UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DO CURSO DE ENGENHARIA ELETRÔNICA
ENGENHARIA ELETRÔNICA

MARCOS PAULO PETTERMANN BRACHT

PROJETO DE UM VEÍCULO AUTÔNOMO CAPAZ DE COBRIR UMA ÁREA POLIGONAL SEM PASSAR MAIS DE UMA VEZ PELA MESMA REGIÃO

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2015

MARCOS PAULO PETTERMANN BRACHT

PROJETO DE UM VEÍCULO AUTÔNOMO CAPAZ DE COBRIR UMA ÁREA POLIGONAL SEM PASSAR MAIS DE UMA VEZ PELA MESMA REGIÃO

Trabalho de Conclusão de Curso de Graduação apresentado na disciplina de Trabalho de Conclusão de Curso 2 (TCC2), como requisito parcial para obtenção do título de Engenheiro Eletrônico, da Coordenação do Curso de Engenharia Eletrônica, da Universidade Tecnológica Federal do Paraná – UTFPR, Campus Toledo.

Orientador: Prof. Me. Jorge Augusto Vasconcelos Alves.

TOLEDO
2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Toledo
Coordenação do Curso de Engenharia Eletrônica

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nº 018

Projeto de um Veículo Autônomo Capaz de Cobrir uma Área Poligonal sem Passar mais de uma Vez pela Mesma Região

por

Marcos Paulo Pettermann Bracht

Esse Trabalho de Conclusão de Curso foi apresentado às 15:50 h do dia **23 de junho de 2015** como requisito parcial para a obtenção do título **Bacharel em Engenharia Eletrônica**. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado **APROVADO**.

_____
Prof. Dr. Fábio Rizental Coutinho
(UTFPR-TD)

_____
Prof. M. José Dolores Vergara Dietrich
(UTFPR-TD)

_____
Prof. M. Jorge Augusto Vasconcelos Alves
(UTFPR-TD)
Orientador

Visto da Coordenação

_____
Prof. M. Alessandro Paulo de Oliveira
Coordenador da COELE

RESUMO

BRACHT, Marcos P. Pettermann. **Projeto de um veículo autônomo capaz de cobrir uma área poligonal sem passar mais de uma vez pela mesma região**. 2015. 71 folhas. Trabalho de Conclusão de Curso 2 (Bacharelado em Engenharia Eletrônica) – Universidade Tecnológica Federal do Paraná. Toledo, 2015.

Este trabalho apresenta um projeto de algoritmos de geração e seguimento de trajetórias aplicado a robôs móveis de acionamento diferencial de modo a percorrer integralmente áreas predefinidas. O movimento é feito com velocidade constante e evitando-se que o robô passe mais de uma vez sobre a mesma região. Controladores proporcional e proporcional-derivativo são usados para ajuste da velocidade angular das rodas e do seguimento de trajetória do robô, respectivamente. Os códigos são testados por meio de um simulador.

Palavras-chave: robôs móveis. Planejamento de movimento. Seguimento de trajetória.

ABSTRACT

BRACHT, Marcos P. Pettermann. **Project of an autonomous vehicle capable of covering a polygonal area without going over the same region more than once**. 2015. 71 folhas. Trabalho de Conclusão de Curso 2 (Bacharelado em Engenharia Eletrônica) – Universidade Tecnológica Federal do Paraná. Toledo, 2015.

This work describes the design of trajectory generation and tracking algorithms applied to differential wheeled mobile robots which must fully cover predefined areas. The movement is done with constant speed and in a way of preventing the robot from going over the same region more than once. Proportional and proportional-derivative controllers are used for wheel speed control and trajectory tracking, respectively. The codes are tested through a simulator.

Keywords: Mobile robots. Motion planning. Trajectory tracking.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AGV	*Automated Guided Vehicle*
DARPA	*Defense Advanced Research Projects Agency*
DC	*Direct Current*
GPS	*Global Positioning System*
IMU	*Inertial Measurement Unit*
NMEA	*National Marine Electronics Association*
PD	*Proportional-derivative*
PWM	*Pulse-Width Modulation*
SLAM	*Simultaneous Localization and Mapping*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

O investimento na criação de robôs autônomos móveis vem do antigo desejo e, por vezes, da necessidade, de reprodução de atitudes humanas por meio de máquinas. Equipamentos dessa natureza substituem trabalhadores em ambientes insalubres ou perigosos, poupando recursos humanos que podem ser aproveitados em áreas mais complexas. Às vantagens do uso desses robôs podemos adicionar o fato de eles serem capazes de realizar tarefas de maneira mais precisa, além de trabalharem quase que ininterruptamente (desde que seja provida alimentação elétrica durante todo o período).

O desenvolvimento de máquinas móveis é vantajoso em relação ao daquelas com base fixa, devido ao fato das primeiras poderem realizar ações em locais diversos. Entretanto, essa melhora vem junto com uma série de novos obstáculos. Entre eles, podemos citar: o reconhecimento e mapeamento de ambientes menos estruturados, desvio de obstáculos, controle sujeito a restrições não-holonômicas ou não integráveis, entre outros. Boa parte desses desafios segue ainda não resolvida[1].

1.1 OBJETIVOS

O objetivo geral deste trabalho consiste no desenvolvimento de um algoritmo gerador de trajetórias e controlador de seguimento de trajetória destinado a robôs móveis que devam percorrer a área interna à uma figura predefinida. Esse movimento deve ser realizado com velocidade constante e evitando-se ao máximo que o veículo passe sobre uma mesma região mais de uma vez.

Adicionalmente à produção dos códigos elencados, deseja-se desenvolver um simulador virtual que seja capaz de expor os resultados fornecidos pelo sistema para uma configuração hipotética predefinida de robô móvel. De modo a facilitar a criação do algoritmo de controle do veículo, busca-se ainda obter um modelo matemático que o represente, ou seja, um conjunto de equações que forneça a sua posição e velocidade baseado apenas em parâmetros do robô e informações dos sensores.

1.2 JUSTIFICATIVA

A ideia inicial deste trabalho era promover uma facilitação nas atividades de espalhamento de produtos agrícolas em lavouras. Desejava-se desenvolver um sistema de pilotagem automática de tratores e outros veículos envolvidos nesse contexto, o qual evitasse ao máximo a passagem da máquina mais de uma vez sobre a mesma área. Isso evitaria o desperdício das substâncias, além de dispensar, quase que totalmente, a necessidade da presença de um operador.

Entretanto, a execução de tal projeto exigiria que se levasse em conta a presença de peças responsáveis pelo espalhamento em si. Além disso, seria preciso adequar os trajetos gerados às limitações de movimento de veículos articulados, algo que aumentaria expressivamente a complexidade do trabalho.

Diante dessas limitações, aliadas ao curto prazo estabelecido para a conclusão do trabalho, optou-se pelo projeto de códigos adequados a um veículo arbitrário, os quais possam ser replicados em máquinas de aplicações diversas. Entre elas, podemos citar: cortadores de grama, robôs de limpeza e os próprios tratores agrícolas citados anteriormente. A implementação de funcionalidades específicas de cada um desses exemplos foge do escopo desta pesquisa. Este trabalho serve também para provar que a automação da pilotagem de veículos que realizam movimentos parecidos com os especificados nesse caso é possível e de modo a produzir uma base para a produção de protótipos completos de máquinas reais desse tipo.

2 REFERENCIAL TEÓRICO

O avançado número de tarefas automatizadas do dia-a-dia atual é resultado de décadas de pesquisas no campo. Para ambientar o leitor em torno de avanços significativos alcançados pela pesquisa e desenvolvimento de robôs autônomos, descreve-se um breve contexto histórico sobre o assunto, além de comentarem-se os principais produtos e trabalhos feitos nessa área. Nas subseções seguintes, é detalhado o tipo de robô para o qual os códigos deste projeto são desenvolvidos. No final desta seção, mostra-se o modelo matemático obtido para o veículo tomado como referência e o método para sua obtenção.

O resumo das evoluções e mercadorias à venda está presente na Seção 2.1; enquanto os estudos relevantes encontrados são apresentados na Seção 2.2. As Seções 2.3 e 2.4 demonstram a configuração de robô e alguns valores escolhidos para este trabalho, respectivamente. Por fim, a modelagem matemática deste veículo está presente na Seção 2.5.

2.1 ROBÔS AUTÔNOMOS COMERCIAIS

As pesquisas no campo da construção de máquinas autônomas tiveram início ainda nos anos 1970. Os primeiros resultados de aplicação prática em dificuldades reais foram o robô de hospital “HelpMate”, da empresa “Pyxus”, e o de segurança da “CyberMotion”, ilustrados nas Figuras 1(a) e 1(b), respectivamente. Esses robôs utilizavam mapas pré-carregados em seu sistema e, através de sensores ultrassônicos, seguiam as paredes para movimentarem-se em construções.

O primeiro deles era um veículo que transportava equipamentos médicos, tais como remédios, equipamentos, refeições, informações de pacientes e resultados de exames de radiologia, e que era capaz de mover-se livremente em hospitais e edifícios anexos, sem o uso de sistemas externos como guias. O robô ainda está disponível no mercado nos dias de hoje, sendo que o modelo atual, alimentado por baterias, pode conectar-se a outros equipamentos sem a necessidade de fios, possui uma tela

sensível ao toque, a qual permite ao usuário escolher a qual lugar do hospital deseja mandá-lo, e é capaz de interagir com elevadores. O processo de se evitar o choque com obstáculos, inicialmente, era feito através de câmeras. Recentemente, as mesmas foram substituídas por *scanners* a laser[2].

Já o Robô de Segurança “SR2” da “CyberMotion” foi lançado com o objetivo de monitorar a presença de invasores em museus dos Estados Unidos. Era composto de 3 rodas, uma estrutura retangular e sensores sonar e infravermelho, os quais possibilitavam à máquina conhecer a qualidade do ar, temperatura e umidade do local, além de detectar movimentos causados por pessoas na sua área de cobertura. A empresa encerrou suas atividades em 2001 e o robô não é vendido atualmente[3].

O desenvolvimento e barateamento de novas tecnologias permitiu que as máquinas autônomas alcançassem o nível experimentado atualmente. O robô “Seekur”, da empresa “Adept MobileRobots”, é um exemplo de veículo capaz de inspecionar, mapear e monitorar áreas internas ou externas através de câmeras, GPS, sensores a laser e comunicação sem fio[4]. A Figura 1(c) mostra uma das versões comercializada do robô.

Entre os veículos autônomos móveis utilizados atualmente podemos ainda citar os “veículos guiados automaticamente” (AGVs). Tratam-se de máquinas destinadas a ambientes industriais, que são capazes de transportar cargas em trajetos predefinidos. Seu movimento é calculado através de uma combinação de *software* e sistemas de localização baseados em sensores, com aceleração e desaceleração controladas[5]. Um exemplo de robô desse tipo é mostrado na Figura 1(d).

(a) Robô de hospital “HelpMate”.
Fonte: Jones (2003).

(c) Robô de aplicações diversas “Seekur”.
Fonte: Adept Technology (2011).

(b) Robô de segurança “SR2”.
Fonte: Wallis (2013).

(d) Exemplo de AGV.
Fonte: The Material Handling Institute (2014).

Figura 1 - Robôs autônomos comerciais.

2.2 PUBLICAÇÕES RELACIONADAS

O desenvolvimento de robôs verdadeiramente autônomos envolve a realização de tarefas em locais de onde se tem nenhuma ou muito poucas informações. No trabalho de Montemerlo e Thrun[6], por exemplo, é tratado do caso em que não se tem um conhecimento prévio do ambiente e tampouco se possui um referencial de localização da máquina. O conjunto desses problemas é designado como “localização e mapeamento simultâneos” (SLAM). Nesses casos, ocorre um dilema tal como o do “ovo e da galinha”, uma vez que, se o próprio robô não sabe onde está, como poderá identificar o mundo à sua volta e dizer onde estão esses elementos? O autor utiliza do Filtro de Kalman Estendido e do Filtro de Partículas para reduzir ao máximo os pontos nos quais o protótipo pode estar localizado.

Uma abordagem alternativa para a solução do problema é proposta por Sujan[7]. O trabalho em questão descreve um algoritmo iterativo, o qual fornece um modelo do ambiente no qual o movimento foi realizado.

Outra pesquisa no campo foi realizada por Miranda e Ribeiro[8]. Nela, é utilizada o chamado “método *bayesiano*” para encontrar a localização instantânea de robôs móveis equipados apenas com *encoders* nas rodas e sensores sonares.

O problema da localização de robôs é tratado ainda nas publicações de Rekleitis[9] e de Thrun[10]. No primeiro, é exposto um tutorial do uso dos filtros de partículas nesse âmbito. Através desse método, são analisadas as estimativas de posição criadas a partir da leitura dos sensores, sendo eliminadas aquelas consideradas pouco prováveis. Já o segundo autor citado descreve e compara as principais técnicas probabilísticas de rastreio de robôs utilizadas na época. Entre elas, estão o filtro de Kalman, o algoritmo de Lu/Milios, mapas multi-planares e abordagens híbridas.

A pesquisa de Erickson[11], bem como aquela realizada por Nourbakhsh[12], refere-se a veículos projetados para se moverem em ambientes internos de construções. Os objetos de estudo são: um robô de limpeza “cego” (equipado apenas de um relógio e sensores de contato) e um de escritório, respectivamente. O primeiro move-se de maneira aleatória em um cômodo qualquer, sendo que, após um intervalo de tempo determinado se movimentando, a possibilidade de ter passado por todo o ambiente é bastante grande. Já a segunda máquina é a vencedora de um concurso feito em 1994 em uma conferência de inteligência artificial. O robô vencedor deveria ser capaz de percorrer trajetórias dentro de uma série de escritórios conectados, sem se perder de maneira irreversível em nenhum instante. Esse objetivo foi alcançado por “Darvish”, uma estrutura móvel composta de diversos sensores sonares.

Exemplos de sistemas de navegação como os apresentados até aqui aplicados em veículos de grande porte podem ser encontrados nas vans *Alice* e *Stanley*, ambos participantes do Grande Prêmio DARPA de 2005. O desafio era criar robôs autônomos capazes de percorrer um longo trajeto em meio a terrenos irregulares. Os dois exemplos citados, equipados com sensores de distância a laser, câmeras, GPS e sistemas de navegação inercial (IMU), concluíram o percurso em tempo aceitável[13] [14].

No Trabalho de Conclusão de Curso de Devenz[15], foi desenvolvido o protótipo de um cortador de grama, uma das máquinas possíveis de serem desenvolvidas a partir deste projeto, conforme citado na Seção 1.2. Nesse caso, porém, o robô se

move em linha reta até que encontre um obstáculo, momento no qual é feita uma curva em uma direção a ser escolhida pelo microcontrolador. Assim, não é feito uso dos sistemas de posicionamento tais como módulo GPS e odometria (termo que designa o processo de estimação de posição e orientação de veículos com base exclusivamente no histórico de ângulo de rotação das rodas) - mas sim de sensores ultrassônicos cujas leituras limitam o movimento do veículo. Não é possível limitar o movimento com base apenas em coordenadas geográficas. Um dos trabalhos utilizados como referência por Deventz, o de um robô aspirador de pó construído por Bissoli e Esteves (2004), realiza movimentos aleatórios e também depende das leituras de sensores ultrassônicos para limitar seu deslocamento.

Já o trabalho de Sandi[16] demonstra o desenvolvimento e a implementação de um sistema de navegação e guiagem de robôs autônomos baseado nas medições de *encoders* das rodas e de uma bússola digital. Relatou-se que uma precisão maior nos movimentos seria possível através do uso de aparelhos capazes de obter, em tempo real, as coordenadas geográficas correspondentes à posição do veículo, entre os quais pode-se citar o módulo GPS.

2.3 ROBÔ MÓVEL CONSIDERADO

O tipo de robô móvel para o qual os códigos mencionados na Seção 1.1 são projetados foi escolhido após uma análise dos equipamentos disponíveis no mercado. Levando em conta o objetivo proposto, uma série de configurações seriam possíveis[17]. Optou-se pela configuração mais acessível que fosse capaz de demonstrar a funcionalidade do sistema.

O chassi escolhido foi um tal qual mostrado na Figura 2. Esse esquema possui duas rodas ativas (acopladas a motores elétricos) e não-orientáveis, além de uma orientável centrada cuja velocidade não pode ser controlada diretamente (popularmente conhecida como “roda boba”), cuja única função é dar sustentabilidade ao veículo. Robôs montados sobre uma carcaça desse tipo possuem acionamento diferencial, característica essa que constitui uma opção barata devido ao menor número de peças móveis presentes e ao fato de rodas universais ou suecas não serem necessárias. As medidas de raio das rodas e de distância entre as mesmas, as

quais serão necessárias para desenvolvimento da modelagem matemática do veículo na Seção 2.5, foram fixadas em 3,2 cm e 13 cm, respectivamente.

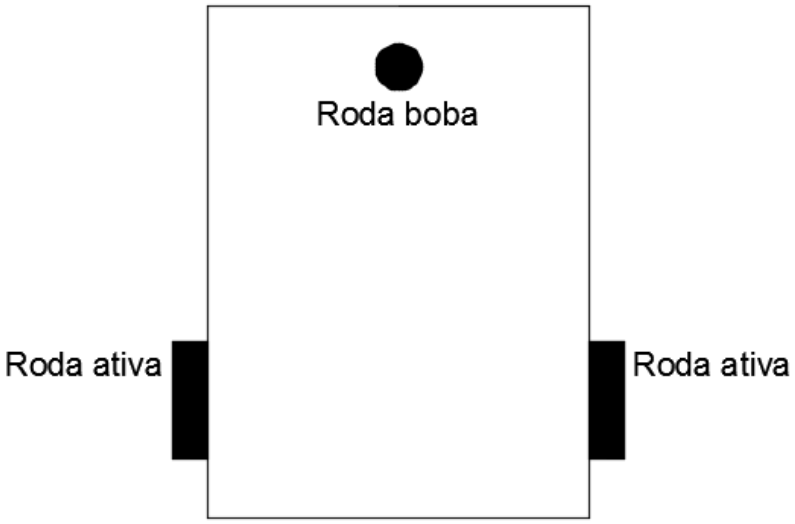

Figura 2 – Esquema do chassi de veículo considerado.

Um controle eficaz de velocidade de rotação das rodas e de posição do robô exige um tempo de amostragem baixo de análise da leitura dos sensores. Portanto, os códigos explicados neste trabalho levam em conta que os dois motores do veículo sejam munidos de caixas de redução. O modelo da Figura 3, por exemplo, oferece uma taxa de transformação de rotação de 1:143[18].

Figura 3 - Exemplo de motor elétrico com caixa de redução.
Fonte: RobotShop (2015).

O Apêndice D contém a descrição da construção de um robô móvel similar ao descrito nesta seção.

2.4 PARÂMETROS DO MOVIMENTO CONSIDERADOS

Levando em conta o veículo especificado na Seção 2.3, alguns parâmetros foram fixados e considerados durante o desenvolvimento dos algoritmos do projeto. Esses valores estão elencados na Tabela 1.

Tabela 1 - Valor dos parâmetros utilizados neste projeto.

Constante	Valor	Unidade
Velocidade linear do robô	0,278	m/s
Período de amostragem da velocidade das rodas	0,005	s
Período da função de controle da velocidade das rodas	0,005	s
Período da função de seguimento de trajetórias	0,020	s

A velocidade a ser mantida constante foi escolhida arbitrariamente. É comum em projetos deste tipo que sejam utilizados períodos de amostragem da ordem de centésimos de segundo. O trabalho de Borenstein[19] e o de Amer[20] são exemplos de pesquisas nas quais são feitas considerações similares. A função de controle de trajetória servirá para que se calculem as velocidades angulares desejáveis para cada rodas. Feito isso, o sistema de controle das velocidades das rodas será chamado algumas vezes, de modo a garantir que os valores projetados foram atingidos. Só então novas frequências de giro serão escolhidas, de acordo com o novo erro de posição estimado. O uso de um modelo cinemático de veículo é comum nestes casos, e o processo de controle de velocidade das rodas é necessário para tornar o modelo válido.

Neste trabalho, sempre que se escrever o índice 1, entenda-se que está se referindo à roda esquerda do robô. Aquela do lado direita será designada pelo número

2. O esquema de uma vista superior do veículo mostrando a nomenclatura de cada roda está ilustrado na Figura 4.

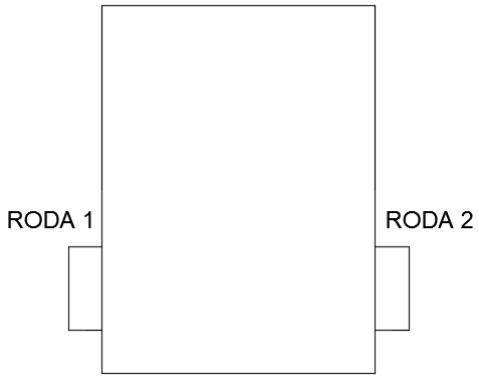

Figura 4 - Esquema de vista superior do veículo, demonstrando a nomenclatura utilizada para cada roda.

Foi preciso ainda especificar o funcionamento dos motores do robô considerado. Digamos que cada motor seja controlado por uma onda PWM, cujo *duty cicle* varie de 0 até 255. Seja ω_i a velocidade angular de uma roda e p_i o nível do sinal aplicado no motor correspondente, com o índice i valendo 1 ou 2. Neste trabalho, levaremos em conta motores cuja relação entre p_i e ω_i seja aquela mostrada na Equação (1).

$$\omega_i = 0{,}2p_i. \tag{1}$$

2.5 MODELAGEM MATEMÁTICA

A obtenção do modelo cinemático do veículo considerado neste projeto passa pelo uso de dois sistemas de coordenadas: um inercial em relação ao ambiente a ser percorrido, e outro que se move juntamente com o carro[17]. As equações que representam os movimentos serão encontradas, primeiro, com base nesse sistema local do robô, e, em seguida, referidas aos eixos imóveis através de uma matriz de transformação. A Figura 5 ilustra o posicionamento dos eixos citados.

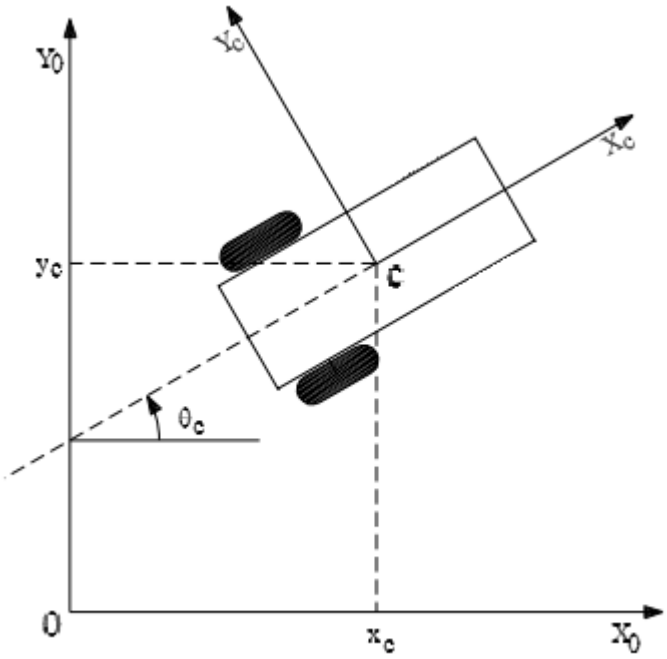

Figura 5 – Sistemas de coordenadas utilizados na modelagem do veículo.
Fonte: Simões (2014).

Assim, consideraremos um sistema inercial $\{X_0, Y_0, \Theta_0\}$, no qual a posição central do veículo possui as coordenadas (x_c, y_c, θ_c), e um sistema móvel $\{X_c, Y_c, \Theta_c\}$, cuja origem será expressa no sistema inercial pelas coordenadas x_c e y_c, e que possuirá um ângulo de rotação θ_c em relação a ele. A origem dos eixos $\{X_c, Y_c, \Theta_c\}$ estará localizada no ponto médio entre as duas rodas controláveis, de modo que o robô só possua velocidade instantânea no eixo X_c. Os cálculos serão realizados com base nos eixos X_c e Y_c, sendo preciso encontrar uma relação entre a expressão de uma posição dessa forma e nos eixos inerciais. Considere-se uma variação de pose qualquer $(x_{c1}, y_{c1}, \theta_{c1})$, definidia como um ponto (x_{c1}, y_{c1}) no sistema $\{X_c, Y_c, \Theta_c\}$ e um deslocamento angular θ_{c1} referente ao sistema de coordenadas local. Levando em conta o esquema da Figura 5, a representação da mesma variação no sistema $\{X_0, Y_0, \Theta_0\}$ pode ser definida como:

$$\begin{aligned} x_{01} &= x_{c1}\cos\theta_{c1} - y_{c1}\,\mathrm{sen}\,\theta_{c1}, \\ y_{01} &= x_{c1}\,\mathrm{sen}\,\theta_{c1} + y_{c1}\cos\theta_{c1}, \\ \theta_{01} &= \theta_{c1}. \end{aligned} \tag{2}$$

O mesmo resultado pode ser mostrado na forma matricial:

$$\begin{bmatrix} x_{01} \\ y_{01} \\ \theta_{01} \end{bmatrix} = \begin{bmatrix} \cos\theta_{c1} & -\operatorname{sen}\theta_{c1} & 0 \\ \operatorname{sen}\theta_{c1} & \cos\theta_{c1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{c1} \\ y_{c1} \\ \theta_{c1} \end{bmatrix}. \quad (3)$$

Uma vez encontrada a matriz de transformação de coordenadas entre os sistemas, podemos deduzir as fórmulas da velocidade do veículo. Nessa etapa, o objetivo é expressar a velocidade total do conjunto em função apenas das seguintes informações de suas rodas: r (raio), L (metade da distância entre elas) e $\dot{\varphi}$ (velocidade de rotação); nesse caso, φ denota o deslocamento angular, sendo que $\dot{\varphi}$ corresponde à sua derivada temporal. Vamos dividir a velocidade total do robô entre suas componentes no sistema local e definir a contribuição de cada uma delas para o valor total, $\overrightarrow{V_c}$, conforme:

$$\overrightarrow{V_c} = (\dot{x_c} \quad \dot{y_c} \quad \dot{\theta_c}), \quad (4)$$

onde $\dot{x_c}$, $\dot{y_c}$ e $\dot{\theta_c}$ correspondem às componentes da velocidade do veículo, descritas no sistema de coordenadas local. Na direção do eixo $\overrightarrow{X_c}$, uma vez que duas das rodas sejam fixas, a componente será a média entre as suas velocidades. Considerando-se que um deslocamento angular pode ser convertido em linear multiplicando-o pelo raio que descreve o movimento, teremos:

$$\dot{x_c} = \frac{v_{r1}+v_{r2}}{2} = \frac{r\dot{\varphi}_1+r\dot{\varphi}_2}{2}, \quad (5)$$

sendo v_{ri} a velocidade linear e $\dot{\varphi}_i$ a velocidade angular de cada roda.

Paralelamente ao eixo $\overrightarrow{Y_c}$, considera-se que não haverá movimento (exceto escorregamentos laterais que não são previstos pelo modelo matemático), lembrando-se novamente que existem duas rodas não-orientáveis na estrutura. Logo,

$$\dot{y_c} = 0. \quad (6)$$

Por fim, o movimento de rotação será representado pela componente $\dot{\theta}_c$. Essa, a exemplo do que ocorreu na direção $\overrightarrow{X_c}$, consistirá na média entre a velocidade das duas rodas. Representa-se:

$$\dot{\theta}_c L = \frac{r\dot{\varphi}_2 - r\dot{\varphi}_1}{2},$$
$$\dot{\theta}_c = \frac{r\dot{\varphi}_2 - r\dot{\varphi}_1}{2L}. \quad (7)$$

Neste caso, o sinal antes dos termos $r\dot{\varphi}_1$ e $r\dot{\varphi}_2$ depende de qual índice corresponde à roda esquerda e qual corresponde à direita. Fixemos como movimentos angulares positivos aqueles realizados no sentido anti-horário. Como neste trabalho estamos chamando de roda 1 aquela localizada no lado esquerdo do robô, se tivermos $r\dot{\varphi}_1$ maior que $r\dot{\varphi}_2$ a velocidade angular total resultante será negativa. Portanto, organizam-se as Equações (7) da forma mostrada. Substituindo-se as Equações (5), (6) e (7) na Equação (4), teremos:

$$\overrightarrow{V_c} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \end{bmatrix} = \begin{bmatrix} \frac{r\dot{\varphi}_1 + r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_2 - r\dot{\varphi}_1}{2L} \end{bmatrix}. \quad (8)$$

A representação no sistema $\{X_0, Y_0, \Theta_0\}$ é obtida substituindo-se a Equação (8) em (3), obtendo-se

$$\overrightarrow{V_0} = \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{\theta}_0 \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & -\operatorname{sen}\theta_0 & 0 \\ \operatorname{sen}\theta_0 & \cos\theta_0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r\dot{\varphi}_1 + r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_2 - r\dot{\varphi}_1}{2L} \end{bmatrix}. \quad (9)$$

Podemos ainda simplificar a Equação (9), expressando cada componente em função da velocidade linear v e angular ω do robô, dadas por

$$v = \frac{r\dot{\varphi}_1 + r\dot{\varphi}_2}{2},$$
$$\omega = \frac{r\dot{\varphi}_2 - r\dot{\varphi}_1}{2L}, \quad (10)$$

obtendo-se

$$\overrightarrow{V_0} = \begin{bmatrix} \dot{x_0} \\ \dot{y_0} \\ \dot{\theta_0} \end{bmatrix} = \begin{bmatrix} v\cos\theta_0 \\ v\,\text{sen}\,\theta_0 \\ \omega \end{bmatrix}. \tag{11}$$

Quando inseridas no código embarcado em robôs móveis, as Equações (9) e (11) possibilitam que a velocidade instantânea do conjunto seja estimada e, consequentemente, que o veículo meça distâncias percorridas. Conforme mostrado na Equação (11), a velocidade linear do robô é composta pelas componentes no sentido dos eixos X e Y, enquanto que sua parcela angular corresponde simplesmente à variação da orientação do veículo.

O modelo cinemático do robô, descrito na Equação (9) e de forma alternativa na Equação (11), será utilizado nos códigos de controle. Na versão discreta dessas equações, a função torna-se dependente da multiplicação entre uma variável inteira e o tempo de amostragem utilizado. Por simplificação, esse produto será designado apenas pela letra k. Utilizam-se os últimos valores medidos de inclinação do robô e de velocidades angulares de cada roda. Esse modelo está demonstrado na Equação (12).

$$\overrightarrow{V_0}[k+1] = \begin{bmatrix} \dot{x_0}[k+1] \\ \dot{y_0}[k+1] \\ \dot{\theta_0}[k+1] \end{bmatrix} = \begin{bmatrix} \cos\theta_0[k] & -\,\text{sen}\,\theta_0[k] & 0 \\ \text{sen}\,\theta_0[k] & \cos\theta_0[k] & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{r\dot{\varphi}_1[k]+r\dot{\varphi}_2[k]}{2} \\ 0 \\ \frac{r\dot{\varphi}_2[k]-r\dot{\varphi}_1[k]}{2L} \end{bmatrix}. \tag{12}$$

3 METODOLOGIA

Os códigos de controle do robô foram projetados com base em um veículo tal qual o especificado nas Seções 2.3 e 2.4. A partir de uma lista de vértices predefinida é formada a figura que deve ser percorrida. A área interna à ela passa por um processo de discretização e por meio disso é gerada a trajetória a ser seguida pelo robô. Essa rota é segmentada e descrita por meio de equações de reta e de curvas. O controle de seguimento do caminho encontrado é feito com base no modelo cinemático do robô.

Cada algoritmo desenvolvido será explicado em uma subseção específica. O planejamento de trajetórias é detalhado na Seção 3.1, o processo de controle de velocidade das rodas na Seção 3.2 e o de seguimento de rotas geradas na Seção 3.3.

3.1 PLANEJAMENTO DE TRAJETÓRIAS

O programa gerador de trajetórias deve, a partir da lista de vértices escolhidas pelo usuário, encontrar o melhor trajeto que faça o carro percorrer toda a área interior à figura sem passar mais de uma vez pelo mesmo ponto. O código produzido analisa cada uma das rotas possíveis de serem feitas respeitando-se algumas restrições, sendo que o modo de avaliar cada uma das possibilidades é através do seu peso associado.

O processo de planejamento de trajetórias possui duas partes constituintes: a discretização do polígono e a geração de trajetórias. Essas etapas estão explicadas nas Seções 3.1.1 e 3.1.2, respectivamente.

3.1.1 Discretização do Polígono

No contexto deste trabalho, discretização é o processo no qual a região interna de um polígono é dividida em quadrados iguais de área predefinida. Isso é feito de modo a tornar finita a dimensão do problema e para facilitar a etapa seguinte do trabalho, que é a escolha da melhor trajetória para o veículo.

A execução desse método depende de um valor constante que define a medida do lado dos quadrados que dividirão a figura. Quanto menor for esse valor, maior será a matriz representativa. Para que a maior porcentagem possível da área seja percorrida sem repetição de regiões visitadas, essa constante deve ser igual à largura do veículo (13 cm, levando em conta o veículo descrito na Seção 2.3). A Figura 6 mostra um exemplo de polígono formado pelos pontos P1 (1,0; 1,0), P2 (2,0; 3,0) e P3 (4,0; 2,0) sobre um fundo quadriculado. Neste caso, a medida do lado dos quadrados de referência foi fixada em meio metro.

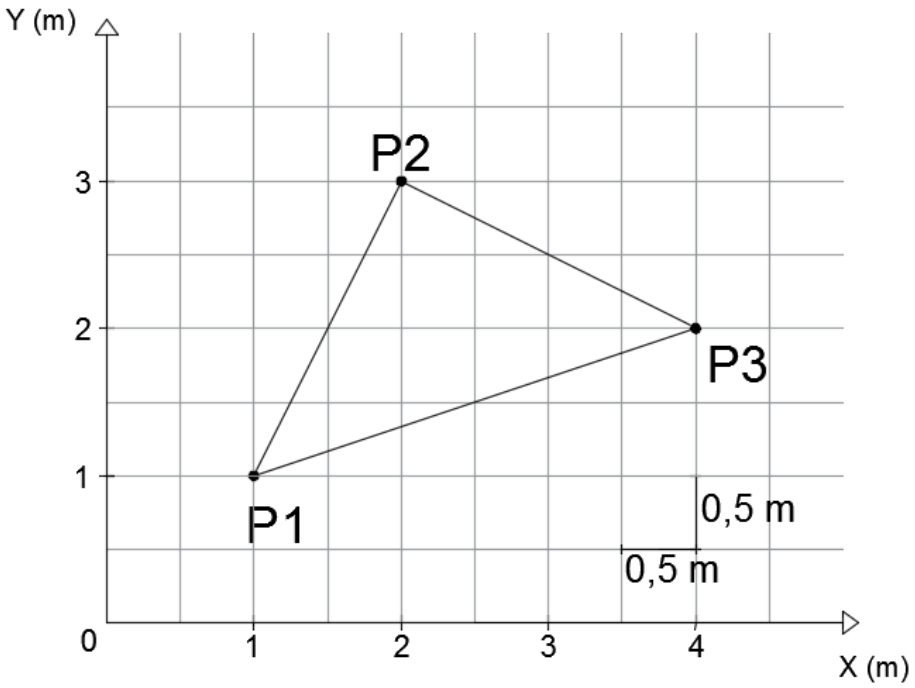

Figura 6 - Exemplo de figura escolhida posicionada entre eixos cartesianos.

Uma vez que a função receba a lista de vértices da figura, é feita uma transformação de coordenadas, similar àquela realizada na modelagem matemática do robô descrita na Seção 2.5. Primeiro, a ordem dos vértices é reorganizada, de

modo que o ponto P1 passe a ser aquele que possui menor coordenada X. Se houver mais de uma posição com mesmo valor no eixo das ordenadas, escolhe-se o que tiver menor índice Y. Já o ponto P2 será o que, entre os demais, garanta que o polígono todo encontre-se no 1º quadrante desse novo sistema.

A Figura 7 relaciona os dois sistemas de coordenadas citados, com a ordem dos pontos já reorganizada. Para a realização dessa alteração nos eixos, podemos utilizar a Equação (3), aplicada apenas para 2 dimensões. Haverá um termo responsável pelo movimento de rotação e outro pelo de translação (necessário já que as origens dos dois eixos não coincidem). Seja $\{X_0, Y_0\}$ o sistema de coordenadas original, $\{X_1, Y_1\}$ o novo que iremos usar, θ o ângulo entre eles e (x_c, y_c) as coordenadas da origem do novo sistema, teremos

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix}. \tag{13}$$

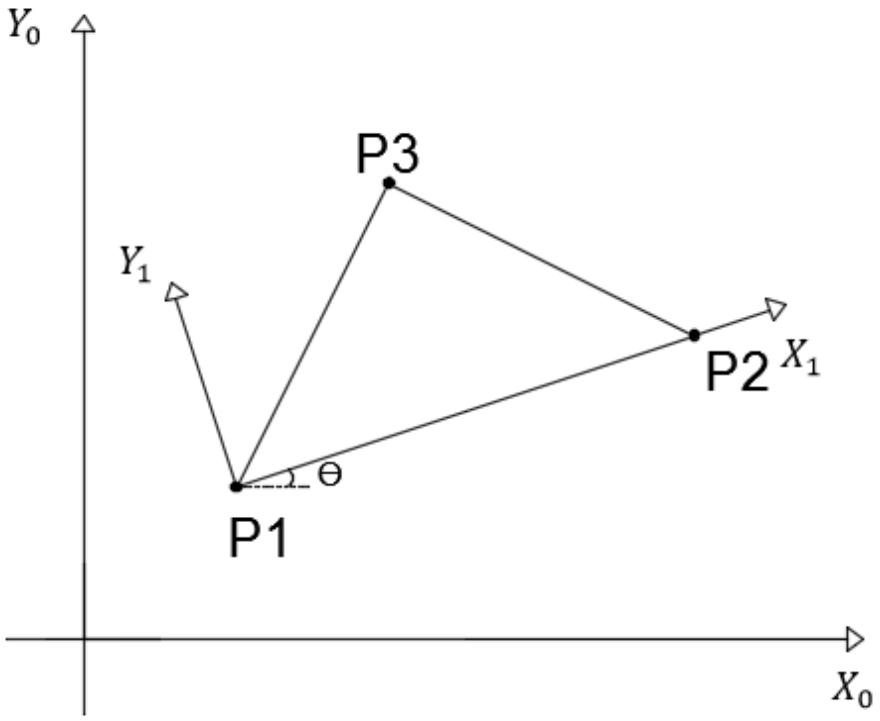

Figura 7 - Relação entre os dois sistemas de coordenadas utilizados.

Seja (x_{Pi}, y_{Pi}) as coordenadas do ponto P_i no sistema original, o ângulo θ pode ser encontrado da seguinte forma

$$\theta = \arctan\left(\frac{y_{P_2} - y_{P_1}}{x_{P_2} - x_{P_1}}\right). \quad (14)$$

Como dessa vez as variáveis desconhecidas são as coordenadas dos vértices no sistema local da figura, precisamos isolá-las na Equação (13). Fazendo isso, encontramos

$$\begin{aligned} x_1 &= (x_0 - x_c)\cos\theta + (y_0 - y_c)\,\mathrm{sen}\,\theta, \\ y_1 &= (x_c - x_0)\,\mathrm{sen}\,\theta + (y_0 - y_c)\cos\theta. \end{aligned} \quad (15)$$

Através das Equações (15), podemos transformar as coordenadas dos vértices escolhidos pelo usuário para o sistema de coordenadas deslocado que criamos. Todos os demais cálculos serão feitos com base nesse sistema.

Após isso, é possível que criemos a matriz de elementos *booleanos* que representa a figura. Os quadrados que dividem a área serão delimitados por linhas paralelas aos eixos cartesianos, conforme mostrado na Figura 8.

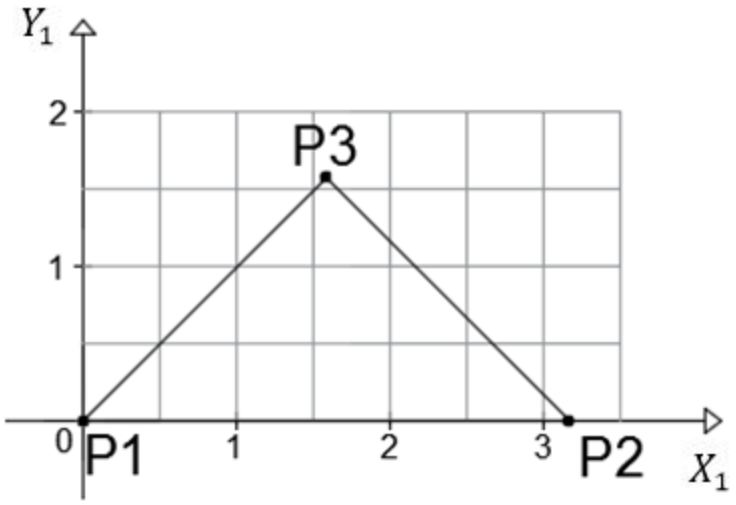

Figura 8 - Figura a ser percorrida expressa no novo sistema de coordenadas escolhido.

As dimensões da matriz podem ser definidas através das Equações (16). Seja n_L o número de linhas, n_C o de colunas, x_m e y_m, respectivamente, as maiores

coordenadas X e Y entre os vértices, e D a constante que define o lado de cada quadrado, conforme explicado anteriormente nesta seção, teremos

$$n_L = \frac{y_m}{D} + 1,$$
$$n_C = \frac{x_m}{D} + 1. \quad (16)$$

O preenchimento da matriz é feito da seguinte forma: primeiro, criou-se uma função responsável por verificar se um ponto faz parte ou não da figura predefinida. Esse método funciona baseado no princípio que diz que, para verificar-se se um ponto faz parte ou não de um polígono, deve-se traçar uma linha que passe pelo ponto, com inclinação qualquer, e contar-se o número de vezes que essa linha atravessa os lados do polígono de um lado e de outro do ponto. Se cada um desses números for ímpar, então o ponto está na figura e a função retorna “verdadeiro”; se os valores forem pares, o retorno será “falso”[22].

Esse método é utilizado para analisar cada ponto central dos quadrados, como os mostrados na Figura 8. Se a função indicar que o ponto pertence ao polígono, a posição correspondente na matriz recebe o valor “verdadeiro”, e “falso” caso contrário. A matriz resultante do processo de discretização para os vértices P1 (1,0; 1,0), P2 (2,0; 3,0) e P3 (4,0; 2,0) usados como exemplo seria a seguinte:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

3.1.2 Geração de Trajetórias

Uma vez que o espaço no qual o veículo irá se mover tenha sido discretizado, pôde-se iniciar o processo de escolha da melhor trajetória que ele deveria seguir para cumprir o objetivo proposto. A criação da matriz de elementos *booleanos* que representa a figura escolhida tornou finito o número de trajetórias possíveis de serem

seguidas no interior da figura especificada, sendo necessário escolher uma das mesmas.

Os movimentos possíveis do carro realizar em cada instante, conforme ilustrado na Figura 9, são 3: seguir no sentido atual, realizar uma curva à direita e realizar uma curva à esquerda. Em todos eles, a distância percorrida em cada passo será de uma posição (o valor em metros depende da constante utilizada na discretização do polígono).

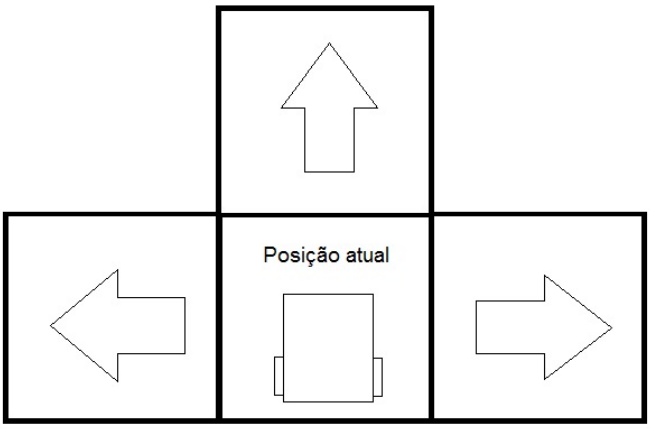

Figura 9 - Opções de movimento entre as quais o robô deve escolher continuamente.

O algoritmo de geração desenvolvido considera apenas o preenchimento da figura em linhas paralelas à reta que une o seu primeiro e segundo vértices (definidos após a reorganização da ordem dos pontos, descrita na Seção 3.1.1). Assim, os trajetos possíveis de serem realizados no interior da figura tomada como exemplo na Seção 3.1.1 são aqueles mostrados na Figura 10.

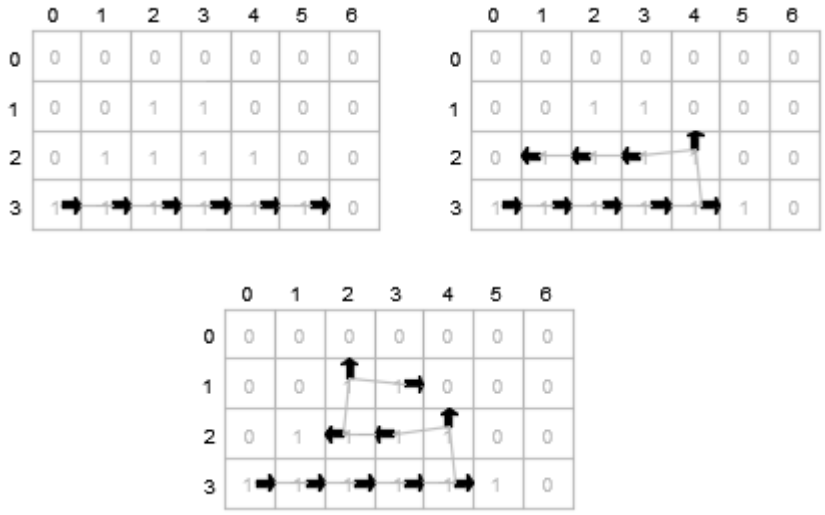

Figura 10 - Trajetos que o código encontraria para a figura tomada como exemplo.

A escolha da melhor dessas trajetórias consiste em um problema de minimização de uma função-custo $f(x)$. Essa associa à cada caminho, representado por um vetor x, um número escalar correspondente ao seu peso. A rota designada para o veículo será aquela que possuir menor custo associado. O problema pode ser formulado seguinte forma

$$\min_x f(x) \tag{17}$$

sujeito à restrição de movimento em linhas descrita anteriormente. Cada deslocamento possível mostrado na Figura 9 possui um custo associado. O cálculo do peso total de um trajeto leva em conta ainda o número n_c de casas que deveriam ser visitadas mas não foram. Seja $v_c = [c_r,\ c_c,\ c_n]$ o vetor de custos utilizado, no qual c_r corresponde a movimentos em linha reta, c_c aos em curva e c_n às posições não preenchidas. O vetor $x = [x_1,\ x_2,\ x_3, \dots, x_n]$, sendo n o número de deslocamentos unitários realizado, caracteriza um trajeto possível. Cada um dos elementos desse vetor pode assumir o valor c_r, se o movimento em questão for em linha reta, ou c_c se for em curva. A função $f(x)$ pode ser escrita como

$$f(x) = \sum_{i=1}^{n} x_i + n_c c_n. \tag{18}$$

O método recursivo responsável por encontrar as trajetórias possíveis foi desenvolvido com base em um simulador executado em um computador. Essa função é chamada toda vez que o veículo se move de uma casa a outra da matriz. A todo momento, são armazenados os índices (linha e coluna) que definem a colocação do robô e variáveis contendo sua orientação vertical e horizontal, o peso acumulado do trajeto atual, a melhor trajetória encontrada até aqui e seu peso associado e uma matriz temporária que informa ao programa as partes já visitadas da figura.

Partindo da consideração de posição inicial do veículo citada anteriormente nesta seção, cada instância da função deve analisar qual o movimento seguinte a ser feito. Em posições da matriz nas quais foram definidos movimentos seguintes em linha reta que não resultaram em fins de trajeto nem em caminhos de largura igual a uma casa, os deslocamentos em curva não são testados. Além disso, todas as curvas devem ser sucedidas obrigatoriamente de outra curva no mesmo sentido. Dessa forma, como dito, o modo como o robô percorre a matriz é fixo: sempre em linhas, partindo do ponto inicial com orientação positiva em X no sistema de coordenadas criado para representar a figura. Todos os movimentos verticais são considerados curvas.

Como pode ser observado na Figura 10, todas as curvas do trajeto são seguidas de outra do mesmo tipo (para a esquerda ou direita do veículo). Sobre os caminhos de largura igual a 1 citados anteriormente nesta seção, tratam-se de retas horizontais cercadas de valores “falso” ou das extremidades da matriz acima e abaixo delas. Esse caso está demonstrado na Figura 11.

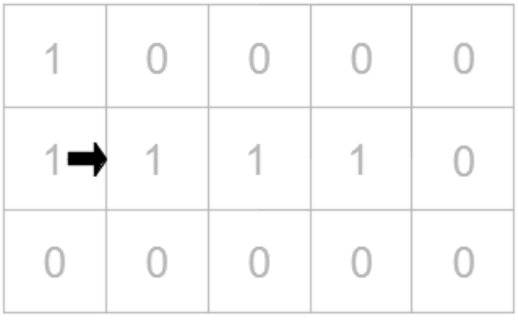

Figura 11 - Exemplo de trecho de caminho de largura igual a uma casa.

Quando o único caminho à frente do robô for cercado de valores “falso” nas linhas imediatamente superior e inferior, é permitido que tente-se realizar curvas

nessa posição. Uma vez que a primeira opção realizável tenha sido encontrada, adiciona-se ao peso total o custo que o veículo tem para realizar esse próximo deslocamento e chama-se recursivamente o mesmo método, passando para ele as novas coordenadas de posição e orientação.

Esse processo se repete até que se alcance um ponto no qual nenhum dos movimentos permitidos é possível. Um exemplo dessa situação está ilustrado na Figura 12.

Figura 12 - Exemplo de um fim de percurso.

Quando um ponto como esse é atingido, o programa entende que o fim de um trajeto foi alcançado. Em seguida, é contabilizado o peso referente às casas da matriz temporária que possuem valor “verdadeiro”, ou seja, aquelas que não foram percorridas. Esse valor é então adicionado à variável que armazena o peso total do trajeto, e essa quantia final é comparada com a menor encontrada até então. Se o número atual for menor do que o outro, este passa a ser o novo menor peso, e as posições percorridas neste trajeto são salvas em um vetor de inteiros.

Após um fim de percurso ser atingido, volta a ser executada uma versão previamente invocada do método. Essa, por sua vez, verifica se há mais algum movimento possível de ser feito a partir daí, fora aquele já realizado e que resultou num fim de trajeto.

Quando a posição inicial do veículo é alcançada novamente e todos os movimentos permitidos a partir daí já tenham sido realizados, o programa chega ao fim. O resultado é um vetor de inteiros salvando o trajeto com menor custo encontrado e uma variável, também inteira, com o seu peso correspondente.

O Apêndice A contém um pseudocódigo que elenca os procedimentos que o método criado realiza. Como foi utilizada a linguagem de programção *C*++, as variáveis globais citadas correspondem a objetos internos da classe à qual a função pertence. Além disso, antes que a primeira instância dessa rotina seja chamada, é necessário carregar a variável “menorPeso” com um grande valor, de modo que se garanta que o custo do primeiro trajeto encontrado sirva de referência para as demais comparações.

O trajeto discreto encontrado, designado pela sequência de posições da matriz que devem ser visitadas pelo veículo, deve ser transformado em contínuo para que possa ser seguido. Isso é feito convertendo-se índices de linha e coluna em coordenadas X e Y expressas em metros. A partir dessas, é possível encontrar as equações de reta e curva que devem ser seguidas. Esse cálculo é feito da seguinte forma:

$$x = Dp_C + \frac{D}{2},$$
$$y = D(n_L - p_L - 1) + \frac{D}{2}, \tag{19}$$

p_C e p_L correspondem, respectivamente, ao número da coluna e da linha onde o robô se encontra. O cálculo da coordenada vertical é mais trabalhoso devido ao fato de o sentido crescente de p_L ser inverso ao utilizado em y. A Figura 13 ilustra um exemplo de reta a ser seguida, com $D = 0{,}5\ \mathrm{m}$. Nesse caso, a equação encontrada seria $y = 0{,}25$ com x de destino valendo 0,25 ou 2,75, dependendo do sentido atual de movimentação.

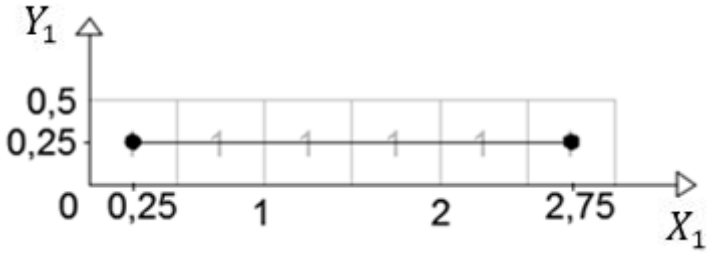

Figura 13 - Trajeto em forma de reta de equação $y = 0,25$ e com $D = 0,5$ m.

Já para movimentos em curva, é necessário especificar o centro (x_c, y_c) e raio de curvatura r_c. Esses valores podem ser obtidos através das seguintes equações:

$$\begin{aligned} x_c &= Dp_C + \tfrac{D}{2}, \\ y_c &= D(n_L - p_L), \\ r_c &= \tfrac{D}{2}, \end{aligned} \tag{20}$$

onde p_C e p_L correspondem aos índices de localização do último ponto do trajeto anterior. A Figura 14 ilustra geometricamente um exemplo de curva gerada dessa maneira.

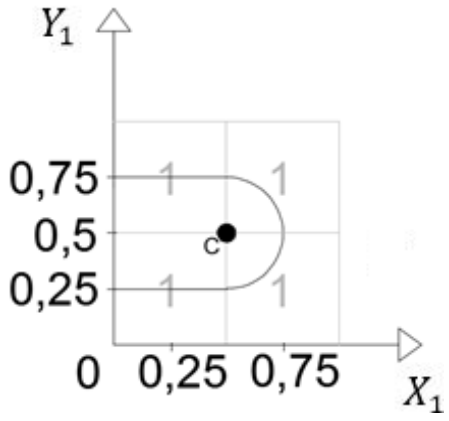

Figura 14 - Exemplo de movimento em curva com $x_c = 0,25$ m, $y_c = 0,5$ m e $r_c = 0,25$ m.

Um caso que foge às regras apresentadas ocorre quando o sentido de movimentação no eixo X não se altera após uma curva. As fórmulas desta seção pressupõem que a posição e a inclinação do veículo são favoráveis no começo de um novo segmento de trajeto. Como isso não ocorre nessa situação, foi preciso se adotar uma técnica de resolução diferente. Curvas que apresentam essa característica são

divididas em duas “semi curvas”, cada qual equivalente a um quarto de circunferência. A Figura 15 ilustra esse problema e sua solução.

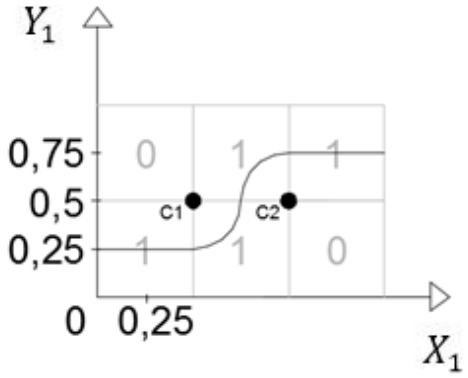

Figura 15 - Exemplo de caso no qual a segmentação de curva foi necessária. C1 e C2 são os centros dos círculos que definem cada trecho do trajeto.

Para a figura usada como exemplo na Seção 3.1.1, a trajetória total seria formada por 3 segmentos de reta e 2 semicírculos. Esse resultado está ilustrado na Figura 16.

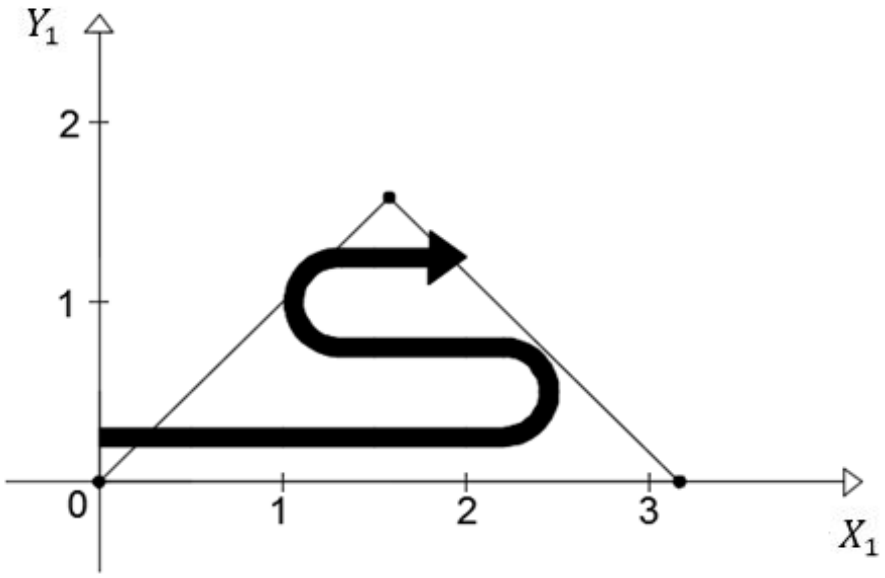

Figura 16 - Trajeto realizado pelo veículo para preencher a figura usada como exemplo na Seção 3.1.1.

3.2 CONTROLE DE VELOCIDADE ANGULAR DAS RODAS

O controle de velocidade de cada uma das rodas envolve duas variáveis básicas: a velocidade angular estimada (valor de entrada) e o nível do sinal PWM enviado ao motor que movimenta a roda (valor de saída). Um controlador do tipo proporcional bastou para a correção das quantidades medidas.

A estrutura básica da função de controle de cada roda é a seguinte: com base na diferença entre a velocidade lida e a desejada naquele instante, calcula-se o incremento ou decremento no nível PWM correspondente; ou seja,

$$p_i = p_i - k_{Pv}\Delta\dot{\varphi}_i, \tag{21}$$

onde p_i é o valor da tensão de saída, k_{Pv} é a constante de controle proporcional, $\Delta\dot{\varphi}_i$ é o erro na velocidade angular e o índice i pode valer 1 ou 2 dependendo de qual roda estamos tratando. O sinal negativo na Equação (21) decorre do fato de se $\Delta\dot{\varphi}_i$ for positivo, p_i terá que ser reduzido para que o erro diminua na próxima medição, e vice-versa.

Foram definidos valores mínimo e máximo para a tensão p_i. Esse valor não deve ser negativo, de modo a garantir apenas rotações que fazem o robô se deslocar para frente. O outro extremo foi predefinido a partir da velocidade linear total que deve ser mantida em 0,278 m/s. Levando em conta essa condição, a maior velocidade angular permitida para uma roda ($\dot{\varphi}_{im}$) é alcançada quando a outra está parada. A partir da Equação (10), que expressa a velocidade linear total do veículo em função das frequências de rotação $\dot{\varphi}_1$ e $\dot{\varphi}_2$, obtemos

$$\dot{\varphi}_{im} = \frac{2V}{R} = \frac{0{,}556}{0{,}032} = 17{,}38\ \text{rad/s}. \tag{22}$$

Com base na reta de modelagem dos motores contida na Equação (1), chegamos ao seguinte valor máximo de PWM

$$p_{im} = \frac{\dot{\varphi}_{im}}{0{,}2} = 86{,}9. \tag{23}$$

Foram escritas funções básicas de cálculo de velocidade a partir de um nível de sinal PWM, definição da frequência de rotação desejada e de controle proporcional com base na Equação (21). O processo desenvolvido para o cálculo da melhor constante k_P é o seguinte: primeiro, um objeto da classe criada para simular o veículo é instanciado, passando-se como parâmetro os valores de velocidade inicial e desejada para cada roda. Em seguida, uma variável auxiliar é criada dentro de um laço para variar entre valores mínimo e máximo predefinidos. A cada execução dessa porção de código, esse número é incrementado em uma outra quantia fixada pelo programador. Todas as constantes obtidas dessa forma são testadas através da função de controle. Com base no intervalo de tempo entre as chamadas desse método que foi fixado em 0,005 segundo, cada k_{Pv} é mantido por cerca de 0,600 segundos (esse intervalo é apenas simulado, o processo todo ocorre instantaneamente), totalizando 120 aferições realizadas. O valor utilizado para avaliar cada uma dessas constantes é a soma do erro absoluto entre as velocidades gerada e desejada; constantes com menor erro acumulado são consideradas melhores que aquelas de erro maior.

Com base no veículo considerado, foi definido que a constante proporcional mais apropriada é de 5,0 para os dois motores. Assim, a forma final da equação (21) foi a seguinte:

$$p_i = p_i - 5{,}0\Delta\dot{\varphi}_i, \tag{24}$$

Após esses cálculos, são feitas verificações que impedem que os valores de p_1 e p_2 ultrapassem os limites mínimos e máximos especificados para cada motor e calculados anteriormente nesta seção.

O Apêndice B contém um pseudocódigo demonstrando a ordem de procedimentos tomados na função de ajuste de velocidade angular das rodas. Como os níveis de PWM enviados aos motores são do tipo inteiro, os limites máximos para cada um deles foram aproximados a partir dos resultados obtidos anteriormente nesta seção.

A Figura 17 exemplifica a atuação do controlador em uma situação em que uma roda parte do repouso e deve alcançar a velocidade de 10 rad/s.

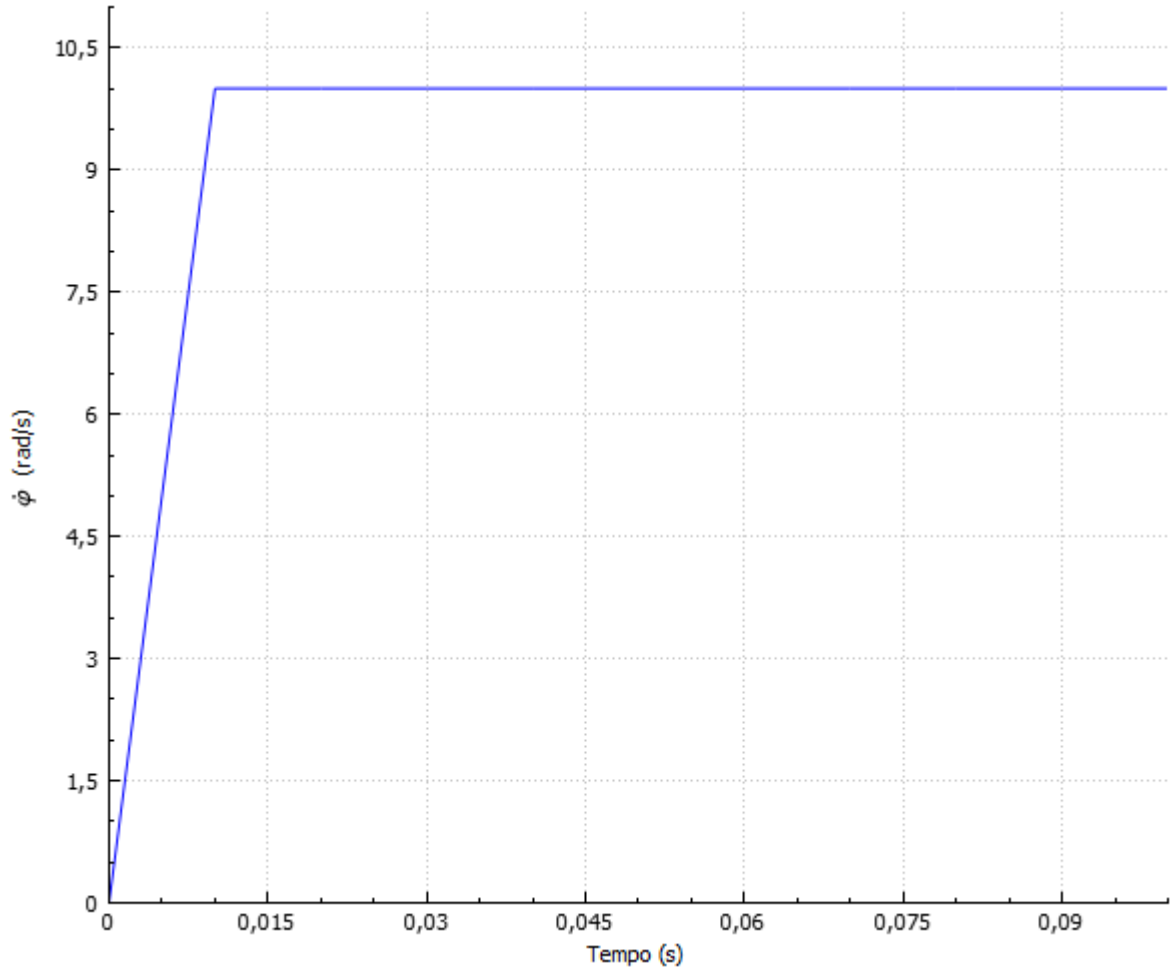

Figura 17 - Resposta da função de controle de velocidade das rodas para um caso em que a roda parte do repouso e deve atingir 10 rad/s.

3.3 SEGUIMENTO DE TRAJETÓRIAS

O seguimento da trajetória escolhida conforme o método explicado na Seção 3.1.2 é feito através de uma função que, com base no desvio estimado do robô até a rota que ele deveria estar seguindo em determinado instante, calcula os valores apropriados de velocidade angular das rodas. Esse é o recurso que o veículo possui para corrigir a sua posição instantânea continuamente. Uma vez que os vértices da figura a ser percorrida tenham sido definidos e que, com base neles, a melhor rota tenha sido encontrada, tem início o processo de medição e controle de velocidade e estimação e controle de posição. A Figura 18 contém um diagrama de blocos

demonstrando a ordem de ações executadas durante o monitoramento e a correção de trajetórias. É importante lembrar-se que o intervalo de tempo entre as chamadas das duas funções de controle mostradas não é o mesmo, conforme explicado na Seção 2.4.

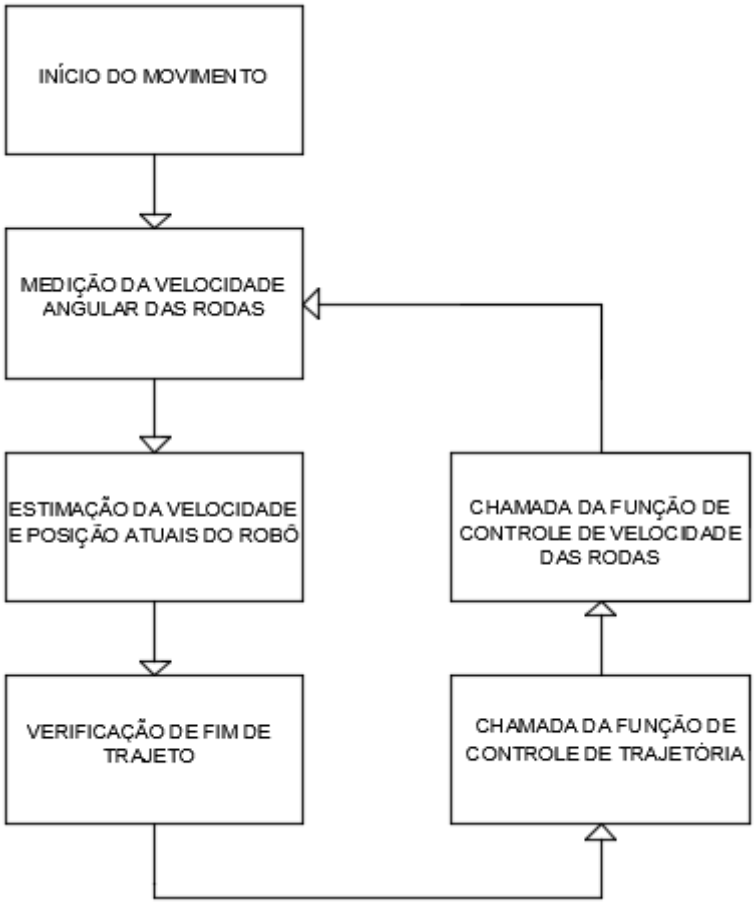

Figura 18 - Diagrama de blocos do processo de monitoramento e ajuste da trajetória efetuada pelo robô.

A Seção 3.3.1 detalha as fórmulas utilizadas para estimação da posição e velocidade atual do robô. Na Seção 3.3.2 apresentam-se as formas de obtenção do erro de posição para cada tipo de trajetória que pode estar sendo seguida. O método controlador projetado está demonstrado na Seção 3.3.3.

3.3.1 Estimação da Posição e Velocidade do Robô

As coordenadas de posição e velocidade total do veículo em cada instante são estimadas em intervalos fixos iguais ao tempo de amostragem citado na Seção 2.4 através do modelo matemático do veículo. Esses valores são obtidos a partir das leituras das velocidades angulares das rodas, da orientação e de constantes geométricas do robô. Com base nas coordenadas de posição (x, y, θ) estimadas pelo programa é feito o controle de seguimento da trajetória escolhida.

Uma vez que as velocidades de cada uma das rodas foram estimadas, o robô calcula, a partir das aferições feitas, as componentes de velocidade e posição atual nos eixos X e Y e suas componentes angulares. Isso é possível graças às Equações (11), para cálculo da velocidade, e às Equações (25), usadas posteriormente para cálculo de posição. Sejam x e y as coordenadas de posição nos eixos X e Y, respectivamente, expressos em metros, θ a inclinação do veículo e T_a o intervalo de tempo entre as execuções desta porção de código. As componentes de posição são estimadas através de

$$\begin{aligned} x[k+1] &= x[k] + v_x[k]T_a, \\ y[k+1] &= y[k] + v_y[k]T_a, \\ \theta[k+1] &= \theta[k] + \omega[k]T_a. \end{aligned} \tag{25}$$

onde v_x e v_y são as componentes de velocidade do veículo em X e em Y, respectivamente, em m/s, e ω é a velocidade angular total, em rad/s. Todos esses valores são calculados através da Equação (12). Conforme definido nas Seções 2.4 e 2.5, considera-se que o sentido positivo de rotação do carro é o anti-horário, que a roda esquerda é a roda 1 e que a direita é a roda 2. O valor de tempo de amostragem T_a é o mesmo da diferença entre os instantes de chamada da função de controle de velocidade, o qual foi fixado em 0,005 segundo conforme citado na Seção 2.4.

3.3.2 Estimação do Erro de Trajetórias

O valor da distância d até o trajeto desejado é obtido de duas formas, dependendo se o veículo estiver seguindo uma reta ou uma curva no momento. A explicação dos métodos utilizados para estimação do erro até uma reta e uma curva estão contidas nas Seções 3.3.2.1 e 3.3.2.2, respectivamente.

3.3.2.1 Trajetórias Retas

O erro de trajetória do robô nos trechos em que ele deveria estar seguindo uma reta é definido como a distância dessa até o centro do veículo. Esse valor deve ser diferente de acordo com o sentido de movimentação adotado no momento. A Figura 19 ilustra o problema.

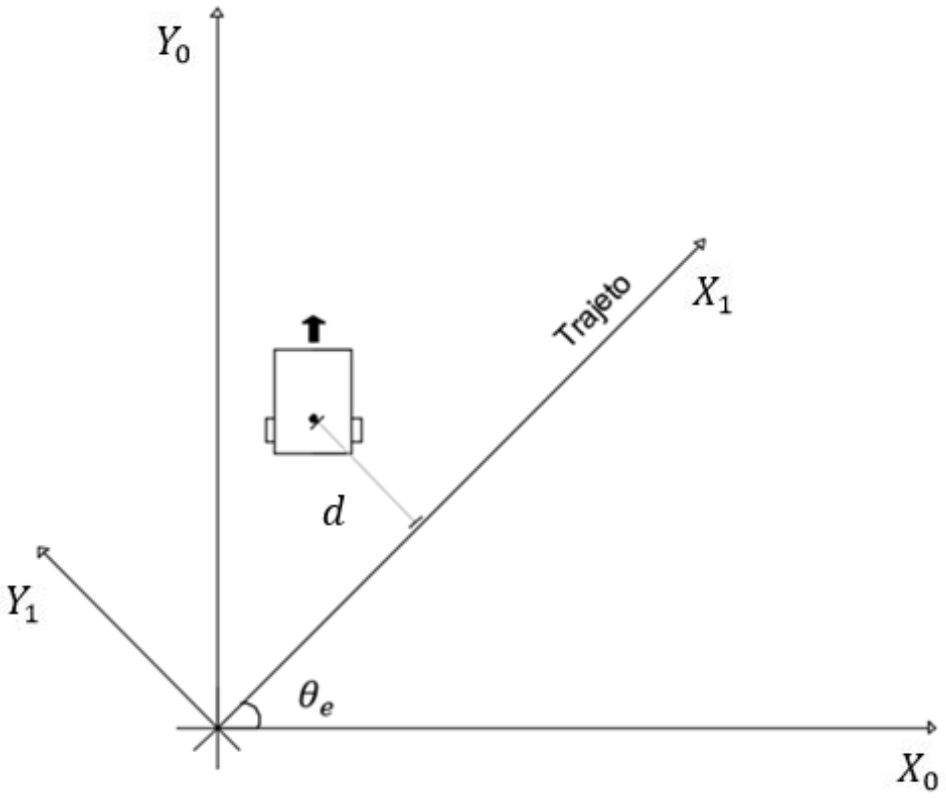

Figura 19 - Relação entre os dois sistemas de coordenadas utilizados, com destaque para a localização do valor de erro de trajetória.

Os cálculos são feitos com base em um sistema de coordenadas transformado. A reta que representa o trajeto passa a ser o eixo X do novo sistema. Assim, sempre que for preciso saber o erro no deslocamento atual, chama-se uma função de transformação, que converte os valores de localização inerciais para seus correspondentes nos novos eixos calculados, e o valor de d é equivalente à nova componente Y obtida. Seja a reta a ser seguida expressa pela Equação (26).

$$y = r_t[1]x + r_t[0], \tag{26}$$

sendo r_t um vetor de duas posições que especifica a linha do trajeto atual. A criação de um novo sistema de coordenadas foi feita de maneira idêntica àquela explicada na Seção 2.5. Através desse mesmo processo, vamos expressar as coordenadas de um ponto (x_0, y_0) do sistema inercial no novo sistema criado. Teremos

$$\begin{aligned} x_1 &= (x_0 - x_c)\cos\theta_e + (y_0 - y_c)\,\mathrm{sen}\,\theta_e, \\ y_1 &= (x_0 - x_c)\,\mathrm{sen}\,\theta_e + (y_0 - y_c)\cos\theta_e, \end{aligned} \tag{27}$$

onde (x_1, y_1) são as coordenadas do ponto convertidas; (x_c, y_c) representam a origem dos novos eixos, e θ_e é o ângulo entre os dois eixos X, ou seja, o ângulo de inclinação da reta que representa o trajeto. O valor de θ_e é diferente de acordo com o sentido de movimentação que deve ser efetuado no momento. Se este for positivo, ou seja, se o veículo se move no sentido crescente do eixo X, teremos

$$\theta_e = \arctan r_t[1]. \tag{28}$$

Se o sentido acompanhar o eixo X negativo, o cálculo será feito com base na Equação (29).

$$\theta_e = \arctan r_t[1] + \pi. \tag{29}$$

Como o que nos interessa é apenas a coordenada convertida Y, a escolha da origem desse novo sistema não é importante. Foram fixados os seguintes valores:

$$x_c = 0,$$
$$y_c = r_t[0]. \quad (30)$$

Feitos esses cálculos, o valor do erro atual de posição d do robô é conhecido e está armazenado em y_1.

3.3.2.2 Trajetórias Curvas

No caso de o trajeto desejado no instante atual ser uma curva, outras fórmulas são utilizadas para obtenção do erro de posição. A curva a ser seguida é representada por uma semicircunferência caracterizada pelo seu ponto central (x_c, y_c) e por um raio r_c. A cada instante, o módulo de d é definido como a distância entre a posição atual do robô e o centro da circunferência que define a curva, subtraído do seu raio. O sinal que precede esse cálculo depende de para qual lado a curva está sendo feita. Quando se tratam de movimentos para a esquerda do robô, d é equivalente ao valor calculado com sinal negativo, ou seja,

$$d = -\left(\sqrt{(x - x_c)^2 + (y - y_c)^2} - r_c\right). \quad (31)$$

Caso contrário, nenhuma alteração de sinal é feita, conforme mostrado na Equação (32).

$$d = \sqrt{(x - x_c)^2 + (y - y_c)^2} - r_c. \quad (32)$$

Essa diferença de tratamento decorre do fato de o deslocamento promovido pela velocidade angular ω depender da orientação do robô no instante em que o movimento é realizado. Como o ponto inicial dos trajetos em curva equivale ao ponto final da rota imediatamente anterior, espera-se que quando o seguimento da semicircunferência tiver início o erro de posição seja pequeno. As Equações (31) e

(32) não foram projetadas para funcionar em situações nas quais o robô encontra-se muito distante da curva especificada.

A Figura 20 contém uma demonstração geométrica do conceito de erro de posição num caso em que se deve realizar uma curva para o lado esquerdo.

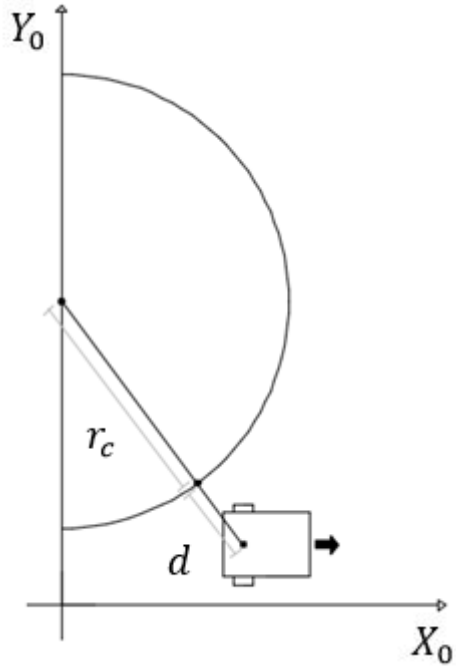

Figura 20 - Interpretação geométrica do erro de posição para trajetórias curvas.

3.3.3 Função de Controle de Seguimento de Trajetórias

A função de ajuste de velocidade é baseada na aplicação de um controlador proporcional-derivativo que possui como entrada o valor da distância estimada do veículo até o trajeto que lhe foi proposto e como saída a nova velocidade angular que o carro deve desenvolver para corrigir a sua colocação. Com base nesse valor calculado, obtêm-se as velocidades angulares recomendadas para cada uma das rodas.

O modelo básico do controlador PD projetado é o seguinte:

$$\omega_n = -k_{Ps} d - k_{Ds} \Delta d, \tag{33}$$

onde ω_n é a nova velocidade angular que o carro deve desenvolver, expressa em rad/s; k_{Ps} e k_{Ds} são as constantes proporcional e derivativa do controlador, respectivamente; d é a distância até o trajeto a ser seguido, em metros, e Δd é a diferença entre d e seu valor anterior, também em metros, ou seja:

$$\Delta d = d_i - d_{i-1}. \tag{34}$$

O valor de ω_n deve ser limitado de forma que não seja definido em níveis inalcançáveis fisicamente para um robô de velocidade linear fixa. Sabendo-se que esse valor foi fixado em 0,278 m/s, o máximo que o módulo de ω_n pode atingir vale:

$$|\omega_{max}| = \frac{2V}{2L} = \frac{0{,}556}{0{,}13} = 4{,}28 \text{ rad/s}. \tag{35}$$

Neste caso, o uso apenas do controle proporcional não soluciona o problema, pois, por mais apropriada que seja a constante k_{Ps}, o valor de d oscila, mantendo um módulo de diferença máximo constante até a quantia desejada. A adição do termo derivativo, conforme demonstrado na Equação (33), garante que a taxa de diminuição de d seja levada em conta, fazendo aproximações mais suaves até o valor ideal (zero).

O controlador a ser projetado deve, através da Equação (33), encontrar o valor de ω que fará o veículo se aproximar da rota desejada. Uma vez que isso tenha sido feito, é preciso descobrir quais velocidades angulares de cada uma das rodas farão o robô desenvolver o ω desejado. Esse cálculo é possível graças a um sistema de equações, criado com base na Equação (12) de estimação das componentes de velocidade do robô. Uma das expressões relaciona $\dot{\varphi}_1$ e $\dot{\varphi}_2$ ao valor de ω projetado, enquanto que a outra é relativa à velocidade linear total que deve ser mantida constante. O sistema a ser resolvido é o seguinte:

$$\begin{aligned} \dot{\varphi}_{2d} - \dot{\varphi}_{1d} &= \frac{\omega_d L}{2R}, \\ \dot{\varphi}_{1d} + \dot{\varphi}_{2d} &= \frac{2V_T}{R}, \end{aligned} \tag{36}$$

onde ω_d e $\dot{\varphi}_{id}$ são a velocidade angular desejada para o robô e para cada uma das rodas, respectivamente, em rad/s, e V_T corresponde à velocidade linear total que o mesmo deveria manter a todo momento, em m/s. Resolvendo essas equações, encontra-se:

$$\dot{\varphi}_{1d} = \frac{V_T - \frac{\omega_d L}{4}}{R},$$
$$\dot{\varphi}_{2d} = \frac{V_T + \frac{\omega_d L}{4}}{R}. \quad (37)$$

Agora que o método de definição de $\dot{\varphi}_1$ e $\dot{\varphi}_2$ a partir do valor de ω_d é conhecido, teve início o desenvolvimento do código capaz de calibrar as constantes do controlador. O processo é análogo àquele utilizado para escolha da constante k_{Pv} das funções controladoras da velocidade angular de cada roda, conforme explicado na Seção 3.2. A sequência de ações tomadas por este programa é a seguinte: antes de tudo, uma classe *C++* é criada contendo todas as fórmulas encontradas de modelagem do robô, bem como o controlador da velocidade angular das rodas. Em seguida, um objeto desse tipo é instanciado, predefinindo-se a posição e velocidade inicias do veículo. Os dois laços de incremento das constantes PD fazem os valores variarem entre limites mínimos e máximos, sendo que eles são alterados continuamente com base em uma taxa fixa. Para cada modificação em k_{Ps}, k_{Ds} é variado entre seus dois extremos, e é testada a resposta do robô simulado para cada combinação dessas duas variáveis. Primeiro, o controlador de posição é chamado, encontrando $\dot{\varphi}_{1d}$ e $\dot{\varphi}_{2d}$. Após isso, é chamada a função de controle de velocidade, a qual calcula os níveis de PWM que devem ser enviados aos motores de modo que as rodas atinjam as velocidades projetadas. Esse procedimento se repete por um intervalo de tempo predefinido em 0,600 segundos. Cada combinação das duas constantes é avaliada através da soma de seu erro d até o trajeto a ser seguido em cada instante.

Como o seguimento de retas e curvas é tratado de maneiras distintas, as constantes das respectivas funções de controle também são diferentes. A porção de código responsável pelo ajuste do controlador foi executada para encontrar as constantes mais adequadas para cada caso.

Para linhas retas, o resultado foi a escolha de um k_{Ps} de valor 686 e de um k_{Ds} igual à 6975, e a Equação (33) assumiu a seguinte forma:

$$\omega_n = -686d - 6975\Delta d. \quad (38)$$

A função foi calibrada novamente para o caso de seguimento de uma curva com raio de curvatura igual à metade da distância entre cada posição da matriz de discretização (único tipo de curva gerada pelo programa). O resultado foi um k_{Ps} de valor 170678 e de um k_{Ds} igual à 159679, com equação de controle

$$\omega_n = -170678d - 159679\Delta d. \quad (39)$$

A função de controle realiza testes finais, os quais garantem que o valor calculado para ω_n não ultrapasse os limites especificados anteriormente nesta seção. O Apêndice C contém o pseudocódigo da função genérica de controle de trajetória, chamada não importa qual o tipo de rota sendo seguida. As constantes do controlador de seguimento de reta ou de curva devem ser passadas por parâmetro de acordo com o caso.

As respostas do robô simulado a esse controlador, funcionando em união com o controle proporcional da velocidade angular das rodas, foram aquelas ilustradas graficamente nas Figura 21 e 22. Na primeira, o veículo parte do ponto $(0{,}0; 0{,}1)$ com inclinação de 0º devendo seguir a reta $y = 0$. Na segunda, ele inicia da origem devendo realizar uma curva de raio igual a 0,065 m à sua esquerda.

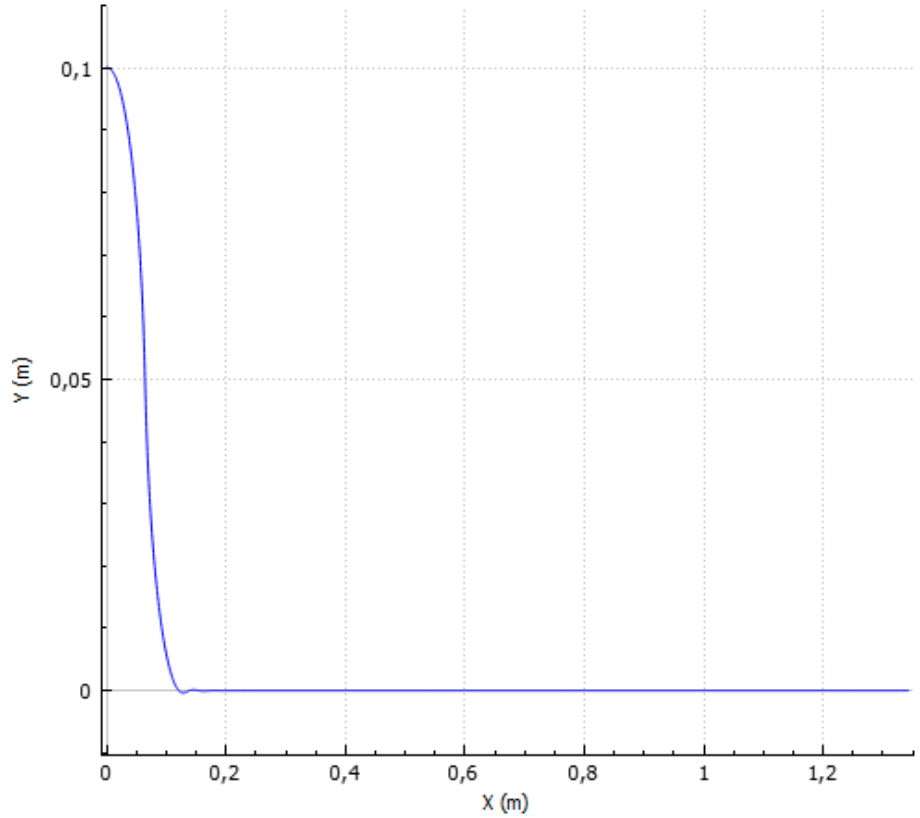

Figura 21 - Resposta da função de ajuste de trajetória para o seguimento de uma reta.

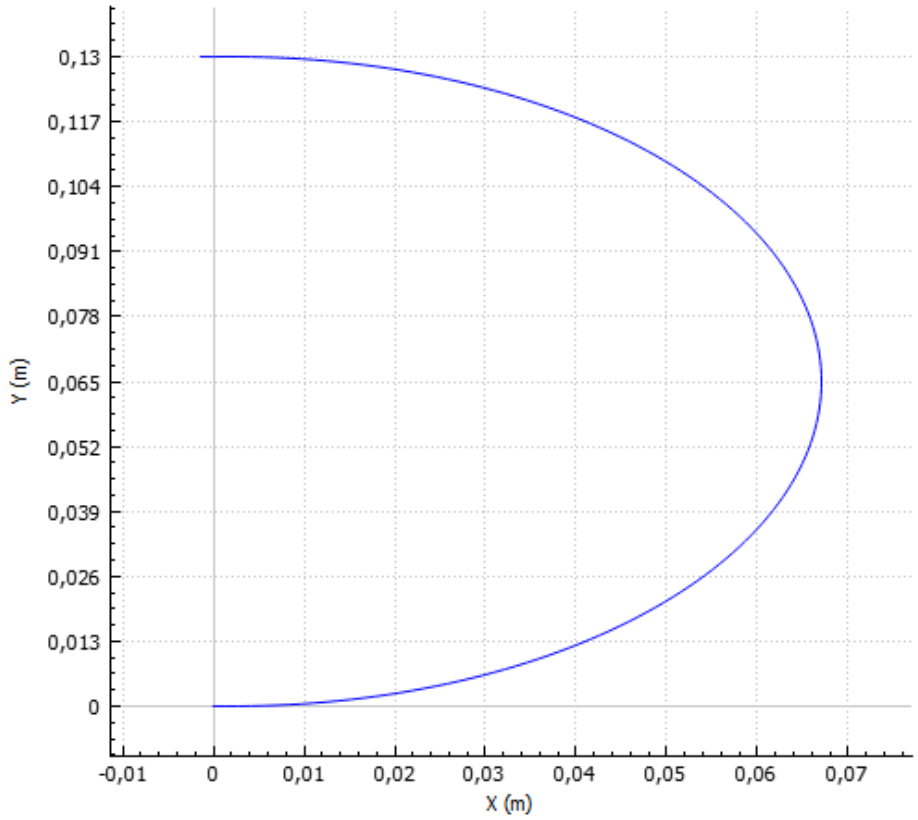

Figura 22 - Resposta da função de ajuste de trajetória para o seguimento de uma curva.

No caso de seguimento das chamadas “semi curvas” por vezes foram encontrados pequenos problemas. Essas situações indesejáveis vêm do fato de a posição inicial do robô no início do segundo quarto de circunferência nem sempre ser favorável. O código de controle utilizado faz com que o robô descreva curvas com raio, inevitavelmente, um pouco maior do que o especificado. Por isso, ao final da primeira semi curva, o veículo necessita se adequar à repentina mudança de trajeto em um intervalo de tempo muito curto, antes que a variável X máxima especificada seja atingida. A Figura 23 ilustra esse problema. A linha contínua descreve o trajeto que deve ser seguido enquanto que a pontilhada mostra uma aproximação do movimento realizado de fato.

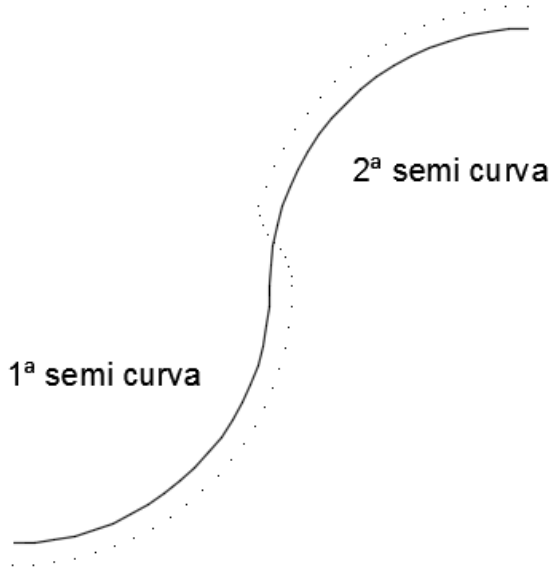

Figura 23 - Ilustração do problema específico do seguimento de curvas segmentadas.

De modo a contornar esse problema, por vezes o raio de curvatura do segundo segmento é definido um pouco abaixo de seu valor usual. Isso faz com que o erro de seguimento de trajetória não se propague para o próximo segmento de trajeto. Para distâncias de discretização suficientemente pequenas, esse efeito é quase imperceptível.

4 RESULTADOS

As fórmulas matemáticas de modelagem dos motores do robô e de estimação da sua posição e velocidade em cada instante, mostradas nas Seções 2.4 e 3.3.1, repectivamente, bem como as funções de geração de trajetórias, de controle de velocidade angular das rodas e de seguimento de trajetórias, presentes nas Seções 3.1 à 3.3, respectivamente, foram inseridas em uma classe da linguagem *C++* que representa o veículo construído. Por meio dessa, é possível simular o comportamento real do veículo, visualizar graficamente o valor de todas as variáveis envolvidas e ajustar as constantes dos controladores.

O simulador foi testado para a geração e seguimento de trajetórias a partir de uma lista de vértices. Em cada caso, foi salvo o erro médio de trajetória de cada estimação de posição. Esse erro diz respeito à diferença média entre a posição do robô e a reta ou curva que ele deveria estar seguindo a cada instante. Em todas as imagens resultantes, a figura a ser preenchida é limitada por linhas contínuas, o trajeto realizado pelo ponto central do veículo é caracterizado por uma linha tracejada e aqueles percorridos por cada roda por linhas pontilhadas. As posição de partida e chegada do veículo são preenchidas com a letra X e com um quadrado, respectivamente. A distância entre cada posição da matriz discretizada foi mantida equivalente à largura total do veículo simulado, a qual vale 13 cm. O vetor de custos utilizado na escolha da trajetória a ser seguida e citado na Seção 3.1.2 foi $v_c = [1,\ 2,\ 5]$, de modo que se dá uma grande importância ao preenchimento do maior número de posições possíveis da matriz de discretização.

O primeiro teste consistiu na definição de um triângulo de vértices (0,0; 0,0), (1,0; 0,0) e (0,5; 1,0). O resultado está exposto na Figura 24. O erro médio encontrado foi de 0,93 mm.

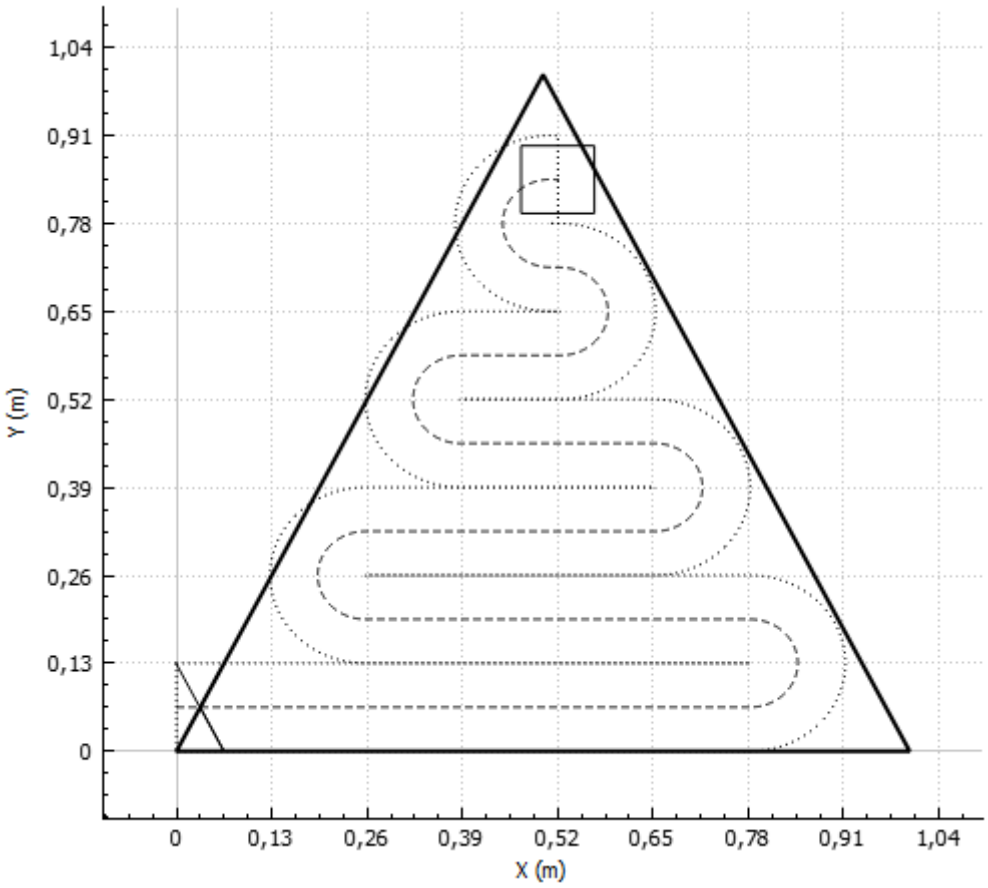

Figura 24 - Movimentação simulada sobre um triângulo.

Em seguida foi passado ao sistema como parâmetro um quadrado de vértices (0,0; 0,0), (2,0; 0,0), (2,0; 2,0) e (0,0; 2,0). O resultado está mostrado na Figura 25. O erro médio de trajeto foi de 0,15 mm.

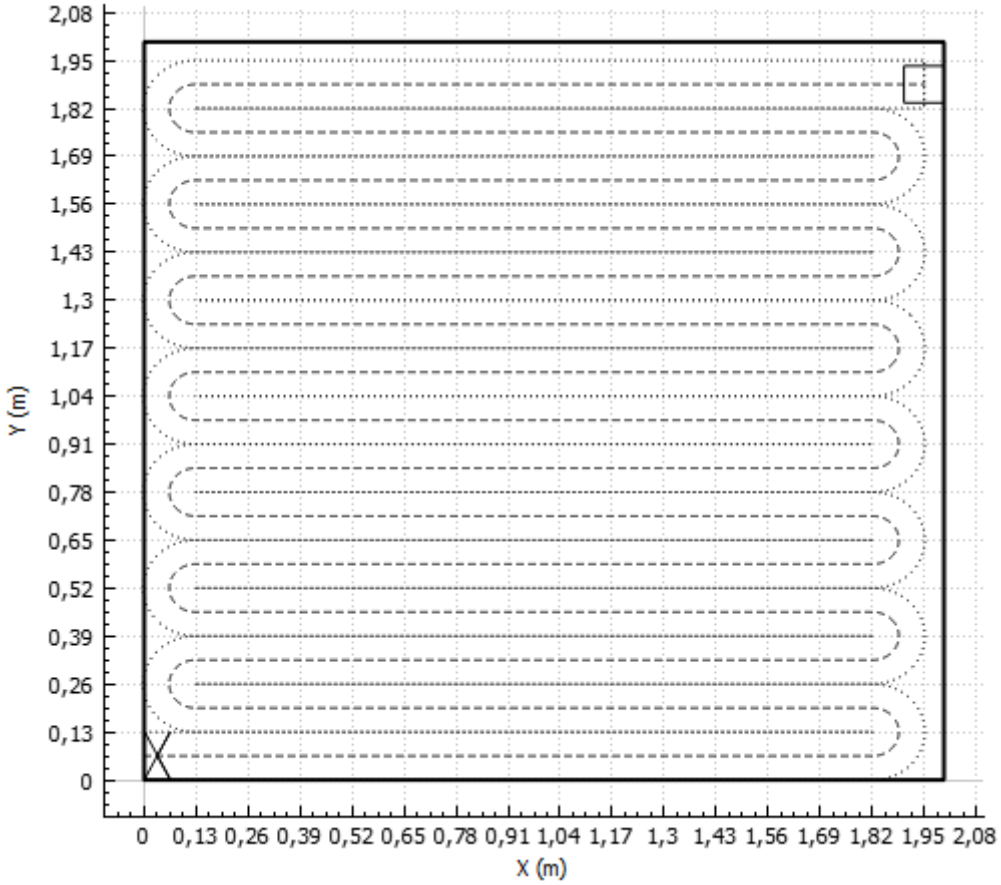

Figura 25 - Movimentação simulada sobre um quadrado.

O terceiro teste envolve a definição de uma figura limitante com grande número de vértices. O resultado está ilustrado na Figura 26 e o erro associado foi de 0,21 mm.

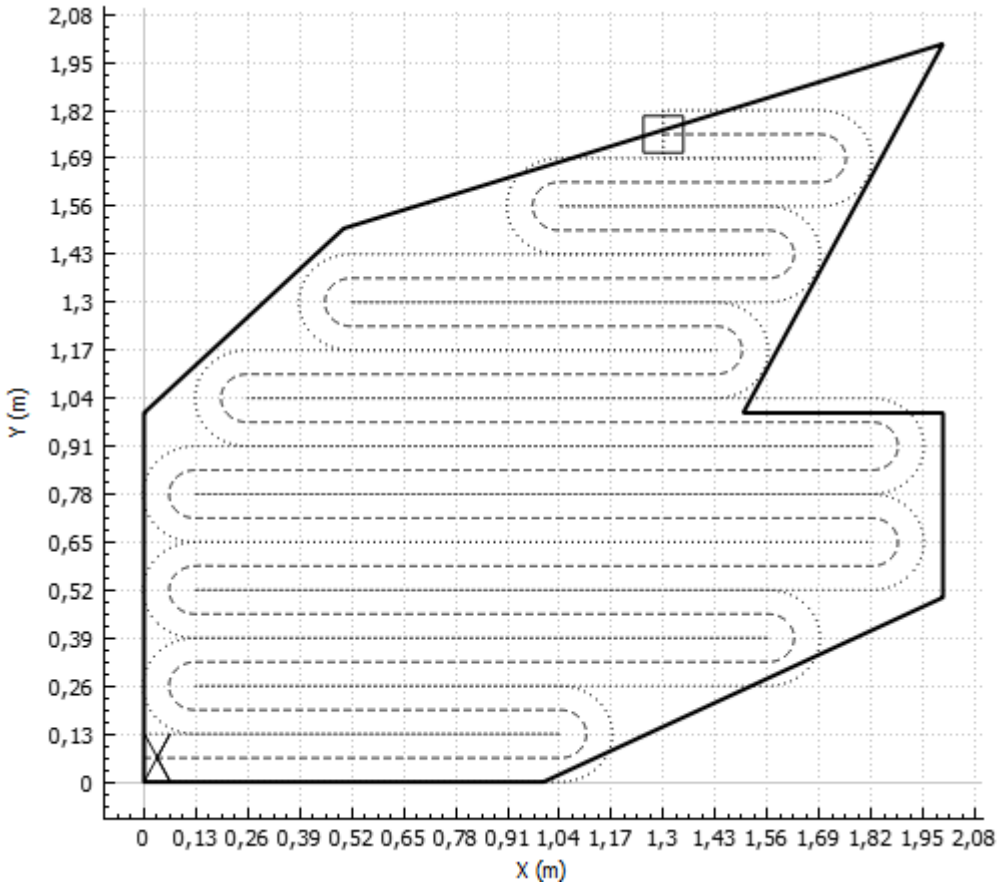

Figura 26 - Movimentação simulada sobre uma figura de 8 lados.

Um teste final foi realizado para verificar a atuação do programa no seguimento de semi curvas. Para isso, foi definida uma figura com vértices (0,0; 0,0), (0,5; 0.0) e (2,0; 1,0). O resultado é mostrado na Figura 27. Três curvas tiveram que ser segmentadas. O erro associado foi de 3,9 mm.

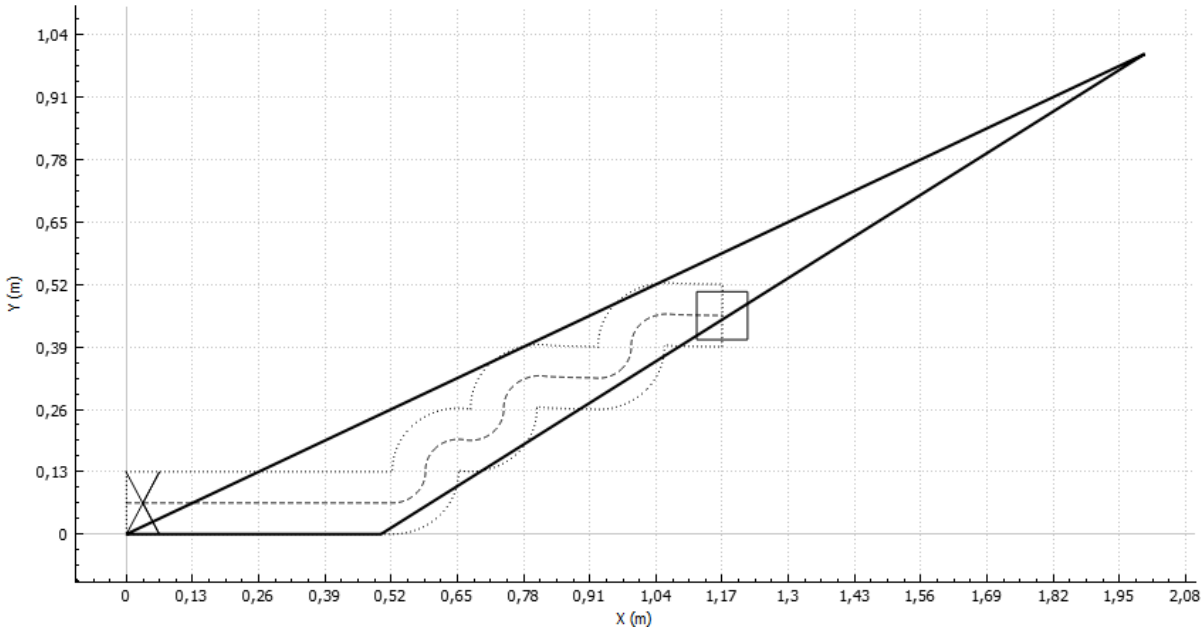

Figura 27 – Movimentação simulada sobre uma figura que exigiu o seguimento de curvas segmentadas.

5 CONSIDERAÇÕES FINAIS

Os resultados mostrados na Seção 4 demonstram que os códigos desenvolvidos de geração e seguimento de trajetórias funcionam com uma precisão adequada para as aplicações para as quais foram direcionados, sendo essas as citadas na Seção 1.2. Em geral, o erro médio de trajetória não alcançou 1 mm, o que corresponde a menos de 0,8 % da largura de veículo considerada. Exceção a essa generalização foi o preenchimento de figuras cujo trajeto envolveu curvas segmentadas, no qual verificou-se erros de alguns milímetros, devido à maior dificuldade no segmento desse tipo específico de trajetória, descrita na Seção 3.3.3.

Além disso, percebe-se que, em alguns dos casos citados na Seção 4, o robô excedeu os limites estabelecidos em alguns milímetros. Esse problema está associado à constante que define o comprimento do lado dos quadrados correspondentes à cada posição da matriz discretizada. Como dito, o valor utilizado nos testes foi igual à largura simulada do veículo. Se essa variável fosse diminuída, a figura original poderia ter sido preenchida de forma mais precisa. Entretanto, essa alteração traria consigo um aumento no número de regiões percorridas mais de uma vez pelo veículo, efeito que se opõe às definições do objetivo do trabalho.

Como o modo de preenchimento das figuras escolhido foi em linhas paralelas a um dos lados das mesmas, o programa pode não funcionar da maneira prevista quando polígonos de largura muito baixa (menores que o dobro da distância de discretização) são definidos, ou seja, em situações em que todos os segmentos de trajeto gerados pelo programa são curvas. Entretanto, nesses casos basta que a ordem dos vértices seja alterada de modo que os movimentos em linha sejam paralelos a outro lado da figura, de modo que os diversos trajetos em curva se transformem em uma única reta.

Acredita-se que a construção de um robô móvel que atenda aos parâmetros citados nas Seções 2.3 e 2.4 e que seja carregado com os códigos descritos neste projeto levaria a resultados semelhantes aos encontrados em simulação. Um problema que pode ocorrer durante testes práticos é o escorregamento de rodas, ocasião na qual o giro de determinada roda não é convertido em movimento do

veículo. Para corrigir os erros de trajeto decorrentes disso é necessário o uso de um referencial inercial de posição, tal como o GPS diferencial.

6 REFERÊNCIAS

1 LAUMOND, Jean-Paul; SEKHAVAT, S.; LAMIRAUX, F. **Robot Motion Planning and Control**. Toulouse: Springer, 1998.

2 JONES, John. **A Robot to Help Make the Rounds**. 2003. Disponível em: <http://spinoff.nasa.gov/spinoff2003/hm_4.html>. Acesso em: 11 nov. 2014.

3 WALLIS, David. **Varied Duties, and Many Facets, in a Guard's Life**. The New York Times; Nova York, EUA; 20 mar. 2013. Disponível em: <http://www.nytimes.com/2013/03/21/arts/artsspecial/museum-guards-on-life-beyond-the-galleries.html>. Acesso em: 11 nov. 2014.

4 Adept Technology Inc. **Seekur**. 2011. Disponível em: <http://www.activrobots.com/ResearchRobots/Seekur.aspx/>. Acesso em: 11 nov. 2014.

5 The Material Handling Institute. **Automatic Guided Vehicles**. 2014. Disponível em: <http://www.mhi.org/fundamentals/automatic-guided-vehicles/>. Acesso em: 15 dez. 2014.

6 MONTEMERLO, Michael; THRUN, Sebastian. **FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics**. Stanford, USA: Springer, 2007.

7 SUJAN, Vivek A. et al. **A New Technique in Mobile Robot Simultaneous Localization and Mapping**. Revista Controle & Automação, vol. 17, no. 2, 2006.

8 MIRANDA, Fábio L. N. de; RIBEIRO, Carlos H. C. **Extração Automática de Mapas de Atributos Baseada em Técnica Bayesiana para Localização de Robôs Móveis**. Revista Controle & Automação, vol. 17, no. 4, 2006.

9 REKLEITIS, Ioannis M. **A Particle Filter Tutorial for Mobile Robot Localization**. Center for Intelligent Machines, McGill University. Montreal, Québec, Canadá, 2003.

10 THRUN, Sebastian. **Robotic Mapping: A Survey**. School of Computer Science, Carnegie Mellon University, Pittsburg, EUA, 2002.

11 ERICKSON, Lawrence H. et al. **Probabilistic Localization with a Blind Robot**. University of Illinois at Urbana-Champaign. Urbana, EUA, 2007.

12 NOURBAKHSH, Illah et al. **DERVISH: An Office-Navigating Robot**. AI Magazine, vol. 16, no. 2. 1995.

13 GILLULA, Jeremy; LEIBS, Jeremy. **How to Teach a Van to Drive: An Undergraduate Perspective on the 2005 DARPA Grand Challenge**. IEEE Control Systems Magazine, 2006.

14 STANLEY: The Robot that Won the DARPA Grand Challenge. Journal of Field Robotics, 2006.

15 DEVENZ, Gabriel. **Desenvolvimento de um Protótipo de Robô Cortador de Grama**. Monografia (Curso de Pós-Graduação em Desenvolvimento de Produtos Eletrônicos) - Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Florianópolis, 2012.

16 SANDI, Franz A. et al. **Sistema para Navegação e Guiagem de Robôs Móveis Autônomos**. SBA Controle & Automação, vol. 9, no. 3, 1998.

17 CAMPION, Guy; BASTIN, Georges; D'ANDRÉA-NOVEL, Brigitte. **Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots**. IEE Transactions on Robotics and Automation, vol. 12, no. 1. 1996.

18 RobotShop. **GM9 - Gear Motor 9 - 90 Degree Shaft**. 2015. Disponível em: <http://www.robotshop.com/en/solarbotics-gm9-gear-motor-9.html>. Acesso em: 29 jun. 2015.

19 BORENSTEIN, Johann. **The CLAPPER: A Dual-drive Mobile Robot With Internal Correction of Dead-reckoning Errors**. IEEE International Conference on Robotics and Automation, San Diego, USA, 1994.

20 AMER, S. I.; ESKANDER, M. N.; ZAKI, Aziza M. **Positioning and Motion Control for Mobile Robot**. International Journal of Emerging Technology and Advanced Engineering, vol. 2, 2012.

21 SIMÕES, Alexandre da Silva. **Aula 12 – Cinemática dos Robôs Móveis**. 2014. Disponível em: <http://www.gasi.sorocaba.unesp.br/assimoes/lectures/rm/Aula12%20-%20Cinematica%20dos%20robos%20moveis%20-%20cinematica.pdf>. Acesso em: 16 nov. 2014. Apresentação de slides.

22 FINLEY, Darel Rex. **Determining Whether A Point Is Inside A Complex Polygon**. Disponível em: <http://alienryderflex.com/polygon>. Acesso em: 01 mai. 2015.

23 Arduino. **Arduino Uno**. Descrição do Produto. 2015. Acesso em: <http://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 19 mai. 2015.

24 Aliexpress. **Chassi de Veículo 2WD**. 2014. Disponível em: <http://pt.aliexpress.com/item/5pcs-lot-Smart-car-chassis-2WD-V15-free-shipping/952336254.html>. Acesso em: 16 nov. 2014. Anúncio.

25 Vishay Semiconductors. **Reflective Optical Sensor with Transistor Output**. Folha de dados. Ago. 2009. Disponível em: <http://www.vishay.com/docs/83760/tcrt5000.pdf>. Acesso em: 19 mai. 2015.

26 STMicroelectronics. **Dual Full-Bridge Driver**. Folha de dados. Jan. 2000. Disponível em: <http://www.st.com/web/en/resource/technical/document/datasheet/CD00000240.pdf>. Acesso em: 20 mai. 2015.

27 Spark n' Go. **Ponte H L298N**. 2015. Disponível em: <http://www.sparkgo.com.br/ponte-h-l298n-rob-00005-p76>. Acesso em: 29 jun. 2015.

28 HobbyKing. **Turnigy 2200mAh 3S 30C Lipo Pack**. 2015. Disponível em: <http://www.hobbyking.com/hobbyking/store/__9394__Turnigy_2200mAh_3S_30C_Lipo_Pack.html>. Acesso em: 21 mai. 2015.

29 ITead Studio. **Arduino GPS Shield**. Folha de dados. Abr. 2012. Disponível em: <ftp://imall.iteadstudio.com/IM120417017_Arduino_GPS_shield/DS_IM120417017_ArduinoGPSshield.pdf>. Acesso em: 19 mai. 2015.

30 Guangzhou HC Information Technology Co. Ltd. **HC-06 Product Datasheet**. Folha de dados. 2011. Disponível em: <http://silabs.org.ua/bc4/hc06.pdf>. Acesso em: 06 mai. 2015.

APÊNDICE A – Pseudocódigo da Função Responsável pela Escolha da Melhor Trajetória

```
VARIÁVEIS GLOBAIS
inteiro: menorPeso, ultimoMov, posLFim
vetor inteiro: menorTrajeto

Procedimento encontraMelhorTrajetoria(matriz bool matrizTemp, vetor inteiro trajeto,
                                      inteiro posL, inteiro posC, inteiro orientL,
                                      inteiro orientC, inteiro pesoTotal)
        inteiro: posL0, posC0, orientL0, orientC0, ultMov0, nMovs;
        bool: fim;
        posL0<-posL,posC0<-posC,orientL0<-orientL,orientC0<-orientC,ultMov0<-ultimoMov;
        fim<-verdadeiro;
        nMovs <- 0;
        SE pesoTotal > menorPeso ENTÃO
                Sair retornando falso;
        FIM SE;
        Adicionar a posição atual no trajeto atual;
        Marcar a posição atual como visitada na matriz temporária;
        PARA mov<-ultimoMov ATÉ 2 FAÇA
                bool: mover;
                mover<-falso;
                SE mov=0 e for possível se mover reto ENTÃO
                        Incrementar a posL e posC de acordo com as orientações;
                        Acrescentar o peso relativo a um movimento reto;
                        mover<-verdadeiro;
                FIM SE;
                SE mov=1 e for possível se mover para a esquerda ENTÃO
                        Incrementar a posL e posC de acordo com as orientações;
                        Acrescentar o peso relativo a um movimento em curva;
                        Definir as novas orientações;
                        mover<-verdadeiro;
                FIM SE;
                SE mov=2 e for possível se mover para a direita ENTÃO
                        Incrementar a posL e posC de acordo com as orientações;
                        Acrescentar o peso relativo a um movimento em curva;
                        Definir as novas orientações;
                        mover<-verdadeiro;
                FIM SE;
                SE mover=verdadeiro ENTÃO
                        bool: proxFim;
                        proxFim <- falso;
                        nMovs <- nMovs+1;
                        SE ultimoMov=0 e mov=0 ENTÃO
                                ultimoMov <- 1;
                        SE NÃO
                                ultimoMov <- 0;
                        FIM SE;
                        Chamar a função novamente com as variáveis atualizadas. Retorno
                        é salvo na variável "proxFim";
                        Remover do peso total aquele referente ao último movimento;
                        posL<-posL0,posC<-posC0,orientL<-orientL0,orientC<-orientC0,
                        ultimoMov<-ultMov0;
                        fim<-falso;
                        SE posL!=posLFim ENTÃO
                                proxFim <- falso;
                        FIM SE;
                        SE proxFim=falso ENTÃO
                                mov <- 2;
                        FIM SE;
                FIM SE;
        FIM PARA;
```

```
        SE fim=verdadeiro ENTÃO
                Adicionar no peso total aquele referente às casas não visitadas;
                SE pesoTotal < menorPeso ENTÃO
                        menorPeso <- pesoTotal;
                        menorTrajeto <- trajeto;
                FIM SE;
                Remover a posição atual do trajeto atual;
                Marcar a posição atual como não visitada na matriz temporária;
                posLFim <- posL;
                Sair retornando verdadeiro;
        FIM SE;
        Remover a posição atual do trajeto atual;
        Marcar a posição atual como não visitada na matriz temporária;
        SE nMovs=1 ENTÃO
                Sair retornando proxFim;
        SE NÃO
                Sair retornando falso;
        FIM SE;
FIM Procedimento.
```

APÊNDICE B – Pseudocódigo da Função de Controle de Velocidade das Rodas

```
VARIÁVEIS GLOBAIS
inteiro: pwm1, pwm2;
real: w1Med, w2Med, w1Des, w2Des;

Procedimento controleMotores()
        real: deltaW1, deltaW2;
        deltaW1 <- w1Med-w1Des, deltaW2 <- w2Med-w2Des;
        pwm1 <- pwm1-5,0*deltaW1, pwm2 <- pwm2-5,0*deltaW2;
        SE pwm1 < 0 ENTÃO
                pwm1 <- 0;
        FIM SE;
        SE pwm1 > 86,9 ENTÃO
                pwm1 <- 86,9;
        FIM SE;
        SE pwm2 < 0 ENTÃO
                pwm2 <- 0;
        FIM SE;
        SE pwm2 > 86,9 ENTÃO
                pwm2 <- 86,9;
        FIM SE;
FIM Procedimento.
```

APÊNDICE C – Pseudocódigo da Função de Controle de Seguimento de Trajetórias

```
VARIÁVEIS GLOBAIS
real: erroAnt, w1, w2;

Procedimento controleTrajetoria(real kps, real kds)
        real: novoW, erro;
        Chamar a função que estima o erro de trajetória atual;
        novoW <- -kps*erro-kds*(erro-erroAnt);
        erroAnt <- erro;
        SE novoW > 4,28 ENTÃO
                novoW <- 4,28;
        FIM SE;
        SE novoW < -4,28 ENTÃO
                novoW <- -4,28;
        FIM SE;
        Calcular os valores desejados de w1 e w2 para que novoW seja atingido;
FIM Procedimento.
```

APÊNDICE D – Robô Móvel Construído

D.1 INTRODUÇÃO

Visando demonstrar algumas das funcionalidades descritas neste trabalho, construiu-se um robô móvel que poderia servir como plataforma de teste para os códigos desenvolvidos. A versão final desse veículo está ilustrada na Figura D.1.

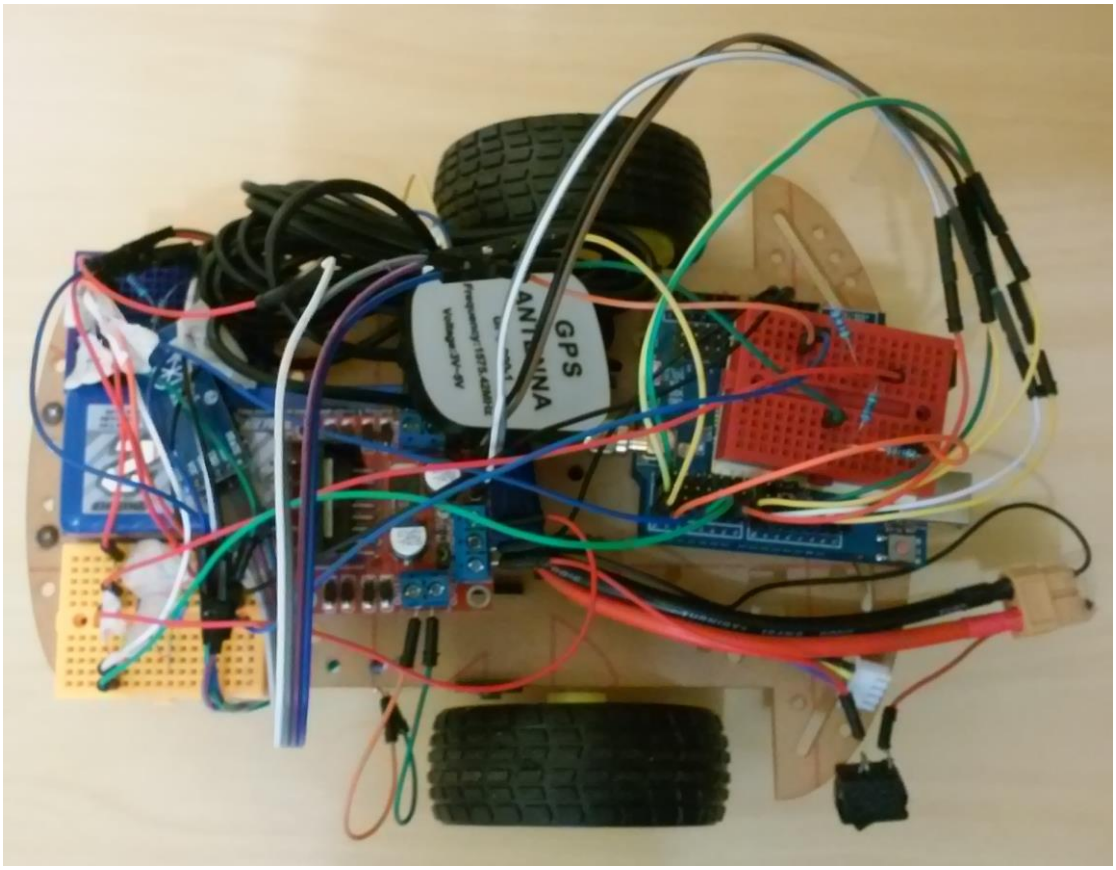

Figura D.1 - Fotografia da versão final do robô utilizado.

D.2 MATERIAIS

As principais partes constituintes do robô construído são as seguintes: placa de desenvolvimento “Arduino Uno”, chassi de veículo, dois *encoders*, duas rodas refletoras anexas aos motores, ponte H, bateria e módulos GPS e *Bluetooth*.

A placa de Desenvolvimento “Arduino” é um componente onde o processador que toma as decisões está presente. Esse é o “cérebro” do projeto, no qual pode ser embarcado o *software* gerador de trajetórias e onde são lidos e interpretados os sinais recebidos de todos os sensores do veículo. A placa utilizada é do modelo “Arduino Uno” e possui um microcontrolador *ATmega328*, além de 14 portas de entrada e saída digitais (sendo 6 do tipo PWM), 6 portas analógicas e conexão USB[23]. Uma fotografia desse conjunto de dispositivos está representada na Figura D.2.

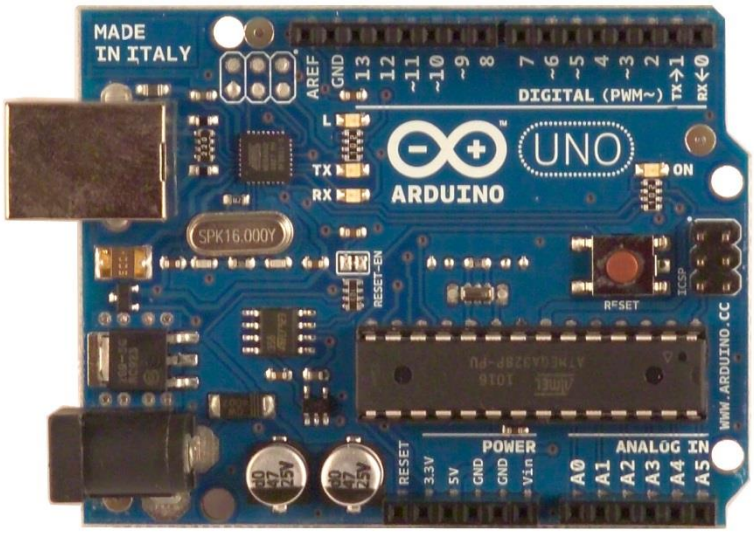

Figura D.2 - Placa de desenvolvimento “Arduino Uno”.
Fonte: Arduino (2015).

Já o chassi de veículo consiste na carcaça do robô, na qual são posicionados e fixados todos os demais equipamentos. A configuração de chassi escolhida foi a mesma sugerida na Seção 2.3 e ilustrada na Figura D.3. As duas rodas controláveis possuem anexas estruturas circulares refletoras, de modo que o número de rotações possa ser contabilizado. O raio das rodas do modelo utilizado é de 3,2 cm, enquanto a distância entre as mesmas foi estimada em 13 cm.

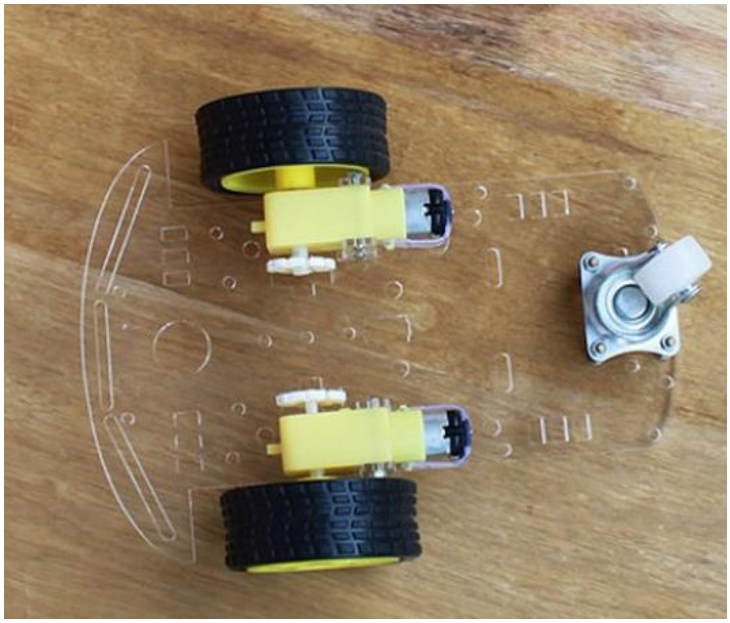

Figura D.3 - Chassi de veículo utilizado.
Fonte: Aliexpress (2014).

Os *encoders* são sensores ópticos refletivos, cuja resposta é uma função da luz incidente neles. Esses dispositivos são instalados um em cada roda. Eles, juntamente com a roda refletora anexa a cada roda do carro, possibilitam a estimação da sua velocidade. O modelo de sensor escolhido foi o *TCRT5000*, composto de um *led* emissor de infravermelho (cor azul) e um transistor fotossensível (cor preta). Cada vez que o feixe de luz é refletido de volta ao dispositivo, o transistor é ativado e a tensão no seu terminal coletor é alterada[25]. Foi necessário projetar um pequeno circuito que garantisse uma tensão de alimentação adequada para as duas partes constituintes do sensor. O *encoder* utilizado e o circuito projetado para possibilitar o seu funcionamento estão ilustrados, respectivamente, nas Figuras D.4 e D.5. Os sensores foram fixados na parte debaixo do veículo, através de uma estrutura de arame.

Figura D.4 - Modelo de *encoder* utilizado.
Fonte: Vishay Semiconductors (2009).

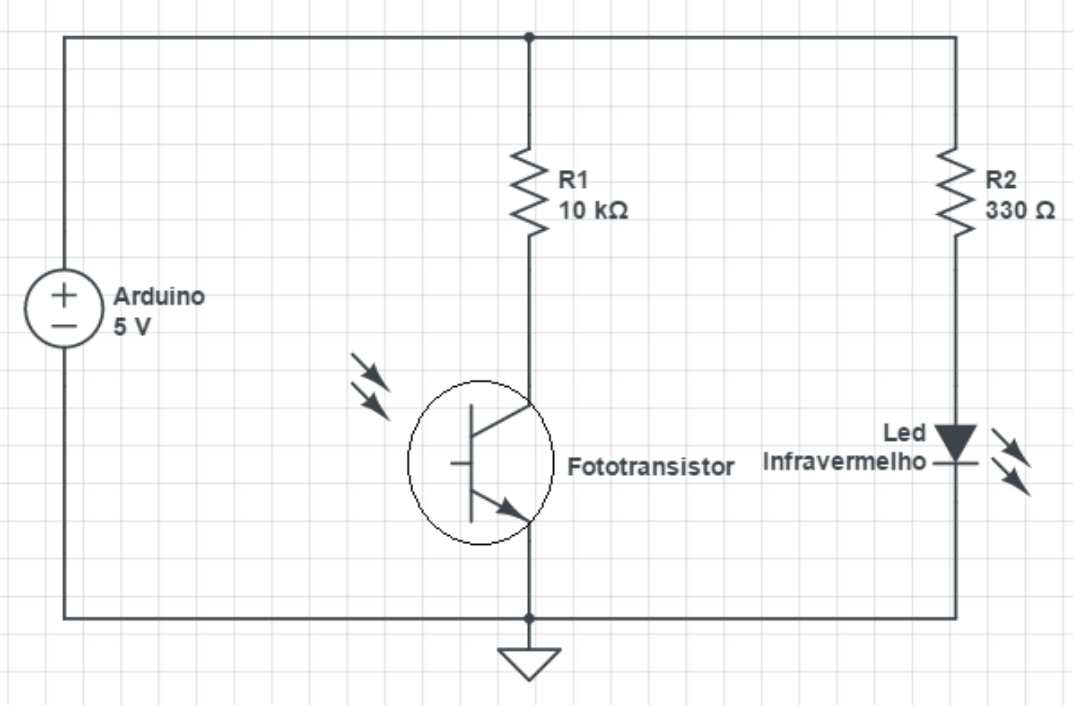

Figura D.5 - Circuito de alimentação de cada um dos *encoders*.

As rodas dentadas são utilizadas em conjunto aos *encoders* e estão esquematizadas na Figura D.6. Um dispositivo desse tipo foi fornecido junto do chassi de veículo comprado, o qual era dividido em 40 partes iguais. Dessa forma, seria possível aferir a velocidade da roda até 40 vezes por volta efetuada. Entretanto, após uma série de experimentos, notou-se que a largura dos buracos da engrenagem era pequena demais para promover alterações no sinal fornecido pelos sensores. Assim, foi necessário alterar a roda refletora para a situação mostrada na Figura D.6, a qual possibilita leituras numa frequência de até 4 vezes por rotação. As superfícies

refletoras são constituídas de lâminas de alumínio que foram fixadas à engrenagem, enquanto as não refletoras são de plástico pintado de cor preta.

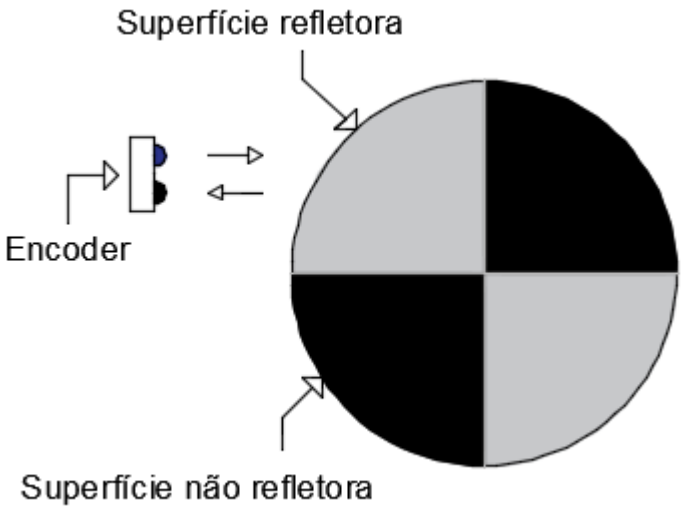

Figura D.6 - Esquema da roda refletora construída.

A ponte H é um circuito elétrico que garante o fornecimento da corrente exigida pelos motores e também a integridade das portas de saída da placa de desenvolvimento utilizada. O modelo utilizado foi o L298N, que possui também um regulador de tensão integrado, disponibilizando uma saída fixa de 5 V para uma alimentação de 6 a 35 V[26]. Esse terminal é utilizado para alimentar o “Arduino”, os módulos e os *encoders*. Através dessa Ponte H são controladas as velocidades dos dois motores do veículo. A tensão fornecida a cada um deles é proporcional à intensidade do sinal PWM proveniente da placa de desenvolvimento. A Figura D.7 ilustra o dispositivo citado.

Figura D.7 - Ponte H com regulador de tensão.
Fonte: Spark n' Go (2015).

A bateria é um dispositivo que energiza o veículo e dispensa a necessidade de fios até tomadas elétricas. O modelo escolhido foi da marca “Turnigy”, composto de 3 células e capaz de fornecer energia durante 1 hora a uma carga que exija corrente de 2,2 A. Suporta picos de corrente de até 40 vezes seu valor nominal por um tempo máximo de 10 segundos. A tensão disponibilizada por ela é de 11,1 V[28]. A Figura D.8 ilustra o modelo adquirido para este projeto.

Figura D.8 - Modelo de bateria embarcado no veículo.
Fonte: HobbyKing (2015).

O módulo GPS é um *kit* composto de módulo e antena, responsável pela conexão com satélites artificiais e obtenção da sua localização atual, expressa por coordenadas geográficas. Os valores de latitude e longitude recebidos são lidos e interpretados pelo processador, de modo que o sentido e velocidade de movimento seguintes possam ser definidos. O módulo utilizado foi o “RoyalTek REB-4216S4”, embutido em uma placa “GPS Shield 1.1”, a qual é projetada para trabalhar em conjunto com o “Arduino”. As informações recebidas de satélite são repassadas no formato NMEA[29]. A Figura D.9 ilustra o *kit* GPS utilizado. O módulo GPS pode ser usado tanto na definição dos vértices da figura como no controle de seguimento de trajetórias.

Figura D.9 - *Kit* de módulo GPS e antena.
Fonte: ITead Studio (2012).

O módulo *Bluetooth* utilizado foi o “JY-MCU HC-06”, operando em modo *slave* e com um limite de distância de transmissão de dados de até 10 metros, perfeitamente adequado para esta aplicação. A sua inclusão no projeto foi feita planejando-se que a definição dos vértices da figura a ser percorrida pudesse ser feita por meio de um aplicativo para o sistema operacional “Android”, que se comunicaria com o robô via *Bluetooth*. Ele necessita de uma tensão de alimentação entre 3,6 V e 6 V, e o nível de sinal enviado é de 3,3 V[30]. A Figura D.10 ilustra um exemplar desse modelo. Como o nível de sinal utilizado por cada um dos dispositivos envolvidos é diferente (5 V no “Arduino” e 3,3 V no módulo) foi necessário aplicar um divisor de tensão que diminuísse o maior desses valores. Sem o uso desse recurso, as portas projetadas

para suportarem uma tensão de 3,3 V poderiam ser danificadas. O modelo de “Arduino” empregado possui duas portas digitais propícias para comunicação serial, as quais foram utilizadas na ligação com o módulo *Bluetooth*. A Figura D.11 mostra o esquema de ligação das portas de recepção (RX) e de transmissão (TX) de cada aparelho. Esses pinos são ligados invertidos propositalmente, de modo que o terminal de transmissão da placa de desenvolvimento esteja conectado ao de recepção do módulo, e vice-versa. Como a conversão de tensão necessária é de 5 V para 3,3 V, a razão entre os dois valores de resistência da Figura D.11 deve ser de, aproximadamente, 1,52. Os resistores utilizados neste caso são de 2,2 kΩ e 1,5 kΩ, satisfazendo, portanto, esse requisito.

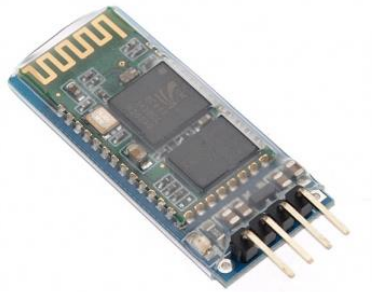

Figura D.10 - Módulo *Bluetooth* utilizado.
Fonte: Guangzhou HC Information Technology (2011).

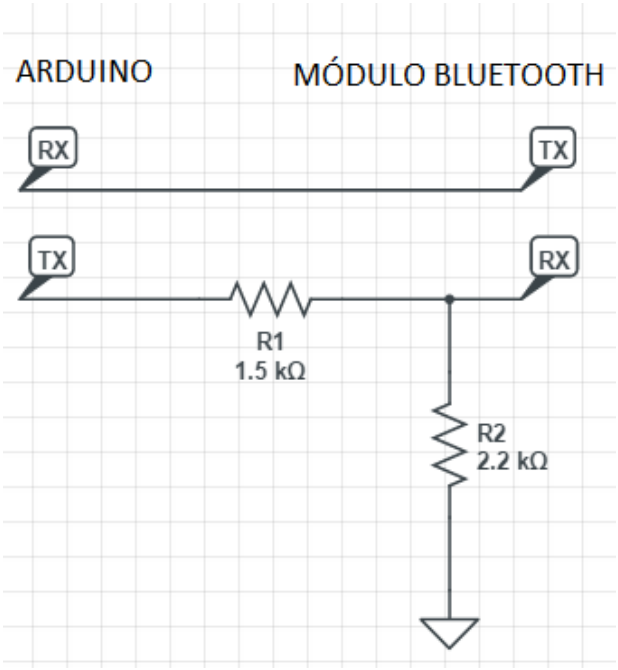

Figura D.11 - Esquema de ligação entre as portas do “Arduino” e do módulo *Bluetooth*.