CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ESTUDO DE ESTRATÉGIAS DE CONTROLE DE FORMAÇÃO DE ROBÔS COM FOCO EM TOLERÂNCIA A FALHAS

MARIANA ATHAYDE GARCIA

Orientador: Prof. Tales Argolo Jesus
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG

Coorientador: Prof. Anolan Yamilé Milanés Barrientos
Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG

BELO HORIZONTE
MARÇO DE 2015

Lista de Figuras

Lista de Quadros

Lista de Abreviaturas e Siglas

CEFET-MG	Centro Federal de Educação Tecnológica de Minas Gerais
DECOM	Departamento de Computação
P	Controlador Proporcional
PI	Controlador Proporcional Integral
PID	Controlador Proporcional Integral Derivativo

Lista de Símbolos

R	Raio de distância do alvo
θ	Sentido no plano cartesiano
v	Velocidade linear
ω	Velocidade angular
(x_a,y_a)	Coordenadas do alvo no plano cartesiano
ω_d	Velocidade angular desejada
(x_r,y_r)	Coordenadas do robô no plano cartesiano
(x_d,y_d)	Coordenadas desejadas do robô no plano cartesiano
e_r	Vetor de erro de posição
r_d	Vetor de distância da origem do plano cartesiano ao ponto desejado
r_r	Vetor de distância da origem do plano cartesiano ao robô
e_x	Erro de posição no eixo x
e_y	Erro de posição no eixo y
θ_d	Sentido desejado no plano cartesiano
θ_r	Sentido real do robô no plano cartesiano
e_θ	Sentido no plano cartesiano
T	Período de rotação ao redor do alvo
ω_c	Velocidade angular passada para primeira malha de controle
ω_r	Velocidade angular real do robô
ω_{dr}	Velocidade angular desejada da roda direita
ω_{dl}	Velocidade angular desejada da roda esquerda
ω_{rr}	Velocidade angular real da roda direita
ω_{rl}	Velocidade angular real da roda esquerda
e_{wr}	Erro de velocidade angular da roda direita

e_{wl}	Erro de velocidade angular da roda esquerda
pwm_r	Potência da roda direita
pwm_l	Potência da roda esquerda

Sumário

1 Introdução

Atualmente, é cada vez mais frequente a participação de robôs na nossa sociedade, desde em seguimentos da indústria, onde esses robôs vêm se mostrando uma solução tanto econômica quanto eficiente, como também em salas cirúrgicas e no nosso cotidiano, na busca de facilitar ainda mais as tarefas (CHEN et al., 2002; CARLES; HERMOSILLA, 2008). Entretanto, existem situações em que a utilização de um único robô é uma solução um tanto quanto lenta e muitas vezes inviável. Como exemplo de uma dessas situações pode-se citar o problema de patrulhamento de fronteira (CORRÊA; JÚNIOR, 2008) : Para proteção das fronteiras de um país, manter diversas patrulhas de policiais circundando a área se torna muitas vezes caro e ineficiente. Uma alternativa é alocar um veículo aéreo não tripulado (*VANT*) vigiando essas fronteiras, entretanto, apenas um *VANT* como vigia deixará uma grande área da fronteira desprotegida por um longo período de tempo. E é por isso que a aplicação de um conjunto de robôs, cooperando entre si, se mostra muitas vezes interessante.

Dentro deste panorama, surge o interesse cada vez mais crescente pelo estudo, não só da robótica, mas também de um segmento mais específico da área da inteligência artificial distribuída, que é o estudo de sistemas multiagentes, que consiste em agentes autônomos que percebem a ação do ambiente e agem de acordo com a percepção da rede de agentes. Ou, segundo os autores Ramchurn et al. (2004, p. 1), "[...]sistemas multiagente são sistemas compostos de agentes autônomos que interagem entre si usando determinados mecanismos e protocolos".

De acordo com Secchi (2008) , "a robótica sempre ofereceu ao setor industrial um excelente compromisso entre produtividade e flexibilidade, uma qualidade uniforme dos produtos e uma sistematização dos processos". Mas, mais importante que maximizar a lucratividade das indústrias, a qualidade dos produtos e facilitar cada vez mais as tarefas cotidianas, os robôs permitem resguardar a vida humana, substituindo seres humanos em situações de risco. Exemplos de aplicação incluem: exploração e mapeamento de áreas desconhecidas, situações de incêndio, onde grupos de pessoas precisam apagar o fogo expondo suas vidas a um risco ou até mesmo em missões de resgate em terrenos perigosos. Daí a importância de que o sistema seja tolerante à falha de um ou mais agentes. Afinal, é de extrema importância que o objetivo seja cumprido.

1.1 Relevância do tema

Hoje em dia, há tarefas que são realizadas em diversas áreas nas quais a presença ou o envolvimento direto de pessoas é algo perigoso, ou até mesmo inviável. Sendo

assim, é crescente a necessidade de se estudar outros meios de acesso a essas situações de risco, sem que isso signifique um risco à vida humana. Diante dessa problemática, o estudo de estratégias de controle de robôs móveis vêm aumentando consideravelmente. Não só para problemas que colocam em risco a vida humana, mas também problemas onde a aplicação dos robôs móveis otimizaria o tempo e eficiência da resolução destes problemas. Dentre estes problemas mencionados pode-se citar (GIRARD et al., 2004; JESUS, 2013; MARJOVI et al., 2009) : o patrulhamento de fronteiras, o controle de incêndio, mapeamento de áreas desconhecidas, busca de pessoas perdidas ou detecção e monitoramento de problemas em determinado alvo, dentre outros.

Como é possível perceber são inúmeras as possibilidades de aplicação dos robôs móveis. Entretanto, devido muitas vezes à urgência e/ou à extensão da cobertura do problema é necessário modular o mesmo e redistribuí-lo entre um sistema multiagente de robôs móveis. Sendo assim, surge aí mais uma demanda por estudos relativos a estratégias de controle de sistemas multiagente constituídos de robôs móveis. Um dos desafios destes sistemas multiagentes não é só o controle de cada agente por si só, mas também como a frota irá se comportar como um todo, para viabilizar a resolução do problema e/ou também maximizar a eficiência na resolução do mesmo. É necessário que se garanta que os robôs não colidam entre si, e trabalhem em um sistema cooperativo de fato, e não atuando individualmente, anulando a vantagem da frota, como se essa fosse constituída de apenas um robô.

Outra questão importante, que inclusive é o foco do tema deste trabalho, é a tolerância a falhas do sistema, isto é, como o sistema irá se comportar, se reestruturar e reorganizar diante da perda de um ou mais robôs, visto que além de ser um ambiente hostil (muitas vezes desconhecido ou até dinâmico), existem outros fatores críticos, dentre eles: falha de comunicação, ou desligamento de um dos agentes devido ao esgotamento de bateria.

1.2 Objetivos

Este trabalho tem como objetivo o estudo de estratégias de controle de formação de uma frota de robôs e seu comportamento em relação à sua estrutura e ao problema, ao se deparar com falhas de um ou mais robôs, bem como a implementação deste sistema multiagente em uma plataforma experimental.

1.3 Infraestrutura Necessária

Para a realização deste trabalho foram utilizados quatro kits da plataforma da *LEGO*: *Lego Mindstorms*, disponível do DECOM (Departamento de Computação do

Centro Federal Tecnológico de Minas Gerais). Cada kit consiste em um microcomputador NXT de 32 bits, três motores, alguns sensores e peças de lego para montagem da estrutura do robô. Além disto, também será utilizado um computador pessoal, com a seguinte configuração: processador *intel core i7*, 8GB de memória *RAM*, 1GB de mémória dedicada e 14", com o *software MATLAB* e a *IDE Bricx Command Center* (SOURCEFORGE, 2001) que é uma plataforma de desenvolvimento para robôs *LEGO Mindstorms* que permite utilizar a linguagem *NXC (Not eXactly C)* para programar os robôs.

2 Trabalhos Relacionados

Com o interesse cada vez mais crescente na área de robótica móvel e sistemas multiagentes, tem uma demanda cada vez maior para estudos nestas áreas. O *Lego®Mindstorms* não é uma plataforma muito interessante de aplicação desses conceitos , entretanto, é uma excelente plataforma a ser utilizada nos estudos dos mesmos. Isto por que, é uma plataforma acessível, existe muita documentação auxiliar, muitos trabalhos relacionados a respeito e é muito simples de ser utilizada. Logo, vê-se muitos trabalhos relacionados a este, que envolvem o estudo da robótica móvel e de sistemas multiagentes.

Existem muitos trabalhos distintos com sistemas multiagentes utilizando-se *Lego®Mindstorms*, com as mais diversas configurações, objetivos distintos, e diferentes estruturas de rede, dentre outros. Entretanto, uma dificuldade reconhecida em todos os trabalhos são as limitações da plataforma, que possui uma quantidade de conexões e tipo de comunicação muito limitada. Para que essa limitação seja superada perde-se uma característica importante da plataforma, que é a simplicidade e facilidade de implementação. O protocolo de comunicação disponível só permite a comunicação *'Master/Slave'* realizada de forma manual.

Outras configurações, requerem uma implementação mais complexa que afeta o custo/benefício de se utilizar essa plataforma, por perder a característica de implementação simples. Pode-se citar como um desses trabalhos, que abordam de forma mais dinâmica e independente a comunicação entre os robôs, Martinez et al. (2009). Seu trabalho consiste em uma sociedade que se configura de forma autônoma, ou seja, indivíduos independentes que se agrupam e formam uma sociedade.

Entre os trabalhos relacionados a este podemos citar, Benedettelli et al. (2009) que propõe uma configuração experimental para utilizar o *Lego®Mindstorms* como ferramenta de estudos de estratégias de controle de sistemas multiagentes. Ele utiliza uma frota composta por quatro robôs, uma *webcam* e o *MATLAB®*. Com o intuito de se obter uma ferramenta de baixo custo para dar aulas de laboratório de robótica, seu trabalho consiste em propor uma configuração que permita a implementação, comparação e o estudo de diversas estratégias de controle e algoritmos. Uma configuração que contempla muitos dos problemas vistos em um cenário real. Utilizando-se de uma unidade central de controle, ele primeiro propõe quatro robôs em pontos diferentes do espaço, orientados em qualquer sentido à rodear um ponto qualquer no espaço, com auxilio da *webcam* e do computador (unidade central de comando). O que se aproxima bastante deste trabalho, diferindo-se principalmente no que diz respeito à *webcam* como sensor de alimentação do sistema.

Outro trabalho também muito interessante que pode-se citar é o do Casini et al. (2011), ele propõe um laboratório remoto utilizando o *LEGO®Mindstorms*. O trabalho se resume a um laboratório remoto para estudos de robótica móvel, em que tem-se um espaço de cerca de 13 metros quadrados que é filmado por duas câmeras, onde os robôs ficam e podem se movimentar. Foi então desenvolvida uma interface gráfica de acesso online ao laboratório, através da qual os usuários podem acessar e utilizar o laboratório para o estudo de robótica móvel.

3 Metodologia

Para a realização deste trabalho, primeiramente será realizado um estudo das estratégias de controle de formação de robôs móveis, das possibilidades de implementação dessas estratégias na plataforma a ser utilizada (no caso, o *LEGO Mindstorms*) e da viabilidade de modelar e implementar o sistema como um sistema distribuído descentralizado.

Após realizados os estudos, será demostrado uma modelagem matemática do problema, na qual serão definidas as restrições da modelagem, bem como as limitações da plataforma, tendo em vista um levantamento das dificuldades que certamente surgirão somente na implementação do sistema no mundo real. A solução para o problema será modelada de forma a viabilizar a correção de erros de sensoriamento que se acumulam a cada iteração.

Para facilitar a implementação, a modelagem será implementada em módulos de controle, com o intuito de simplificar o entendimento e a validação do sistema. A implementação será feita da malha mais interna à malha mais externa, sendo testado e validado módulo à modulo de controle, bem como suas integrações.

Posteriormente, serão realizadas simulações, através da ferramenta *MATLAB*, à fim de comparar o modelo real com o modelo idealizado do sistema.

3.1 Fundamentação Teórica

Será feito nesse capítulo, um levantamento das diversas estratégias de controle de formação, e das diversas formas de se planejar a rede. Será analisada a viabilidade de implementação dessas estratégias na plataforma *LEGO Mindstorms*, visto que esta se trata de uma ferramenta de baixo custo usada para fins didáticos, por isso, apresenta muitas limitações.

Além disso, apresenta-se no mesmo, outros conceitos que acredita-se essencial para a inserção do leitor no entendimento do texto, tais como: o que é um modelo matemático não holonômico, o que é controle de formação e faz-se um pequeno levantamento de *softwares* disponíveis para implementação do modelo pretendido.

3.2 Modelagem matemática

De acordo com a abordagem escolhida na etapa anterior será feita a modelagem matemática do problema, onde serão considerados, dentre outros fatores: as variáveis

existentes no mundo real e principalmente as restrições do problema.

Na modelagem matemática será definida a abordagem do problema, se a estrutura da rede será centralizada ou não, como será implementado o *feedback* (realimentação) para correção de erros que ocorrerão devido à imprecisão dos encoders óticos acoplados aos motores do kit *LEGO Mindstorms*, que é da ordem de ± 1 grau por rotação. É nesta etapa que também será definido o melhor formato de posicionamento dos robôs e se será possível implementar mais de um formato.

Outro fator importante desta etapa é que o sistema será modelado de acordo com os devidos parâmetros, para que o mesmo seja tolerante a falhas e para que sua estrutura varie, otimizando a resolução do problema, de acordo com as falhas que possam vir a surgir em um ou mais pontos da rede.

3.3 Implementação na plataforma experimental

Durante a pesquisa, inicia-se a fase de implementação não só da estrutura dos robôs, como também da estrutura do ambiente real e o desenvolvimento do código-fonte do sistema multiagente.

Conforme o problema vai sendo modelado, será implementado na plataforma, a fim de se obter dados que indiquem se a modelagem escolhida, juntamente com os artifícios utilizados, foram suficientes como solução para o problema. Outros métodos serão estudados quando os resultados não forem satisfatórios ou quando a melhoria dos resultados apresentar um custo/benefício razoável, e então será realizado uma análise comparativa para se escolher o método mais adequado.

3.4 Simulações computacionais

Após a implementação do sistema serão feitas simulações em ambiente virtual controlado com as mesmas estratégias de controle implementadas na plataforma, com o intuito de se realizar uma comparação entre ambas. Para tanto, será utilizada como ferramental o *software MATLAB*.

3.5 Análise de Resultados

Nas etapas de simulação e implementação do sistema, após ele ser validado e testado, será feita a coleta dos dados que serão submetidos a análise. Para comparar ambos os resultados será coletado em tempo de execução os dados necessários para descrever a trajetória do(s) robô(s) e os mesmos serão plotados no *MATLAB*.

3.6 Conclusão do trabalho

Pretende-se nesta etapa, a partir da análise de dados feita na etapa anterior, comparar as estratégias utilizadas e contextualizar os resultados obtidos em um problema real.

4 Cronograma

Para execução do trabalho as seguintes etapas serão seguidas:

1. Revisão bibliográfica das estratégias de controle e tecnologias necessárias para elaboração deste trabalho.
2. Modelagem Matemática.
3. Implementação na plataforma experimental.
4. Escrita do TCC I.
5. Simulações Computacionais.
6. Análise de resultados.
7. Conclusão do trabalho.
8. Escrita do TCC II.
9. Apresentação do trabalho.

Quadro 1 – Cronograma de Desenvolvimento do Projeto

Cronograma										
Tarefa	Mês									
	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ
1	X	X	X	X	X	X	X			
2		X	X	X	X					
3			X	X	X	X	X			
4		X	X	X	X					
5					X	X				
6						X	X			
7							X	X		
8						X	X	X	X	
9										X

5 Fundamentação Teórica

Para que se possa compreender tal trabalho é importante, primeiramente, conhecer alguns conceitos. Para tanto, faz-se neste capítulo uma breve contextualização teórica para auxiliar o leitor ao longo deste trabalho.

5.1 Modelos Matemáticos: Sistemas Não Holonômicos

Os robôs utilizados neste trabalho são considerados modelos não holonômicos e para entender o que é um modelo matemático não holonômico é necessário entender o que é o grau de liberdade de um sistema. Como é visto na literatura (TAYLOR et al., 2013), o grau de liberdade de um sistema é igual ao "número de coordenadas que podem variar independentemente em um pequeno deslocamento". Dito isto pode-se dizer que um sistema holonômico é um sistema onde o número de coordenadas utilizadas para descrever as configurações do sistema é igual ao grau de liberdade do sistema (TAYLOR et al., 2013). Ou seja, o sistema pode se movimentar livre e independentemente em qualquer um dos seu eixos (os eixos referentes à configuração do sistema).

Sendo assim, os sistemas não-holonômicos são sistemas em que pode-se chegar à qualquer outro ponto do espaço, entretanto, com restrições. Visto que as variáveis não podem se mover independentemente. Como exemplo de um sistema não-holonômico podemos citar um veículo, que pode alcançar qualquer ponto do espaço bidimensional, entretanto, para alcançar um ponto qualquer deslocado apenas em seu eixo x é necessário um movimento não só ao longo do seu eixo x mas, também do seu eixo y. Já que, um veículo não pode se mover lateralmente (GOUVÊA, 2011).

5.2 Controle de Formação

Com o crescente avanço da robótica, começou-se o interesse por sistemas robóticos cooperativos, onde muitos robôs agem em conjunto para alcançar o mesmo objetivo. Para que um sistema multiagente possa executar uma tarefa em conjunto é preciso que cada robô esteja na posição correta, para tal, é necessário o controle de formação. O controle de formação é essencial para sistemas robóticos multiagentes pois permite que cada robô esteja em seu devido lugar no momento certo. Existem diversos tipos de estruturas de formação, tanto no que diz respeito a formação física da rede, como do ponto de vista lógico da rede. Como é visto na literatura (MARTINEZ et al., 2009), pode-se classificar uma rede de multiagentes de diversas formas: Homogênea ou não, no que diz respeito aos tipos de unidades, no que diz respeito à estrutura da rede como

indivíduo, se a rede é formada por indivíduos independentes ou se é o mesmo robô, quanto à estrutura organizacional, dentre outros.

Do ponto de vista físico podemos citar alguns tipos de estruturas de formação, como referenciado em Balch e Arkin (1998), tais como, estrutura em*line*, *column*, *diamond* e *wedge*. A primeira delas consiste em uma formação em linha horizontal (*line*)como, o próprio nome revela. A segunda, em uma linha vertical (*column*), a terceira *diamond*, que consiste em uma rede em formato de um losango e a quarta, *wedge* em formato de *'V'*, o que se parece com a estrutura de um *flock*.

Figura 1 – Controle de Formação: classificação quanto à estrutura física

(a) *Line*,(b) *Column*,(c) *Diamond* e (d) *Wedge*

Fonte: Balch e Arkin (1998)

Além disso tem a classificação quanto a estrutura da rede lógica, centralizada, descentralizada e híbrida e quanto a seu grupo de arquitetura, no caso deste trabalho, a classificação seria móvel.(MARTINEZ et al., 2009). Outras classificações interessantes que serão abordadas posteriormente são, as técnicas de referência para controle de formação.

5.3 Controle Proporcional Integral Derivativo e a Técnica de Controle em Cascata

O controlador Proporcional Integral Derivativo, ou simplesmente, *PID* consiste em uma técnica com ações proporcionais, integrais e derivativas. A ação proporcional (*P*), consiste em minimizar o erro, enquanto a ação integral (*I*) tende a zerar este erro e a ação derivativa (*D*) tende à reduzir o tempo de resposta. A equação que o modela está indicada pela Equação (1). Ou seja, a ação proporcional consiste em multiplicar o erro pelo ganho, minimizando o erro, a ação integral consiste em multiplicar pelo ganho a soma dos erros durante a execução do sistema e por fim, a ação derivativa que consiste em multiplicar o ganho pela derivada do erro, tentando assim, antecipar a resposta do sistema.

$$ct = k_p e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \tag{1}$$

Uma técnica de controle importante que também foi utilizada neste trabalho, foi a técnica conhecida como controle em cascata, esta técnica consiste em passar como referência para o controlador interno, a saída do controlador mais externo, que possui uma referência independente, e a saída do sistema como um todo, é a saída do controlador mais interno. Como será visto mais a frente, é o caso deste trabalho.

5.4 Plataformas

Para o desenvolvimento deste trabalho foram consideradas diversas plataformas de desenvolvimento e gerênciamento de robôs móveis que são compatíveis com *Lego MindStorms*. Abaixo serão citados algumas dessas plataformas com suas características.

5.4.1 ROS

ROS(*Robotic Operating System*) é um sistema operacional robótico opensource que dispões de ferramentas e bibliotecas desenvolvidas para criar aplicações robóticas.Ele permite a comunicação entre o computador e o *lego*, permite a implementação de uma rede centralizada de até quatro robôs, embora, possua funções específicas para área de robótica esse sistema ainda não dispõe de muitos materiais para pesquisa, por isso, optou-se por não utilizá-lo.

5.4.2 NXT-G

É uma plataforma gráfica que vem com o próprio kit *Lego Mindstorms*, até mesmo pela sua natureza gráfica, ela é muito simples. Entretanto, não permite a comunicação com o computador em tempo de execução. Devido a sua limitação e simplicidade, optou-se por não utilizá-lo.

5.4.3 Simulink

O *Simulink* possui um pacote compatível com o *Lego Mindstorms* que permite desenvolver e simular algoritmos para plataformas robóticas. Entretanto, só é permitido a comunicação via *bluetooth* entre dois robôs, portanto não atende às demandas requeridas por este trabalho.

5.4.4 LABView

O *LABVIEW* possui um módulo para programar e controlar robôs *Lego Mindstorms*. É uma ferramenta que permite a comunicação entre o robô e o computador e entre os robôs. Entretanto, é necessário possuir uma licença para utilizar desta ferramenta, por isso, a ferramenta não será utilizada neste trabalho.

5.4.5 RWTH Aachen MINDSTORMS NXT Toolbox

O *MATLAB* possui uma ferramenta opensource desenvolvida para controlar robôs *Lego Mindstorms NXT*, conhecida como: *RWTH Aachen NXT Toolbox*. Permite a comunicação entre robô e o computador ou entre um conjunto de até 4 robôs, no modelo de comunicação mestre/escravo.

5.4.6 BRICX Command center

O ambiente de desenvolvimento integrado conhecido como *BRICX Command Center* é utilizado para o desenvolvimento de aplicações para todas as versões do *Lego MindStorms*, do *RCX* ao *EV3*, incluindo o modelo utilizado neste trabalho que é o *NXT*. Suporta diversas linguagens como: *Not eXactly C* (NXC), *Next Byte Codes* (NBC) e permite o desenvolvimento em *java*, usando o *firmware LeJos*.

6 Abordagem e Modelagem do Problema

Existem diversas maneiras de se implementar um sistema como este, tanto do ponto de vista do sistema distribuído e sua rede de comunicação, quanto do ponto de vista de controle e realimentação das malhas. Neste capítulo faz-se um detalhamento da abordagem do problema e do funcionamento do sistema como um todo e a modelagem escolhida para abordar o problema. Será utilizado uma abordagem já mencionada anteriormente, que é o controle em cascata, utilizado para modularizar o problema e assim, tornar mais simples a implementação e o entendimento do mesmo.

Serão três malhas de controle: A primeira e mais interna será responsável pelo controle da velocidade angular de cada roda, para se chegar à posição (x,y) desejada. Esta malha estará presente em cada um dos robôs da frota que estarão à circular o alvo. A segunda, malha intermediária, será responsável para que cada robô chegue à um determinado ponto (x,y) no espaço, portanto, será responsável por corrigir o posicionamento do robô. Esta malha também estará presente em cada robô que circundar o alvo. A terceira malha, e portanto, a mais externa é responsável pela coordenação da frota, fornecendo a cada robô as informações necessárias para que o mesmo saiba o ponto (x,y), onde deve ficar para consolidar e manter a formação.

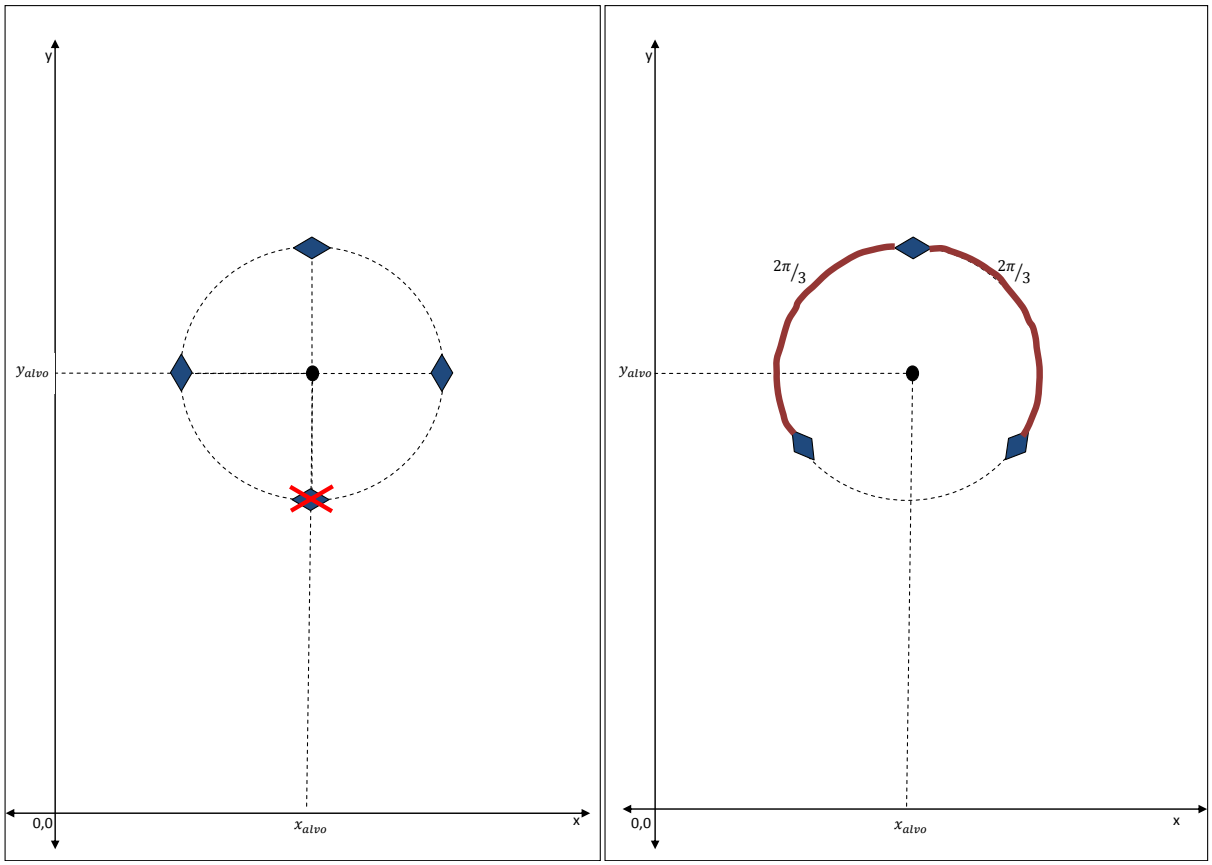
Faz-se então neste capítulo, primeiramente, a modelagem do problema. Para que então, possa-se modelar as malhas de controle e estabelecer a modelagem da rede de comunicação da frota, que não será modelada nesta primeira etapa do trabalho. Portanto, não serão apresentados detalhes sobre ela. Para estabelecer o modelo matemático do problema, será considerado o modelo de robô mostrado na Figura 2.

Figura 2 – Modelo do Robô

6.1 Modelo Matemático

Este trabalho tem como intuito modelar uma malha de controle que guie uma frota de robôs a circundar, a uma distancia R, um alvo localizado em uma determinada posição (x,y) do plano. Esse controle deve, também, ser responsável pela formação da tropa de robôs que, com a falha de um ou mais robôs, deve se reestruturar para continuar cobrindo com eficiência a fronteira. Ou seja, caso um ou mais robôs saiam da rede, a frota ira se reajustar para que cada robô tenha a mesma distância entre si e assim, não fique uma grande parte da fronteira sem cobertura, como demostrado na Figura 3. Que representa uma frota de quatro robôs andando ao redor do alvo, quando então, um dos robôs falha. E o sistema se reajusta para adaptar-se à rede de apenas três robôs.

Figura 3 – Modelagem do Sistema



Para introduzir a dinâmica dos robôs móveis utilizados, inicialmente o robô será considerado como um uniciclo, um elemento pontual. A dinâmica de um robô móvel não-holonômico do tipo uniciclo, desconsiderando a dinâmica, pode ser descrita pelas equações abaixo:

$$\dot{x} = v\cos(\theta) \tag{2}$$

$$\dot{y} = v \sin(\theta) \tag{3}$$

$$\dot{\theta} = \omega \tag{4}$$

sendo:

- (x,y) as coordenadas da posição do robô no plano cartesiano;
- θ o sentido do robô no plano cartesiano;
- v e ω indicam a velocidade linear e angular do robô, respectivamente.

Derivadas dessas equações, surgem as equações 5,6 e 7 modeladas baseadas no robô real, que não é um elemento pontual no espaço e sim, um modelo não holonômico. Elas serão utilizadas a princípio para visualizar a trajetória do robô no ambiente *MATLAB*.

$$x_{k+1} = x_k + \frac{D_r + D_l}{2} \cos(\theta_k) \tag{5}$$

$$y_{k+1} = y_k + \frac{D_r + D_l}{2} \sin(\theta_k) \tag{6}$$

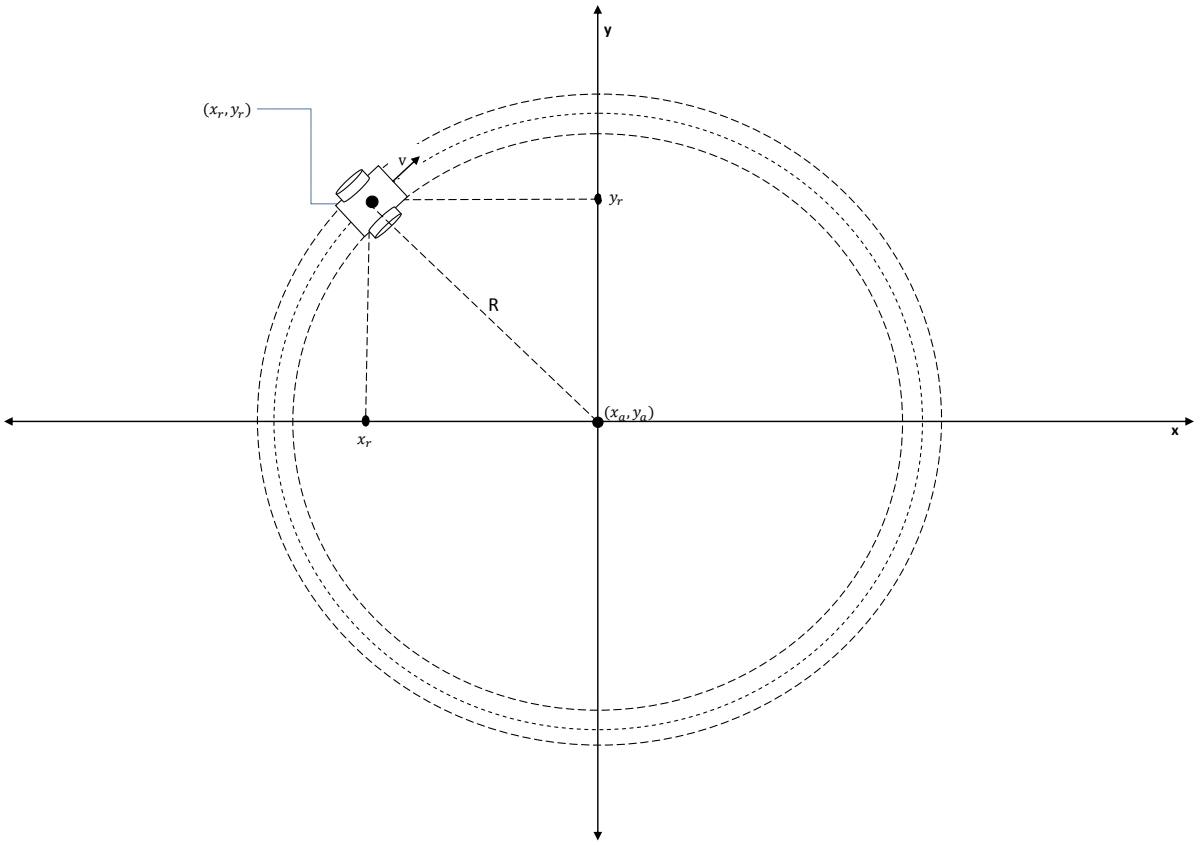
$$\theta_{k+1} = \theta_k + \frac{D_r - D_l}{L} \tag{7}$$

sendo:

- x_{k+1} e x_k a coordenada x do robô no instante k e no instante $k+1$;
- y_{k+1} e y_k a coordenada y do robô no instante k e no instante $k+1$;
- θ_{k+1} e θ_k o sentido do robô no instante k e no instante $k+1$;
- D_r e D_l a distância que a roda direita e esquerda percorreram no instante de tempo entre k e $k+1$, respectivamente;
- L o tamanho do eixo do robô;

Considerando que à medida que o sistema se estabiliza, o robô tende entra em movimento circular uniforme ao redor do alvo. Ou seja, a velocidade linear (v) tende a se igualar a velocidade angular (ω) vezes o raio (R) de distância do alvo.

Figura 4 – Representação do sistema estável



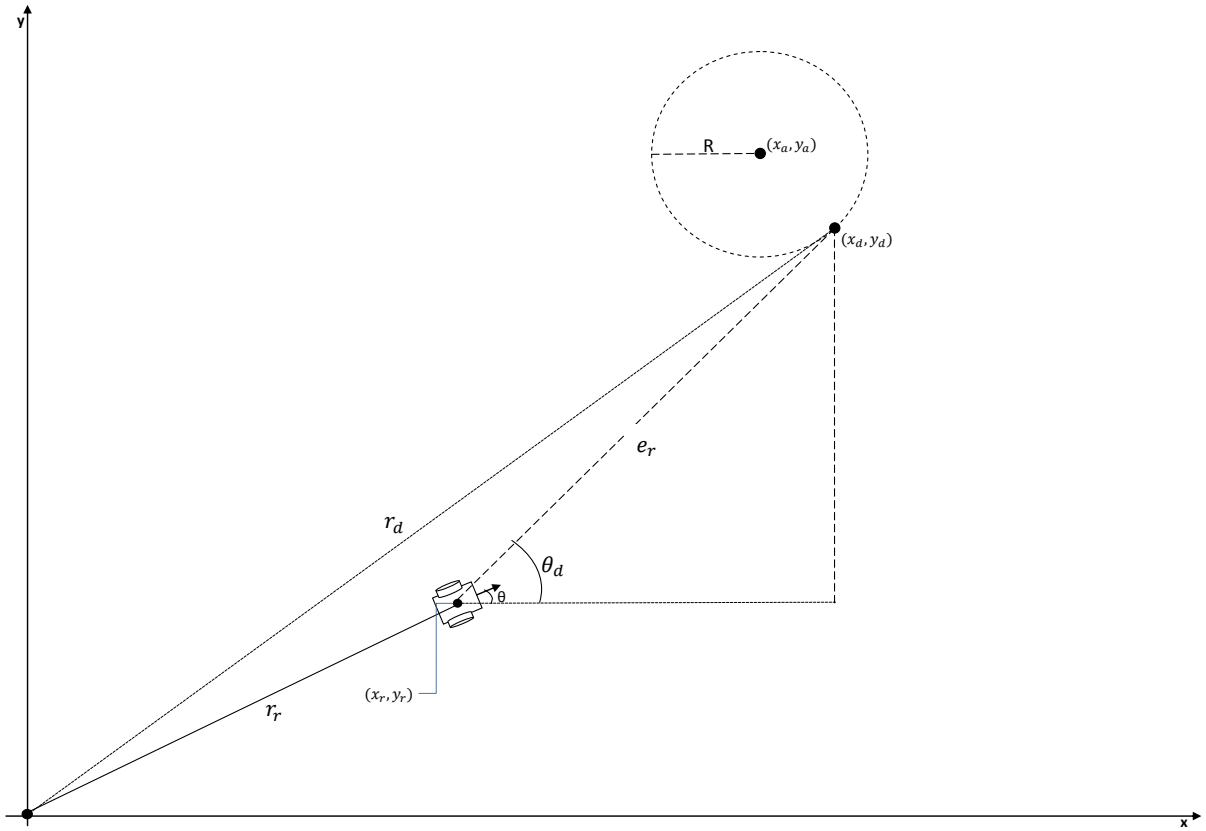
6.2 Malha de Controle 2: Posicionamento

O trabalho tem como objetivo, dado a posição *x*,*y* do alvo (x_a,y_a), fazer com que a frota de robôs se desloque para a região no espaço do alvo e o circule com uma distância *R*, à uma dada velocidade angular (ω_d) desejada. Para que a malha de controle 3, que consiste no controle de formação da tropa funcione, primeiramente é necessário, implementar o controle de posicionamento de cada robô da frota. Ou seja, a malha de controle de formação irá passar para cada robô os parâmetros necessários para o ajuste da estrutura, tais como, velocidade e posicionamento.

Como pode ser visto na Figura 5, o problema a ser solucionado pela segunda malha de controle, consiste em, dado um sistema de coordenadas cartesianas, onde têm-se um alvo de posição (x_a,y_a) e um robô móvel de posição (x_r,y_r), cujo sentido (θ) é indicado pela sua variação dado o eixo *x* do sistema, onde pretende-se fazer com que o robô chegue ao ponto desejado (x_d,y_d, recebido da malha de controle 3), que se distância do alvo à uma distância *R*, e então, fazê-lo circular ao redor do alvo. Ou seja, consiste em fazer com que o robô vá até o alvo e o circule à uma distância *R*.

Para tal, a malha 2 funciona da da seguinte maneira: Recebe da malha 3 os parâmetros necessários para o cálculo da posição desejada do robô (x_d,y_d), a partir

Figura 5 – Esquema do Sistema do Ponto de Vista de Apenas Um Agente



daí é achado o erro de posição do robô (e_x,e_y), fazendo-se a diferença entre a posição desejada e a posição real do robô (x_r,y_r), que é obtida através dos *encoders* do Lego, que se mostrou suficientemente precisos.

$$e_r = r_d - r_r \tag{8}$$

$$e_x = x_d - x_r \tag{9}$$

$$e_y = y_d - y_r \tag{10}$$

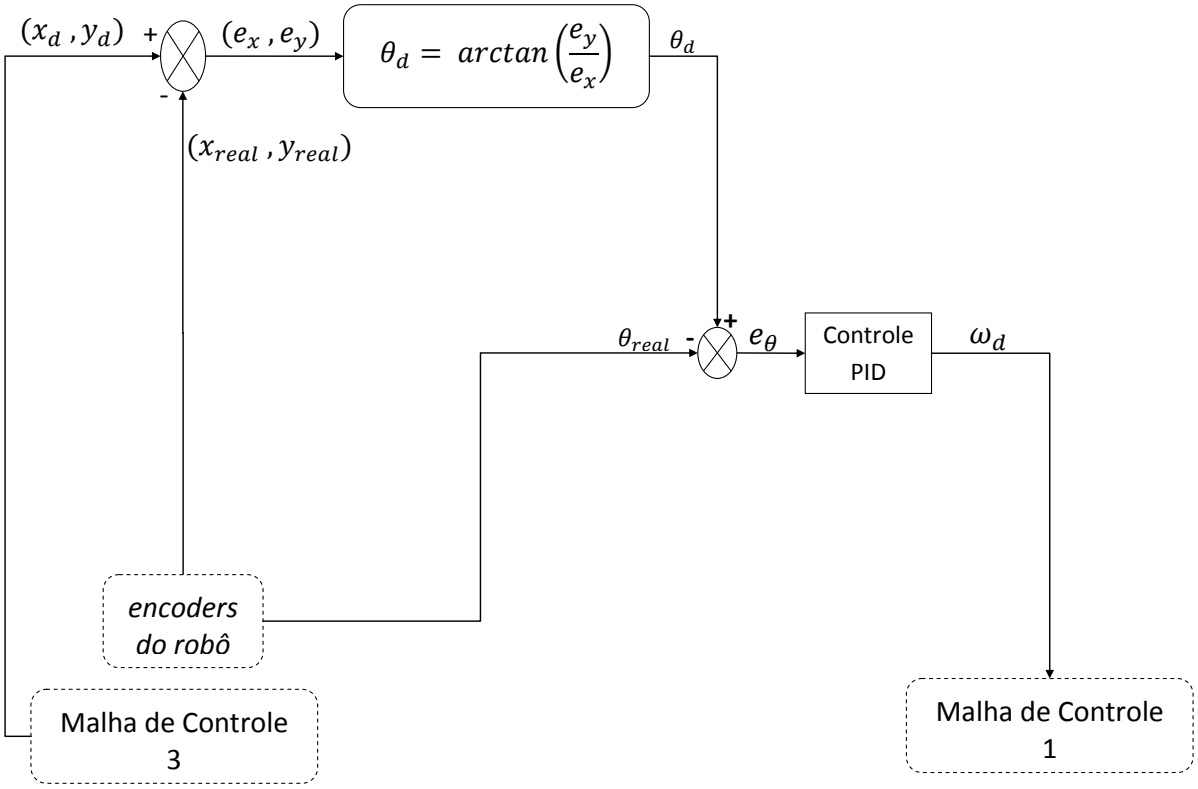
Através do erro de posicionamento do robô, encontra-se o sentido desejado (θ_d), como mostrado na Equação (11). Então, o erro de sentido do robô é obtido, através da Equação (12) abaixo.

$$\theta_d = \arctan(\frac{e_y}{e_x}) \tag{11}$$

$$e_\theta = \theta_d - \theta_r \tag{12}$$

O erro de sentido (e_θ) é passado para o controlador que retorna a velocidade angular da ação de controle, que será passada para a malha mais interna. Posteriormente, será feita uma comparação entre os controladores *PI* e *PID* e será definido o controlador a ser utilizado nesta malha.

Figura 6 – Segunda malha de Controle do Sistema - Controle de Posicionamento



6.3 Malha de Controle 1: Velocidade Angular das Rodas

Como já dito anteriormente neste trabalho, este sistema de controle consiste em um controle de três malhas, e agora será abordado sobre a primeira malha. Ela controla os motores para atingir a velocidade angular desejada (ω_d). Ou seja, deseja-se circular o alvo em um período de T segundos, a malha de controle de velocidade vai receber a velocidade angular desejada (ω_d), que é uma derivação do período de circulação desejado, como mostrado na equação abaixo:

$$\omega = \frac{2\pi}{T} \tag{13}$$

É importante ressaltar que como o robô não é um elemento pontual[1], como considerado na Seção 6.1 ao descrever as equações do modelo, temos que descrever a velocidade

[1] Veja a Figura 4 para visualizá-lo como um elemento não pontual, que depende da variação de velocidade de cada roda para definir a velocidade e o sentido do robô.

angular (ω) e linear (v) do robô em função de cada roda, para sabermos a potência a ser aplicada em cada uma delas para que o robô obtenha a velocidade e o sentido desejados.

A partir daí o módulo calcula velocidade angular desejada de cada roda, como demonstrado nas equações abaixo:

$$\omega_{dr} = \frac{2v + \omega_d L}{2r_p} \tag{14}$$

$$\omega_{dl} = \frac{2v - \omega_d L}{2r_p} \tag{15}$$

onde:

- v é a velocidade linear (m/s) desejada do robô;
- ω_{dr} é a velocidade angular desejada da roda direita;
- ω_{dl} é a velocidade angular desejada da roda esquerda;
- r_p o raio do pneu (m);
- L o tamanho do eixo do robô (m).

Com a velocidade angular de cada roda, dada pelos *encoders* do robô, é calculado o erro das velocidades, como mostrado nas equações abaixo, e o controlador *PI*, os recebe como entrada, retornando as ações de controle que serão passadas como potência para cada uma das rodas.

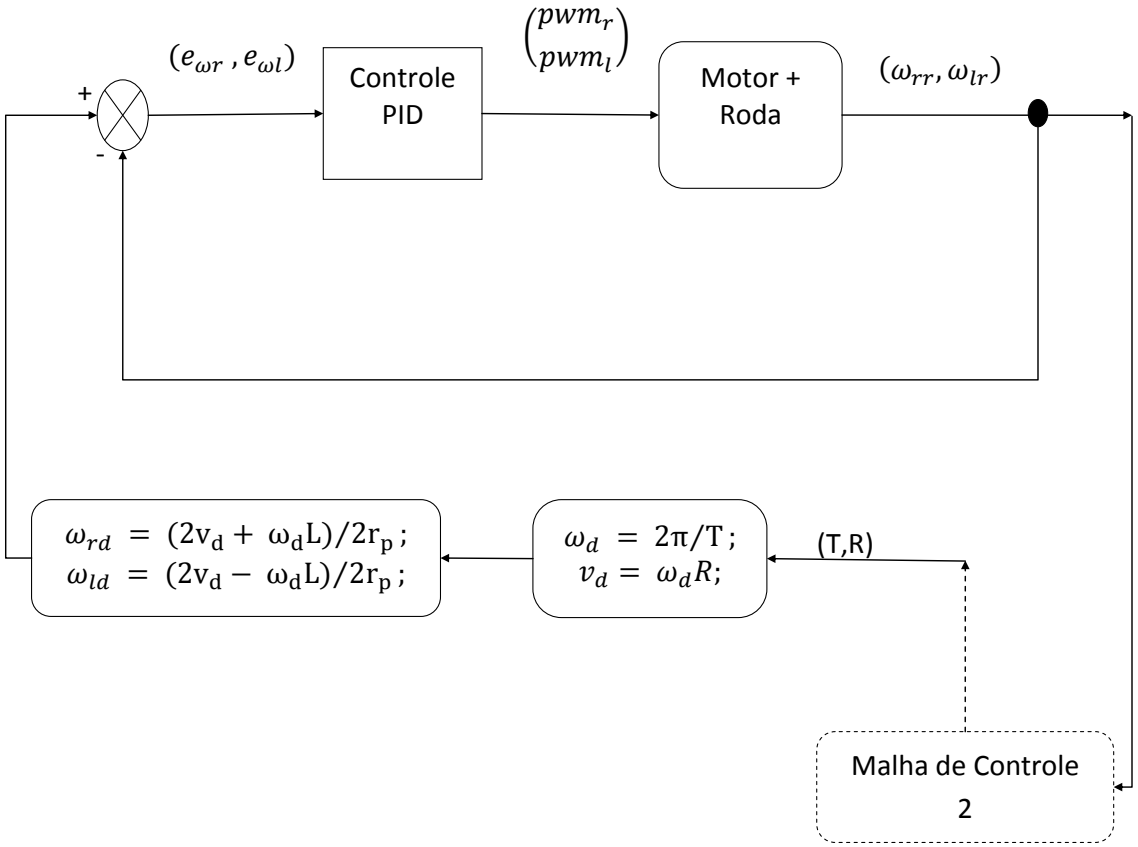
$$e_{wr} = \omega_{dr} - \omega_{rr} \tag{16}$$

$$e_{wl} = \omega_{dl} - \omega_{rl} \tag{17}$$

onde:

- e_{wr} é o erro da velocidade angular da roda direita;
- e_{wl} é o erro da velocidade angular da roda esquerda;
- ω_{rr} é a velocidade angular real da roda direita;
- ω_{rl} é a velocidade angular real da roda esquerda.

Figura 7 – Primeira malha de Controle do Sistema - Controle da velocidade angular



7 Resultados Preliminares

Para alcançar o objetivo final deste trabalho, fez-se uma modularização do problema que será validada e integrada, módulo a módulo. Para a primeira parte do trabalho, supõe-se que o robô já tenha encontrado o alvo e precisa simplesmente circular ao seu redor. Ou seja, precisa realizar um movimento circular uniforme. Para tal, foi necessário elaborar a malha de controle 1 que é responsável pelo controle de velocidade de cada roda.

Sendo assim, foi estabelecido o raio (R) e o período (T) em que o robô pretende circular ao redor do alvo. Definindo-se R igual a $15cm$ e $T = 5s$. Tem-se que a velocidade angular desejada do robô (ω_d) será de $1{,}256rad/s$, conforme Equação (13). E a velocidade linear desejada (v_d) do robô será de $0{,}1884m/s$, supondo o ideal que é um sistema já estabilizado. Ou seja, em movimento circular uniforme onde tem-se que:

$$v = \omega R \tag{18}$$

A partir destas definições foram utilizados os *encoders* do próprio *Lego Mindstorms* para obter a velocidade real (ω_{rr},ω_{rl}) de cada roda e as equações 14 e 15, para calcular a velocidade angular desejada de cada roda (ω_{dr},ω_{dl}), para que o robô entre em movimento circular uniforme. E assim, foram utilizadas as equações 16 e 17, para o cálculo do erro da velocidade angular (e_{wr},e_{wl}). Este erro alimenta o controlador que retorna como saída, a potência (pwm_r, pwm_l) que será passada a cada motor.

Foram implementados três tipos de controladores diferentes, afim de comparar o desempenho dos mesmos para resolução do problema. Primeiro, implementou-se um controlador simples que acresce de uma unidade a potência do motor quando a um erro maior que zero, ou decresce, caso haja um erro menor que zero. É importante ressaltar que os motores do *kit Lego Mindstorms* aceitam comandos de potência que variam de -100 a 100. Logo, poderiam haver situações em que a potência passada aos motores excederia aos limites do motor, para evitar a saturação dos atuadores, foi definido um limite para a saída do controlador que respeite às limitações da plataforma.

Ao utilizar a ferramenta *MATLAB* para visualizar o erro de velocidade angular, percebe-se que ela converge para zero como esperado, como mostrado na Figura 8.

Entretanto, ao observar a trajetória do robô no plano e ao plotar no *MATLAB* a trajetória do mesmo,utilizando-se as equações 5, 6 e 7, percebe-se que ele demora um pouco para realizar o movimento circular, como demonstrado na Figura 9.

Feito isso, para fins de comparação, implementou-se um controlador PI, de

Figura 8 – Erro de velocidade angular - Controle Simples

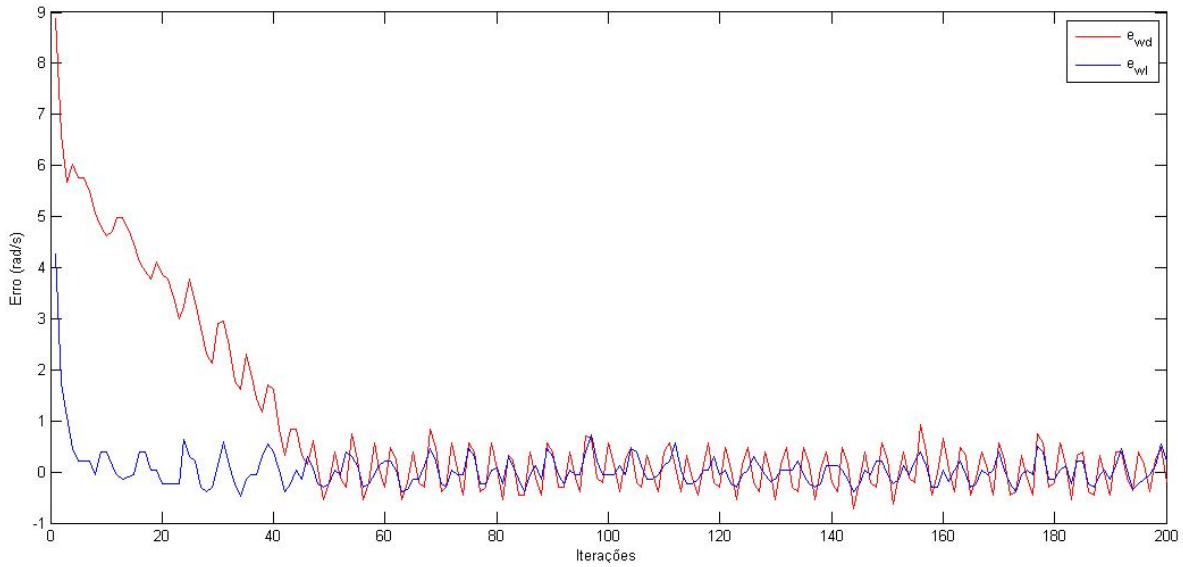
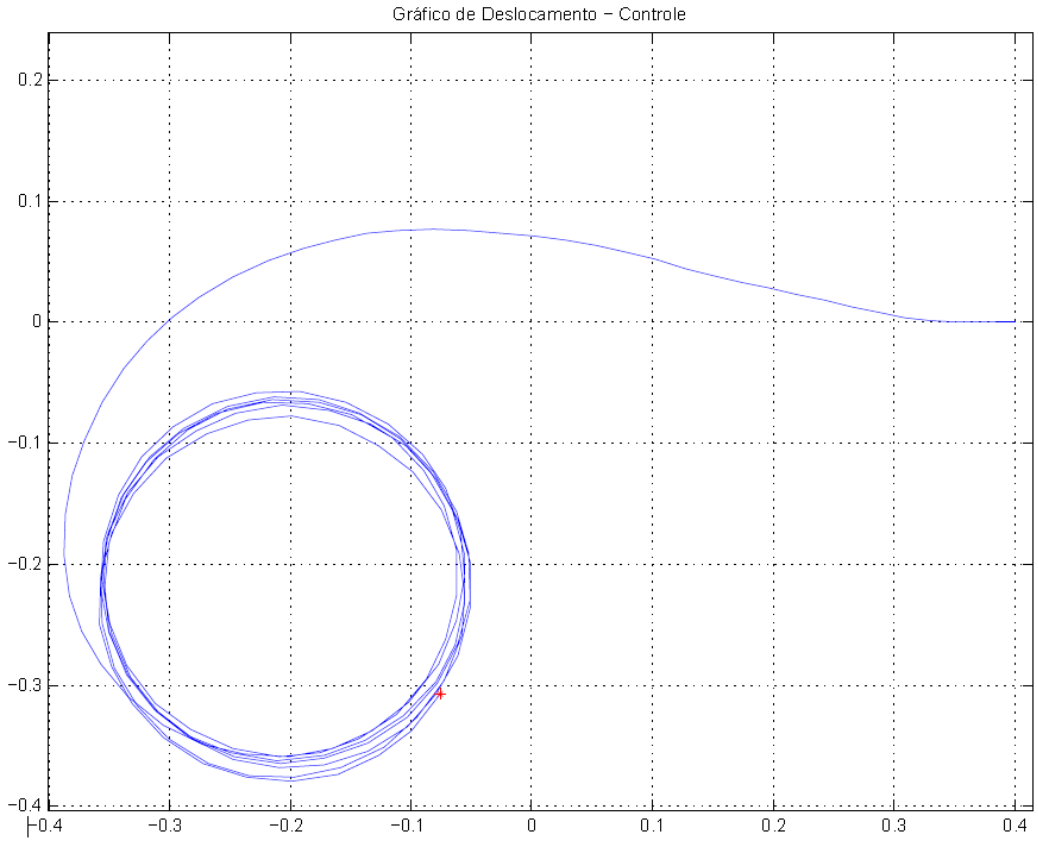


Figura 9 – Trajetória do Robô (R: $15cm$) - Controle Simples



ganho proporcional (k_p) igual a 10 e ganho integral (k_i) igual a 1 e obteve-se os erros e a trajetória demonstrados pelas figuras 10 e 11.

Observando as figuras 8 e 10 é possível notar que ambos os controladores ten-

Figura 10 – Erro de velocidade angular - Controle PI: $k_p = 10$,$k_i = 1$ - Raio 15cm

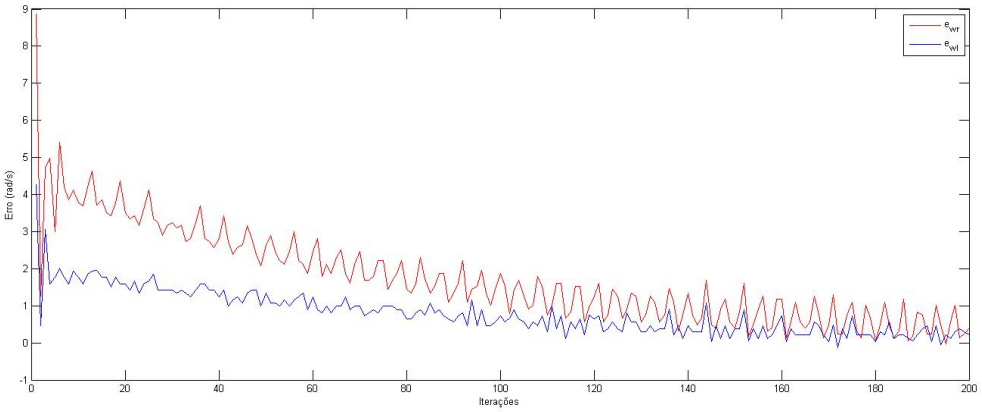
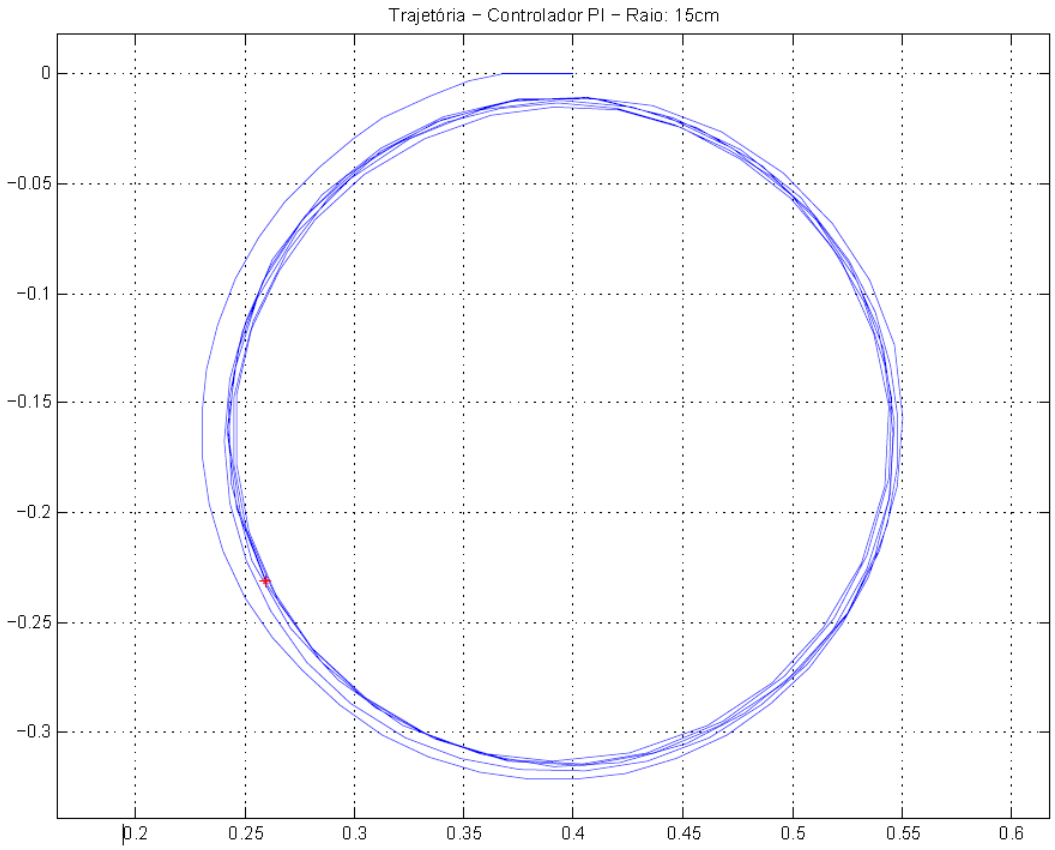


Figura 11 – Trajetória do Robô (R: $15cm$) - Controle PI: $k_p = 10$,$k_i = 1$



dem à diminuir o erro de velocidade. Contudo, ao verificar na figura comparativa Figura 12 é possível perceber que o controlador simples converge mais rapidamente que o controlador *PI*. Entretanto, embora o controlador simples convirja mais rapidamente, o que se mostra no comparativo da trajetória (Figura 13) de ambos os controladores, é que o controlador *PI*, converge mais rapidamente para a trajetória desejada. Sendo assim, foram realizados mais experimentos com variações do controlador *PI*, como será

visto a seguir.

Figura 12 – Comparativo - Erro de velocidade angular - Controle PI e Controle Simples - Raio 15cm

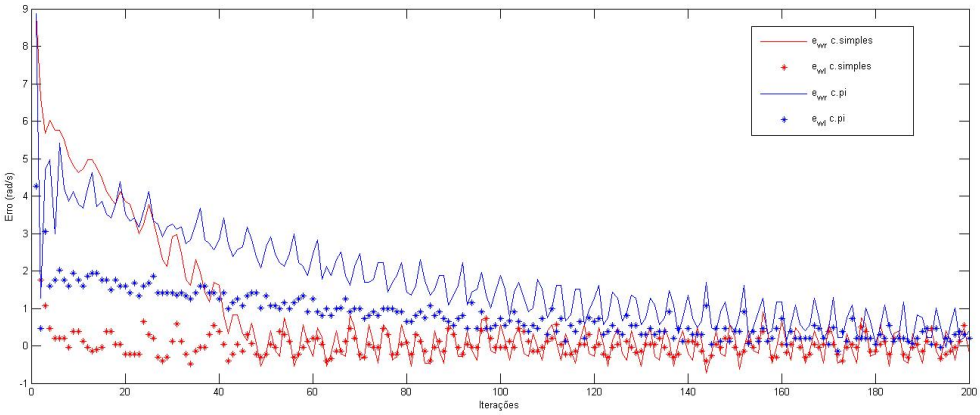
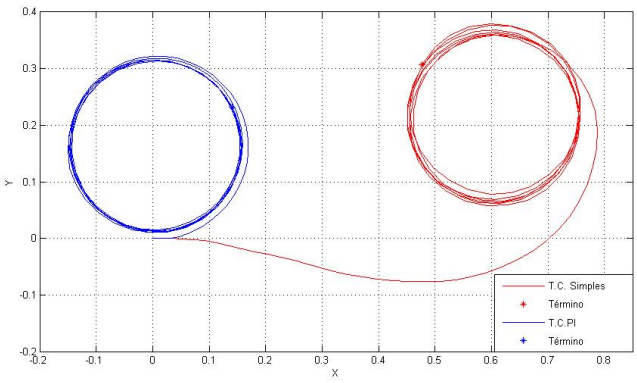


Figura 13 – Comparativo da Trajetória do Robô (R: $15cm$) - Controle PI e Controle Simples



Primeiro, fez-se testes variando o ganho proporcional (k_p). Notou-se, como comprovado pela Figura 15 que ao dobrar k_p, o sistema se tornava instável, como previsto na Seção 5.3, que indica que quanto maior o ganho, mais o sistema tende à se instabilizar. Depois, foi realizada uma comparação, alterando-se os parâmetros de k_p e k_i, como mostrado na **??**. E por fim, foram realizados experimentos variando o raio do perímetro ao redor do alvo, como exemplificado na Figura 16, onde observa-se que a trajetória é suficientemente precisa, conforme o sistema se estabiliza.

Figura 14 – Erro de velocidade angular - Controle PI: Sistema Instável - Raio 15cm

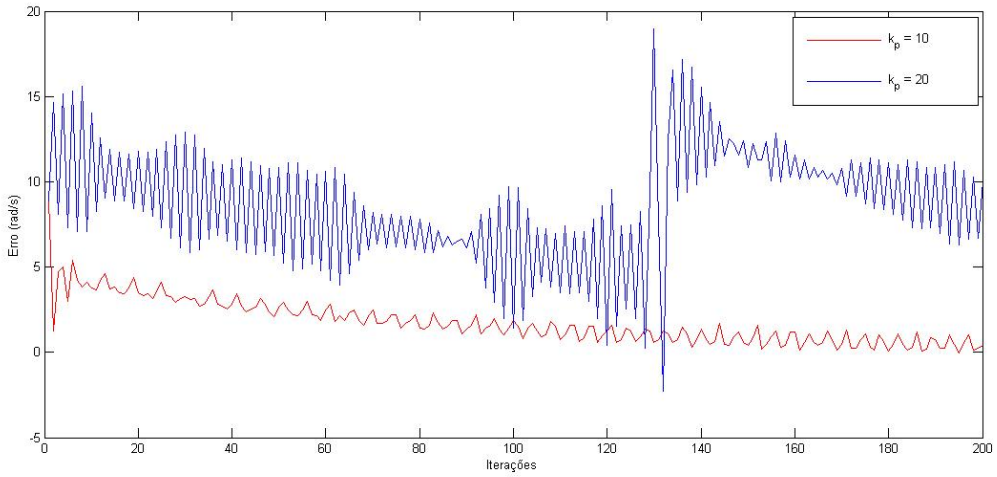


Figura 15 – Comparativo: Erro de velocidade angular - Controladores PI

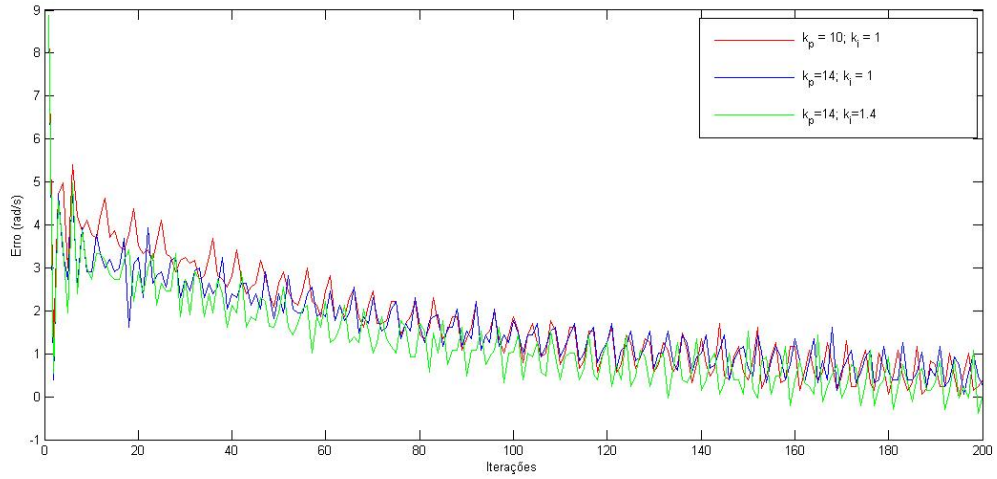
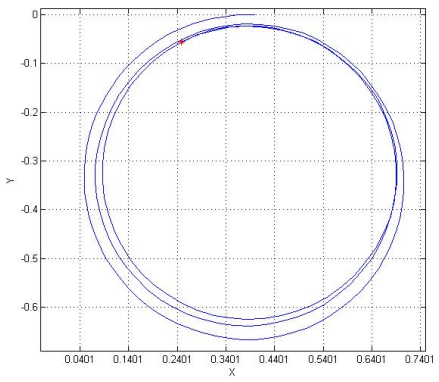


Figura 16 – Trajetória do Controladores PID - Raio $30cm$



8 Conclusão

Após realizados os testes, pode-se concluir que os *encoders* são suficientemente precisos, até então, para serem utilizados como ferramenta de medição e alimentador do sistema. Entretanto, devem ser realizados mais experimentos afim de aprimorar mais os ganhos do controlador para que possa-se assim ter uma maior segurança quando forem realizados os testes na malha 2, que no momento, apresenta-se em fase de implementação e testes.

Referências

BALCH, T.; ARKIN, R. C. Behavior-based formation control for multirobot teams. **Robotics and Automation, IEEE Transactions on**, IEEE, v. 14, n. 6, p. 926–939, 1998. Citado na página 11.

BENEDETTELLI, D.; CASINI, M.; GARULLI, A.; GIANNITRAPANI, A.; VICINO, A. A lego mindstorms experimental setup for multi-agent systems. In: **Control Applications, (CCA) Intelligent Control, (ISIC), 2009 IEEE**. [S.l.: s.n.], 2009. p. 1230–1235. Citado na página 4.

CARLES, M.; HERMOSILLA, L. O futuro da medicina: nanomedicina. **Revista Científica Eletrônica de Medicina Veterinária**, v. 6, n. 10, p. 1–7, 2008. Citado na página 1.

CASINI, M.; GARULLI, A.; GIANNITRAPANI, A.; VICINO, A. A lego mindstorms multi-robot setup in the automatic control telelab. In: **Proceedings of 18th IFAC World Congress, Milano, Italy**. [S.l.: s.n.], 2011. v. 28. Citado na página 5.

CHEN, H.; SHENG, W.; XI, N.; SONG, M.; CHEN, Y. Automated robot trajectory planning for spray painting of free-form surfaces in automotive manufacturing. In: **Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on**. [S.l.: s.n.], 2002. v. 1, p. 450–455 vol.1. Citado na página 1.

CORRÊA, M. A.; JÚNIOR, J. B. C. **Estudos de veículos aéreos não tripulados baseado em sistemas multi-agentes e sua interação no espaço aéreo controlado**. [S.l.]: Sitraer, 2008. Citado na página 1.

GIRARD, A. R.; HOWELL, A. S.; HEDRICK, J. K. Border patrol and surveillance missions using multiple unmanned air vehicles. In: IEEE. **Decision and Control, 2004. CDC. 43rd IEEE Conference on**. [S.l.], 2004. v. 1, p. 620–625. Citado na página 2.

GOUVÊA, J. A. **CONTROLE DE FORMAÇÃO DE ROBÔS NÃO-HOLONÔMICOS COM RESTRIÇÃO DE CURVATURA UTILIZANDO FUNÇÃO POTENCIAL**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2011. Citado na página 10.

JESUS, T. A. **Estratégias de Guiagem e Cooperação de Robôs Aéreos Sujeitos a Restrições nas Entradas e/ou nos Estados**. Tese (Doutorado) — Universidade Federal de Minas Gerais, 2013. Citado na página 2.

MARJOVI, A.; NUNES, J. G.; MARQUES, L.; ALMEIDA, A. de. Multi-robot exploration and fire searching. In: IEEE. **Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on**. [S.l.], 2009. p. 1929–1934. Citado na página 2.

MARTINEZ, D. L. et al. Reconfigurable multi robot society based on lego mindstorms. Helsinki University of Technology, 2009. Citado 3 vezes nas páginas 4, 10 e 11.

RAMCHURN, S. D.; HUYNH, D.; JENNINGS, N. R. Trust in multi-agent systems. **Knowl. Eng. Rev.**, Cambridge University Press, New York, NY, USA, v. 19, n. 1, p. 1–25, mar. 2004. ISSN 0269-8889. Disponível em: <http://dx.doi.org/10.1017/S0269888904000116>. Citado na página 1.

SECCHI, H. **Uma Introdução a Robôs Móveis**. [S.l.]: Argentina, 2008. Citado na página 1.

SOURCEFORGE. **Bricx Command Center**. 2001. Disponível em: <http://bricxcc.sourceforge.net/>. Acesso em: 30 de março de 2009. Citado na página 3.

TAYLOR, J.; DÍAZ, J.; GRAU, A. **Mecánica clásica**. Reverté, 2013. ISBN 9788429143126. Disponível em: <http://books.google.com.br/books?id=6QJjngEACAAJ>. Citado na página 10.