotação da plataforma, caso contrário dá-se a translação da mesma. Nos dados iniciais é inserido o número de iterações a realizar (número de pontos em que a trajetória se divide). O programa está estruturado de forma a dividir cada iteração em duas, uma de rotação e outra de traslação, ou seja, se existirem cinco iterações ao longo da trajetória, no programa existem dez, cinco de rotação (pares) e cinco de translação (impares).

Após essa decisão, são atribuídos os valores de potência dos motores. Enquanto o número de graus contado pelos encoders não atingir o valor calculado de "pulsosB" e "pulsosC", são enviados valores de potência. Se qualquer umas destas variáveis for negativa, o que acontece quando se dá a rotação da plataforma, é atribuído um valor negativo (vel_r), ou seja, a roda vai rodar num sentido inverso, caso contrário, é atribuído o mesmo valor a cada uma das rodas (vel_l).

A implementação desta estratégia revelou-se pouco eficiente, devido ao tempo que o veículo demorava a percorrer a trajetória. A trajetória aproximada e a pretendida podem, no entanto, ser aproximadas se o número de iterações for suficientemente elevado, levando a que os incrementos sejam mais reduzidos.

5.2 Implementação de uma Trajetória com Controlo em Velocidade

Com o objetivo de melhorar a estratégia anterior, foi implementada uma nova estratégia com comando de velocidade para as rodas. Desta forma, podem ser eliminadas as descontinuidades presentes no controlo em posição, assim como, minimizar o tempo de execução da trajetória.

O movimento deixa de ser limitado pelo sinal dos encoders e passa a ser controlado por um perfil de velocidade que calcula qual a velocidade a atribuir a cada uma das rodas em cada incremento de tempo, dependendo do tempo total de execução (T) e do tempo de incremento [t(i)], o qual é limitado pela capacidade do processador. Na Figura 27, estão representados o ponto inicial e final, mas na verdade, existem vários pontos correspondentes ao número de incrementos realizados ao longo da trajetória.

Foram realizados vários testes no sentido de determinar qual o menor tempo de incremento [t(i)] que era possível utilizar. A *toolbox* permite utilizar uma funcionalidade chamada "*overrun detection"* que avisa quando os controladores entram em sobrecarga, podendo originar um programa mal-executado e consequentemente, uma trajetória indesejada. Após alguns testes com vários tempos de ciclo, chegou-se à conclusão que tempos inferiores a 0.1s não deverão ser utilizados no sentido de não sobrecarregar o controlador.

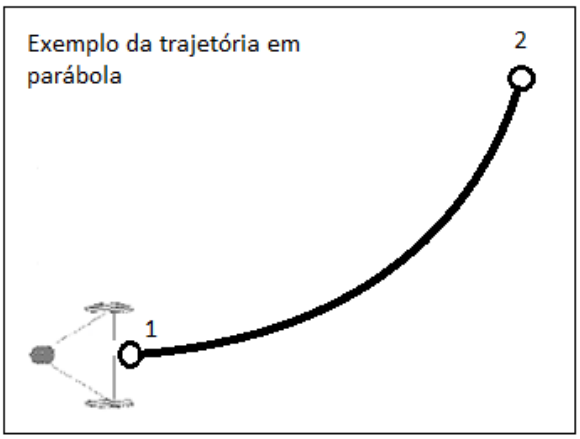

Figura 27 – Exemplo da trajetória percorrida (controlo em velocidade)

Além de ter sido implementado o mesmo tipo de trajetórias da estratégia anterior, foram implementados dois perfis de velocidade, sinusoidal (Figura 28) e trapezoidal (Figura 29).

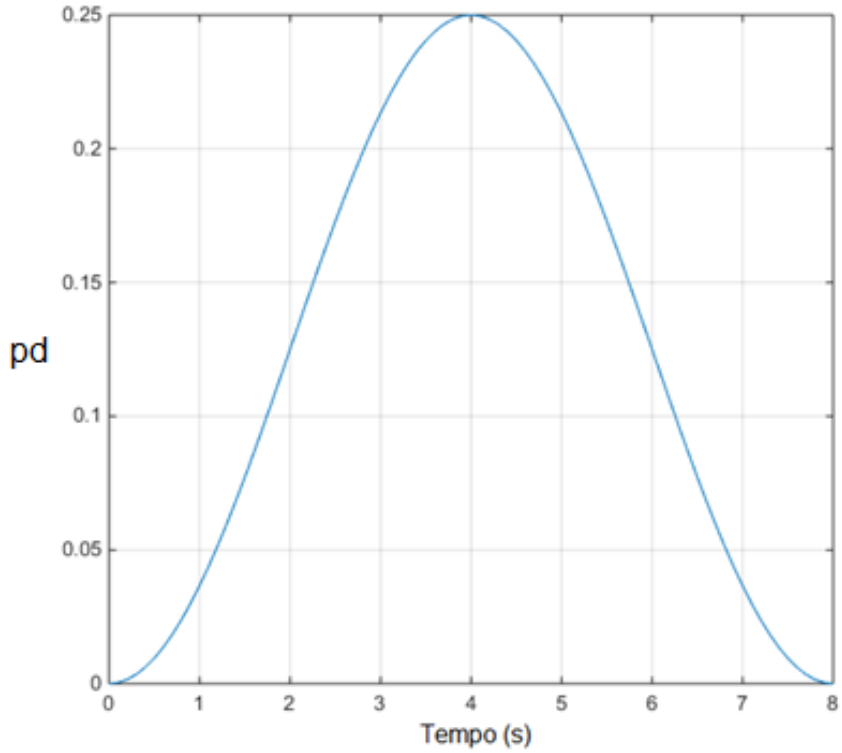

Figura 28 – Perfil de velocidade sinusoidal

O perfil de velocidade trapezoidal rege-se pela seguinte equação:

$$\begin{cases} \dfrac{k_1 \,.\, t}{T.\ c_1}, & t \le T\ c_1 \\ k_1, & T\ c_1 < t \le T - T\ c_2 \\ \dfrac{k_1 . (T - t)}{T . c_2}, & t > T\ c_2 \end{cases} \tag{14}$$

na qual, são usados os seguintes parâmetros:

$$\begin{cases} c_1 = 1/3 \\ c_2 = 1/3 \\ k_1 = \dfrac{2}{T\,(2 - c_1 - c_2)} \end{cases} \tag{15}$$

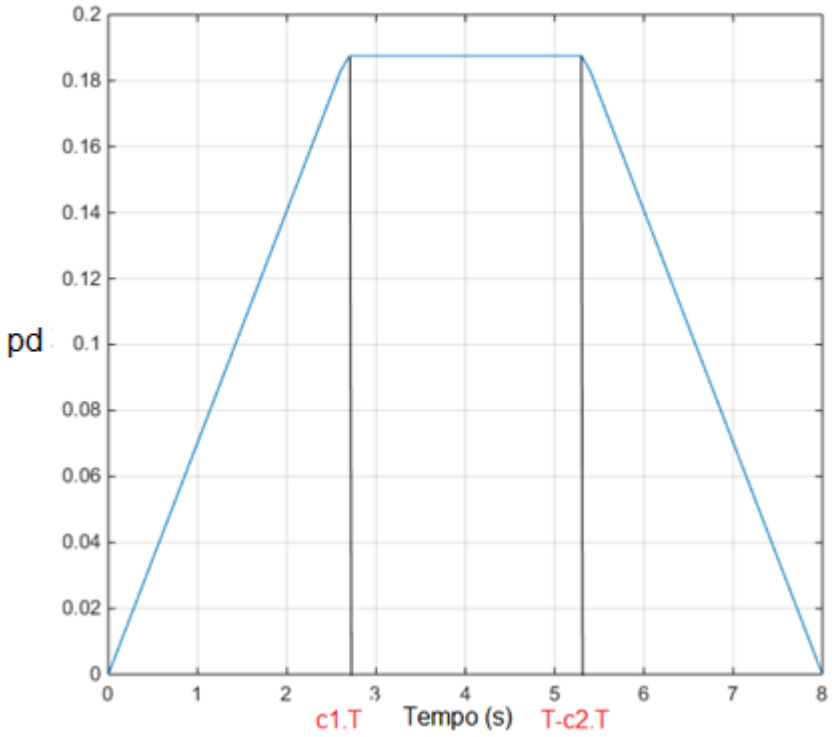

Figura 29 – Perfil de velocidade trapezoidal

Ambos os perfis foram implementados com o objetivo de diminuir a possibilidade de escorregamento entre as rodas e o pavimento.

O fluxograma da Figura 30 descreve o algoritmo implementado para controlar o robô ao longo da parábola. Os dados iniciais necessários para o programa executar o cálculo são:

- Diâmetro das rodas (d1, d2);
- Distância entre eixos (disteixos);
- Configuração inicial do robô (x_i, y_i, θ_i);
- Configuração final do robô (x_f, y_f);
- Tempo do movimento (T);
- Incremento de ciclo [t(i)].

Após o cálculo dos perfis de posição e velocidade, o programa calcula a trajetória a realizar. Em cada incremento de tempo são calculados os valores de x_i, y_i *e* θ_i. Derivando as expressões em ordem ao tempo, é obtida a velocidade no referencial global. Utilizando a equação 1 (matriz de rotação ortogonal), são obtidas as velocidades no referencial local, para posteriormente realizar o cálculo segundo o modelo cinemático do robô e obter as velocidades angulares de cada roda.

Como já foi referido na secção 5.1, não é possível enviar valores de velocidade diretamente para o controlador do robô, sendo necessário converter esses valores para potência (descrito na secção 5.5).

Nas figuras 30 e 31 são apresentados os fluxogramas do movimento do veículo NXT segundo uma parábola, com perfis sinusoidal e trapezoidal respetivamente. Para o caso do EV3, os fluxogramas são iguais, alterando apenas o bloco onde estão definidas as potências fornecidas a cada roda.

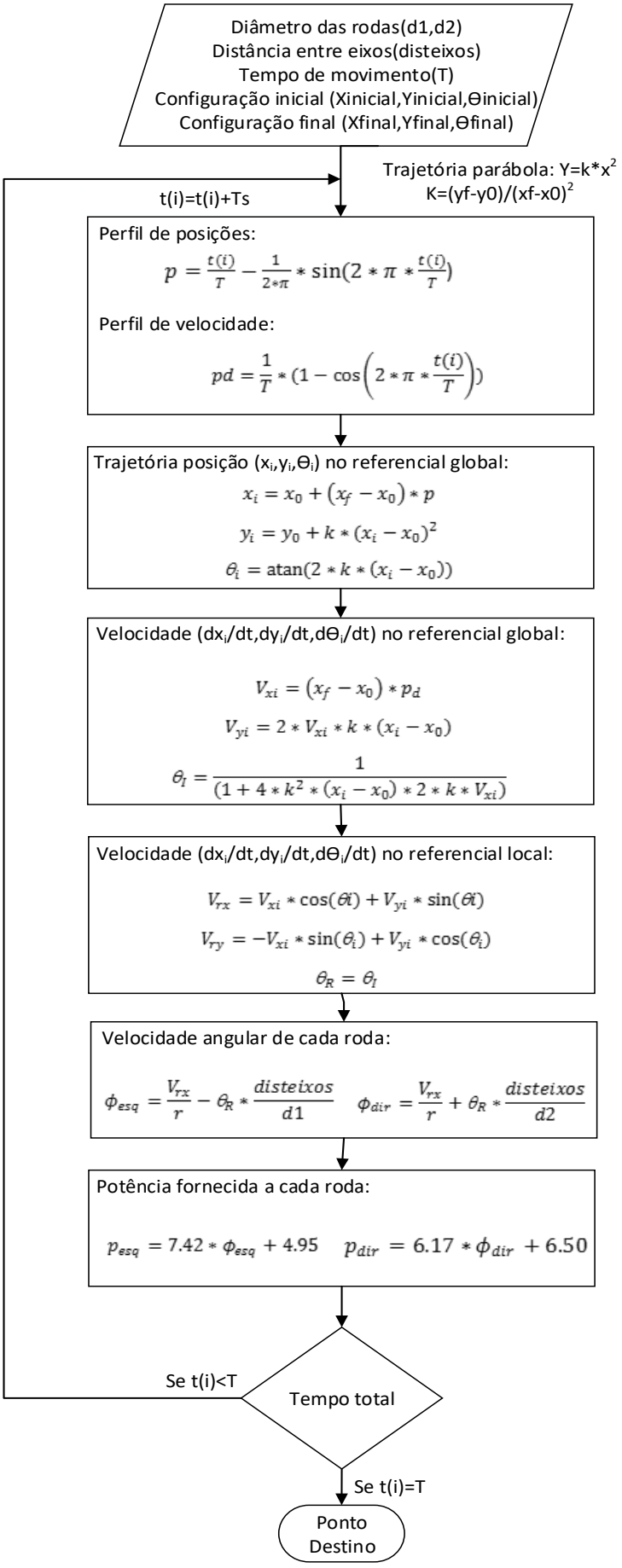

Figura 30 - Fluxograma do movimento do veículo (NXT), segundo uma parábola com controlo em velocidade (perfil sinusoidal)

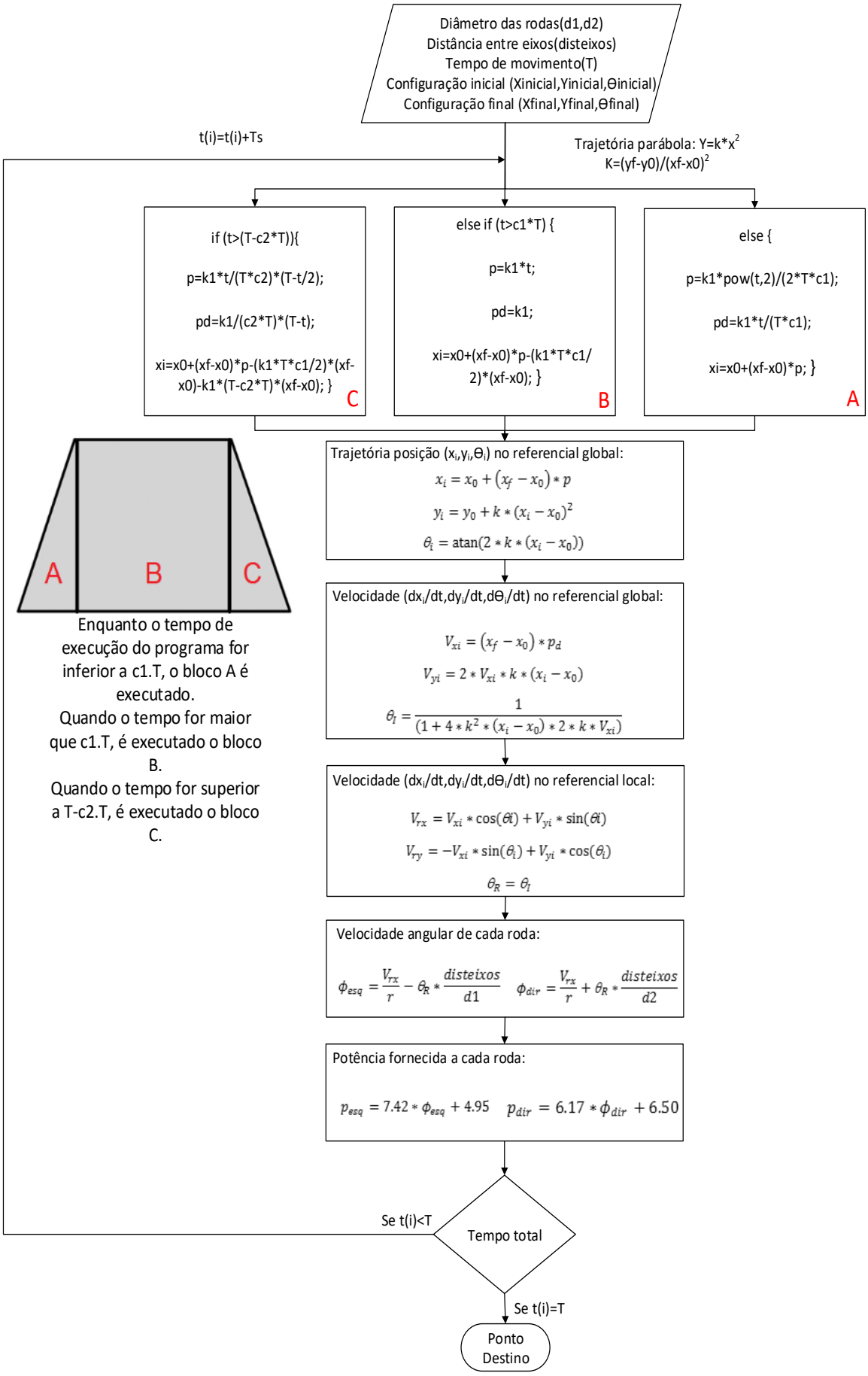

Figura 31 – Fluxograma do movimento do veículo (NXT), segundo uma parábola com controlo em velocidade (perfil trapezoidal)

5.3 Localização do Veículo

Após serem implementadas as estratégias descritas previamente, foi necessário utilizar um modelo que estimasse a posição do robô por odometria.

Como referido no capítulo 2.4, existem inúmeros erros que podem contribuir para o desvio do robô da trajetória pretendida, assim como soluções que minimizam esses erros. No entanto, este trabalho focou-se na estimativa da localização com base nos encoders presentes nos motores dos robôs.

O modelo utilizado para realizar essa estimativa é descrito nas equações seguintes [1]:

$$\Delta x = \Delta s . \cos\left(\theta + \frac{\Delta\theta}{2}\right) \tag{16}$$

$$\Delta y = \Delta s . \sin\left(\theta + \frac{\Delta\theta}{2}\right) \tag{17}$$

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{18}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{disteixos} \tag{19}$$

onde:

- (Δx; Δy; $\Delta\theta$) – variação da posição em relação ao intervalo de tempo anterior;
- Δs_r; Δs_l – distâncias percorridas pela roda direita e esquerda respetivamente;
- disteixos – distância entre eixos de um veículo diferencial.

O cálculo da posição atualizada (**p**') é feito recorrendo à seguinte equação, em que **p** representa a posição anterior:

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \mathbf{p} + \begin{bmatrix} \Delta s . \cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \Delta s . \sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s . \cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \Delta s . \sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \Delta\theta \end{bmatrix} \tag{20}$$

5.4 Programas Implementados

A implementação das estratégias previamente descritas teve por base uma validação em Matlab, na qual foram simuladas e testadas, e posteriormente foram implementadas em *Simulink* no controlador de ambos os robôs.

No diagrama da Figura 32, podem ser observadas as estratégias utilizadas. Os testes realizados em cada uma delas estão presentes na secção 5.5.

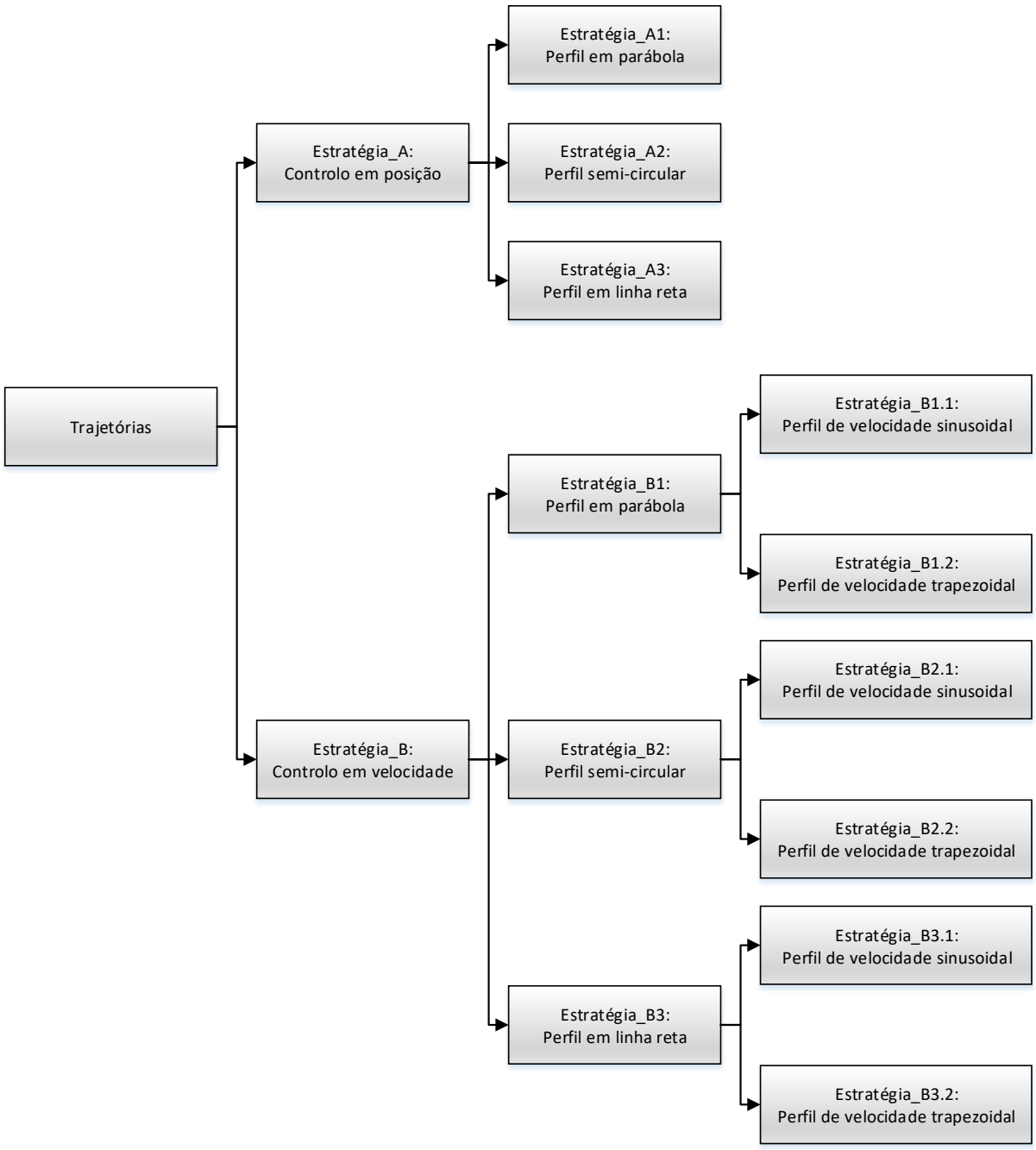

Figura 32 - Estratégias implementadas

Na Figura 33 é apresentado o diagrama do programa em simulink que executa a parábola com controlo em posição.

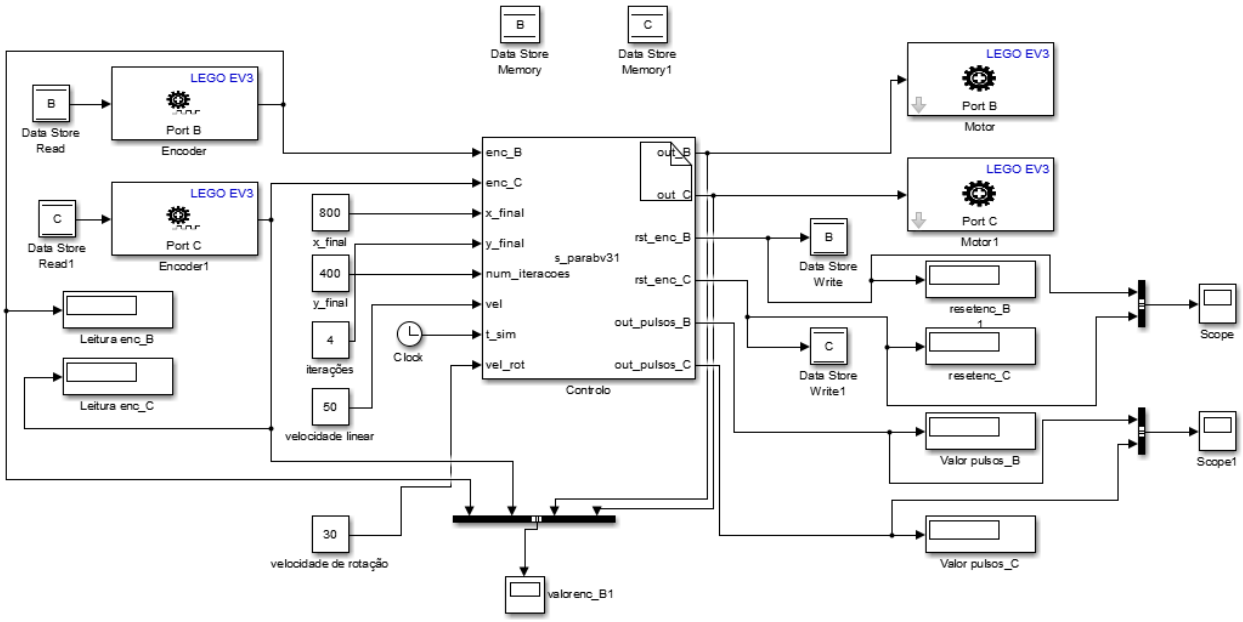

Figura 33 - Programa realizado para a parábola com controlo em posição

5.5 Experiências e Testes Realizados

A necessidade de serem utilizados valores de potência (*power*) na estratégia com controlo em velocidade, implicou a realização de vários testes aos motores, de modo a traçar as várias caraterísticas (potência vs. velocidade) em diferentes situações.

A obtenção das curvas foi feita através do valor do deslocamento angular dos motores ao longo de 10s. No final desse período, para cada valor de potência, é obtido um número de graus/s realizados por cada motor. Convertendo esse número para rad/s, é possível traçar as várias caraterísticas apresentadas.

Todos os testes foram realizados com as baterias carregadas (sensivelmente 7.9 Volt para o NXT e 10V para o EV3) e na mesma superfície onde foram realizadas as trajetórias, por forma a minimizar ao máximo fatores externos que pudessem influenciar os resultados.

Foram aplicadas gamas de potência dos 10% aos 100%. Abaixo dos 10%, nem sempre é certo que os motores consigam mover a plataforma.

O motor B corresponde ao motor direito e o motor C ao motor esquerdo.

O primeiro teste foi realizado com um motor imobilizado e outro com potência atribuída, fazendo com que o robô se deslocasse em torno da roda imobilizada (figura 34 a 37).

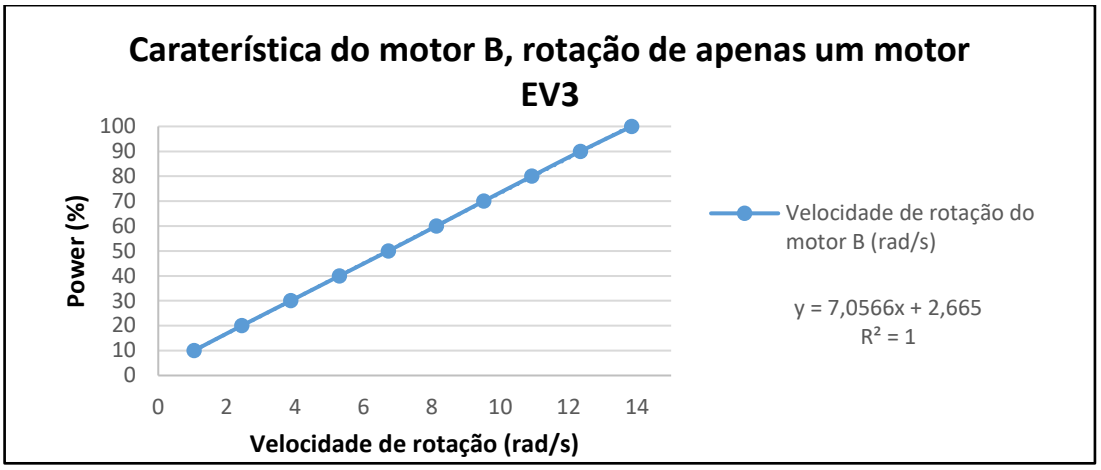

Figura 34 – Caraterística do motor B, rotação de apenas um motor (EV3)

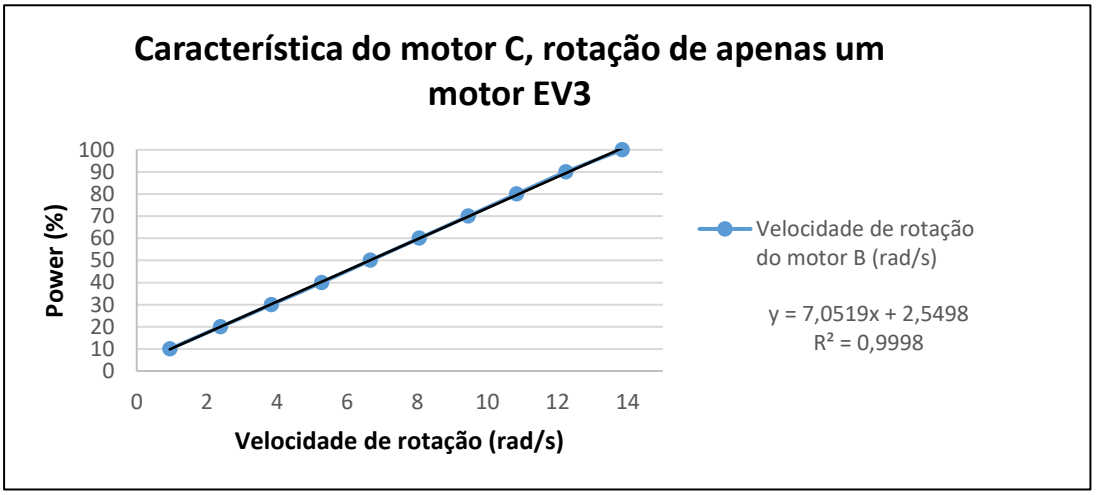

Figura 355 - Caraterística do motor B, rotação apenas de um motor (EV3)

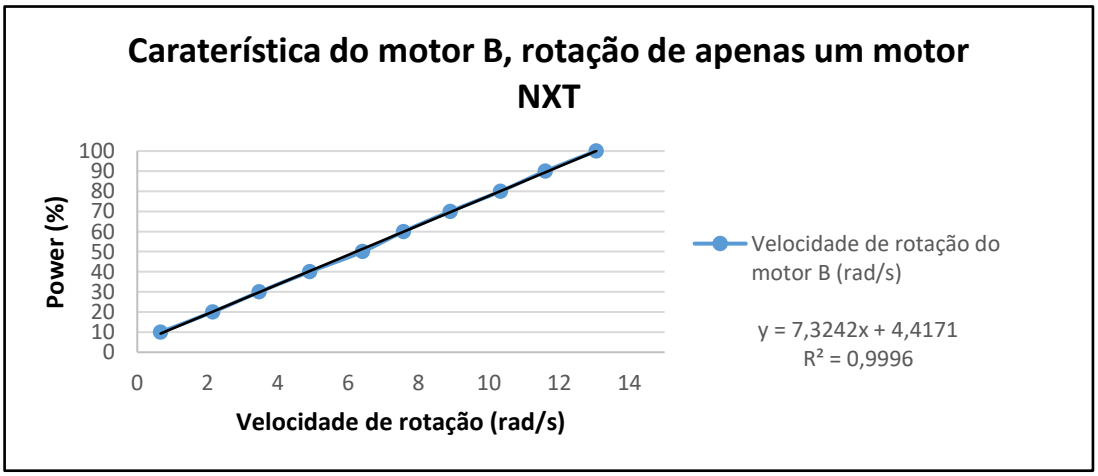

Figura 36 – Caraterística do motor B, rotação apenas de um motor (NXT)

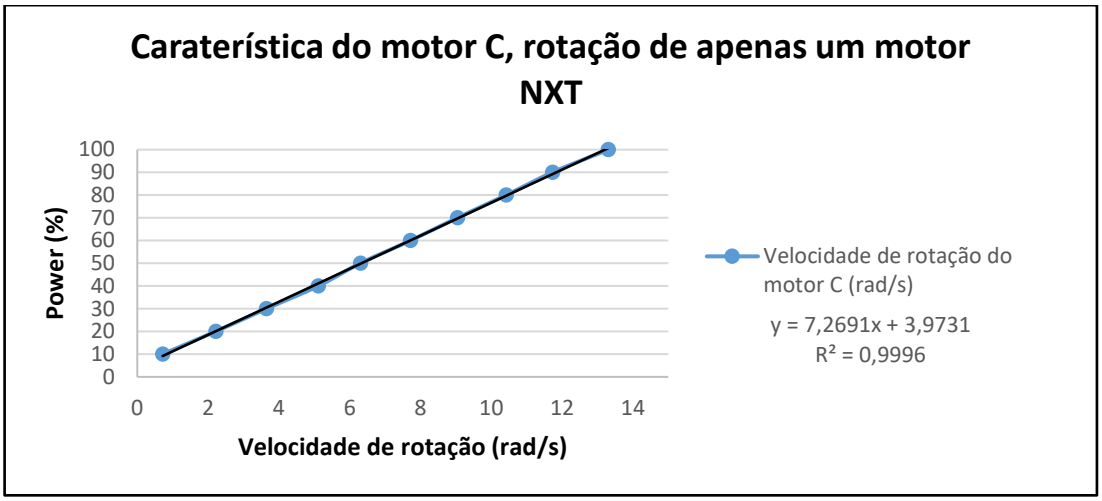

Figura 36 – Caraterística do motor C, rotação de apenas um motor (NXT)

No segundo teste realizado foram aplicados valores de potência iguais para ambos os motores (figura 38 a 41).

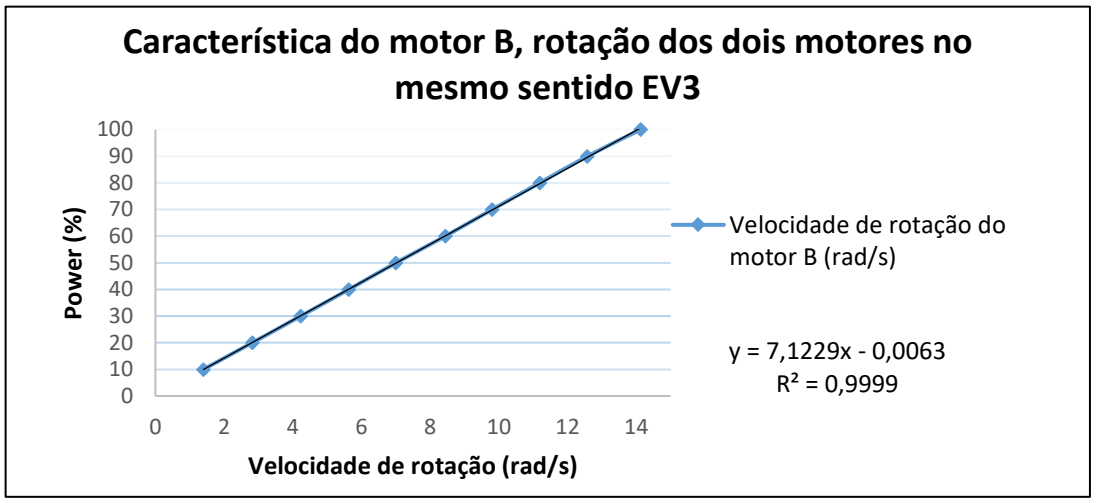

Figura 37 – Característica do motor B, rotação de ambos os motores no mesmo sentido (EV3)

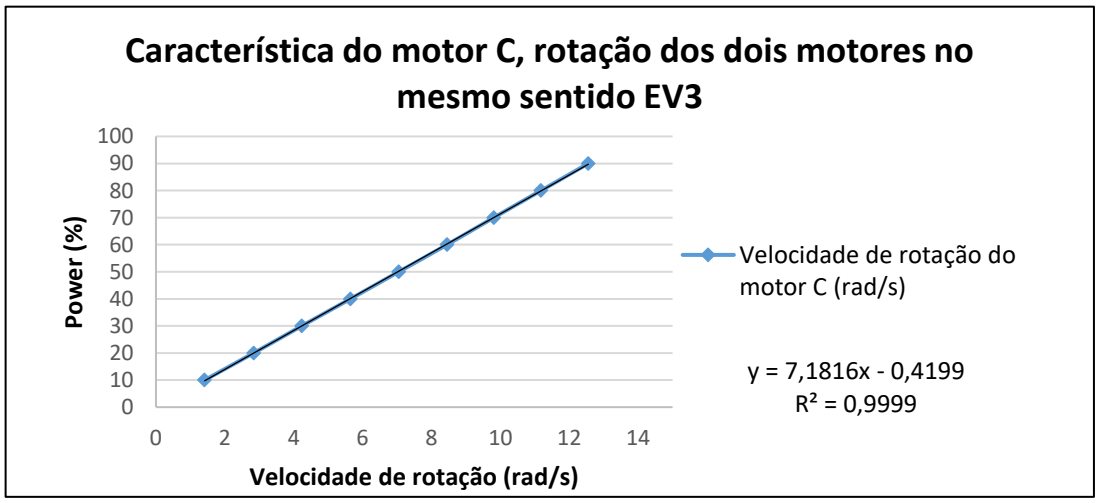

Figura 38 – Característica do motor C, rotação de ambos os motores no mesmo sentido (EV3)

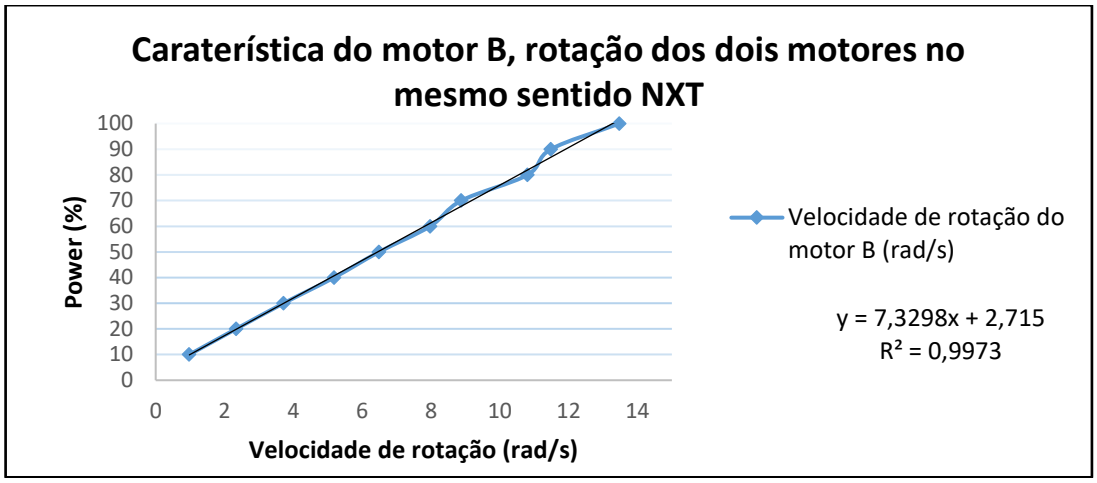

Figura 39 – Caraterística do motor B, rotação de ambos os motores no mesmo sentido (NXT)

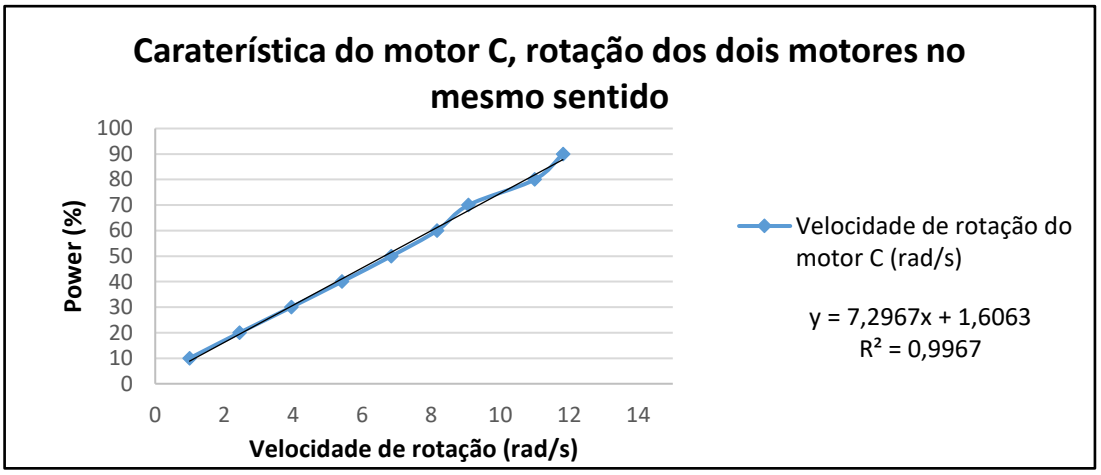

Figura 40 – Caraterística do motor B, rotação de ambos os motores no mesmo sentido (NXT)

No último teste realizado foram aplicadas potências iguais em ambas as rodas, mas sentidos opostos (figura 42 a 45).

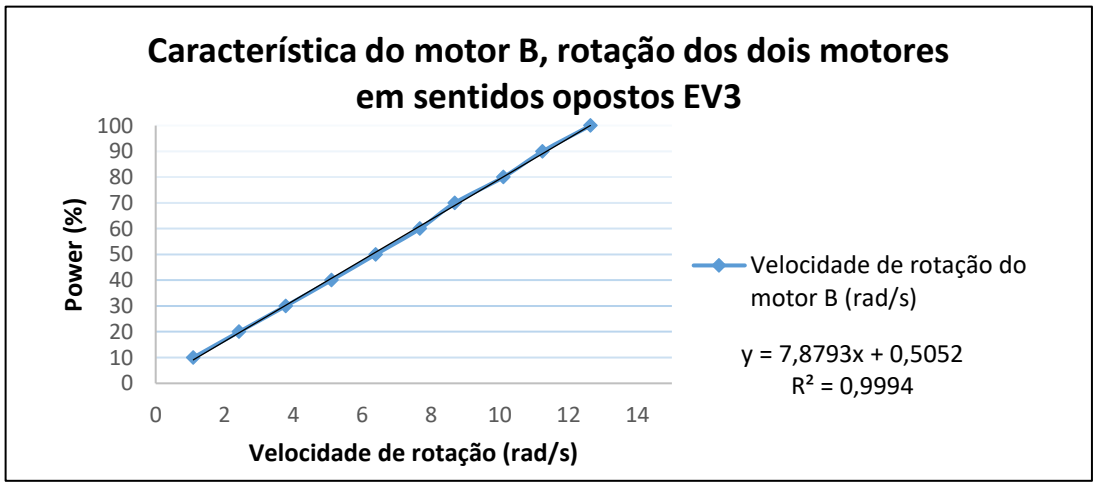

Figura 41 – Característica do motor B, rotação de ambos os motores em sentidos opostos (EV3)

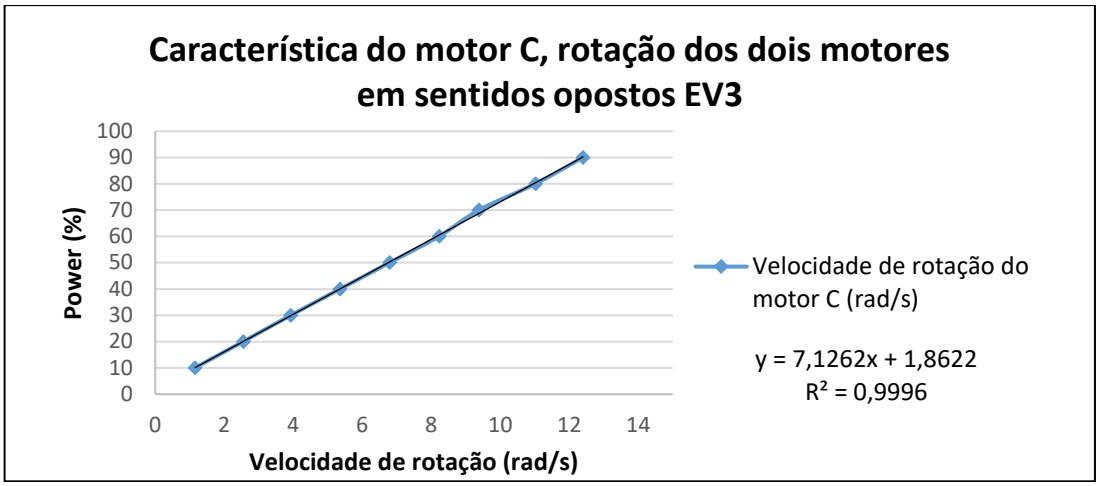

Figura 42 – Característica do motor C, rotação de ambos os motores em sentidos opostos

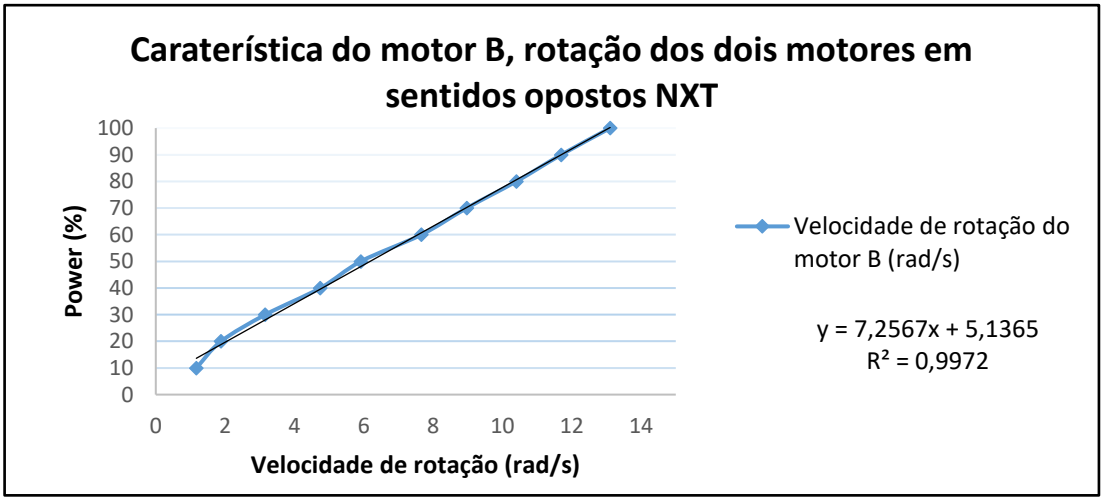

Figura 43 – Característica do motor B, rotação de ambos os motores em sentidos opostos (NXT)

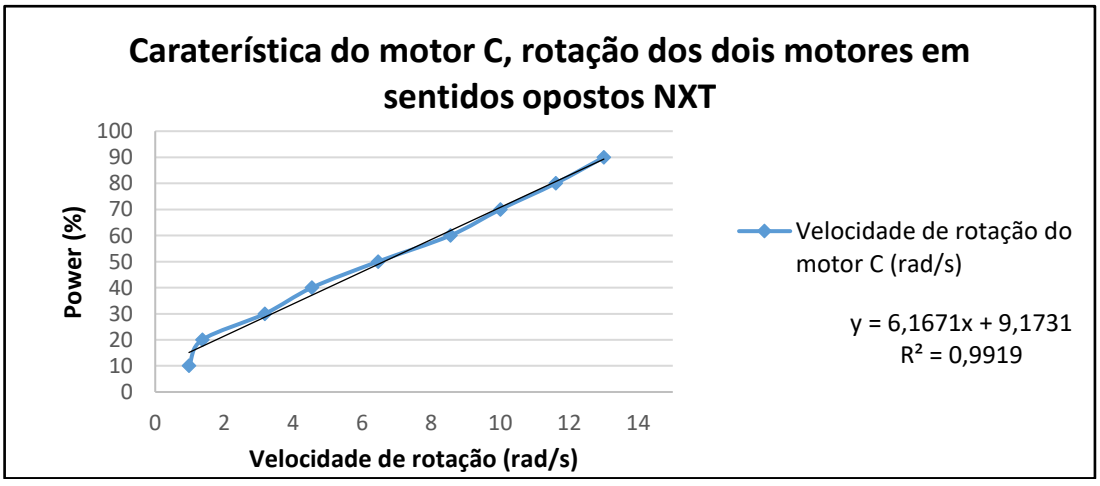

Figura 44 – Característica do motor C, rotação de ambos os motores em sentidos opostos (NXT)

Após a obtenção das caraterísticas para diferentes cenários, foram utilizadas as curvas de potência vs. velocidade para converter o valor da velocidade em potência. A situação que apresenta maior esforço para o robô acontece quando existe apenas rotação de uma só roda, razão pela qual foram implementadas as respetivas caraterísticas.

5.6 Resultados Obtidos

A apresentação dos resultados obtidos segue a designação correspondente à esquematização apresentada na Figura 32.

O deslocamento dos robôs deu-se sobre uma tábua de 1000X1000mm com uma escala quadriculada de 100x100mm. Considerando as limitações dos métodos utilizados (régua e transferidor) para efetuar as medições da posição, foi admitida uma incerteza de ±5mm (posição) e ±2 graus (orientação) nos valores medidos em todas as estratégias.

Estratégia A – Controlo em Posição

A primeira estratégia implementada, como já foi referido, baseia-se em dois movimentos distintos, rotação e translação.

- **Estratégia A1 – Deslocamento em Parábola**

Os valores iniciais utilizados são apresentados na Tabela 2.

Tabela 2 – Valores utilizados na implementação da estratégia A1

	NXT	EV3
Nº iterações [n]	4	4
Diâmetro das rodas (d1, d2) [mm]	56	56
Distância entre eixos (disteixos) [mm]	112	119

	NXT/EV3	
	Caso 1	Caso 2
Posição inicial (x_0) [mm]	0	0
Posição inicial (y_0) [mm]	0	0
Orientação inicial (θ_0) [graus]	0	0
Posição final (x_f) [mm]	800	1000
Posição final (y_f) [mm]	200	1000
Orientação final (θ_f) [graus]	23,63	60,25

Na Tabela 3 são apresentados os resultados para cada caso, tendo sido realizados dois testes para cada um.

Tabela 3 – Resultados da implementação da trajetória em parábola com controlo em posição

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	790	780	956	990	760	790	1000	980
Erro (x_f) [mm]	-10	-20	-44	-10	-40	-10	0	-20
Posição final (y_f) [mm]	225	250	1000	970	240	230	900	960
Erro (y_f) [mm]	+25	+50	0	-30	+40	+30	-100	-40
Orientação final (θ_f) [graus]	26	28°	58°	61°	29°	27°	65°	66°
Erro (θ_f) [graus]	+2,37°	+4,37°	-2,25°	+0,75°	+5,37°	+3,37°	+4,75°	+5,75°

Analisados os valores obtidos pelos encoders e recorrendo ao modelo descrito no capítulo 5.3, foram refeitas as trajetórias com base em odometria, como pode ser observado nas figuras 46 a 49.

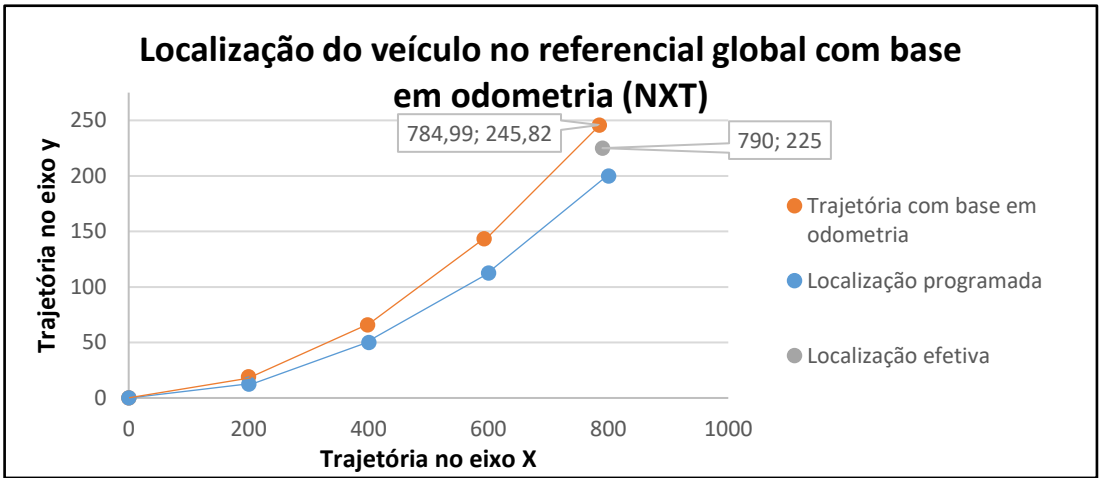

Figura 45 – Localização do veículo com recurso a odometria, caso 1, NXT

A localização efetiva é a posição real atingida medida laboratorialmente.

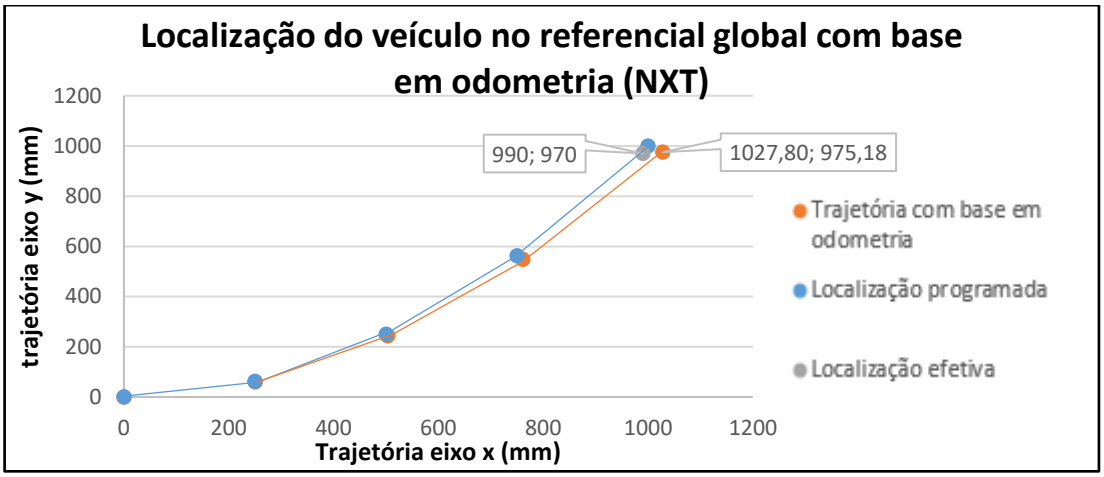

Figura 46 – Localização do veículo com recurso a odometria, caso 2, NXT

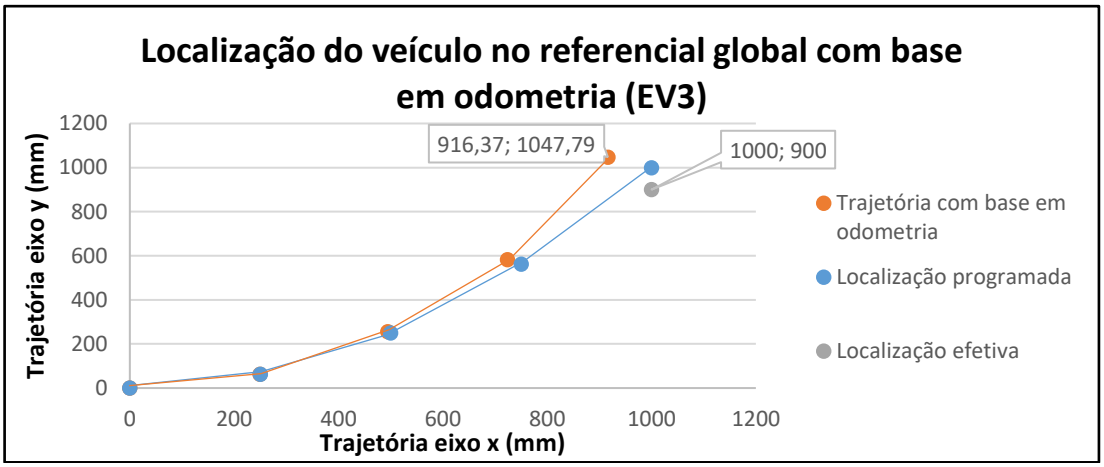

Figura 47 – Localização do veículo com recurso a odometria, caso 2, EV3

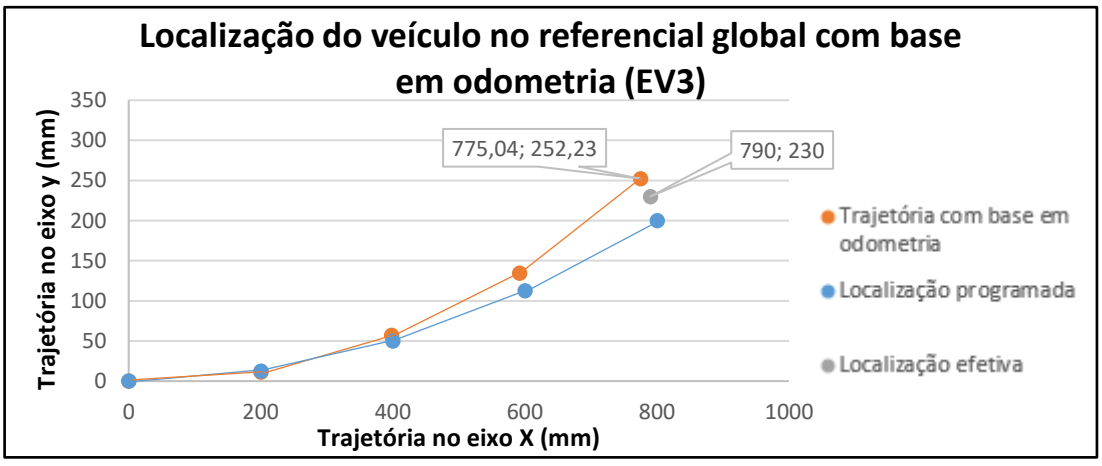

Figura 48 – Localização do veículo com recurso a odometria, caso 1, EV3

- **Estratégia A2 – Deslocamento Semicircular**

Alterando a trajetória para um perfil de deslocamento semicircular, foram obtidos os valores apresentados na Tabela 4.

Tabela 4 – Valores utilizados na implementação da estratégia A2

	NXT/EV3	
	Caso 1	Caso 2
Posição inicial (x_0) [mm]	0	0
Posição inicial (y_0) [mm]	0	0
Orientação inicial (θ_0) [graus]	90°	90°
Posição final (x_f) [mm]	500	900
Posição final (y_f) [mm]	0	0
Orientação final (θ_f) [graus]	-90°	-90°

Os resultados obtidos podem ser observados na Tabela 5.

Tabela 5 – Resultados da implementação da trajetória em semicírculo com controlo em posição

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	440	490	860	925	480	505	890	900
Erro (x_f) [mm]	-60	-10	-60	+25	-20	+5	-10	0
Posição final (y_f) [mm]	-30	10	10	40	-10	5	0	5
Erro (y_f) [mm]	-30	+10	-10	+40	-10	+5	0	+5
Orientação final (θ_f) [graus]	-82	-88°	-93°	-85°	-92°	-88°	-90°	-88°
Erro (θ_f) [graus]	-8°	-2°	+3°	-5°	+2°	-2°	0°	-2°

As trajetórias refeitas com base em odometria podem ser observadas nas figuras 50 a 53.

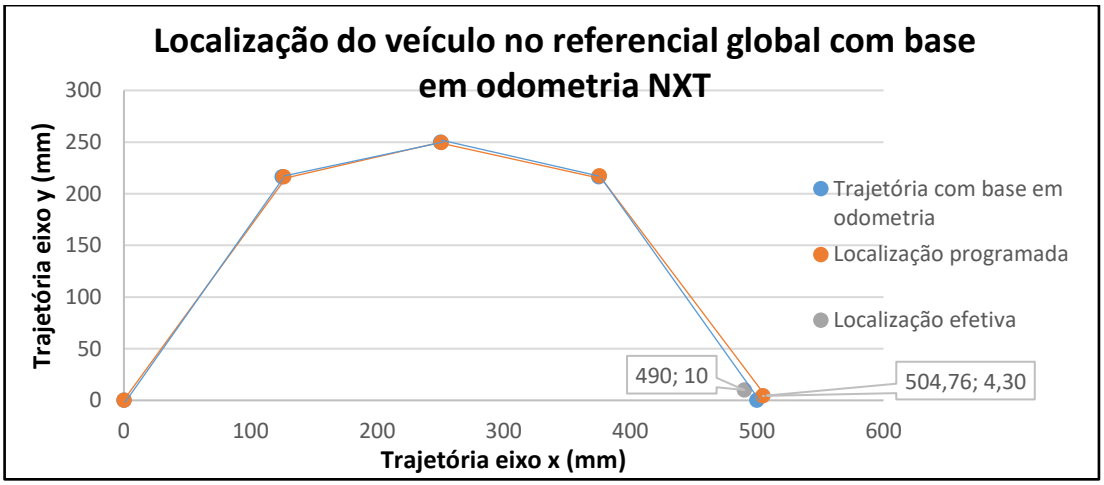

Figura 49 – Localização do veículo com recurso a odometria, caso 1, NXT

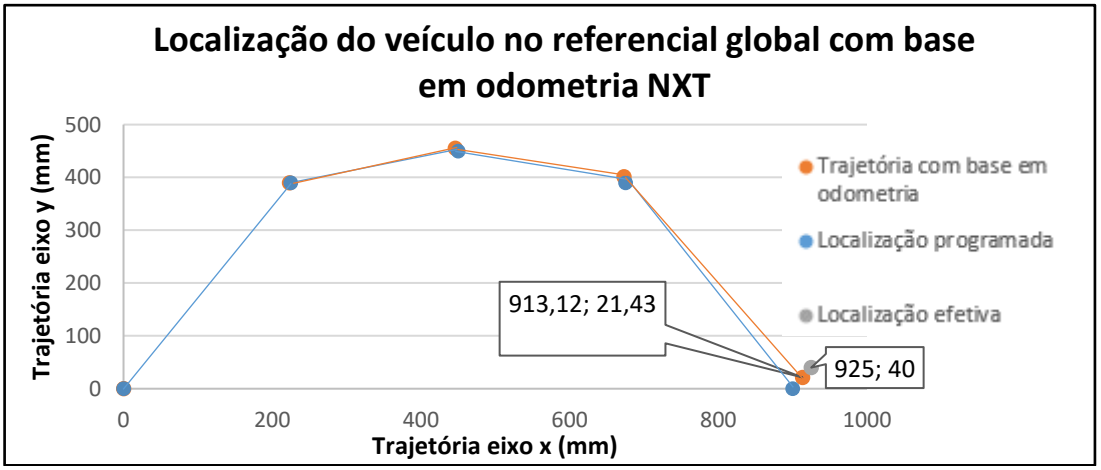

Figura 50 – Localização do veículo com recurso a odometria, caso 2, NXT

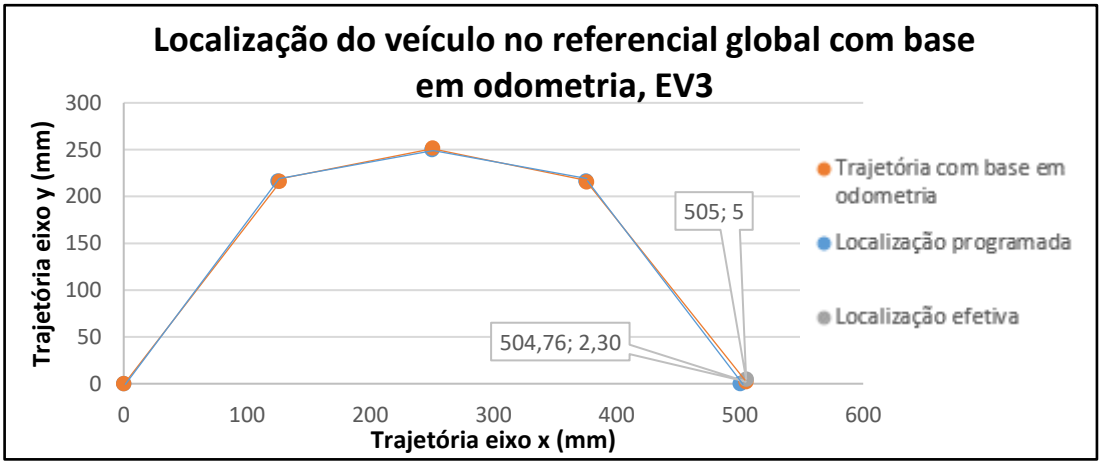

Figura 51 – Localização do veículo com recurso a odometria, caso 1, EV3

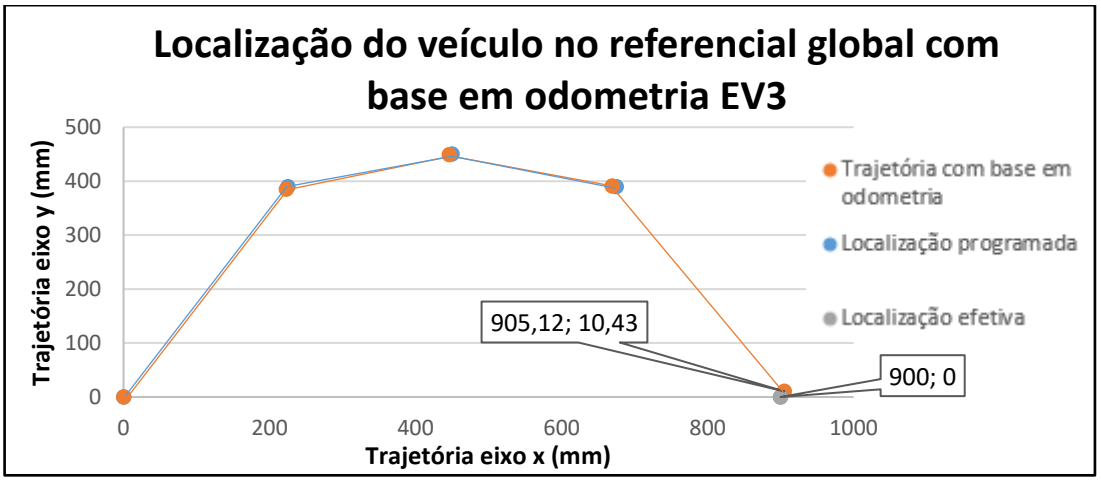

Figura 52 – Localização do veículo com recurso a odometria, caso 2, EV3

Comparando a estratégia A1 com a A2, os resultados do deslocamento semicircular revelaram-se mais eficazes. Em relação aos dois robôs, o EV3 mostrou-se mais eficiente comparativamente ao NXT no que diz respeito à trajetória calculada com base em odometria e à localização efetiva, tanto no caso 1 como no caso 2.

- **Estratégia A3 – Deslocamento em Linha Reta**

Por último, foi implementado um deslocamento em linha reta nos dois sistemas. Os resultados obtidos, uma vez que não existe acumulação de erros devido às diferentes iterações, sendo a trajetória realizada numa única iteração, foram melhores. As pequenas diferenças encontradas entre a posição desejada e a obtida devem-se provavelmente às diferenças existentes nas caraterísticas dos motores e/ou à incerteza do alinhamento das rodas.

Os valores utilizados foram os mesmos apresentados na estratégia A2 e os resultados encontram-se na Tabela 6.

Tabela 6 – Resultados da implementação da trajetória em linha reta com controlo em posição

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	495	492	886	905	501	498	894	902
Erro (x_f) [mm]	-5	-8	-14	+5	+1	-2	-6	+2
Posição final (y_f) [mm]	-5	-8	-30	-45	8	5	16	20
Erro (y_f) [mm]	-5	-8	-30	-45	+8	+5	+16	+20
Orientação final (θ_f) [graus]	-3°	-4°	-8°	-10°	5°	2°	10°	15°
Erro (θ_f) [graus]	-3°	-4°	-8°	-10°	+5°	+2°	+10°	+15°

Estratégia B – Controlo em Velocidade

A estratégia B foi implementada com o objetivo de melhorar as trajetórias realizadas na estratégia A, melhorando o tempo de execução de cada uma, realizando um movimento continuo do ponto de origem ao ponto de destino.

Foram ainda implementados dois perfis de velocidade, conseguindo desta forma variar as fases de aceleração e desaceleração dos veículos, tentando minimizar os possíveis erros que ocorrem nestas zonas de movimento devido ao escorregamento entre as rodas e o pavimento.

- **Estratégia B1 – Deslocamento em Parábola**

Na Tabela 7 são apresentados os valores dos parâmetros iniciais utilizados em ambos os perfis de aceleração, sinusoidal e trapezoidal.

Tabela 7 – Valores utilizados na implementação da estratégia B1

	NXT/EV3	
	Caso 1	Caso 2
Posição inicial (x_0) [mm]	0	0
Posição inicial (y_0) [mm]	0	0
Orientação inicial (θ_0) [graus]	0°	0°
Posição final (x_f) [mm]	400	800
Posição final (y_f) [mm]	800	400
Orientação final (θ_f) [graus]	79,25°	50,75

Tempo de ciclo [s]	8
Tempo de iteração [s]	0,1

- **Estratégia B1.1 – Perfil de Velocidade Sinusoidal**

A estratégia B1.1 foi implementada com um perfil de velocidade sinusoidal apresentado na Figura 28 do capítulo 5.2. O perfil sinusoidal é o perfil mais suave que pode ser aplicado, sendo expectável apresentar melhores resultados comparativamente ao trapezoidal. A desvantagem é que não permite ajustar a rampa de aceleração/desaceleração.

Os valores obtidos nesta estratégia são apresentados na Tabela 8.

Tabela 8 – Valores medidos na implementação laboratorial da estratégia B1.1

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	390	385	770	768	380	378	820	828
Erro (x_f) [mm]	-10	-15	-30	-32	-20	-22	+20	+28
Posição final (y_f) [mm]	820	818	440	446	850	850	360	362
Erro (y_f) [mm]	+20	+18	+40	+46	+50	+50	-40	-38
Orientação final (θ_f) [graus]	81°	83°	53°	54°	84°	84°	48°	49°
Erro (θ_f) [graus]	+1,75°	+3,75°	+2,25°	+3,25°	+4,75°	+4,75°	-2,75°	-1,75°

Por forma a validar a implementação das trajetórias, são apresentadas nas figuras 54 a 56 e 56 a 60, a evolução das trajetórias resultantes dos cálculos efetuados no programa em Matlab assim como pelos cálculos realizados no Simulink e posteriormente enviados para o controlador dos robôs.

Caso 2 – NXT

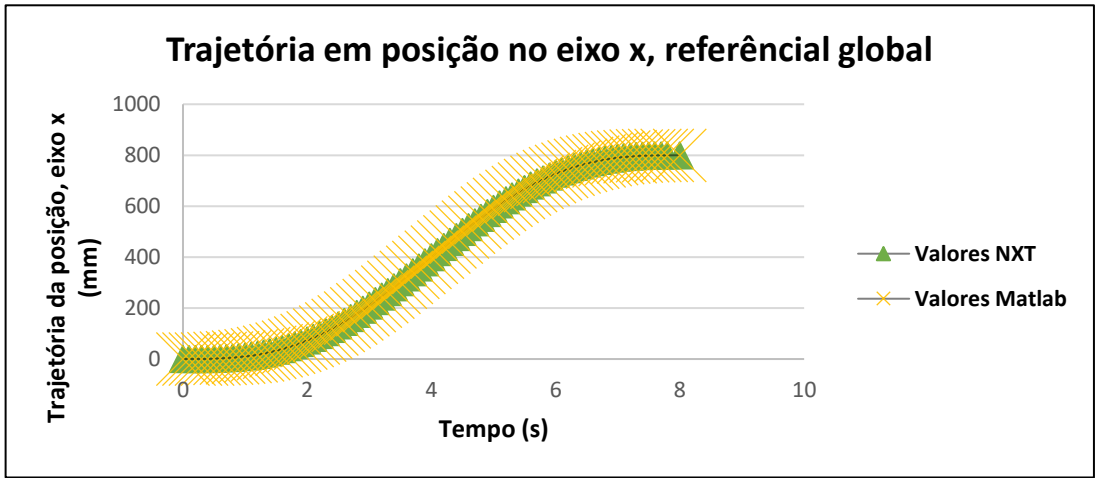

Figura 53 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

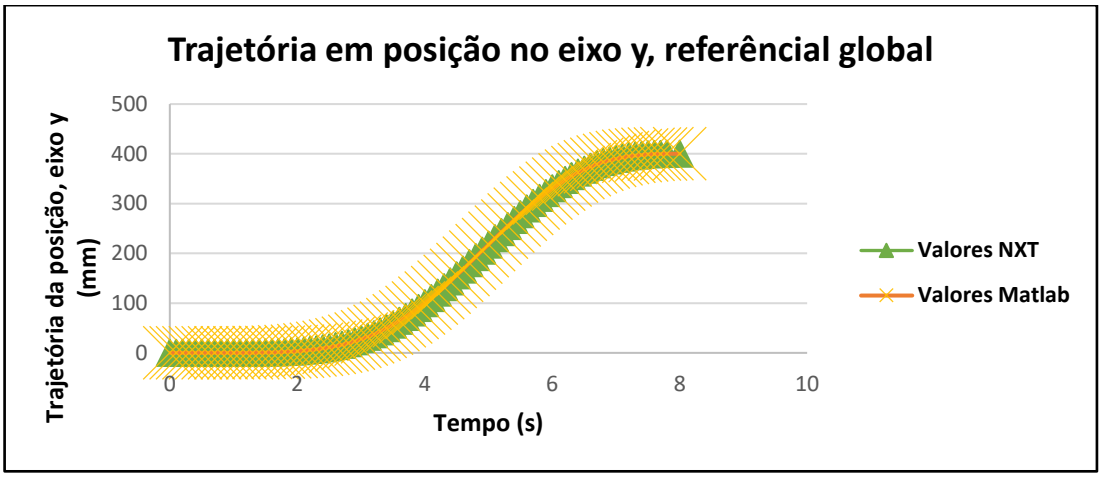

Figura 54 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

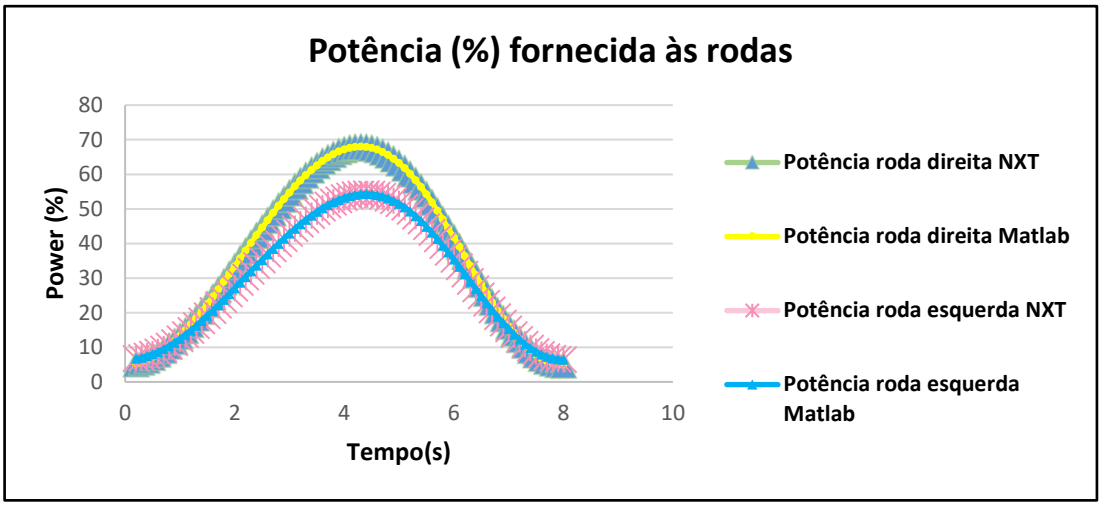

Figura 55 – Valores da potência calculada pelo Matlab e NXT para cada roda

Recorrendo à odometria e à leitura dos encoders, é possível reconstruir a trajetória que o veículo percorreu ao longo do tempo (Figura 56).

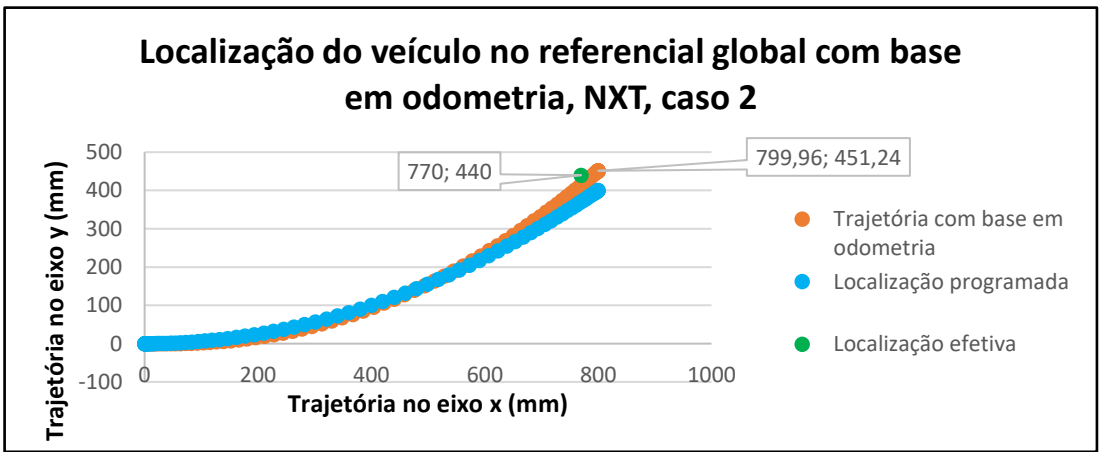

Figura 56 – Localização do veículo NXT com recurso a odometria, caso 2

Analisando a Tabela 8 e o gráfico da Figura 56, é possível observar que existem algumas diferenças na posição final do veículo. Erros como por exemplo, escorregamento das rodas, assim como a resolução dos encoders e erros associados à medição laboratorial, podem ser considerados como causas dessas discrepâncias.

Caso 2 – EV3

O mesmo programa foi utilizado no veículo EV3, retificando as caraterísticas dos motores e a distância do entre eixo.

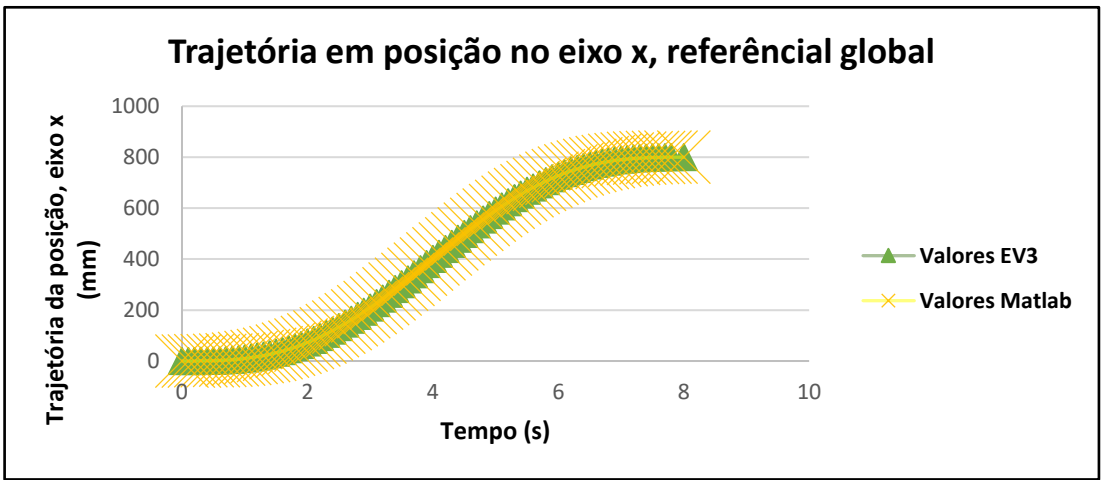

Figura 57 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

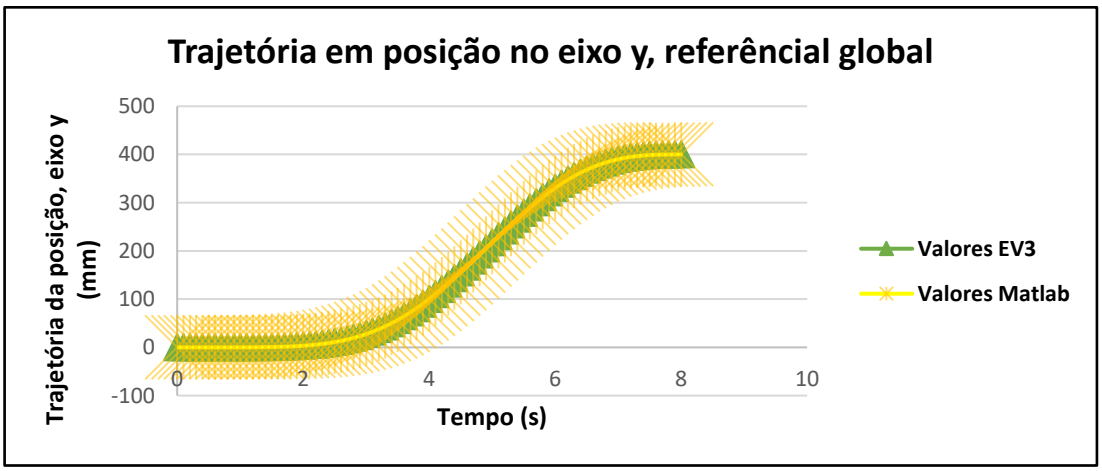

Figura 58 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

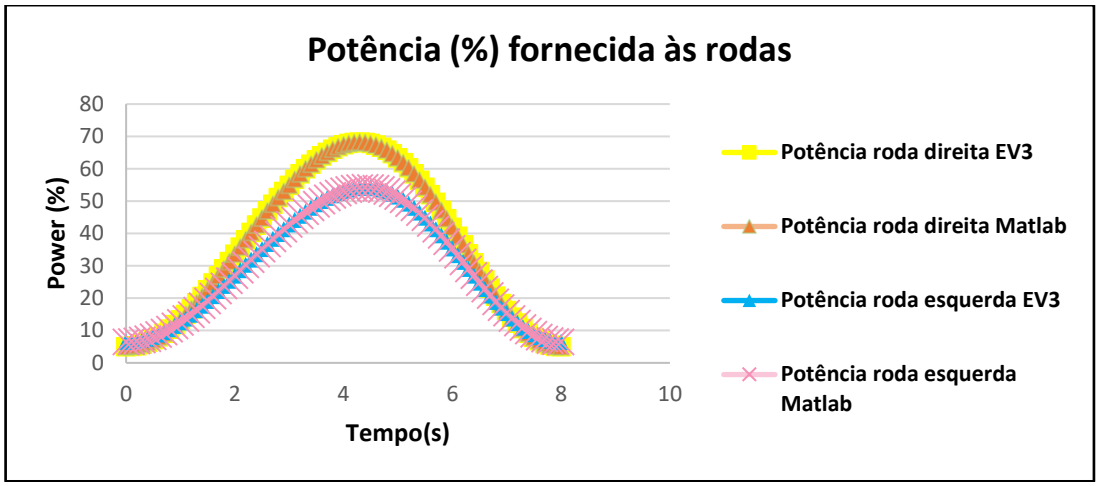

Figura 59 – Valores da potência calculada pelo Matlab e EV3 para cada roda

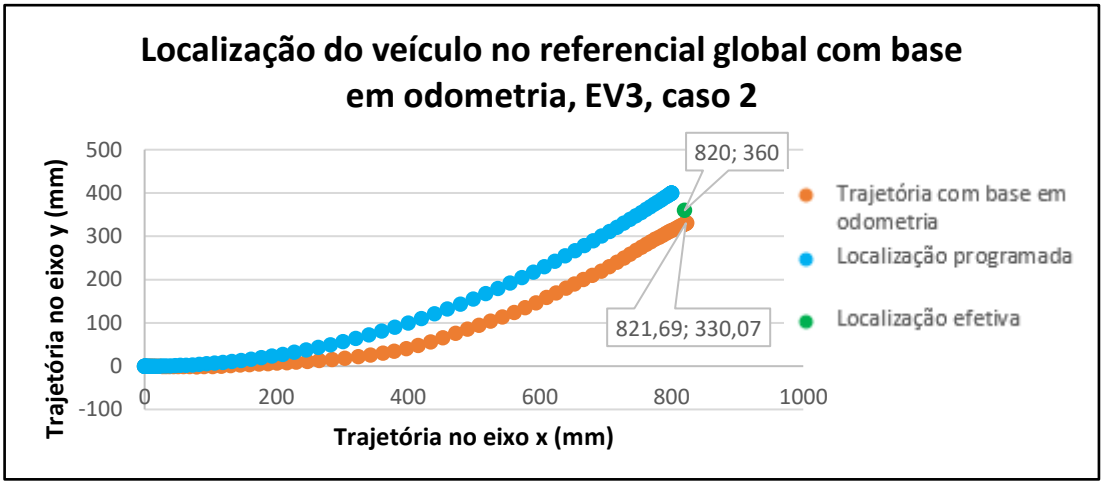

Figura 60 – Localização do veículo EV3 com recurso a odometria, caso 2

Comparando as duas plataformas, o NXT apresenta uma curva de odometria mais próxima da localização programada. O EV3 fica bastante aquém do esperado, principalmente na coordenada y. Quanto à diferença entre a localização efetiva e a trajetória com base em odometria, o NXT apresenta uma maior, embora ligeira, diferença.

- **Estratégia B1.2 – Perfil de Velocidade Trapezoidal**

A alteração do perfil de velocidade sinusoidal para trapezoidal permite impor com maior rigor, a aceleração, velocidade máxima e desaceleração dos veículos, conseguindo ao mesmo tempo, minimizar os efeitos de escorregamento das rodas com o pavimento. Os resultados podem ser observados na Tabela 9.

Tabela 9 – Valores medidos na implementação laboratorial da estratégia B1.2

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	410	415	740	740	400	403	845	850
Erro (x_f) [mm]	+10	+15	-60	-60	0	+3	+45	+50
Posição final (y_f) [mm]	810	807	500	502	830	834	390	393
Erro (y_f) [mm]	+10	+7	+100	+102	+30	+34	-10	-7
Orientação final (θ_f) [graus]	78°	76°	58°	59°	80°	79°	49°	51
Erro (θ_f) [graus]	-1,25°	-3,25°	+7,25°	+8,25°	+0,75°	-0,25°	+1,75°	-0,25°

Caso 2 – NXT

A validação pode ser observada nas figuras 62 a 64.

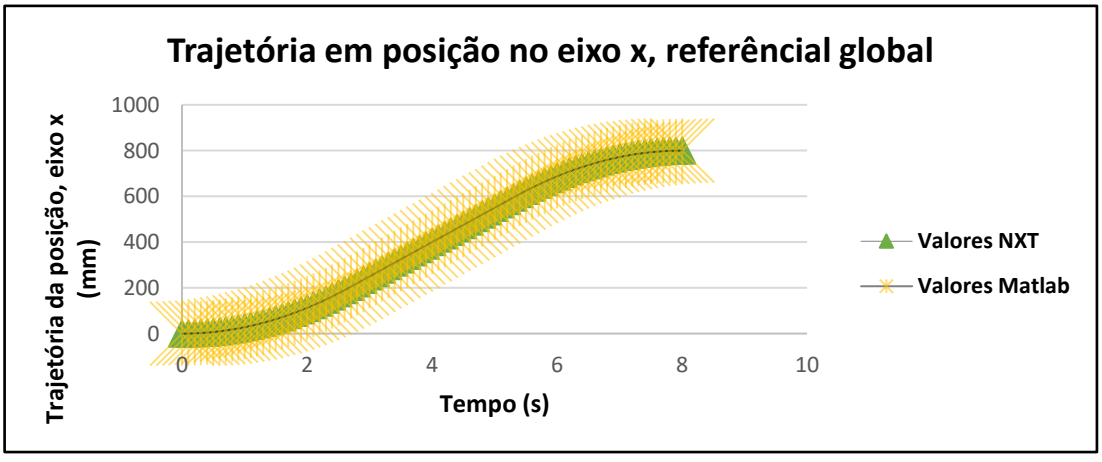

Figura 61 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

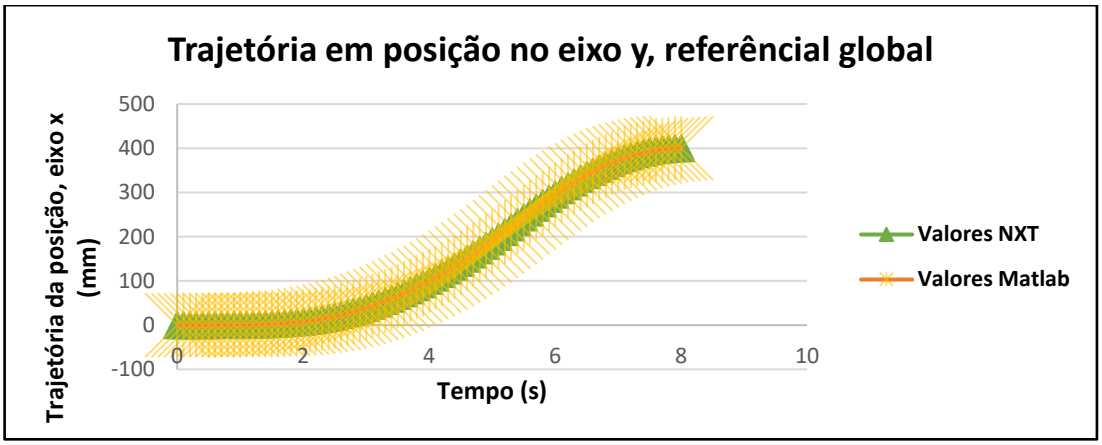

Figura 62 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

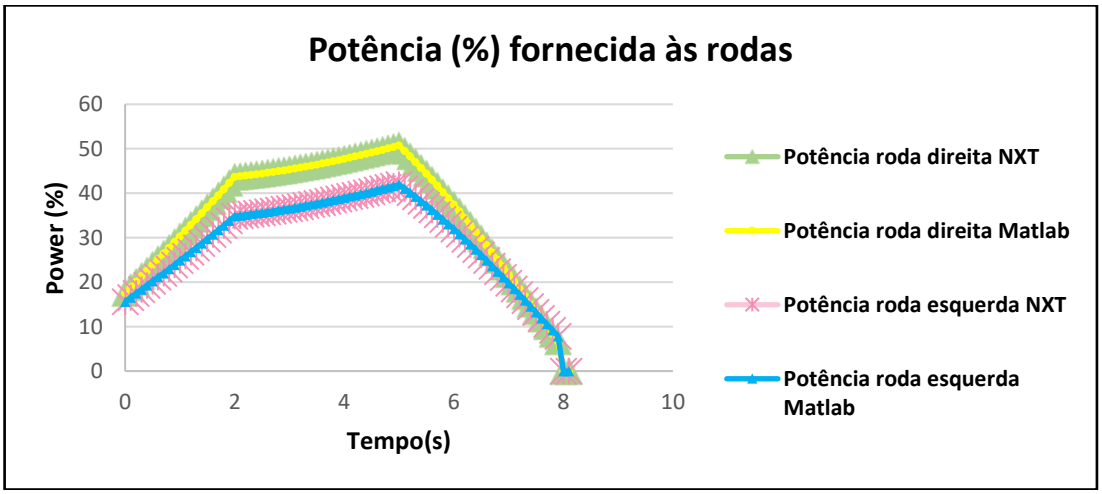

Figura 63 – Valores da potência calculada pelo Matlab e NXT para cada roda

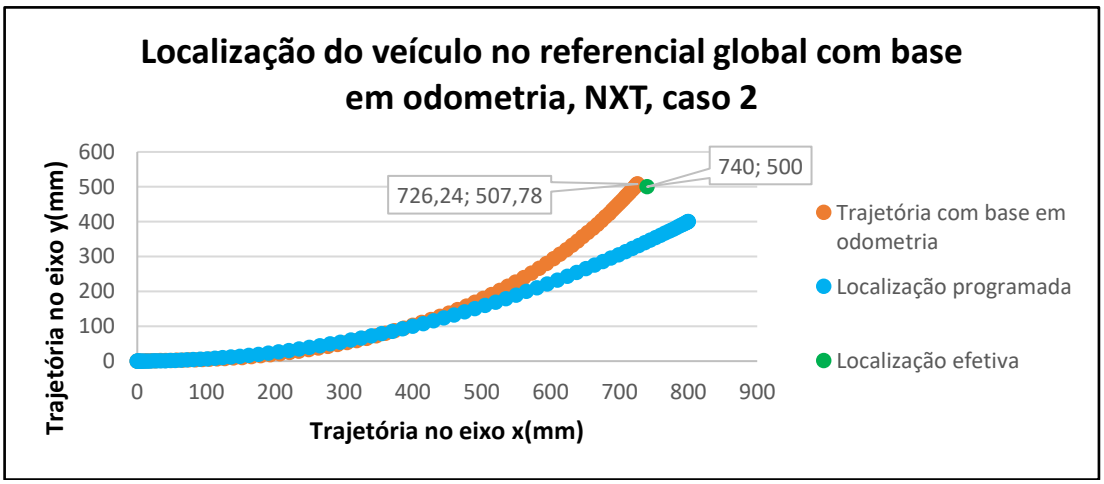

Figura 64 – Localização do veículo NXT com recurso a odometria, caso 2

Caso 2 – EV3

A validação da implementação realizada, pode ser observada nas figuras 66 a 68.

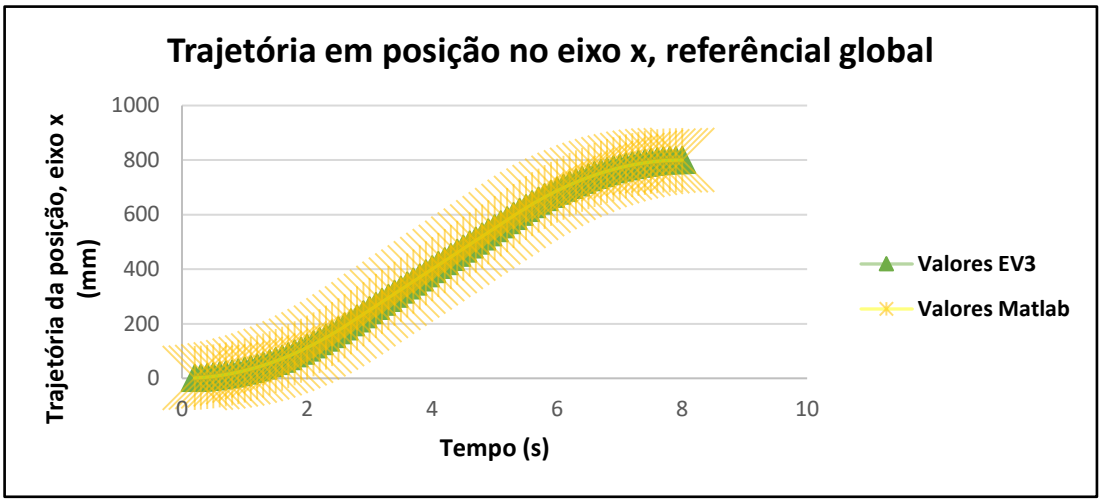

Figura 65 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

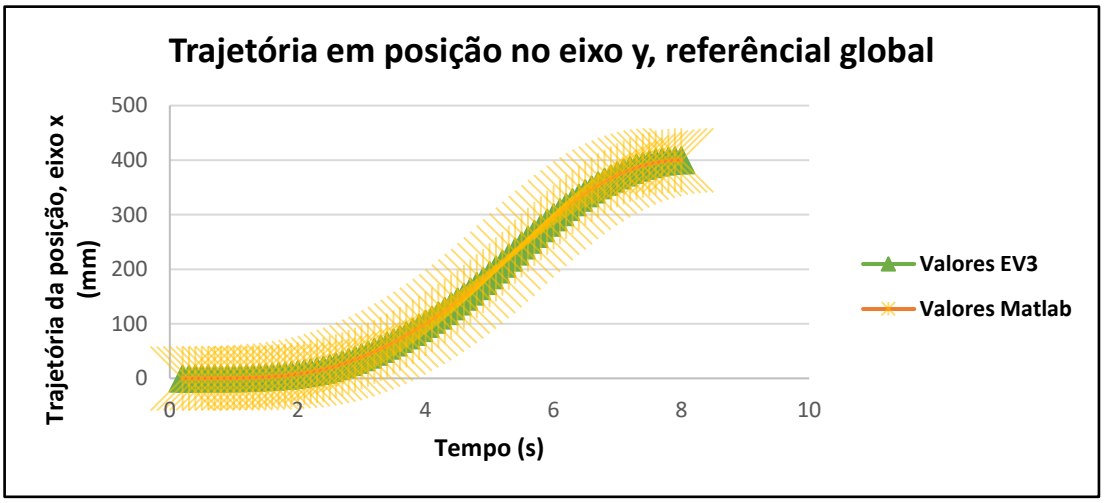

Figura 66 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

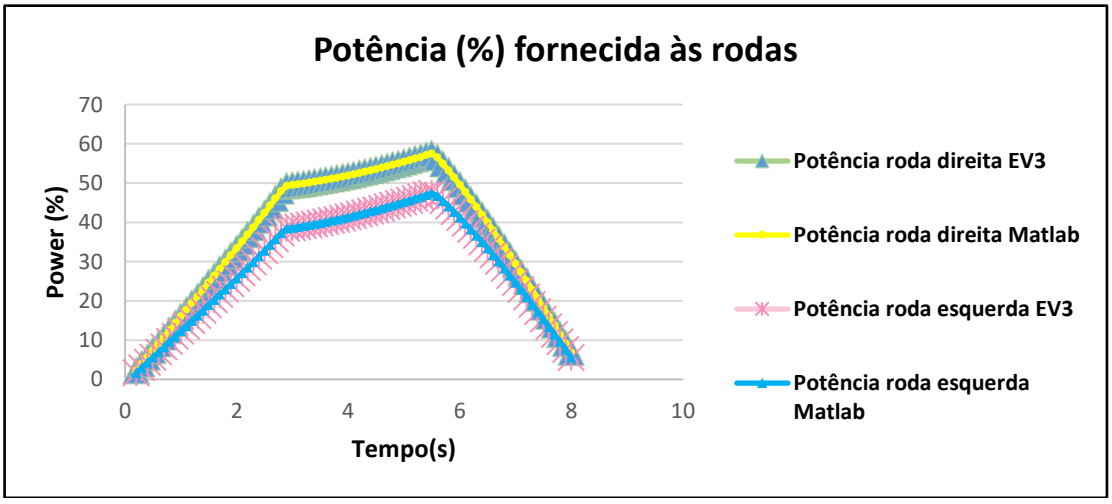

Figura 67 – Valores da potência calculada pelo Matlab e EV3 para cada roda

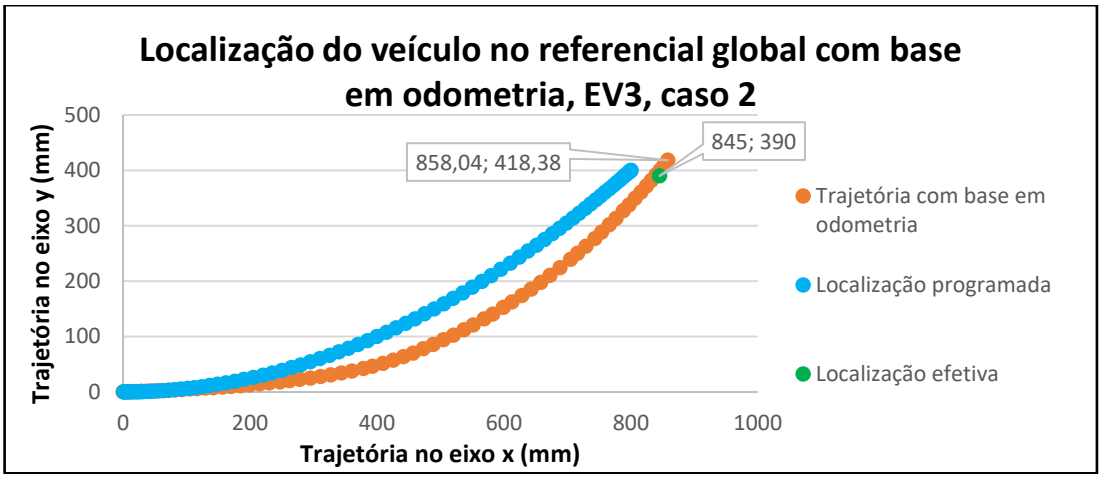

Figura 68 - Localização do veículo EV3 com recurso a odometria, caso 2

Fazendo uma análise aos resultados obtidos laboratorialmente e posteriormente com recurso a odometria, é possível observar que o deslocamento em parábola com perfil de velocidade trapezoidal piorou a trajetória no que diz respeito ao NXT, levando-o a uma localização efetiva mais afastada da programada. Em relação ao EV3, as coordenadas da localização efetiva apresentaram uma maior diferença das coordenadas com base em odometria, afastando-se ligeiramente uma da outra.

Estratégia B2 – Deslocamento Semicircular

A estratégia seguinte foi implementar um deslocamento semicircular. De forma análoga à estratégia anterior, foram implementados perfis de velocidade sinusoidal e trapezoidal.

Os parâmetros utilizados são apresentados na Tabela 10.

Tabela 10 – Valores utilizados na implementação da estratégia B2

	NXT/EV3	
	Caso 1	Caso 2
Posição inicial (x_0) [mm]	0	0
Posição inicial (y_0) [mm]	0	0
Orientação inicial (θ_0) [graus]	90°	90°
Posição final (x_f) [mm]	500	800
Posição final (y_f) [mm]	0	0
Orientação final (θ_f) [graus]	-90°	-90°

Tempo de ciclo [s]	8
Tempo de iteração [s]	0,1

- **Estratégia B2.1 – Perfil de Velocidade Sinusoidal**

Os valores obtidos na implementação das trajetórias semicirculares com perfil sinusoidal podem ser analisados na Tabela 11.

Tabela 11 – Valores medidos na implementação laboratorial da estratégia B2.1

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	520	525	860	848	560	567	840	830
Erro (x_f) [mm]	+20	+25	+60	+48	+60	+67	+40	+30
Posição final (y_f) [mm]	3	5	50	40	5	5	0	6
Erro (y_f) [mm]	+3	+5	+50	+40	+5	+5	0	+6
Orientação final (θ_f) [graus]	-95°	-92°	-70°	-76°	-88°	-87°	-90°	-89°
Erro (θ_f) [graus]	+5°	+2°	-20°	-6°	-2°	-3°	0°	-1°

Caso 1 – NXT

A validade da implementação pode ser analisada nas figuras 70 a 72 e 74 a 76.

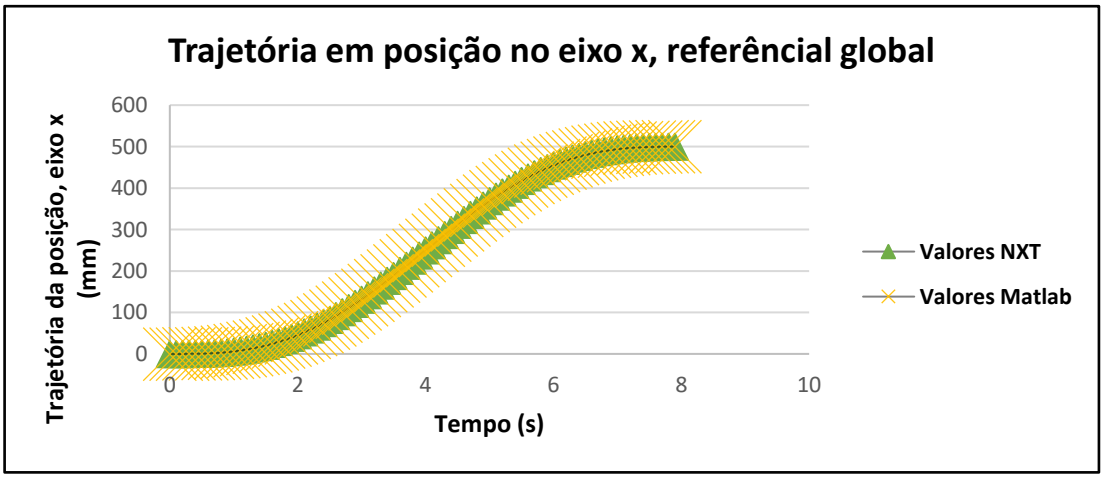

Figura 69 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

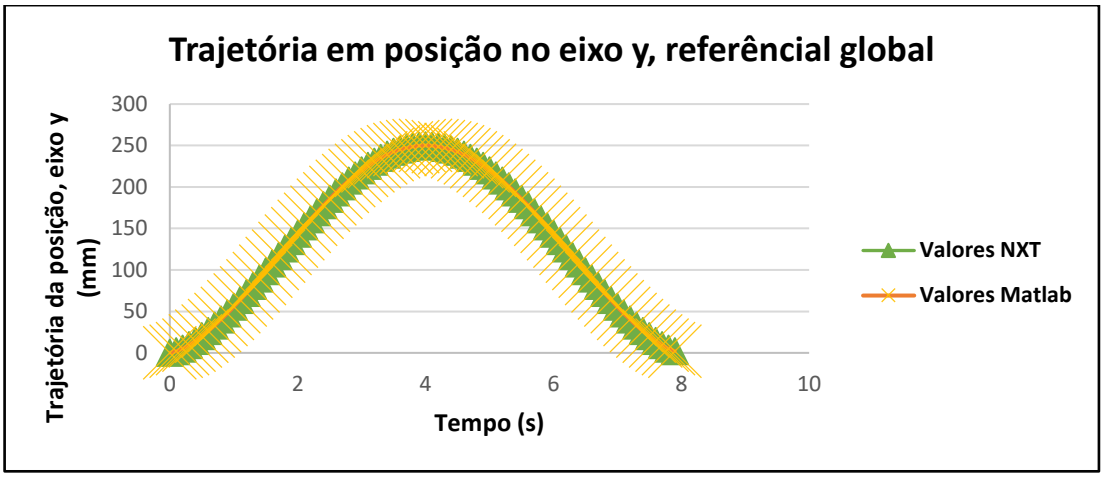

Figura 70 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

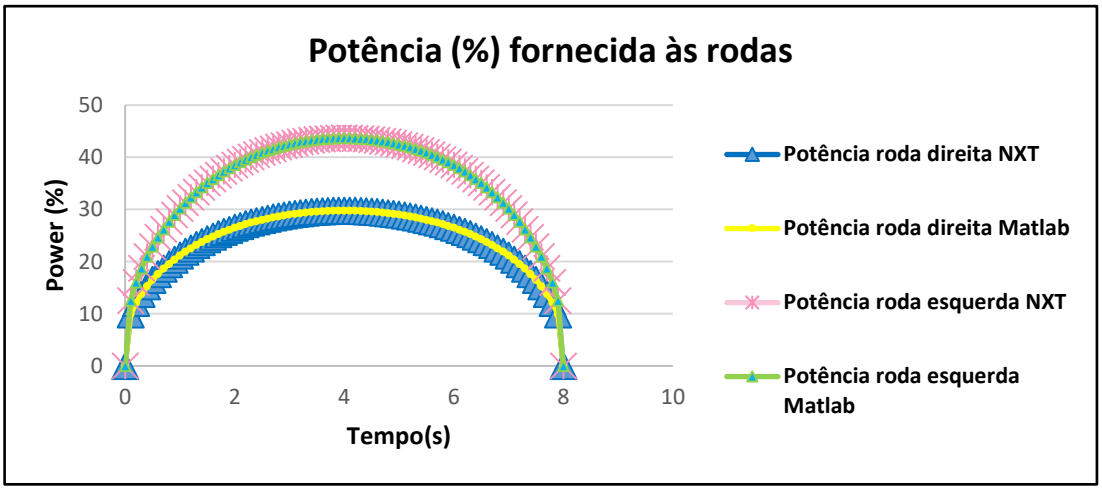

Figura 71 – Valores da potência calculada pelo Matlab e NXT para cada roda

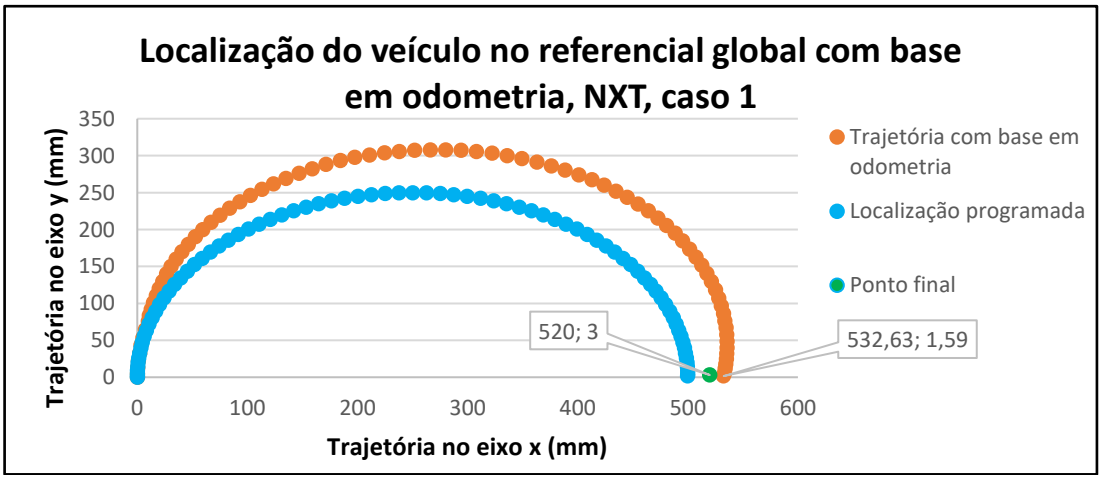

Figura 72 – Localização do veículo NXT com recurso a odometria, caso 1

Caso 1 – EV3

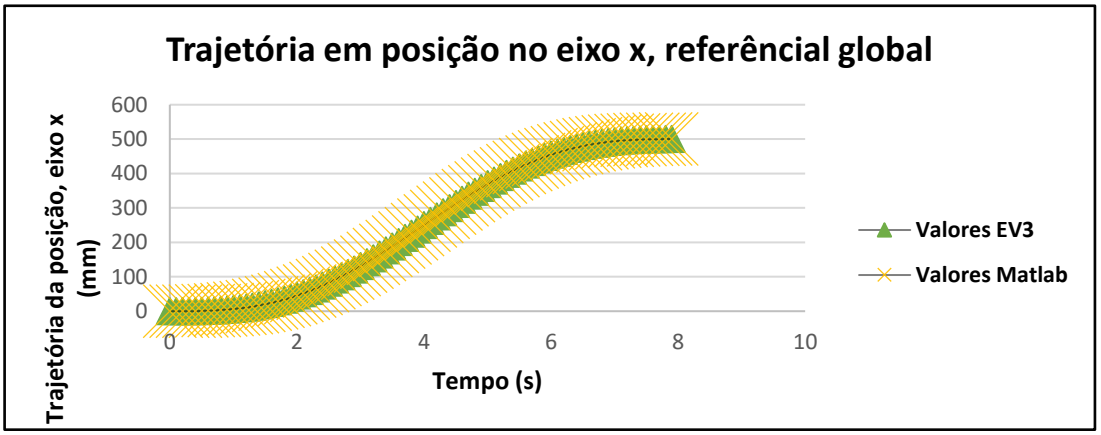

Figura 73 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

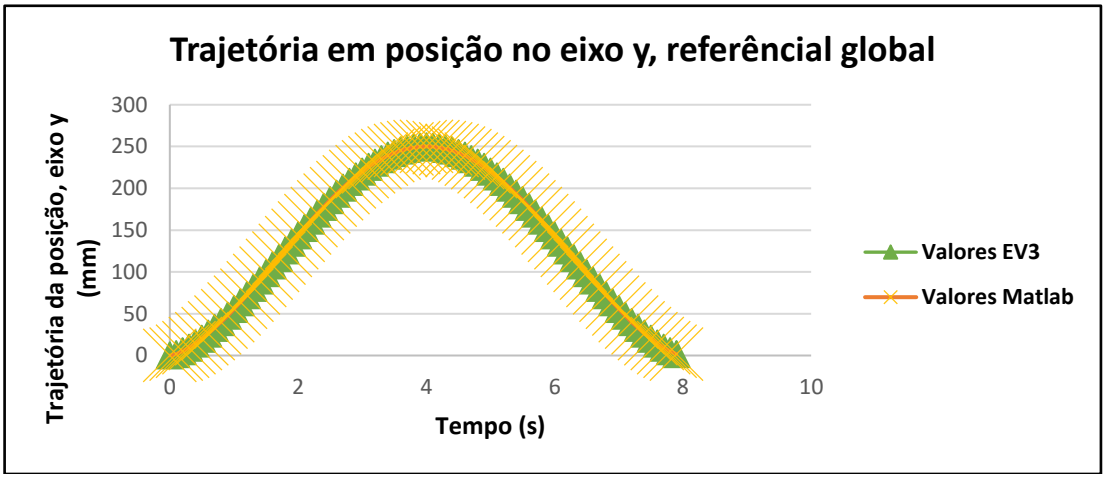

Figura 74 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

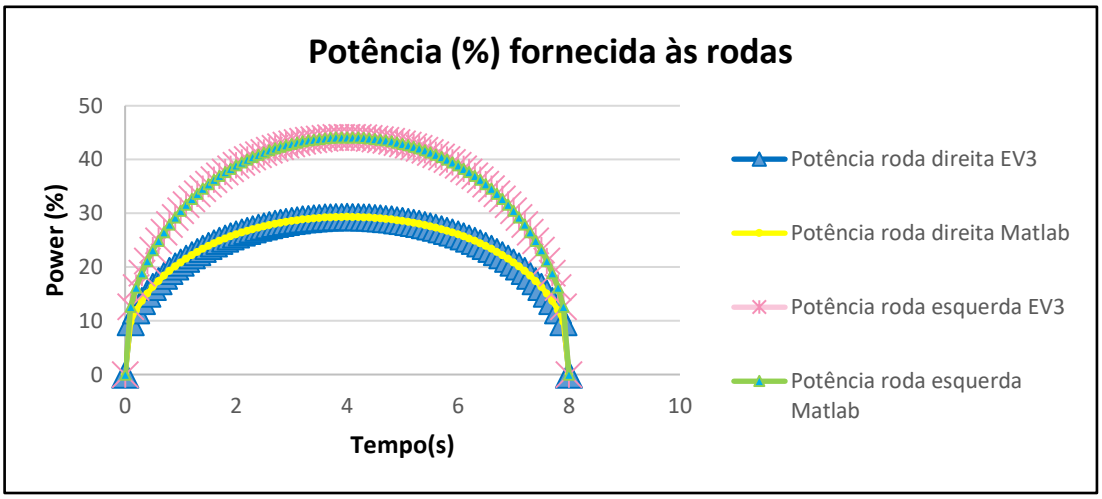

Figura 75 – Valores da potência calculada pelo Matlab e EV3 para cada roda

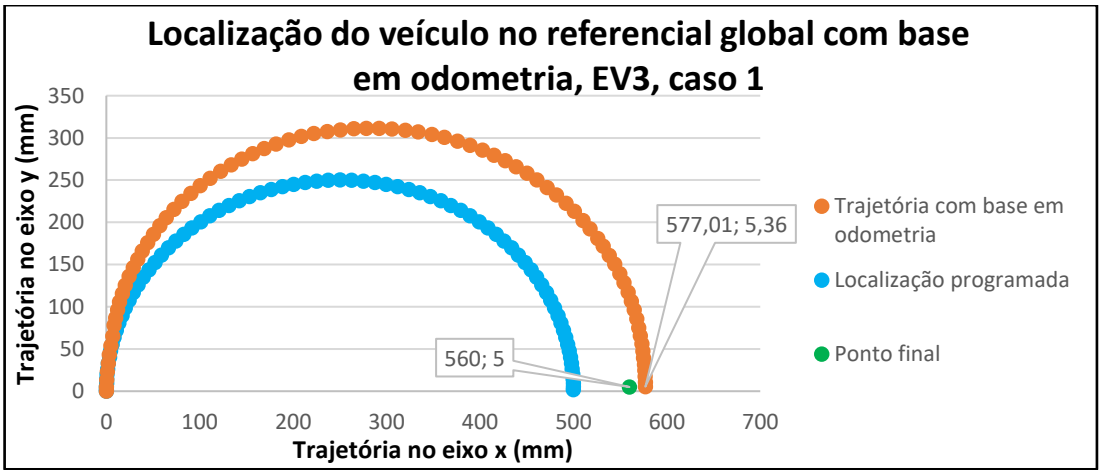

Figura 76 – Localização do veículo EV3 com recurso a odometria, caso 1

- **Estratégia B2.2 – Perfil de Velocidade Trapezoidal**

Os resultados obtidos com a alteração do perfil sinusoidal para trapezoidal podem ser analisados na Tabela 12.

Tabela 12 – Valores medidos na implementação laboratorial da estratégia B2.2

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	550	554	840	846	550	550	810	815
Erro (x_f) [mm]	+50	+54	+40	+46	+50	+50	+10	+15
Posição final (y_f) [mm]	20	20	60	50	0	2	0	4
Erro (y_f) [mm]	+20	+20	+60	+50	0	+2	0	+4
Orientação final (θ_f) [graus]	-91°	-90°	-80°	-78°	-90°	-89°	-88°	-89°
Erro (θ_f) [graus]	+1°	0°	-10°	-12°	0°	-1°	-2°	-1°

Caso 1 – NXT

A validade da implementação volta a ser apresentada nas figuras 78 a 80 e 82 a 84.

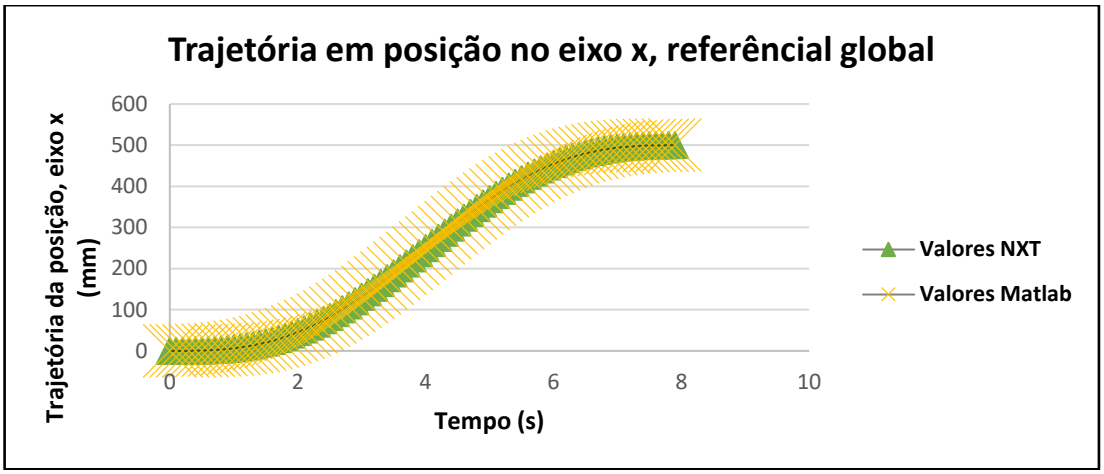

Figura 77 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

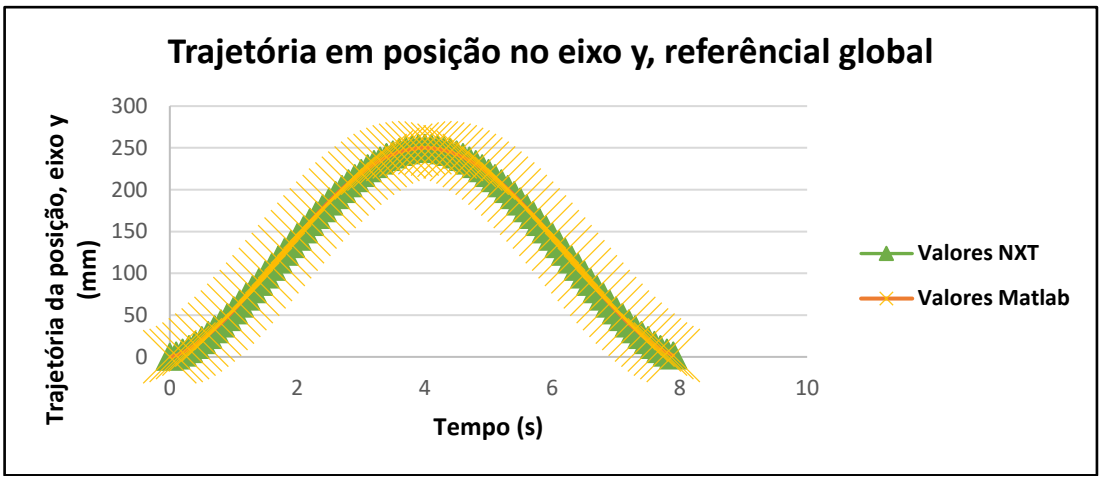

Figura 78 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

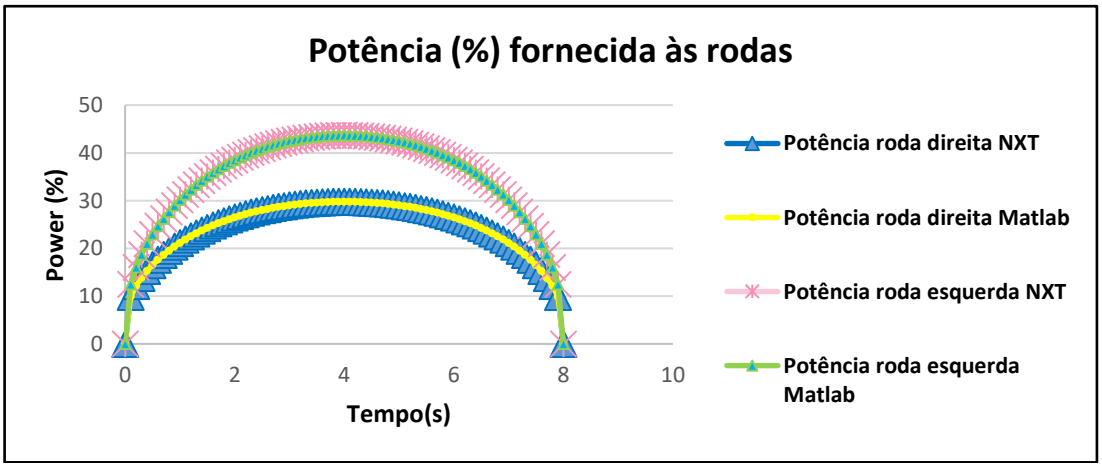

Figura 79 – Valores da potência calculada pelo Matlab e NXT para cada roda

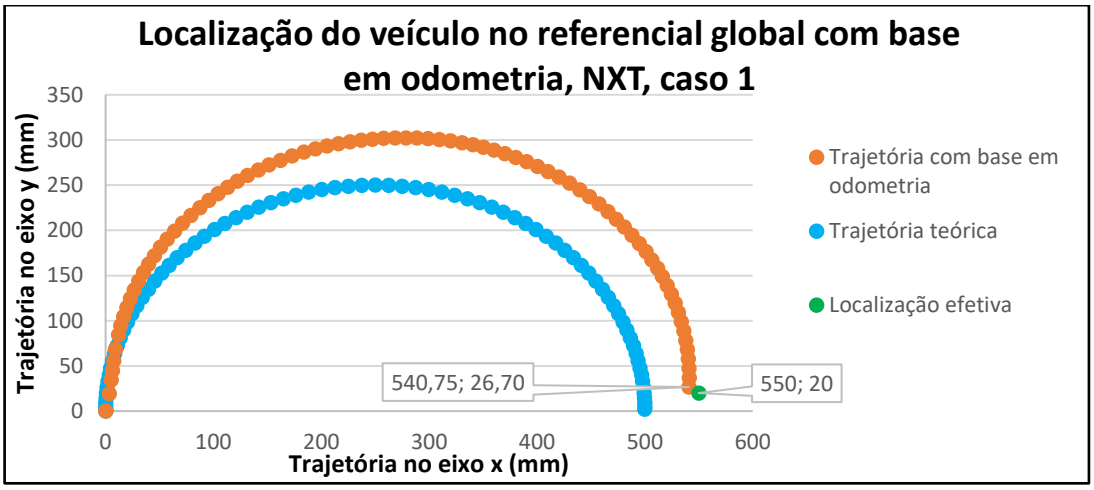

Figura 80 – Localização do veículo NXT com recurso a odometria, caso 1

Caso 1 – EV3

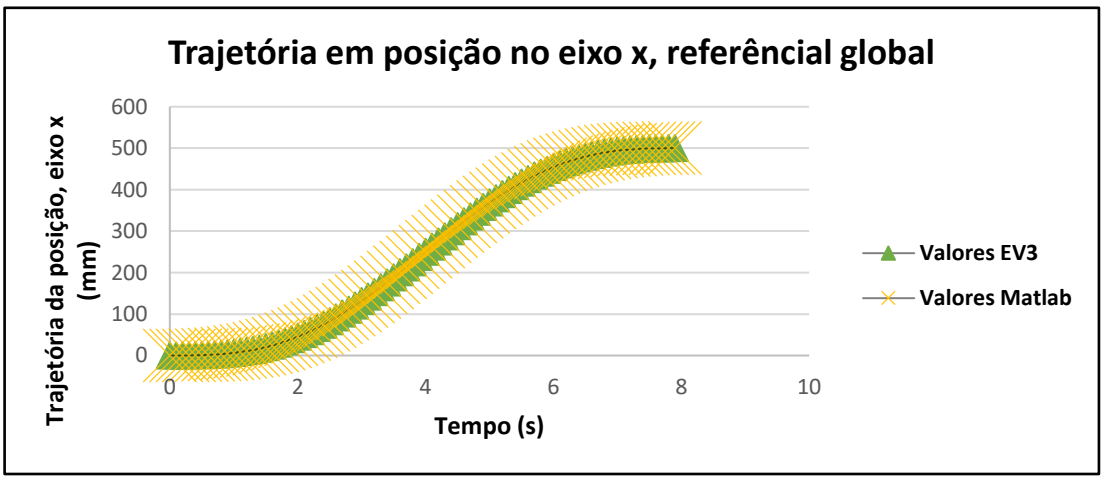

Figura 81 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

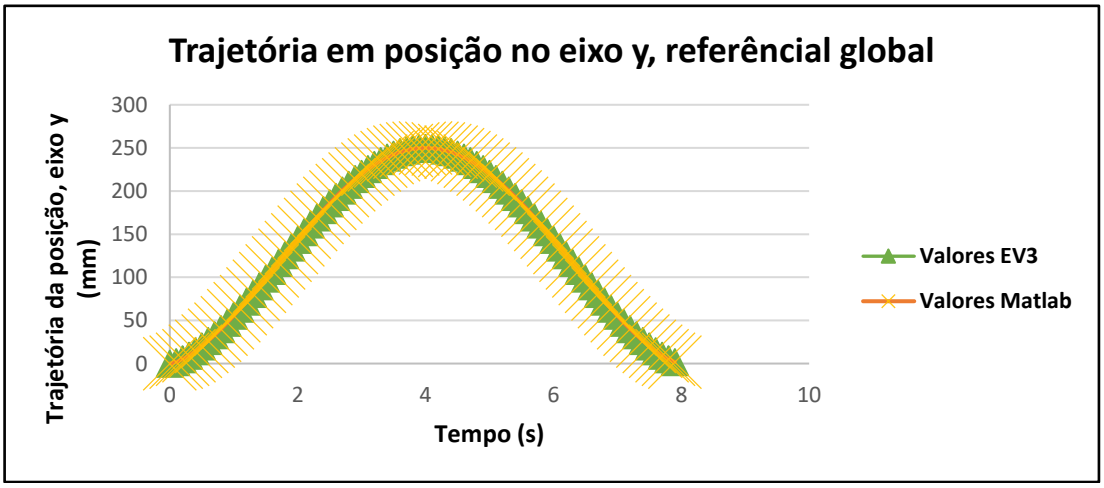

Figura 82 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

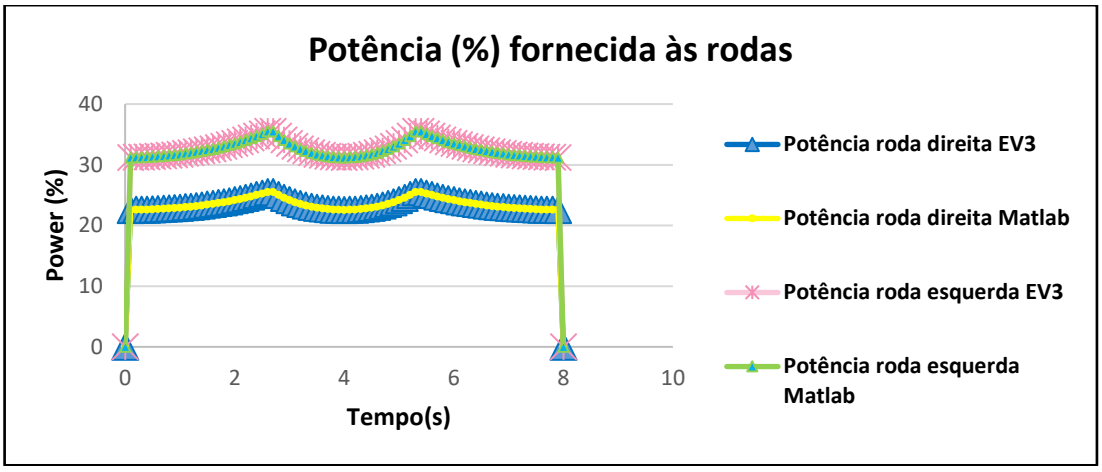

Figura 83 – Valores da potência calculada pelo Matlab e EV3 para cada roda

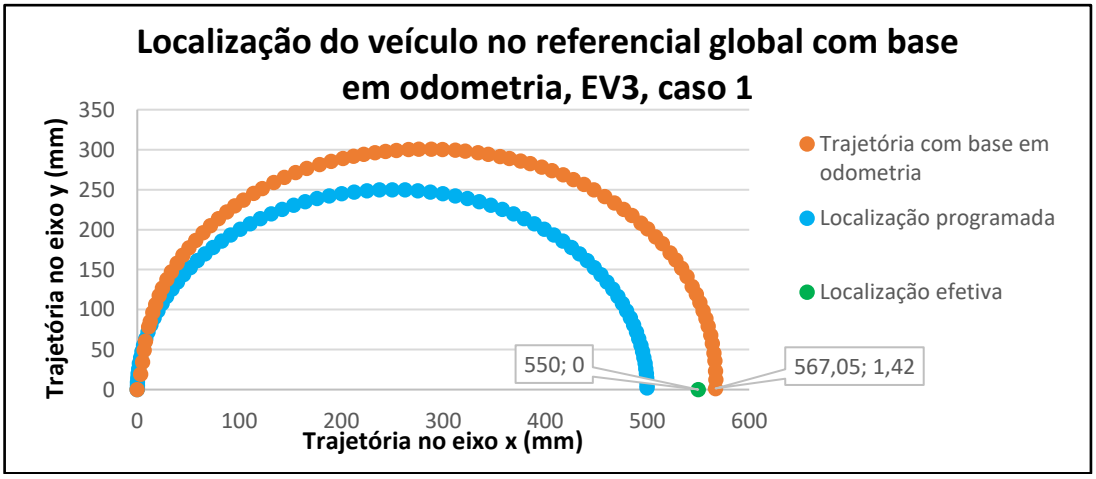

Figura 84 – Localização do veículo EV3 com recurso a odometria, caso 1

Apesar das diferenças entre as estratégias B2.1 e B2.2 serem pequenas, o perfil trapezoidal piorou ligeiramente os resultados no NXT e melhorou da mesma forma no veículo EV3.

- **<u>Estratégia B3 – Deslocamento em Linha Reta</u>**

A última estratégia aqui apresentada trata da implementação do deslocamento em linha reta.

O principal objetivo foi observar a influência das diferenças que os motores e as incertezas do alinhamento das rodas podem introduzir nos resultados finais, medindo a posição e orientação final das plataformas.

Os parâmetros iniciais aplicados apresentam-se na Tabela 13.

Tabela 13 – Valores utilizados na implementação da estratégia B3

	NXT/EV3	
	Caso 1	Caso 2
Posição inicial (x_0) [mm]	0	0
Posição inicial (y_0) [mm]	0	0
Orientação inicial (θ_0) [graus]	0°	0°
Posição final (x_f) [mm]	500	800
Posição final (y_f) [mm]	0	0
Orientação final (θ_f) [graus]	0°	0°

Tempo de ciclo [s]	8
Tempo de iteração [s]	0,1

- **Estratégia B3.1 – Perfil de Velocidade Sinusoidal**

Os valores obtidos na implementação do deslocamento em linha reta com perfil de velocidade sinusoidal podem ser observados na Tabela 14.

Tabela 14 – Valores medidos na implementação laboratorial da estratégia B3.1

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	460	460	785	786	500	505	810	805
Erro (x_f) [mm]	-40	-40	-15	-14	0	+5	+10	+5
Posição final (y_f) [mm]	-1	-2	-5	-6	25	23	35	30
Erro (y_f) [mm]	-1	-2	-5	-6	+25	+23	+35	+30
Orientação final (θ_f) [graus]	-1°	-1°	-10°	-10°	15°	14°	20°	18°
Erro (θ_f) [graus]	-1°	-1°	-10°	-10°	+15°	+14°	+20°	+18°

Seguindo o método utilizados nas estratégias B1 e B2, a validação da implementação da trajetória em linha reta com perfil sinusoidal, pode ser observada nas figuras 86 a 88 e 90 a 92.

Caso 1 – NXT

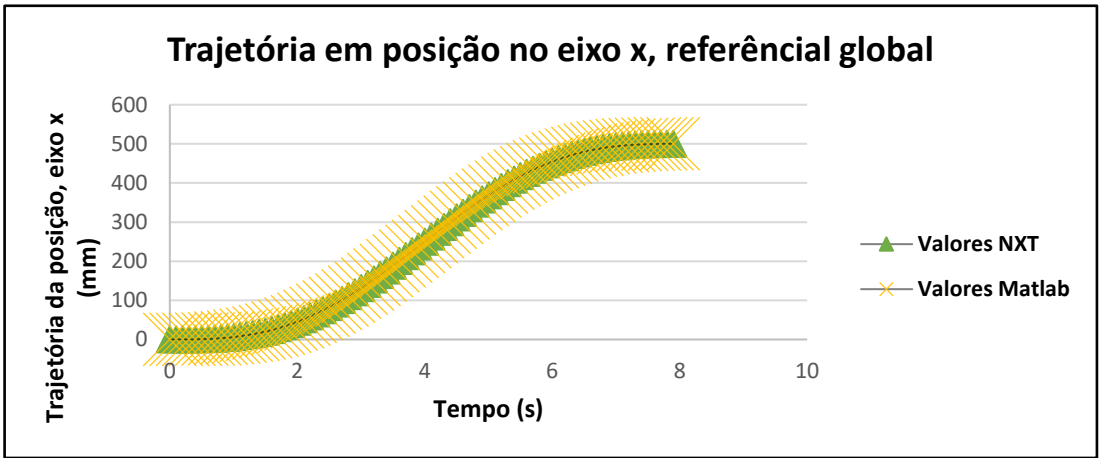

Figura 85 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

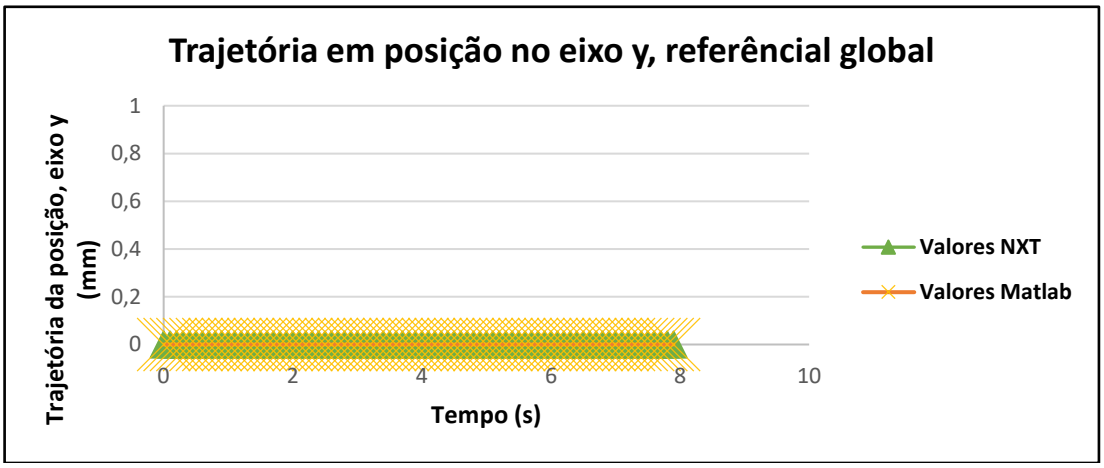

Figura 86 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

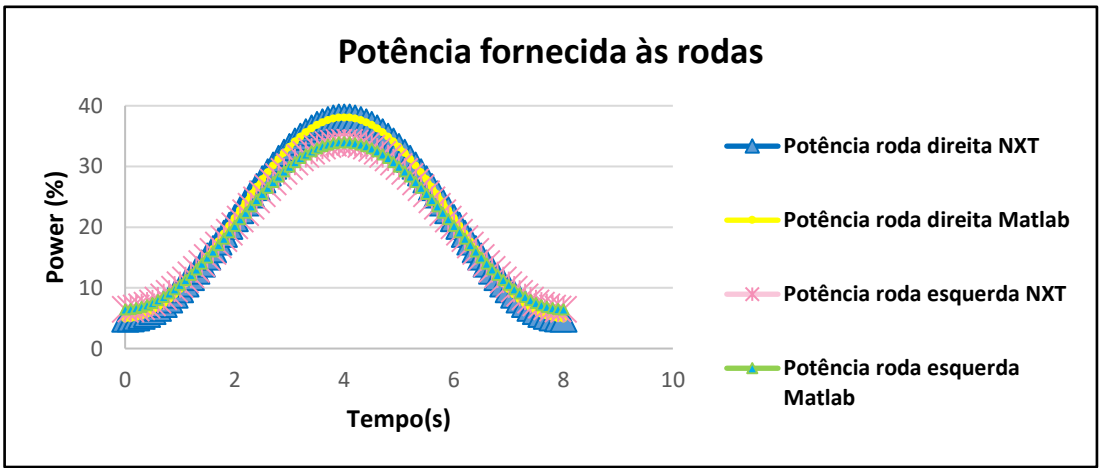

Figura 87 – Valores da potência calculada pelo Matlab e NXT para cada roda

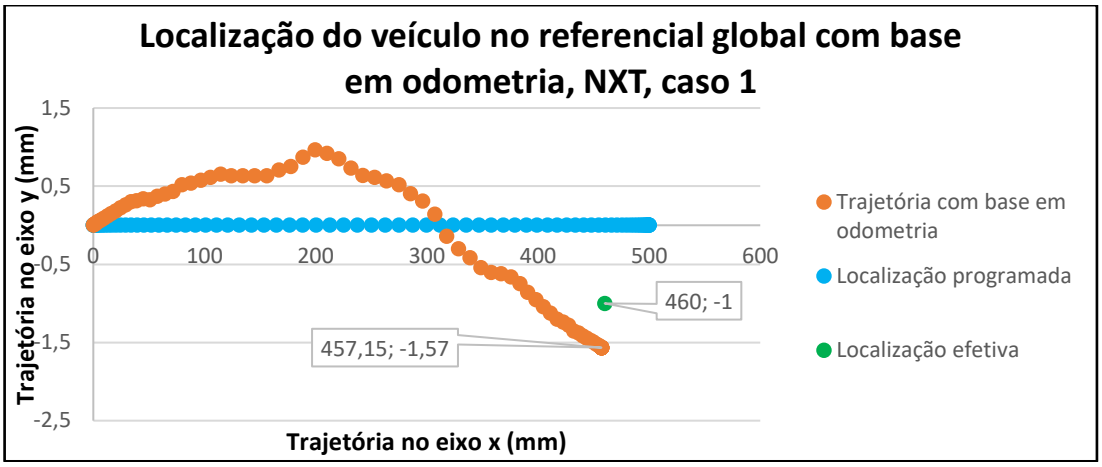

Figura 88 – Localização do veículo NXT com recurso a odometria, caso 1

Caso 1 – EV3

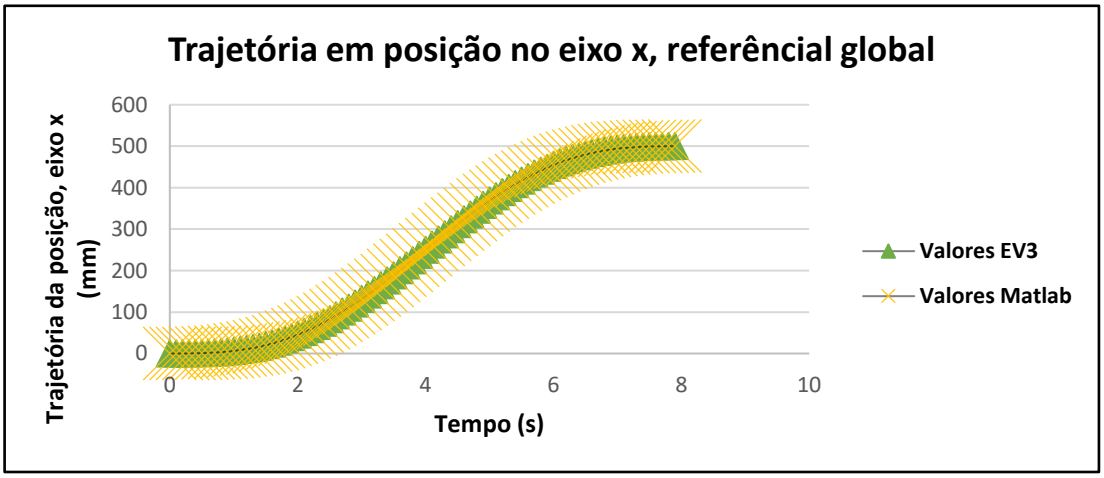

Figura 89 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

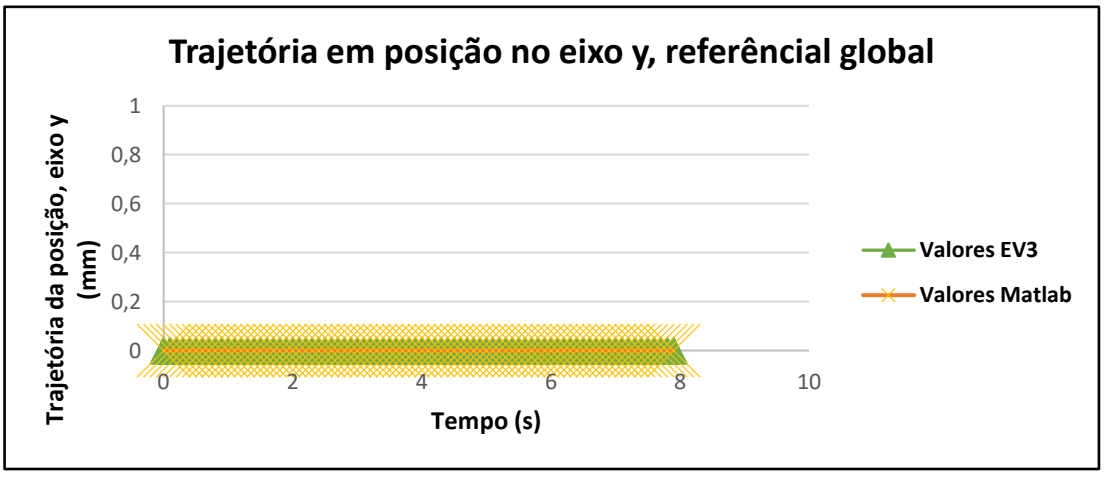

Figura 90 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

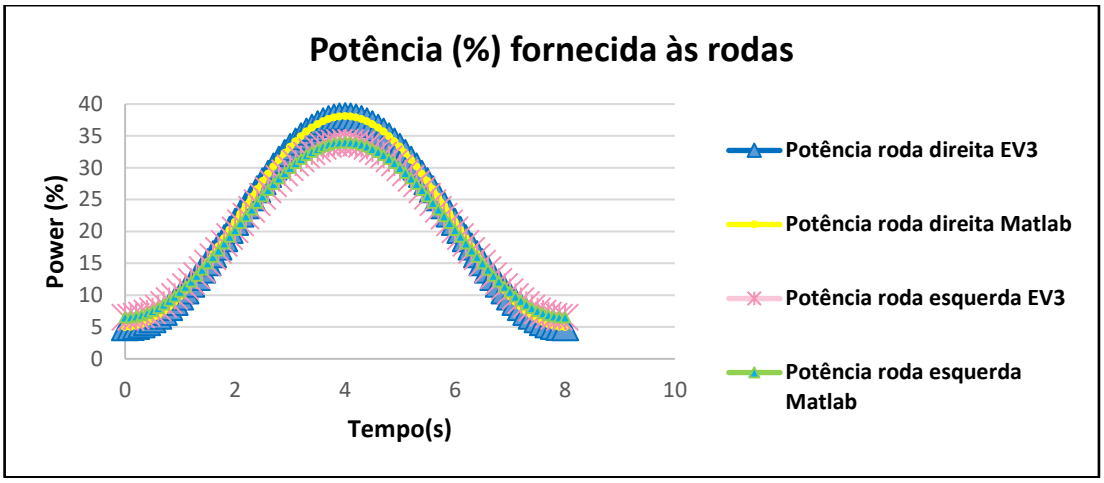

Figura 91 – Valores da potência calculada pelo Matlab e EV3 para cada roda

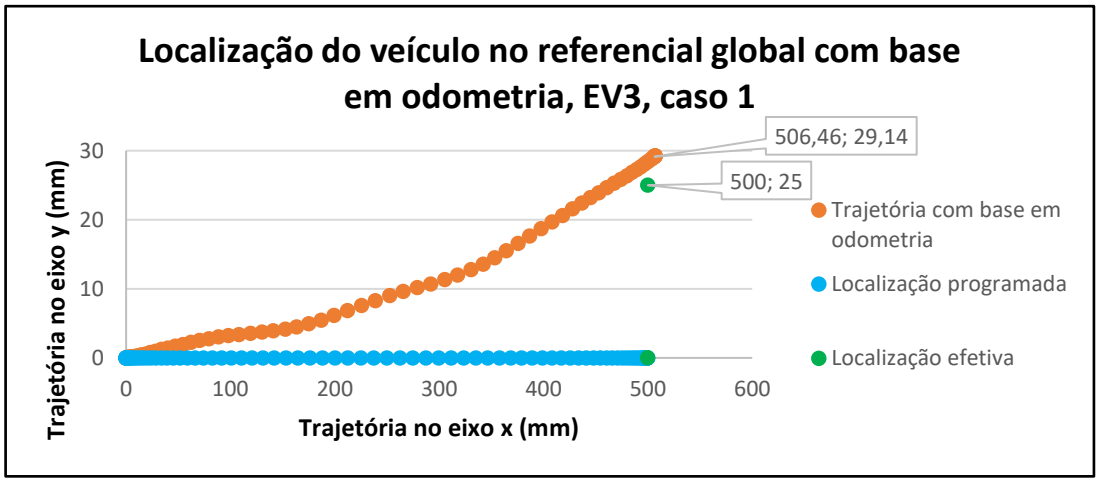

Figura 92 – Localização do veículo EV3 com recurso a odometria, caso 1

- **<u>Estratégia B3.2 – Perfil de Velocidade Trapezoidal</u>**

Os resultados da estratégia B3.2 são apresentados na Tabela 15.

A validade da implementação das trajetórias em linha reta com perfil trapezoidal pode ser observada nas figuras 94 a 96 e 98 a 100.

Tabela 15 – Valores medidos na implementação laboratorial da estratégia B3.2

	NXT				EV3			
	Caso 1		Caso 2		Caso 1		Caso 2	
	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2	Teste 1	Teste 2
Posição final (x_f) [mm]	450	449	770	774	500	505	790	798
Erro (x_f) [mm]	-50	-51	-30	-26	0	+5	-10	-2
Posição final (y_f) [mm]	-6	-8	-25	-22	-6	-9	-15	-12
Erro (y_f) [mm]	-6	-8	-25	22	-6	-9	-15	-12
Orientação final (θ_f) [graus]	-5°	-6°	-16°	-14°	-5°	-8°	-10°	8°
Erro (θ_f) [graus]	-5°	-6°	-16°	-14°	-5°	-8°	-10°	+8°

Caso 1 – NXT

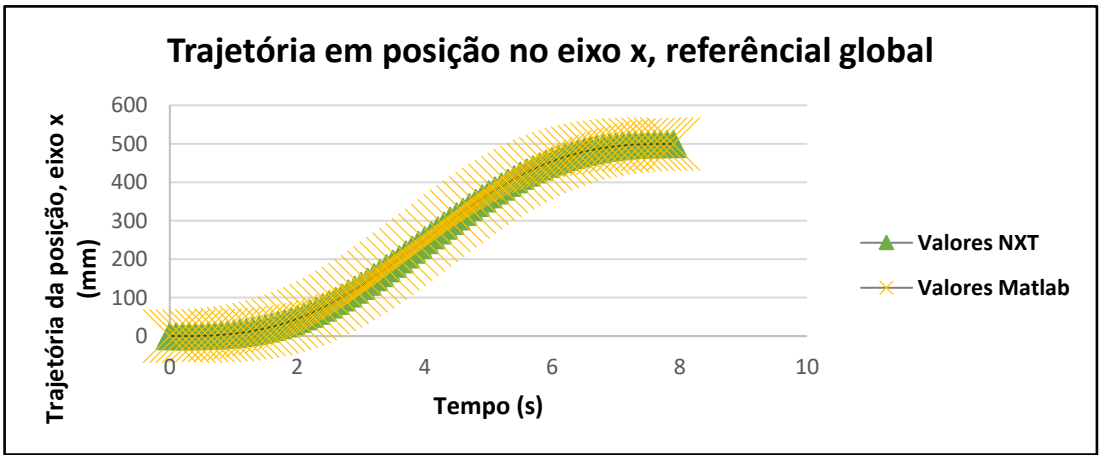

Figura 93 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

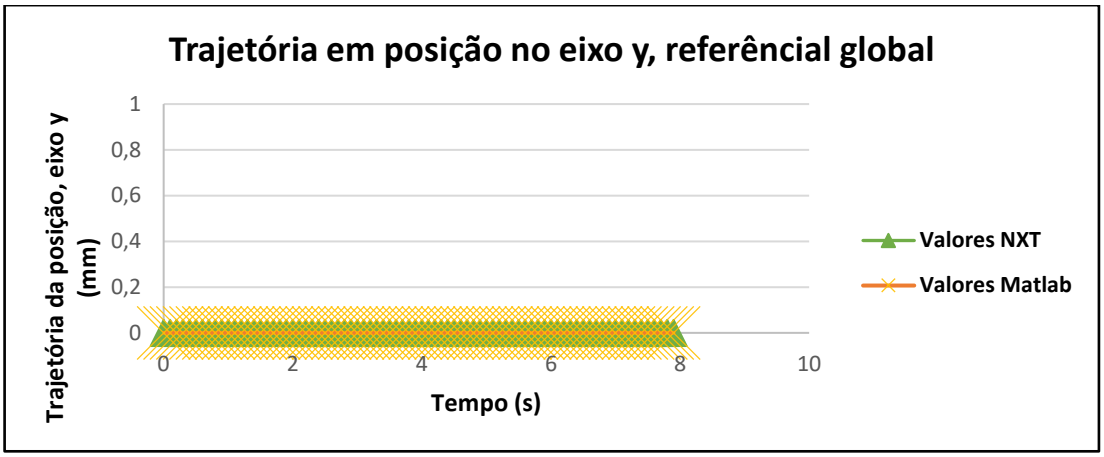

Figura 94 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

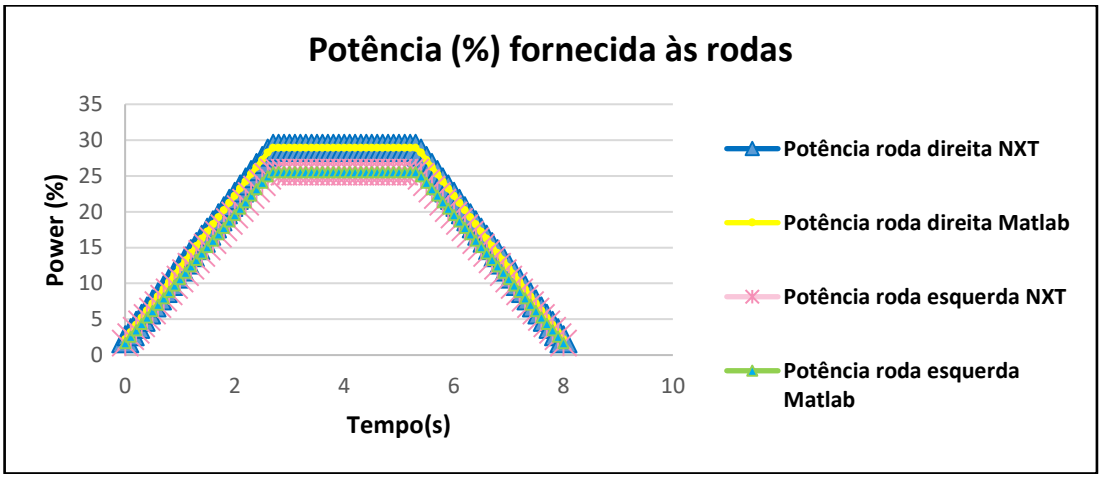

Figura 95 – Valores da potência calculada pelo Matlab e NXT para cada roda

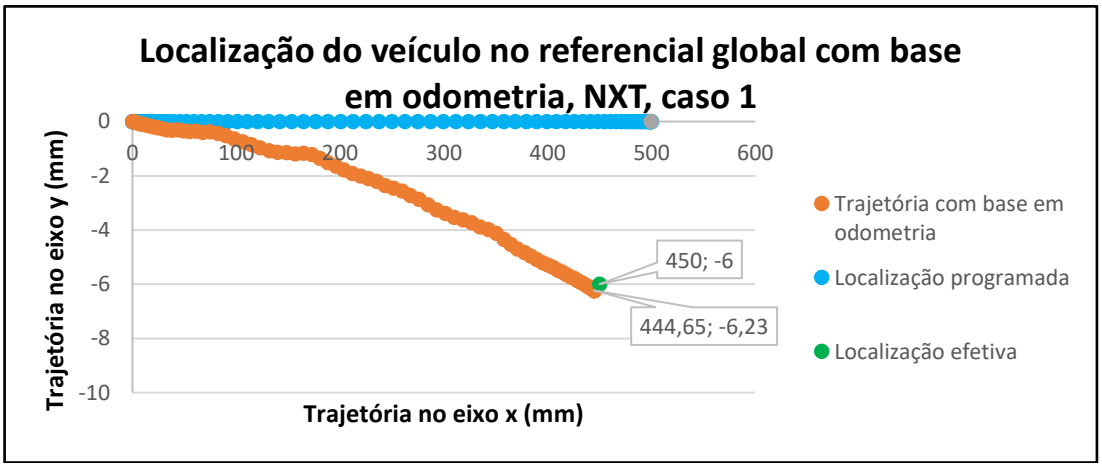

Figura 96 – Localização do veículo NXT com recurso a odometria, caso 1

Caso 1 – EV3

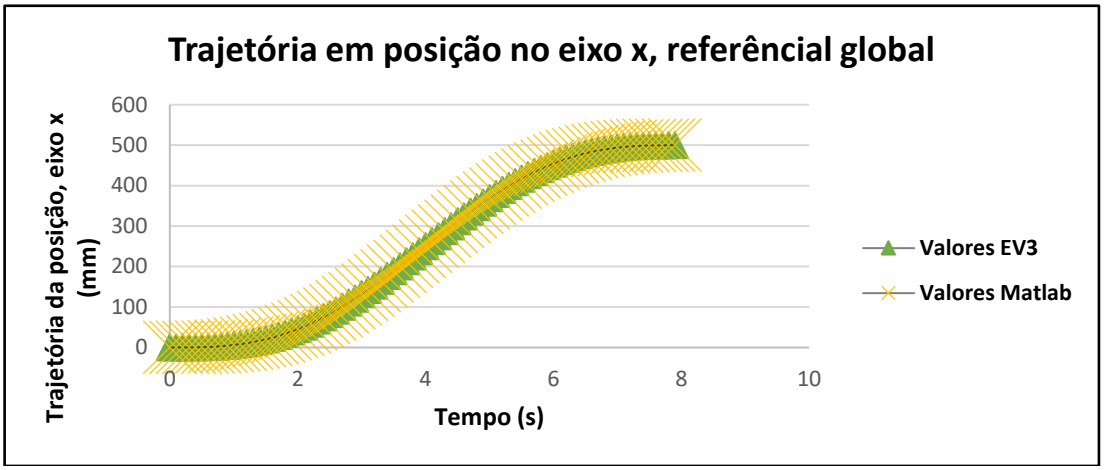

Figura 97 – Evolução da trajetória do veículo em posição, no eixo x do referencial global

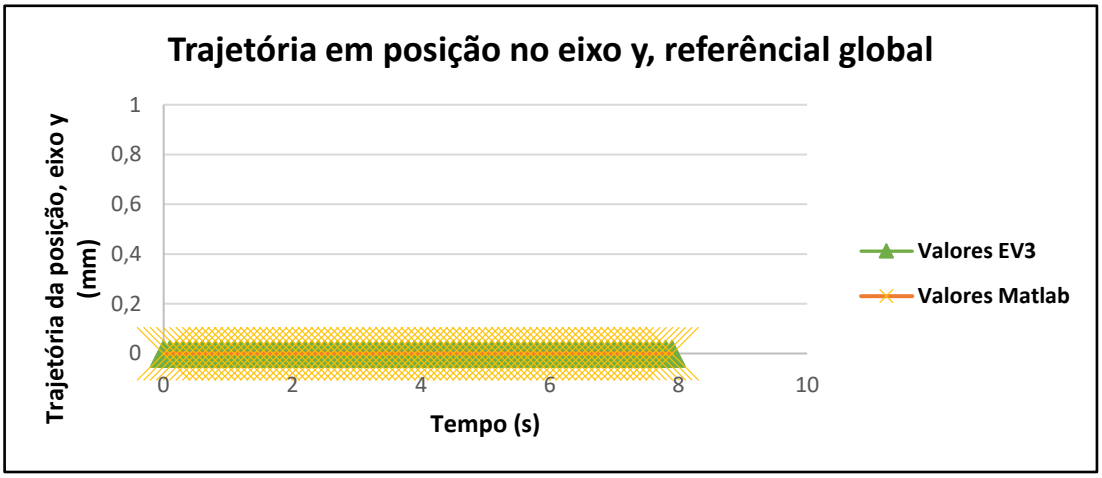

Figura 98 – Evolução da trajetória do veículo em posição, no eixo y do referencial global

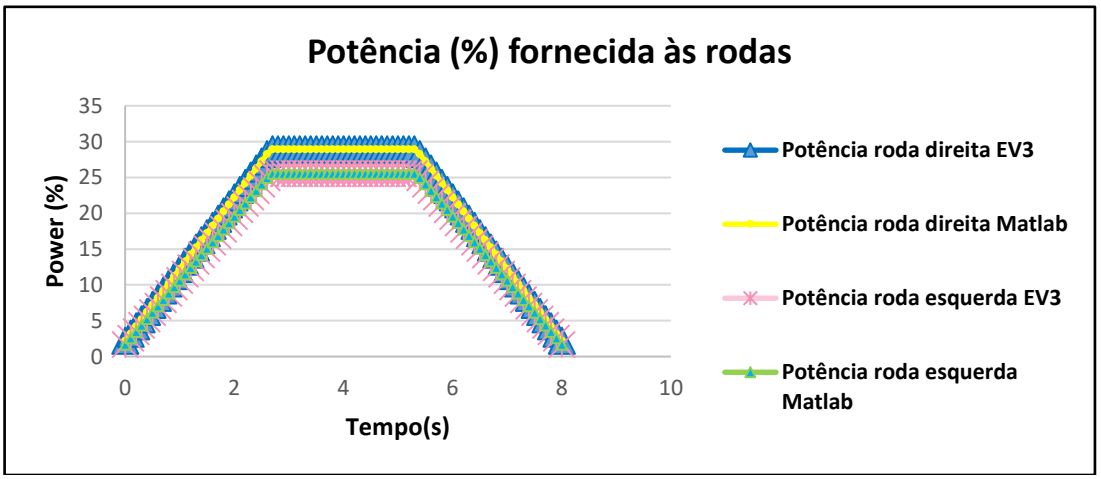

Figura 99 – Valores da potência calculada pelo Matlab e NXT para cada roda

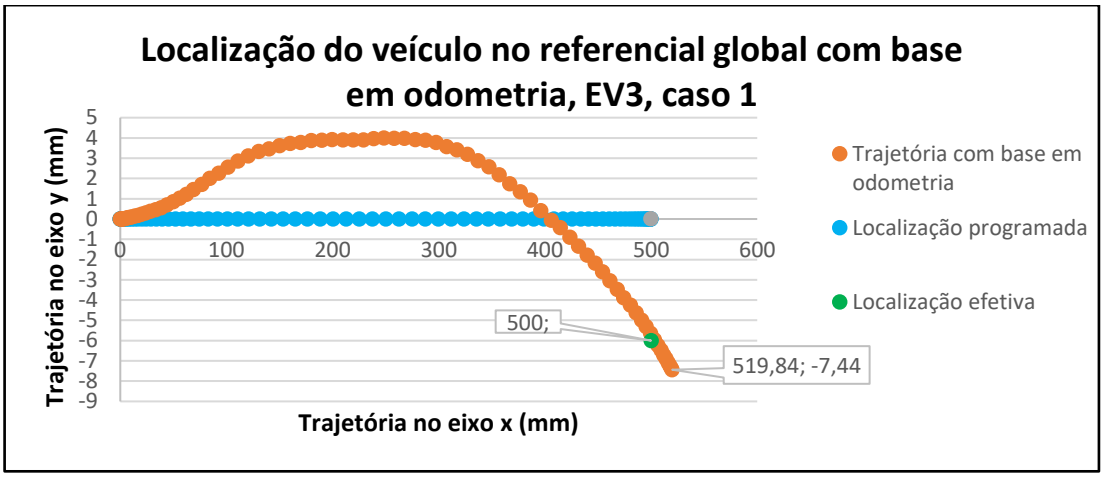

Figura 100 – Localização do veículo EV3 com recurso a odometria, caso 1

Fazendo uma análise geral aos resultados obtidos na estratégia B3, mais uma vez se observa que o perfil trapezoidal apresenta algumas melhorias em relação ao sinusoidal no veículo EV3. Enquanto que na estratégia sinusoidal existe um desvio considerável da reta (sensivelmente de 0mm a 29mm), essa diferença vê-se atenuada no perfil trapezoidal, variando entre 4mm a -7,5mm no eixo y. Quanto ao NXT, os resultados voltam a piorar com a estratégia trapezoidal.

6 Conclusões e Perspetivas de Trabalho Futuro

O desenvolvimento de um sistema de movimentação autónoma é uma tarefa exigente e complexa, envolvendo várias etapas intermédias, necessárias ao bom funcionamento do sistema.

Este trabalho teve como foco a implementação de trajetórias em veículos de baixo custo (Lego Mindstorms NXT e EV3), com recurso ao *software* Matlab/Simulink.

Foram implementados vários tipos de trajetórias com duas abordagens diferentes, controlo em posição, no qual o veículo se desloca com base em movimentos de rotação e translação independentes, e controlo em velocidade, originando um movimento de rotação e translação simultâneo.

Um dos objetivos do trabalho, foi a utilização de um *software* diferente do proposto pela Lego. O Simulink revelou-se bastante intuitivo na ótica do utilizador, sendo necessária a instalação de uma toolbox para o controlo dos robôs da Lego. A utilização de s-functions escritas em código C permitiu simplificar os programas realizados, evitando que a linguagem utilizada fosse exclusivamente gráfica. A linguagem C revelou-se bastante expedita e versátil, sendo possível adicionar várias bibliotecas dentro da s-function, caso tenham de ser utilizadas no programa. Neste caso, apenas foi utilizada a biblioteca "math.h", na qual se incluem várias funções matemáticas necessárias para o cálculo das trajetórias.

A validação dos programas executados foi feita com recurso ao Matlab, sendo possível comparar os valores obtidos pelos controladores dos veículos.

A localização do veículo com recurso exclusivo a odometria, apesar de ser extremamente útil, revelou que não é totalmente eficaz na estimativa da posição. Sensores como por exemplo bússolas digitais deverão ser utilizados em conjunto com a odometria para a obtenção de resultados mais corretos.

A medição laboratorial foi dificultada pela ausência de meios expeditos para a verificação de erros de localização. Foi possível observar que as trajetórias com perfil de velocidade trapezoidal minimizam os problemas de comando dos veículos a baixa velocidade, melhorando desta forma, o desempenho global.

Visto que os sistemas utilizados são de baixo custo, existem alguns problemas que afetam o comportamento dos veículos ao longo das trajetórias, como por exemplo, baixa rigidez estrutural, incertezas nos parâmetros dimensionais assim como algumas folgas. Além destes problemas, a baixa resolução dos encoders (1°) e o comportamento dos motores a baixa velocidade introduzem algumas dificuldades no cálculo das trajetórias comparativamente ao Matlab.

Um dos principais fatores de erro presentes é a diferença entre os motores esquerdo e direito de ambos os veículos. Apesar de se terem determinado as caraterísticas de todos os motores em diversos cenários, existem sempre erros que não se conseguem eliminar. Desta forma, como um possível trabalho futuro, a utilização de um controlador PID para garantir que os robôs se consigam deslocar numa linha reta perfeita, seria interessante.

Outra sugestão para ser implementada no futuro é a utilização de um algoritmo "*Pure-Pursuit*" [18], por forma a melhorar e minimizar as diferenças das trajetórias obtidas. O algoritmo calcula a curvatura necessária para mover o veículo da posição atual para uma posição de destino dentro da trajetória. Desta forma, sempre que o veículo se desvia da trajetória programada, o algoritmo calcula o arco necessário para levar o veículo de novo para a trajetória correta.

Referências

[1] – Wallén, J. 2008, "The history of the industrial robot", Sweden, último acesso: abril 2016, http://liu.diva-portal.org/smash/get/diva2:316930/FULLTEXT01.pdf.

[2] – Lima, P., Ribeiro, M.I. 2002, "Mobile Robotics", Instituto Superior Técnico, Lisboa, último acesso: maio 2016, http://users.isr.ist.utl.pt/~mir/cadeiras/robmovel/Introduction.pdf.

[3] – Manetas, M. 2011, "In My Computer", United States of America, último acesso: abril 2016.

[4]–Moravec,H.P. 2016, "Encyclopaedia Britannica", último acesso: maio 2016, http://www.britannica.com/technology/robot-technology

[5] – Jafri, Ali R. 2009, "Robotics: From Industrial Robots to Humanoids", Paquistão, último acesso: março 2016.

[6] – Rebecca, J. R. 2011, "Unimate: The Story of George Devol and the First Robotic Arm", The Atlantic, último acesso: março 2016.

[7]– Siegwart, R., Nourbakhsh, I. R. 2004, "Introduction to autonomous mobile robots", A Bradford Book, MIT press, Cambridge, US, último acesso: Junho 2016.

[8]–http://www.terex.com/port-solutions/en/products/automated-guided-vehicles/agv/index.htm, último acesso: abril 2016.

[9] – Curley, R. 2010, "The Britannica Guide to Inventions that Changed the Modern World", Britannica Educational Publishing, último acesso: abril 2016.

[10] – Nilsson, Nills J. 1984, "Shakey the Robot", SRI International, último acesso: abril 2016, http://www.ai.sri.com/pubs/files/629.pdf.

https://en.wikipedia.org/wiki/Lunar_rover, último acesso: março 2016.

[11] – https://en.wikipedia.org/wiki/Lunar_rover, último acesso: março 2016.

[12] – http://www.gebocermex.com, último acesso: abril 2016.

[13] – http://www.egemin-automation.com/en/

[15] – http://www.ec21.com/product-details/Automated-Guided-Vehicle--8969284.html

[16] – www.leonardo-da-vinci-models.com

[17] – Borenstein, J., Everett, H.R., Feng, L. 1996, "Where am I? Sensors and Methods for Mobile Robot Positioning", University of Michigan, United States of America, último acesso: Junho 2016.

[18] – Samuel, M., Hussein, M., Mohamad, Maziah, B., "A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle", International Journal of Computer Applications, Volume 135, último acesso: Julho 2016.

[19] – Silva, P. M., Movimentação autónoma de Robôs Móveis de Baixo Custo, com Base no Sistema NXT da LEGO®, Dissertação de Mestrado Integrado em Engenharia Mecânica, Faculdade de Engenharia da Universidade do Porto, 2010.