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
JULHO 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
IDE	*Integrated Development Environment*
NBC	Next Byte Codes
NXC	Not eXactly C
P	Controlador Proporcional
PI	Controlador Proporcional Integral
PID	Controlador Proporcional Integral Derivativo
VANT	Veículo Aéreo Não Tripulado

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
v_d	Velocidade linear desejada
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 Definição do Problema

Visando cumprir esse objetivo, definiu-se dois problemas. O primeiro e mais simples, consiste na coordenação de uma frota de robôs para que a mesma se alinhe

paralelamente e sigam em linha reta, como em problemas de varreduras de área em paralelo. E o segundo, consiste em um controle de uma frota de N robôs, para que a mesma localize um dado alvo no espaço e o circunde a uma distância R e em um período T, que se reajustará conforme o tamanho da frota.

1.4 Infraestrutura Necessária

Para a realização deste trabalho foram utilizados três 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, por exemplo, 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 outros 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, além de tudo, contemple 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 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.

3.1 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).

3.2 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, centralizada ou não, no que diz respeito

à estrutura da rede como indivíduo, até mesmo se a rede é formada por indivíduos independentes ou se é o mesmo robô, quanto à estrutura organizacional e 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.

3.3 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.

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

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.

3.4 Plataformas

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

3.4.1 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.

3.4.2 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.

3.4.3 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.

3.4.4 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.

3.4.5 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.

3.4.6 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*.

4 Metodologia

Para a realização deste trabalho, primeiramente fez-se 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. Onde não tem-se um mestre definido, apenas uma sociedade de robôs que conforme precisam executar uma tarefa, vão recrutando e formando uma frota.

Após realizados os estudos, concluiu-se que, apesar de viável como pode ser visto no capítulo 2, a implementação de um sistema distribuído descentralizado seria muito complexo para ser abordado no período proposto para a realização deste trabalho, que tem como objetivo principal o estudo de estratégias de controle de formação de múltiplos robôs móveis. Para tanto, foi adotada uma estrutura de rede centralizada denominada 'Mestre/Escravo', utilizando a comunicação via *Bluetooth*, a qual a própria plataforma e linguagem (*NXT*) dão suporte.

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. Foram escolhidas duas estratégias diferentes de formação de múltiplos robôs móveis e fez-se então, duas abordagens distintas. Uma delas que atende apenas à um dos problemas e uma outra abordagem mais genérica que pode ser adaptada para a resolução de ambos os problemas.

Para estabelecer o modelo matemático do problema, foi considerado o modelo de robô mostrado na Figura 2, que consistem em um modelo não holonômico, onde tem-se duas rodas unidirecionais e uma roda orientável. Para definição do modelo matemático levou-se em consideração que os *encoders* seriam utilizados para odometria e então, foram definidas as restrições da modelagem matemática do problema que, desconsidera problemas como: saturação do atuador, derrapagem das rodas, os erros de medição dos *encoders* bem como, as limitações da plataforma.

Após feita a modelagem do problema, foram feitas as implementações e os testes, onde seria possível observar se a odometria feita seria suficiente para alimentar o sistema com precisão e tornar a solução viável. Levando-se em consideração que para tanto foram utilizados os *encoders* óticos acoplados aos motores do *kit Lego Mindstorms*® que, por sua vez, possuem uma imprecisão que é da ordem de $\pm$ 1 grau por rotação

Concomitantemente, foram realizadas simulações, através da ferramenta *MATLAB*®, a fim de prever e validar a modelagem feita do problema, desconsiderando os problemas práticos como, falha na comunicação e falta de sincronismo entre os robôs, erros

Figura 2 – Modelo do Robô

dos *encoders* e problemas como saturação do motor e derrapagem das rodas. Bem como fazer uma comparação entre o modelo real e o modelo idealizado do sistema.

4.1 Modelo Matemático

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 para visualizar a trajetória do robô no ambiente *MATLAB*® e verificar se a trajetória é compatível com o caminho percorrido pelo robô no mundo real.

Tendo em vista, que a odometria será feita utilizando-se os *encoders* da própria plataforma e o sistema será realimentado com essas medidas, é de extrema importância que a trajetória observada no mundo real e a registrada pelo robô, através das equações e utilizando-se os *encoders*, sejam consideravelmente semelhantes. Caso o contrário, a realimentação do sistema estará incorreta, comprometendo seriamente, e por que não dizer inviabilizando, o controle do sistema.

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

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

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

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;

Dado que a velocidade linear e angular de um robô como o do modelo utilizado neste trabalho é dada pela velocidade angular de cada uma das suas rodas unidirecionais, tem-se nas equações 8 e 9 a função que descreve a velocidade linear e angular do robô baseado na velocidade angular de suas rodas.

$$v = \frac{(\omega_r + \omega_l)rp}{2} \tag{8}$$

$$\omega = \frac{(\omega_r - \omega_l)rp}{L} \tag{9}$$

Em que:

- v é a velocidade linear do robô;

- ω é a velocidade angular do robô;
- ω_r é a velocidade angular da roda direita do robô;
- ω_l é a velocidade angular da roda esquerda do robô;
- rp é o raio da roda do robô;
- L é a distância entre as rodas unidirecionais do robô.

4.2 Problema 1: *In line*

O primeiro problema consiste em: dado uma frota de *N* robôs, esses robôs devem se alinhar horizontalmente e seguir em linha reta, andando paralelamente, utilizando a estrutura já citada na seção 3.2, denominada *"In Line"*. Primeiro, este problema será modelado considerando uma modelagem mais simples que será melhor descrita a seguir.

Considerando-se *N* robôs separados por uma distância Δy no eixo y, cada um em um ponto distinto no eixo x, como mostrado na Figura 3 (a), a tropa deve se alinhar paralelamente e seguir andando paralelamente com uma velocidade v_d constante.

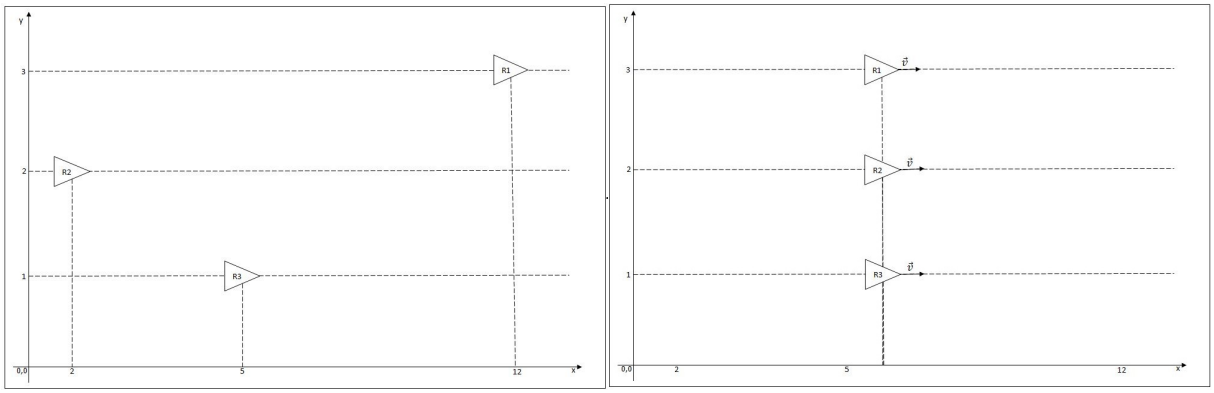

Figura 3 – Problema 1: Fig.(a): Sistema no instante i. Fig.(b): Sistema no instante i+k.

Supondo uma tropa de três robôs, orientados todos no mesmo sentido do eixo x, as velocidades de cada robô devem variar de acordo com o erro de posicionamento do mesmo no eixo x, conforme a distância entre os robôs. Conforme mostrado nas equações 10, 11 e 12. Essas equações fazem com que o robô mais adiantado da frota, imprima uma velocidade menor que os outros robôs, tendendo a se aproximar dos mesmos, em contrapartida os robôs que estão "atrasados"imprimem uma velocidade maior, até que o erro entre eles seja zero e os mesmos caminhem com velocidade constante.

$$v_{Robo1} = v_d + k \times erro_{2,1} \tag{10}$$

$$v_{Robo2} = v_d + k \times (erro_{1,2} + erro_{3,2}) \tag{11}$$

$$v_{Robo3} = v_d + k \times erro_{1,3} \tag{12}$$

Onde,

- v_{Robo1}, v_{Robo2} e v_{Robo3} são as velocidades que cada robô deve possuir para assumir a formação *in line*;
- v_d é a velocidade que a frota deve assumir após estar alinhada;
- k é a constante de ganho proporcional do controlador;
- E as variáveis de erro são calculadas, obtendo se a diferença entre a posição no eixo x entre os robôs, como demonstrado na Equação (13).

$$erro_{i,j} = x_i - x_j \tag{13}$$

É importante notar que ao abordar o problemas desta maneira elimina-se o problema de colisão entre os robôs, que estão inicialmente separados no eixo y e alinhados no mesmo sentido, e assim seguem, visto que os mesmos não imprimem velocidade angular. Desta forma, tornando o problema bem didático para ser abordado inicialmente. Posteriormente, será mostrado o problema sendo abordado de outra forma.

4.3 Problema 2: Rodeando um alvo

O segundo problema consiste em guiar uma frota de robôs a localizar e circundar, a uma distância *R*, um alvo localizado em uma determinada posição (*x*,*y*) do plano. E tem como objetivo secundário, ajustar a formação da tropa de robôs que deve se reajustar de acordo com o tamanho da frota, para que continue 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 4. 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.

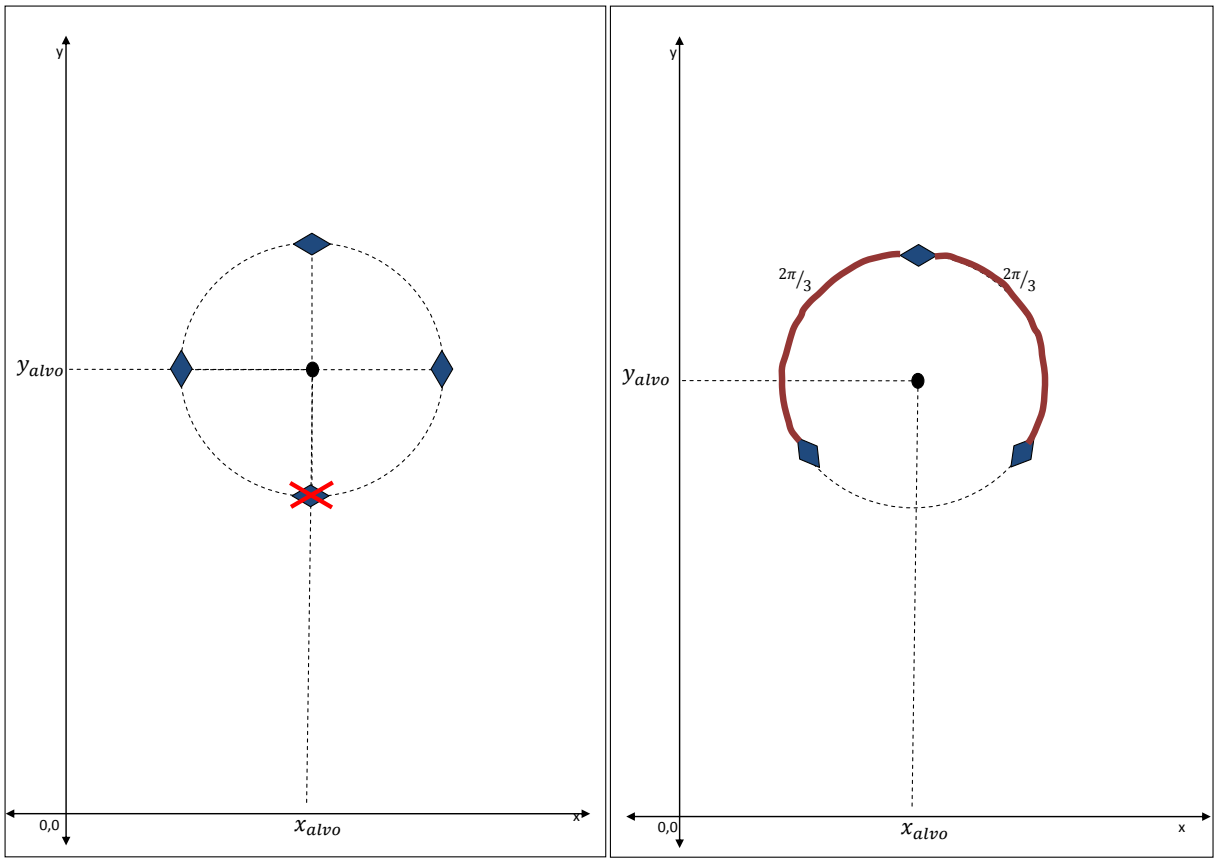

Figura 4 – Modelagem do Sistema

5 Abordagem e Modelagem do Problema: Malhas de Controle

Como citado no Capítulo 4, serão utilizadas duas abordagens diferentes, uma para resolver apenas o primeiro problema e a outra uma abordagem mais genérica que permite adaptação para resolução de ambos os problemas. A primeira abordagem, utilizará uma malha de controle de velocidade e uma malha de controle que será responsável pela comunicação entre os robôs e definirá a velocidade de cada robô, baseado na posição dos outros integrantes da frota.

Já a segunda abordagem, 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. 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. Ambas as malhas estarão presentes em cada um dos robôs da frota. 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 um detalhamento das malhas de controle utilizadas e das modificações realizadas para que a segunda abordagem atenda à ambos os problemas. Bem como, uma descrição de como funcionam as malhas responsáveis pela comunicação e pelo controle de formação para a resolução de cada um dos problemas e faz-se um levantamento dos possíveis problemas que podem surgir na implementação do sistema.

5.1 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 de cada roda (ω_{dr},ω_{dl}). Ou seja, a malha de controle vai receber a velocidade linear e angular que se deseja imprimir no robô e retornar a "potência"que deve ser aplicada a cada um dos motores para se atingir as velocidades desejadas. Como demonstrado pela /autoreffig:malha1.

É importante ressaltar que como o robô não é um elemento pontual[1], como considerado na Seção 4.1 ao descrever as equações do modelo, temos que descrever a velocidade angular (ω) e linear (v) do robô em função de cada roda, para sabermos a

[1] Veja a **??** 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.

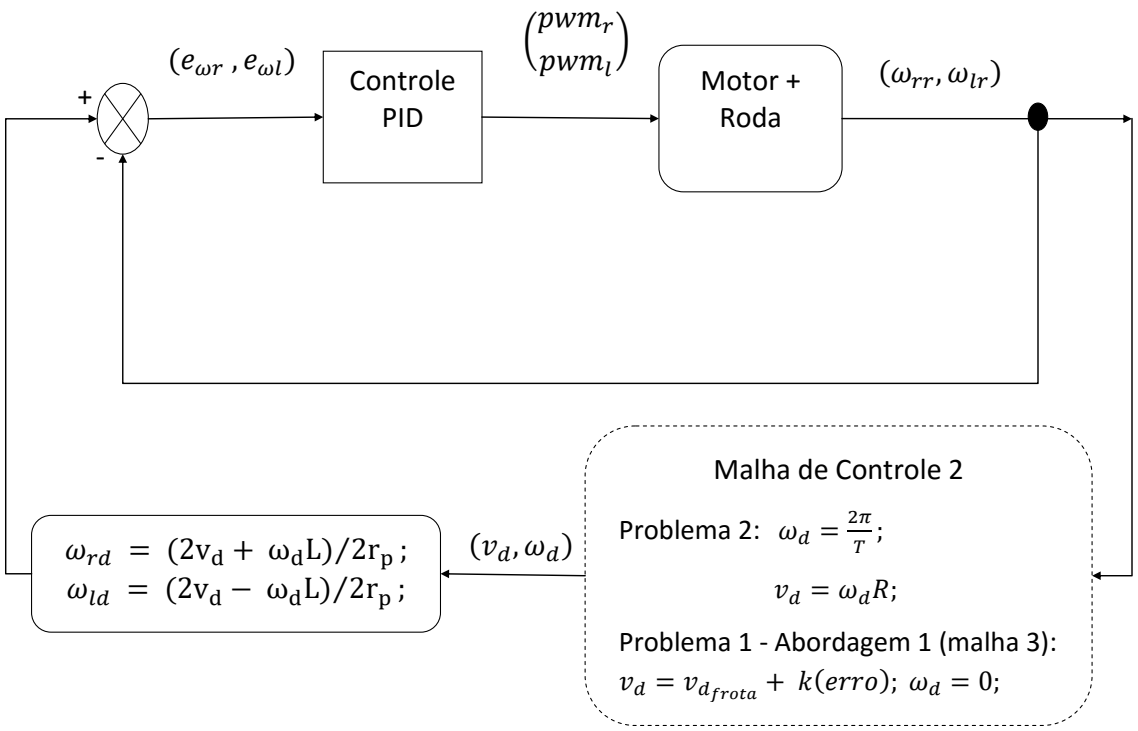

Figura 5 – Primeira malha de Controle do Sistema - Controle da velocidade angular

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

Com a velocidade angular de cada roda do robô, calculada através dos dados fornecidos pelos *encoders*, é calculado o erro das velocidades, como mostrado nas equações abaixo, e o controlador os retorna 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} e e_{wl} são os erros da velocidade angular da roda direita e esquerda, respectivamente;
- ω_{rr} é a velocidade angular real da roda direita e esquerda, respectivamente.

5.2 Malha de Controle 2: Posicionamento

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. Ou seja, para que o robô cumpra a missão, é necessário que a malha de controle de formação passe para cada robô os parâmetros necessários para o ajuste da estrutura, tais como, posicionamento, velocidade e número de robôs da frota. Esse por sua vez, deve conseguir se posicionar na posição correta, recebida pela malha de controle 3,o que só será possível se a malha de controle 1 conseguir controlar adequadamente os motores para que eles imprimam a velocidade correta em cada roda.

Como pode ser visto na **??**, o problema a ser resolvido pela segunda malha de controle, consiste em, dado um sistema de coordenadas cartesianas, onde têm-se 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). Para tanto, a malha 2 funciona da seguinte maneira: Recebe da malha 3 os parâmetros necessários para o cálculo da posição desejada do robô (x_d,y_d), a partir 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.

$$e_r = r_d - r_r \tag{18}$$

$$e_x = x_d - x_r \tag{19}$$

$$e_y = y_d - y_r \tag{20}$$

Através do erro de posicionamento do robô, encontra-se o sentido desejado (θ_d), como mostrado na Equação (21). Então, o erro de sentido do robô é obtido, através da Equação (22) abaixo.

$$\theta_d = \arctan(\frac{e_y}{e_x}) \tag{21}$$

$$e_\theta = \theta_d - \theta_r \tag{22}$$

O erro de sentido (e_θ) é passado para o controlador que retorna a velocidade linear e angular da ação de controle, que será passada para a malha mais interna.

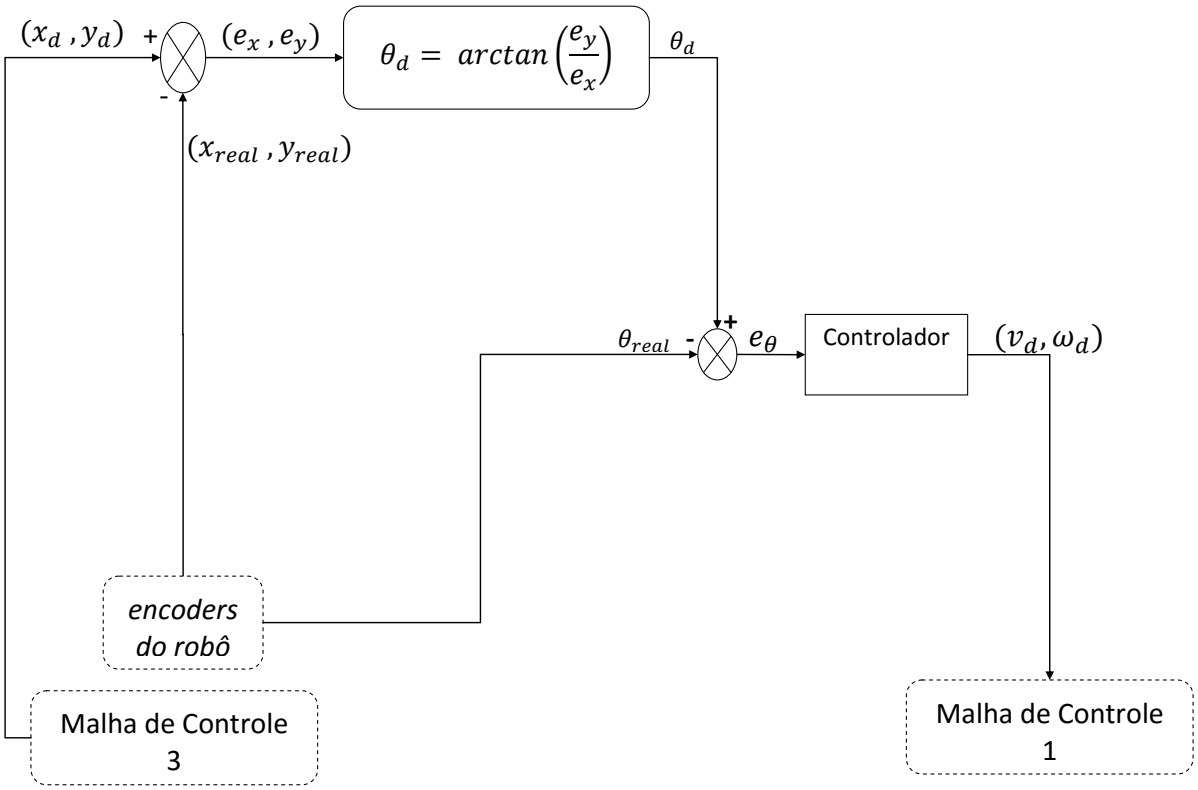

Figura 6 – Segunda malha de Controle do Sistema - Controle de Posicionamento

5.3 Malha de Controle 3: Controle de formação

6 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 **??**. 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:

$$v = \omega R \tag{23}$$

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 7.

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 8.

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

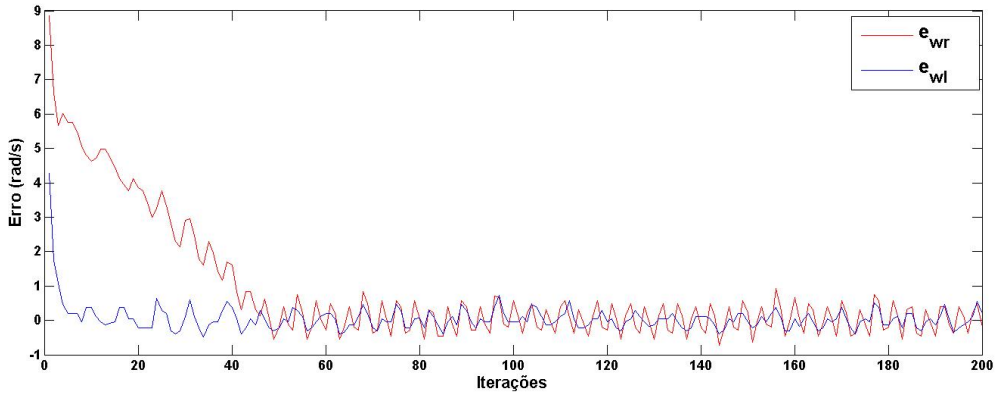

Figura 7 – Erro de velocidade angular - Controle Simples

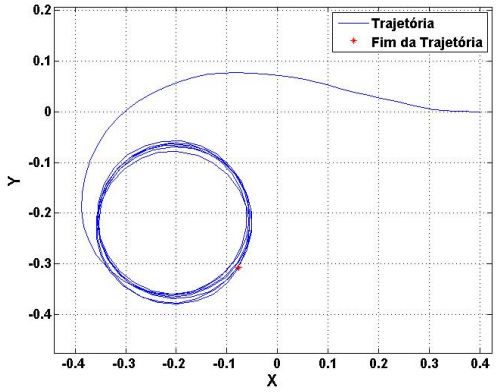

Figura 8 – Trajetória do Robô (R: $15cm$) - Controle Simples

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 9 e 10.

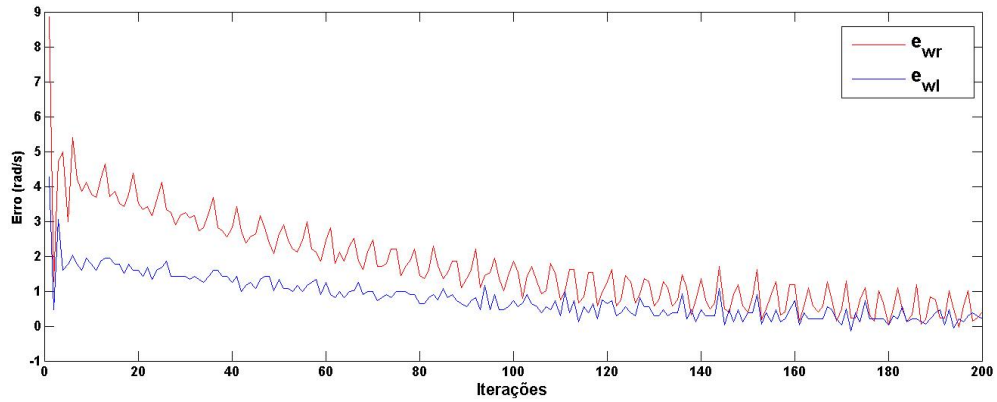

Figura 9 – Erro de velocidade angular - Controle PI: $k_p = 10$,$k_i = 1$ - Raio 15cm

Observando as figuras 7 e 9 é possível notar que ambos os controladores tendem à diminuir o erro de velocidade. Contudo, ao verificar na figura comparativa Figura 11 é possível perceber que o controlador simples converge mais rapidamente que o controlador *PI*. Entretanto, embora o controlador simples convirja mais rapidamente, o que

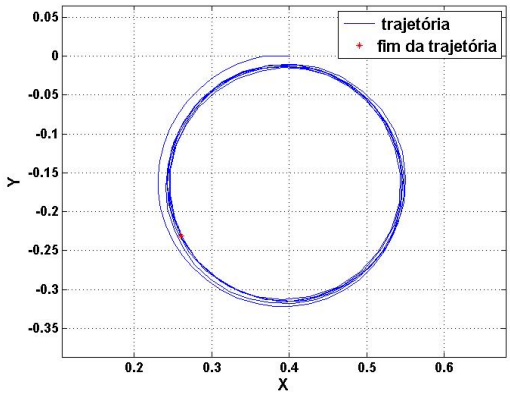

Figura 10 – Trajetória do Robô (R: $15cm$) - Controle PI: $k_p = 10$,$k_i = 1$

se mostra no comparativo da trajetória (Figura 12) 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.

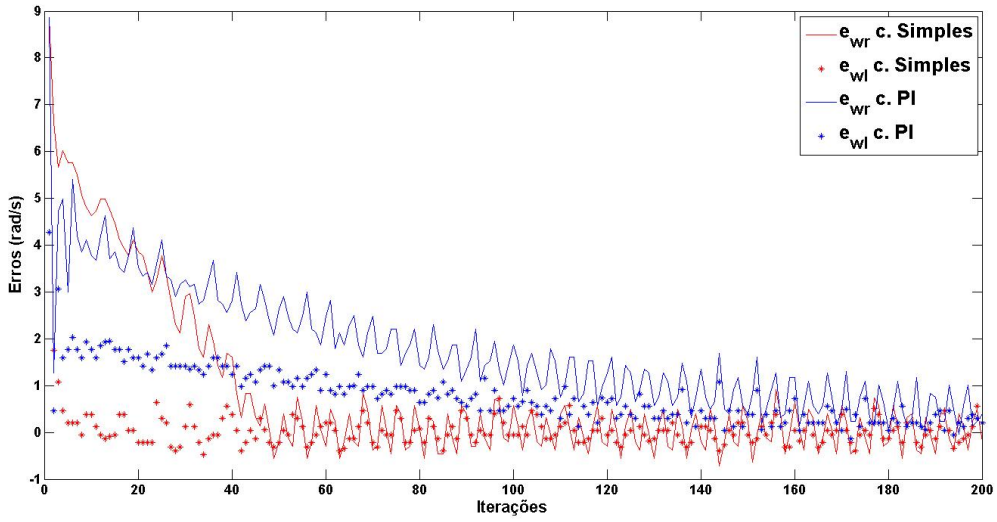

Figura 11 – Comparativo - Erro de velocidade angular - Controle PI e Controle Simples - Raio 15cm

Primeiro, fez-se testes variando o ganho proporcional (k_p). Notou-se, como comprovado pela Figura 13 que ao dobrar k_p, o sistema se tornava instável, como previsto na Seção 3.3, 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 Figura 14. E por fim, foram realizados experimentos variando o raio do perímetro ao redor do alvo, como exemplificado na Figura 15, onde observa-se que a trajetória é suficientemente precisa, conforme o sistema se estabiliza.

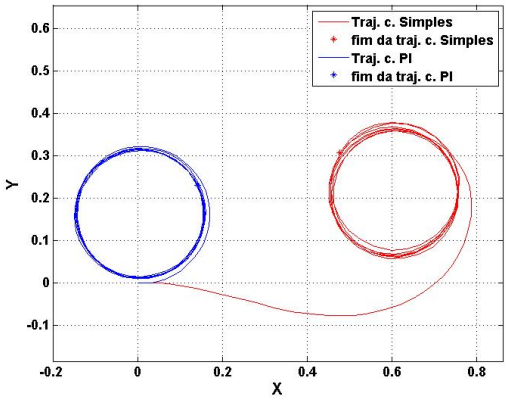

Figura 12 – Comparativo da Trajetória do Robô (R: $15cm$) - Controle PI e Controle Simples

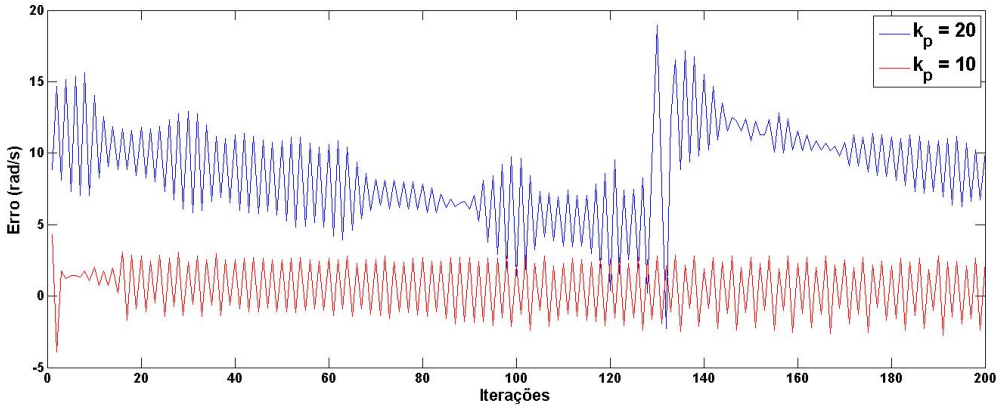

Figura 13 – Erro de velocidade angular - Controle PI: Sistema Instável - Raio 15cm

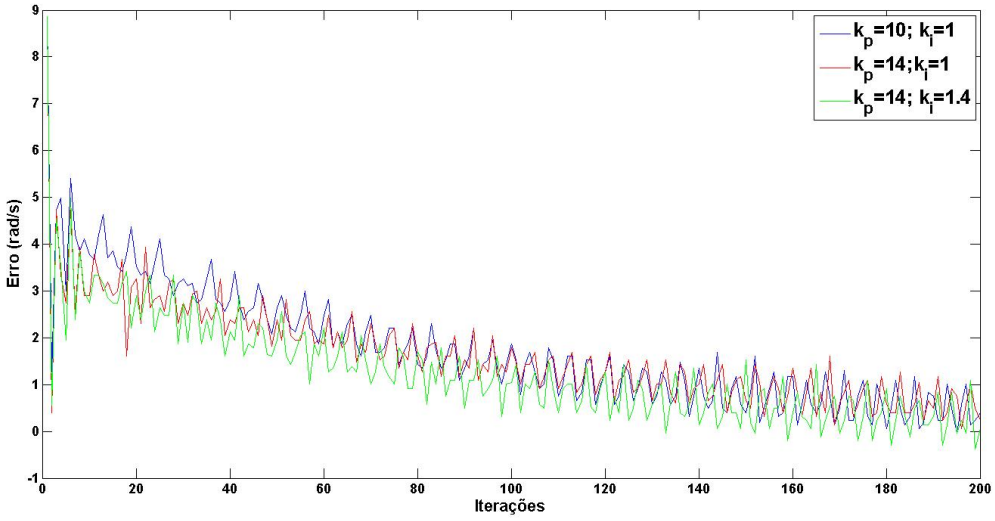

Figura 14 – Comparativo: Erro de velocidade angular - Controladores PI

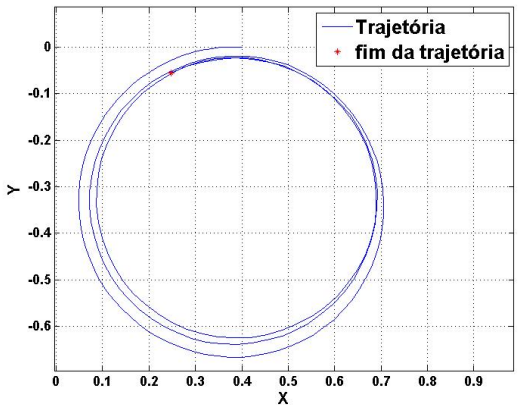

Figura 15 – Trajetória do Controladores PID - Raio $30cm$

7 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.

Na segunda etapa deste trabalho serão realizados mais estudos a respeito das possibilidades de estruturação da rede do *Lego Mindstorms*® e será realizada sua implementação afim de se concluir com o objetivo deste trabalho.

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 7.

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 6.

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, 6 e 7.

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 6.