



Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia


Sender Rocha dos Santos

**Sistemas Inteligentes Adaptativos Aplicados a um Robô
Auto-Equilibrante de Duas Rodas**

Rio de Janeiro
2015

Sender Rocha dos Santos

**Sistemas Inteligentes Adaptativos Aplicados a um Robô Auto-Equilibrante de
Duas Rodas**



Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientadores: Prof. Dr. José Franco Machado do Amaral
Prof. Dr. Jorge Luís Machado do Amaral

Rio de Janeiro

2015

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

S237 Santos, Sender Rocha dos.
Sistemas inteligentes adaptativos aplicados a um robô auto-
equilibrante de duas rodas / Sender Rocha dos Santos. - 2015.
171f.

Orientadores: José Franco Machado do Amaral; Jorge Luís
Machado do Amaral

Dissertação (Mestrado) – Universidade do Estado do Rio de
Janeiro, Faculdade de Engenharia.

1. Engenharia Eletrônica. 2. Sistemas inteligentes de controle
- Dissertações. 3. Inteligência artificial - Dissertações. I. Amaral,
José Franco Machado do. II. Amaral, Jorge Luís Machado do. III.
Universidade do Estado do Rio de Janeiro – Faculdade de
Engenharia. IV. Título.

CDU 004.89:007.52

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial
desta tese, desde que citada a fonte.

Assinatura

Data

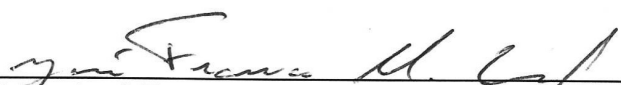
Sender Rocha dos Santos

**Sistemas Inteligentes Adaptativos Aplicados a um Robô Auto-Equilibrante de
Duas Rodas**

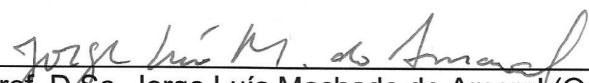
Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em: 25/02/2015


Banca Examinadora:




Prof. D.Sc. José Franco Machado do Amaral (Orientador)
Faculdade de Engenharia – UERJ



Prof. D.Sc. Jorge Luís Machado do Amaral (Orientador)
Faculdade de Engenharia – UERJ



Prof. Ph.D. Pedro Henrique Gouvêa Coelho
Faculdade de Engenharia – UERJ



Prof. D.Sc. Douglas Mota Dias
Departamento de Engenharia Elétrica - PUC-Rio

Rio de Janeiro

2015

DEDICATÓRIA

À minha família e aos Professores.

AGRADECIMENTOS

Aos meus orientadores, Prof. Doutor José Franco Machado do Amaral e ao Prof. Doutor Jorge Luís Machado do Amaral pela amizade, pela orientação, pelos ensinamentos, pela paciência e pela contribuição fundamental para o desenvolvimento deste trabalho.

Aos professores Pedro Henrique Gouvêa e Douglas Mota Dias, pela importante contribuição para a conclusão deste trabalho.

À UERJ e aos seus professores, pelos ensinamentos, dentro e fora da sala de aula.

Aos meus colegas de mestrado, pelo companheirismo e pelo inegável apoio quando necessário.

À minha família, em especial à minha esposa, pelo companheirismo, apoio e dedicação.

A todos aqueles, que embora não citados nominalmente, contribuíram direta e indiretamente para a execução deste trabalho.

RESUMO

SANTOS, Sender Rocha. **Sistemas inteligentes adaptativos aplicados a um robô auto-equilibrante de duas rodas**. 2015. 171f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2015.

Com o avanço no desenvolvimento e utilização de veículos e robôs auto-equilibrantes, faz-se necessário a investigação de controladores capazes de atender os diversos desafios relacionados à utilização desses sistemas. Neste trabalho foi estudado o controle de equilíbrio e posição de um robô auto-equilibrante de duas rodas. O interesse particular nesta aplicação vem da sua estrutura e da riqueza de sua dinâmica física. Por ser um problema complexo e não trivial há grande interesse em avaliar os controladores inteligentes. A primeira parte da dissertação aborda o desenvolvimento de um controle clássico do tipo PID, para em seguida ser comparado com a implementação de dois tipos de controladores inteligentes: *On-line Neuro Fuzzy Control* (ONFC) e *Proportional-Integral-Derivative Neural-Network* (PID-NN). Também é apresentada a implementação dos controladores em uma plataforma de hardware, utilizando o kit LEGO Mindstorm, e numa plataforma de simulação utilizando o MATLAB-Simulink. Em seguida, dois estudos de casos são desenvolvidos visando comparar o desempenho dos controladores. O primeiro caso avalia o controle de equilíbrio e posição do robô auto-equilibrante de duas rodas sobre um terreno plano tendo como interesse observar o desempenho intrínseco do sistema sob ausência de fatores externos. O segundo caso estuda o controle de equilíbrio e posição do robô em terrenos irregulares visando investigar a resposta do sistema sob influência de condições adversas em seu ambiente. Finalmente, o desempenho de cada um dos controladores desenvolvidos é discutido, verificando-se resultados competitivos no controle do robô auto-equilibrante de duas rodas.

Palavras-chave: Robô de duas rodas; Auto-equilibrante; Controle *neuro-fuzzy*; Controle PID; Redes neurais artificiais; Sensores inerciais.

ABSTRACT

SANTOS, Sender Rocha. ***Adaptive Intelligent Systems applied to one two-wheeled robot***. 2015. 171f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2015.

The advances and the development of vehicles and autobalance robots make necessary the investigation of controllers able to meet the various challenges related to the use of these systems. The focus of this work is to study the equilibrium and position control of one two-wheeled robot. The particular interest in this application comes from its structure and its rich physical dynamics. Since this is a complex and non trivial problem, there is great interest in to analyze intelligent controllers. The first part of this dissertation discusses the development of a classic PID controller. Then it is compared with two types of intelligent controllers: On-line Neural Fuzzy Control (ONFC) and Proportional-Integral-Derivative Neural-Network (PID-NN). Also it is presented the implementation of controllers in a hardware platform using the LEGO Mindstorm kit and in a simulation platform using the MATLAB-Simulink. Two case studies are developed. The first one investigates the control of equilibrium and position of two-wheeled robot on a flat terrain to observe the intrinsic performance in lack of external factors. The second case studies the equilibrium and position control of the robot in irregular terrains to investigate the system response under influence of hard conditions in its environment. Finally, the performance of each controller developed is discussed and competitive results in the control of two-wheeled robot are achieved.

Keywords: Two-wheeled robot; Autobalance; Neuro-fuzzy control; PID control; Artificial neural net; Inercial sensors.

LISTA DE FIGURAS

Figura 1 – Sistemas Auto-Equilibrantes: (A) Ballbot (LAUWERS et al., 2006); (B) nBot (NGUYEN et al., 2004); (C) Segway (NGUYEN et al., 2004); (D) JOE (GRASSER et al., 2002).....	22
Figura 2 – Controlador PID para robô auto-equilibrante de duas rodas. (CHAN et al., 2013)	23
Figura 3 – Treinamento supervisionado no controle por rede neural artificial. (CHAN et al., 2013)	24
Figura 4 – Mapa de controladores usados por pesquisadores em robôs auto-equilibrantes sobre duas rodas. (CHAN et al., 2013)	25
Figura 5 –Pêndulos Invertidos. (a) Pêndulo rotacional de um único link (ÅSTRÖM et al., 2000), (b) Carrinho de pêndulo invertido (YOSHIDA, 1999), (c) Pêndulo invertido duplo (FURUTA et al.,1978).....	27
Figura 6 – Plataformas de desenvolvimento e simulação.	33
Figura 7 – Kit lego.	34
Figura 8 – Diagrama de blocos do NXT.	34
Figura 9 – Servomotor NXT. (a) Vista lateral; (b) Vista Interior.	35
Figura 10 – Sensores que acompanham o kit: sensor de toque, sensor de luz, sensor som e sensor ultrassônico.....	36
Figura 11 – NXT Gyro Sensor, da <i>HiTechnic Products</i>	36
Figura 12 – Estrutura do giroscópio de estado sólido XV-3500CB, da <i>Epson Toyocom Corp.</i> e seu princípio de funcionamento.	37
Figura 13 – Software RobotC da Carnegie Mellon University.	38
Figura 14 – Ambiente de desenvolvimento integrado do RobotC.	39
Figura 15 – Ambiente de desenvolvimento Simulink-MATLAB.	40
Figura 16 – Coordenadas do robô (CHAN, 2013).	41
Figura 17 – Diagrama do auto-equilibrante (BAGEANT, 2011).....	43
Figura 18 – Diagrama de corpo livre do corpo do auto-equilibrante.....	44
Figura 19 – Diagrama de corpo livre da roda.	45
Figura 20 –Parâmetros do NXT (DUNG, 2010). (a) vista 3D do robô; (b) visão lateral.	47
Figura 21 – Diagrama de controle do Auto-equilibrante.	50

Figura 22 – Variáveis de entrada e saída do Auto-equilibrante.....	51
Figura 23 – Diagrama de controle de equilíbrio em destaque.	52
Figura 24 – Diagrama do controle PID.	53
Figura 25 – Oscilação sustentada com período crítico P_c	54
Figura 26 – Operação de Auto-equilíbrio.	55
Figura 27 – Variáveis de entrada e saída do Auto-equilibrante para controle de posição.....	56
Figura 28 – Diagrama de controle de posição em destaque.	56
Figura 29 –Operação do controle de posição. (A) Posição inicial; (B) Posição final.	57
Figura 30 –Determinação do ganho K para controle de posição.....	58
Figura 31 – Exemplo do problema de <i>drift</i> do sensor durante o tempo (Witzand, 2009).	59
Figura 32 – Exemplo da medida de uma Unidade de Medida Inercial (Siciliano, 2008).	60
Figura 33 – Gyro estacionário: (A) Giga de teste; (B) Saída do gyro, amostrada a cada 3 milisegundos.....	61
Figura 34 – Relação entre tensão e offset do gyro.....	62
Figura 35 – Robô Auto-equilibrante de 02 rodas. (a) vista frontal; (b) vista traseira ; (c) vista direita; (d) vista esquerda.	63
Figura 36 – Detalhe do sensor.	64
Figura 37 –Bateria Recaregável NXT. (a) Montagem sem bateria recarregável; (b) Montagem com bateria recarregável; (c) Bateria recarregável; (d) Carregador bateria.	65
Figura 38 – Teste do método de integração.....	67
Figura 39 – Processo de cálculo do off-set do gyro.	68
Figura 40 – Auto-equilibrante em operação.	70
Figura 41 – Deslocamento angular do auto-equilibrante em operação.	70
Figura 42 – Velocidade angular do auto-equilibrante em operação.	71
Figura 43 – Deslocamento linear do auto-equilibrante em operação.	71
Figura 44 – Velocidade linear do auto-equilibrante em operação.	72
Figura 45 – Rejeição do robô para distúrbio externo.....	72
Figura 46 – Rejeição à distúrbio externo.....	73
Figura 47 – Plataforma MATLAB-Simulink de simulação do auto-equilibrante NXT.	74
Figura 48 – Variáveis de entrada na plataforma de simulação.	75

Figura 49 – Seleção do tipo de controle na plataforma de simulação.	76
Figura 50 – Malha fechada de controle do auto-equilibrante NXT.	77
Figura 51 – Modelo dinâmico do auto-equilibrante NXT.	78
Figura 52 – Controlador: (A) PID; (B) ONFC; (C) PIDNN.	80
Figura 53 – Resposta dinâmica do sensor gyro.	81
Figura 54 – Modelo do sensor gyro: (A) Bloco Simulink do ruído do sensor gyro. (B) Resposta dinâmica do bloco.	81
Figura 55 – Ganhos de realimentação.	82
Figura 56 – integração.	83
Figura 57 – Derivação.	83
Figura 58 – Implementação do filtro de <i>bias</i> no Simulink e efeito na resposta da velocidade angular.	84
Figura 59 – Sistemática de validação do modelo.	85
Figura 60 – Deslocamento angular: (A) real e simulado para operação de controle PID estacionário; (B) Distribuição de Frequência; (C) Correlação entre simulado e real.	86
Figura 61 – (A) Velocidade angular real e simulado para operação de controle PID estacionário. (B) Distribuição de Frequência. (C) Correlação entre simulado e real.	87
Figura 62 – (A) Deslocamento linear real e simulado para operação de controle PID estacionário. (B) Distribuição de Frequência. (C) Correlação entre simulado e real.	88
Figura 63 – (A) Velocidade linear real e simulada para operação de controle PID estacionário. (B) Distribuição de Frequência. (C) Correlação entre simulado e real.	89
Figura 64 – Representação de uma estrutura NFN.	90
Figura 65 – Representação das funções de pertinência de uma estrutura NFN (CARVALHO, 2010).	91
Figura 66 – Diagrama de blocos do controle com o ONFC.	93
Figura 67 – Estrutura do controlador ONFC com 3 pesos.	93
Figura 68 – Funções de pertinência complementares usadas no controlador ONFC.	94
Figura 69 – Implementação da malha de controle do controlador ONFC para avaliação de controle e análise de parâmetros.	106
Figura 70 – Detalhe do bloco dos pesos do controlador ONFC.	106
Figura 71 – Detalhe do bloco de regularização do controlador ONFC.	107

Figura 72 – Ajuste dos pesos controlador ONFC. (A) Sem regularização; (B) Com regularização.....	108
Figura 73 – Resposta do deslocamento angular para diferentes taxas de aprendizado.....	111
Figura 74 – Ajuste dos pesos do controlador ONFC.....	111
Figura 75 – Deslocamento angular para o controle ONFC estacionário.	112
Figura 76 – Deslocamento linear para o controle ONFC estacionário.	112
Figura 77 – Implementação do filtro de média móvel: (A) teta sem filtro; (B) teta com filtro.	113
Figura 78 – Implementação do filtro de média móvel: (A) pesos sem filtro; (B) pesos com filtro.....	113
Figura 79 – Representação da rede neural artificial multicamadas.....	114
Figura 80 – Estrutura do controlador PIDNN.....	117
Figura 81 – Malha de controle da sintonia <i>on-line</i> (FERRARI, 2011).....	119
Figura 82 – Implementação no Simulink do controlador PIDNN para treinamento <i>off-line</i>	121
Figura 83 – Atualização dos pesos <i>off-line</i> na simulação.....	122
Figura 84 – Comparativo entre o deslocamento angular real do controle PIDNN e do PID.	123
Figura 85 – Malha de controle para o treinamento online.	124
Figura 86 – Implementação no Simulink do controlador PIDNN para treinamento online.....	124
Figura 87 – Variação dos ganhos K_p , K_i e K_d na simulação.	125
Figura 88 – Controle de Equilíbrio dos controladores: (A) ONFC; (B) PIDNN;(C) PID: deslocamento angular.	128
Figura 89 – Controle de Equilíbrio dos controladores ONFC, PIDNN e PID: deslocamento linear.	129
Figura 90 – Comparação entre o desempenho do PID e do ONFC para rejeição à perturbação externa: (A) Resposta do sistema com controlador PID; (B) Resposta do sistema com controlador ONFC.	130
Figura 91 – Atualização em tempo real dos pesos do ONFC durante rejeição à pequena perturbação externa.	131
Figura 92 – Controle de Posição ONFC, PIDNN e PID: deslocamento linear.	132
Figura 93 – Controle de Posição ONFC, PIDNN e PID: deslocamento angular.....	132

Figura 94 – Exemplo subida em rampa por um auto-equilibrante (CHAN et al., 2013).	134
Figura 95 – Exemplo de subida sobre um degrau por um auto-equilibrante (CHAN et al., 2013).	134
Figura 96 – Operação do auto-equilibrante em terreno inclinado. (A) rampa subida; (B) terreno plano; (C) rampa descida.	135
Figura 97 – Ajuste dos pesos do controlador ONFC durante deslocamento em terreno inclinado.	136
Figura 98 – Saída de controle dos controladores PID, PIDNN e ONFC durante operação em terreno inclinado.	137
Figura 99 – Deslocamento angular dos controladores PID, PIDNN e ONFC durante operação em terreno inclinado.	137
Figura 100 – Deslocamento linear dos controladores PID, PIDNN e ONFC durante operação em terreno inclinado.	138
Figura 101 – Subida sobre um degrau pelo auto-equilibrante.	139
Figura 102 – Saída dos controladores durante subida ($t=17s$) sobre um degrau.	139
Figura 103 – Ajuste dos pesos do controlador ONFC durante subida sobre um degrau.	140
Figura 104 – Deslocamento linear durante subida sobre um degrau.	140
Figura 105 – Deslocamento angular controle ONFC e controle <i>neuro-fuzzy</i> (Su et al., 2010).	141
Figura 106 – Robô auto-equilibrante de duas rodas (Su et al., 2010).	142
Figura 107 – Desempenho do NXT para diferentes condições iniciais.	143
Figura 108 – Resposta do deslocamento angular para diferentes condições iniciais.	143
Figura 109 – Interferência do cabo de comunicação durante operação do NXT.	144

LISTA DE TABELAS

Tabela 1 – Parâmetros do NXT.....	48
Tabela 2 – Parâmetros e equações do NXT.	49
Tabela 3 – Ganho das variáveis.....	50
Tabela 4 – Ganho crítico e período crítico.	53
Tabela 5 – Ganho PID inicial pelo método de Ziegler-Nichols.	54
Tabela 6 – Ganho PID final.	55
Tabela 7 – Análise de robustez da sintonia do γ em relação aos parâmetros α e o universo de discurso.	109
Tabela 8 – Análise de robustez da sintonia do α em relação aos parâmetros γ e o universo de discurso.	109
Tabela 9 – Análise do universo de discurso em relação aos parâmetros α e γ	109
Tabela 10 – Análise de robustez da sintonia do α em relação aos parâmetros γ e o universo de discurso.	110
Tabela 11 – Pesos sintonizados pelo treinamento off-line.	122
Tabela 12 – Pesos sintonizados pelo treinamento online.	126
Tabela 13 – Comparativo entre as sintonias dos ganhos dos controladores.	133

LISTA DE ABREVIATURAS E SIGLAS

MATLAB	<i>Matrix Laboratory®</i>
ONFC	<i>On-line Neural Fuzzy Control</i>
PID	<i>Proportional-Integral-Derivative</i>
PIDNN	<i>Proportional-Integral-Derivative Neural Network</i>
SCARA	<i>Selective Compliance Assembly Robot Arm</i>

SUMÁRIO

	INTRODUÇÃO	18
1	SISTEMA AUTO-EQUILIBRANTE	27
1.1	O mecanismo de auto-equilíbrio	28
1.2	O problema do pêndulo invertido	28
1.3	Aplicações	29
1.4	Próximos desafios	30
2	REQUISITOS DE PROJETO E RECURSOS NECESSÁRIOS	33
2.1	Plataforma de desenvolvimento de hardware.....	33
2.2	Plataforma de desenvolvimento de software: ROBOTC	37
2.3	Plataforma de simulação: MATLAB-Simulink	40
3	MODELAGEM MATEMÁTICA E ESTRATÉGIA DE CONTROLE	41
3.1	Modelagem matemática	41
3.1.1	<u>Considerações iniciais</u>	42
3.1.2	<u>Modelagem física</u>	42
3.1.3	<u>Desenvolvimento das equações</u>	46
3.2	Parâmetros da planta	47
3.3	Sistema de controle	49
3.3.1	<u>Controle de equilíbrio</u>	52
3.3.2	<u>Controle de posição</u>	55
4	DESENVOLVIMENTO, PROGRAMAÇÃO E VALIDAÇÃO	59
4.1	Caracterização do sensor gyro	59
4.2	Montagem mecânica do robô	63
4.3	Código de controle e programação.....	65
4.4	Testes e validação experimental	69
5	DESENVOLVIMENTO DE AMBIENTE DE SIMULAÇÃO	74
5.1	Modelagem computacional.....	74
5.2	Testes e validação do modelo computacional.....	85
6	CONTROLE INTELIGENTE	90
6.1	Controle <i>neuro fuzzy</i>	90
6.1.1	<u>Determinação da ação de controle</u>	93
6.1.2	<u>Ajuste de pesos</u>	95
6.1.3	<u>Análise da convergência do ONFC</u>	97

6.1.4	<u>Características de desempenho</u>	98
6.1.5	<u>Regularização dos pesos</u>	101
6.1.6	<u>Avaliação da influência dos parâmetro alfa e gama no desempenho do sistema</u>	104
6.1.7	<u>Implementação do controle ONFC</u>	105
6.2	Controle neuro artificial	114
6.2.1	<u>Equações de ajuste dos pesos do PIDNN</u>	118
6.2.2	<u>Implementação do controle PIDNN</u>	121
6.3	Estudo de Caso 1	126
6.3.1	<u>Controle de Equilíbrio em terreno plano</u>	126
6.3.2	<u>Controle de Posição em terreno plano</u>	131
6.4	Estudo de Caso 2	133
6.4.1	<u>Controle em terreno inclinado</u>	135
6.4.2	<u>Controle em terreno com degrau</u>	138
6.5	Discussão dos resultados	141
7	CONCLUSÕES	145
7.1	Recomendações de trabalhos futuros	146
	REFERÊNCIAS	148
	ANEXO: código de programação no RobotC	157

INTRODUÇÃO

Robôs autônomos estão desempenhando um papel cada vez mais importante na indústria para atender a alta demanda de sistemas automatizados, e espera-se deles a capacidade de detectar informações sobre o ambiente, processar a informação, e realizar ações adequadas para uma grande quantidade de tarefas.

No que diz respeito aos acionamentos robóticos, um dos grandes desafios para esses robôs é que as técnicas de controle tradicionais geralmente exigem um modelo matemático preciso do sistema e do seu ambiente, e uma modelagem inadequada irá, naturalmente, causar efeitos negativos relacionados ao desempenho. As técnicas de Inteligência Computacional vêm sendo empregadas, particularmente a computação neural (MILLER, 1990) (LEWIS, 1998), (KIMURA et al., 2007), a computação evolucionária (SIM et al., 2002) (DAVIDOR, 1991) e a lógica *fuzzy* (LEE, 1990), pois são poderosas ferramentas para o desenvolvimento de um sistema de controle melhor e mais eficiente sem a necessidade de modelos matemáticos precisos. Em especial, técnicas adaptativas podem, a partir de uma estrutura geral de controle com parâmetros associados ajustáveis, obter métricas de desempenho relevantes. Tais métricas podem vir a cobrir os requisitos de velocidade e acurácia, assim como outras importantes especificações, tais como: estabilidade, confiabilidade e segurança.

O projeto de controladores robóticos por técnicas tradicionais de controle costuma ser baseada na construção de um modelo do sistema e do seu ambiente, para, então, usar uma técnica de planejamento adequada para o controlador executar e realizar as tarefas desejadas num ambiente estruturado. Tipicamente, isto poderia envolver controle de posição e forças exercidas por um manipulador robótico, com restrições na trajetória e com suavidade nos movimentos. Claramente, tais métodos baseados somente em modelos não são adequados para robôs autônomos que trabalham em ambientes dinâmicos e desconhecidos, e que lidam com fatores tais como variações de carga imprevistas, degradação da planta, variação dinâmica de obstáculos em tempo real, ambientes não estruturados, variação da localização de objetos e etc. Sendo assim, a abordagem baseada em sensores torna-se uma alternativa comum. O robô coleta os dados do seu ambiente em tempo real e processa esses dados para gerar ações adequadas. A evidente

desvantagem de tal método baseado somente em sensoriamento é que o robô necessita captar dados precisos no ambiente desconhecido.

O filtro de Kalman provê uma boa técnica para lidar com o problema de controle robótico (GREWAL & ANDEWS, 1993). O filtro linear de Kalman é muito atraente para o projeto de controladores para sistemas dinâmicos lineares devido à sua simplicidade e baixo custo computacional. Sua desvantagem é que ele só pode ser usado efetivamente para sistemas lineares. O filtro de Kalman não linear pode ser usado no acompanhamento da dinâmica de robôs, mas ainda assim é muito difícil estimar a dinâmica e os ambientes desconhecidos num robô autônomo. Isto porque há incerteza e o ambiente desconhecido é, geralmente, não linear, com ruído não-gaussiano nos dados do sensor e o filtro de Kalman não garante a convergência. Técnicas inteligentes, entretanto, podem ser prontamente aplicadas em ambientes com ruídos, não linearidades e não estacionário (BULLINARIA & LI, 2005).

Uma técnica inteligente muito utilizada é a rede neural artificial cujo poder computacional é bem entendido do ponto de vista teórico. Um ponto importante é que a rede neural pode processar entradas de várias fontes de diferentes tipos e aprender como usar melhor esses recursos para produzir ações apropriadas. Também apresenta robustez contra ruídos, degradação e pode generalizar bem de seu conjunto de treinamento para operar em novas situações. O processamento paralelo pode também agilizar sua capacidade computacional, e fazer esta técnica mais viável para controle em tempo real (HAM & KOSTANIC, 2001).

A ideia básica do controle por rede neural tem sido atualmente usada em várias aplicações reais no mundo e descrições detalhadas vêm sendo publicadas. Quatro exemplos indicam a gama de possibilidades: Walter & Schulten (1993) desenvolveram dois algoritmos para controle de servovisão de um robô industrial Puma 562, um usando uma rede “*neural-gas*”, e o outro baseado num mapa auto-organizável. As redes neurais codificam a transformação entre as coordenadas da imagem de entrada e os ângulos das juntas do robô. Esses sistemas foram capazes de aprender como posicionar o efetuador do robô usando a informação obtida somente de um par de câmeras. Em (ER & LIEW, 1997) é apresentado um sistema de controle neural para um Robô SCARA (*Adept One SCARA Robot*) usando ângulos das juntas, velocidades angulares e erros de posição como entradas das

redes, com comandos de motor para os vários atuadores nas saídas, e mostrou as vantagens desse sistema. Clark & Mills (2000) descrevem a questão técnica de se usar rede neural em tempo real aprendendo num robô CRS *Robotics Corporation A460 robot* com 6 graus de liberdade e mostrou como a rede neural poderia prover um sinal para compensar o erro permanente no sistema controlador PID (Proporcional-Integral-Derivativo). Gong & Yao (2001) apresenta a formulação de algoritmos de controle robusto adaptativo para sistemas não lineares e aplicou-os para um motor linear, tendo cuidado particular para evitar os efeitos de desestabilização potencial do aprendizado *on-line*.

Apesar do sucesso da abordagem por redes neurais, essa técnica tem algumas limitações. O aprendizado baseado em gradiente pode ser difícil de configurar, em particular, as taxas de aprendizado (CLARK & MILLS, 2000) e os valores dos parâmetros iniciais (BULLINARIA, 2003). Cabe ressaltar que existe ainda a possibilidade do processo de aprendizado terminar num mínimo local.

Os sistemas baseados em Lógica *Fuzzy* têm demonstrado sua capacidade de resolver diversos tipos de problemas em várias aplicações, inclusive em robótica. Existe um interesse crescente em torná-los ainda melhores dotando-os de uma capacidade de aprendizado, pois isto viabiliza a incorporação de ambos os tipos de conhecimento (subjetivo e objetivo) de um modo matemático unificado (MENDEL, 1995). Muitas aplicações robóticas de sistemas *fuzzy* apresentam hibridizações com outras técnicas inteligentes, em particular, com redes neurais como no sistema NEFCON (*Neural Fuzzy Controller*) (NÜRNBERGER et al., 1999) e com Algoritmos Genéticos como no SFE (Sistema Fuzzy Evolucionário) (AMARAL, 2003).

A computação evolucionária trata os problemas pelo emprego de ideias chave da evolução biológica pela seleção natural. Há uma grande variedade de formas em que tais processos evolucionários podem ser implementados para aplicações práticas, como em (EIBEN & SMITH, 2003). A ideia geral é manter uma população de sistemas (exemplo, rede neural ou sistemas de controle tradicionais), em que cada um é especificado por um “genótipo” consistindo de um conjunto de parâmetros representados de modo conveniente. Então, dependendo dos valores dos parâmetros, cada indivíduo terá uma “aptidão” associada. Os melhores indivíduos (sistemas) em cada geração tem mais chance de serem selecionados para gerar a prole, usando formas apropriadas de operadores de cruzamento e mutação. Os

tipos de cruzamento e mutação irão depender do problema e de como os parâmetros são representados. A técnica evolutiva tende a obter um bom indivíduo (sistema) adequado para solução do problema após certo número de gerações.

Dois exemplos de uso da abordagem evolucionária para controle robótico industrial ilustram o que pode ser alcançado: Moriarty & Miikkulainen (1996) evoluíram o comportamento para desvio de obstáculo de um controle por rede neural de um braço robótico OSCAR-6 que recebeu visão e sensor de entrada. Duas redes neurais foram desenvolvidas: uma permitia que o braço robótico se aproximasse do alvo, e a outra permitia movimentos finais mais precisos. A evolução de todos os sistemas foi guiada por uma medida simples do melhor indivíduo sobre a tarefa inteira, envolvendo ambas as habilidades de alcançabilidade do alvo e desvio de obstáculos. Em (GÓMEZ & EGGENBERGER, 2004) foi evoluído um sistema de rede neural para controlar um manipulador movendo objetos coloridos, usando representação de cores de entrada e detecção de movimento e informação do ângulo da junta. Experimentos foram realizados com sucesso num manipulador robótico Mitsubishi MELFRA RV-2JA, com seis graus de liberdade e um sistema de visão estéreo.

Outro exemplo de hibridização são os Sistemas *Fuzzy-Genéticos* que são basicamente sistemas *fuzzy* com procedimento de aprendizado baseado em Algoritmos Genéticos. Em (AMARAL, 2003) é apresentado um projeto de uma aplicação mecatrônica no qual um controlador *fuzzy* é desenvolvido para controlar um dos servomotores de um braço robótico. O sistema recebe sinais provenientes de dois sensores de entrada que medem a intensidade luminosa e sua saída aciona um servomotor. O objetivo é fazer com que o sistema “procure” a luz.

Uma área que merece destaque dentro da pesquisa em robótica é a robótica móvel. Devido à necessidade de operar em ambientes desconhecidos, os robôs móveis necessitam de um nível muito maior de inteligência para que possam aprender a se adaptar com sucesso no seu ambiente dinâmico. Inteligência num robô móvel é considerada como um comportamento adaptativo que faz um robô adaptar-se e agir inteligentemente em seu ambiente.

Na última década começou a crescer o interesse pelos robôs auto-equilibrantes de duas rodas (TAKAHASHI, OGAWA & MACHIDA, 2000), (DEEGAN, THIBODEAU & GRUPEN, 2006), (STILMAN, OLSON & GLOSS, 2010). Muito desse

interesse é inspirado pela Plataforma Robótica Móvel Segway (NGUYEN et al., 2004).

Existem robôs que se equilibram em duas rodas tais como o *Segway Robotic Mobility Platform* (RMP) (NGUYEN et al., 2004) (figura 1-C), o nBot (NGUYEN et al., 2004) (figura 1-B) e o JOE (GRASSER et al., 2002) (figura 1-D). Existem também robôs holonômicos, tais como o Ballbot, que se equilibram em torno de uma única bola esférica (LAUWERS et al., 2006) (figura 1-A).



Figura 1 – Sistemas Auto-Equilibrantes: (A) Ballbot (LAUWERS et al., 2006); (B) nBot (NGUYEN et al., 2004); (C) Segway (NGUYEN et al., 2004); (D) JOE (GRASSER et al., 2002).

Controladores PID são muito utilizados individualmente ou em conjunto com outros controladores. Eles são mundialmente usados em muitas áreas na indústria, porque somente exigem a sintonia dos três ganhos. Robôs auto-equilibrantes de duas rodas tem números de estados: um para o deslocamento angular, um para velocidade linear e, se for aplicável, um para rotação. Os ganhos PID podem ser razoavelmente definidos somente por experimentação. No entanto, o método de Ziegler-Nichols pode ser usado para um ajuste inicial dos ganhos, como mostra (NASIR, 2011). Nasir (2011) usa dois controladores PID separados: um para velocidade longitudinal e um para o deslocamento angular (figura 2).

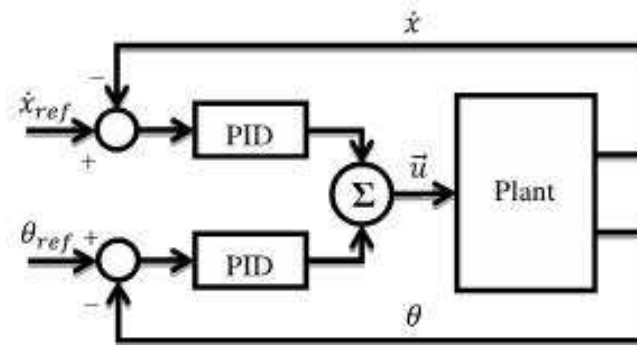


Figura 2 – Controlador PID para robô auto-equilibrante de duas rodas. (CHAN et al., 2013)

O controle *fuzzy* é aplicado em sistemas não lineares. Dependendo do número de conjuntos *fuzzy*, ele pode aproximar qualquer lei de controle não linear. Entretanto, devido à natureza da lei de controle, a estabilidade por Lyapunov pode não ser diretamente provada. Devido à complexidade do sistema auto-equilibrante de duas rodas, o número de conjuntos *fuzzy* necessários pode ser grande. Portanto, o sistema normalmente consistirá de duas malhas fechadas de controle *fuzzy*: uma para o controle de equilíbrio e outra para o controle de posição/ velocidade; similar a topologia do controle PID da figura 2. Cada controle *fuzzy* controla somente dois estados como em (CHIU & PENG, 2006). Caso considere-se a rotação do robô, o sistema consistirá de três malhas fechadas de controle *fuzzy* (NASIR et al., 2011). (WU, 2011) usou um controlador *fuzzy* na malha de controle de posição, e um controle PID para o controle de equilíbrio.

Redes neurais artificiais são muito utilizadas para a estratégia de controle adaptativa para robôs auto-equilibrantes de duas rodas. Redes neurais podem aproximar e controlar sistemas não lineares. Para treinar os pesos dos neurônios, um treinamento supervisionado é necessário (figura 3). O equilíbrio vai melhorando ao longo do tempo com o aprendizado da rede neural.

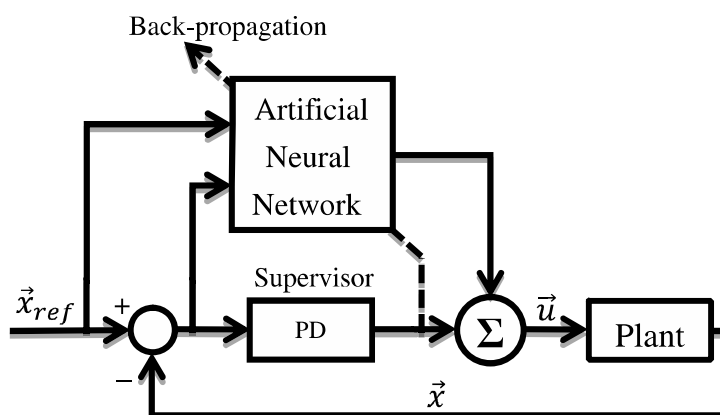


Figura 3 – Treinamento supervisionado no controle por rede neural artificial. (CHAN et al., 2013)

A rede neural de base radial (em que a função de ativação é uma função de base radial) foi utilizada por Tsai, Huang e Lin (2010) para uma *scooter* de duas rodas. A função de base radial proporciona algum grau de generalização local. Tsai (2010) utilizou uma rede neural para auto-equilíbrio, e outra rede neural separada para o controle de rotação. Atualmente a rede neural promove ações de controle para baixas velocidades. Não está claro se a rede neural terá um bom desempenho para altas velocidades.

Em muitos casos, os pesquisadores tem somente investigado uma única rede neural e estratégia de controle, então não é claro como cada tipo de rede neural pode ser comparado com outra rede neural e/ou estratégia de controle, apesar de que algumas tendências gerais podem ser inferidas do uso dessas redes neurais em outros problemas.

Como mostra a figura 4, uma grande variedade de controladores têm sido investigados para o robô auto-equilibrante de duas rodas (CHAN et al., 2013). Resultados experimentais são particularmente escassos para controles não lineares. Mesmo que simulações possam ser usadas para mostrar robustez a distúrbios e incertezas nos modelos, podem haver problemas durante a implementação na prática, tais como: ruído do sensor, tempo de amostragem e o atraso na resposta do sistema. Dados experimentais deveriam ser melhores em relação ao desempenho para mostrar a robustez dos controladores para estes efeitos.

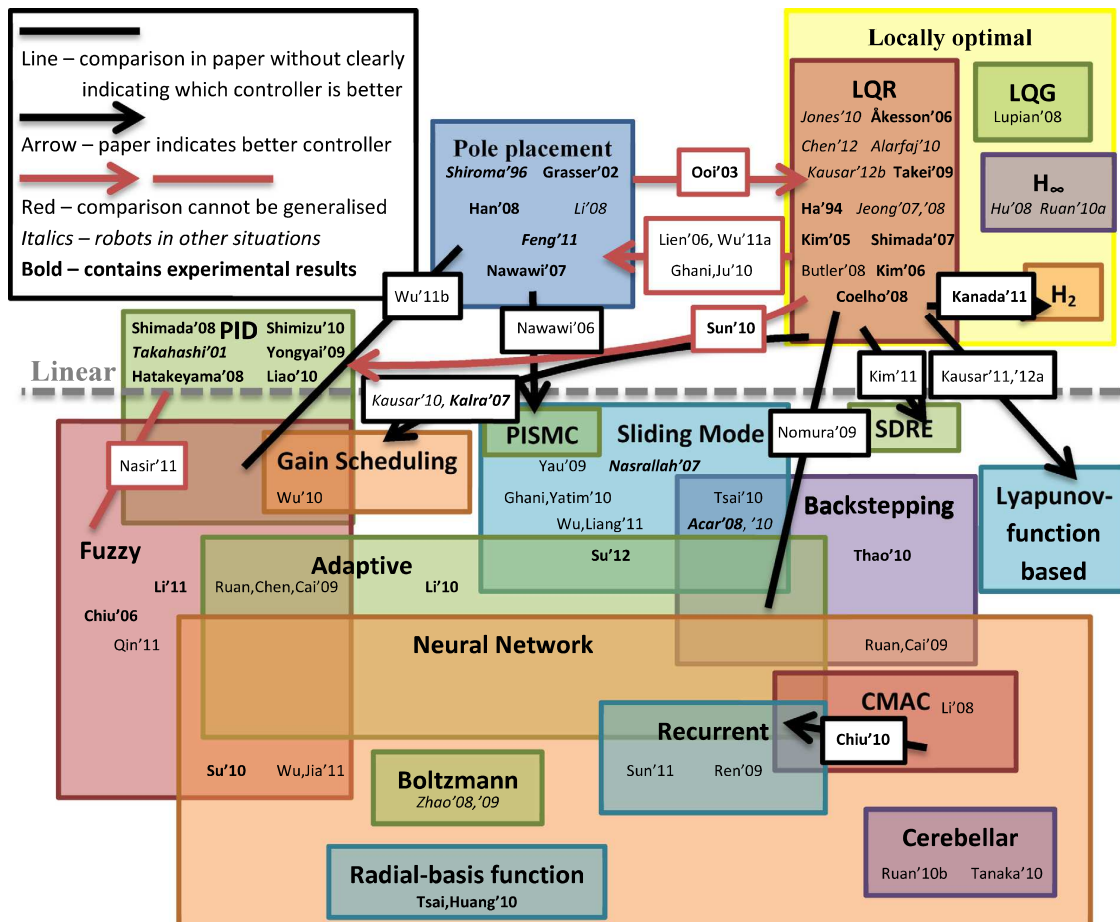


Figura 4 – Mapa de controladores usados por pesquisadores em robôs auto-equilibrantes sobre duas rodas. (CHAN et al., 2013)

A proposta deste trabalho é implementar algoritmos utilizando técnicas de inteligência computacional para controle de um sistema robótico auto-equilibrante de duas rodas capaz de percorrer uma trajetória em terrenos com certo grau de irregularidade.

O desenvolvimento dos algoritmos contempla uma avaliação de problemas práticos no projeto de controladores robóticos, em especial, no que diz respeito à combinação de simplicidade e robustez. O aspecto adaptativo inteligente deve conferir uma adaptação automatizada para ser aplicada aos parâmetros do controle inteligente do sistema robótico. A utilização de técnicas visa a melhorar o desempenho do controlador robótico.

Objetivos

O foco principal do presente trabalho é o desenvolvimento experimental e por simulação de sistemas inteligentes para o controle de um sistema robótico auto-equilibrante de duas rodas autônomo. Especificamente, serão estudados um controle *neuro-fuzzy* e um controle híbrido com rede neural artificial.

Tais sistemas deverão ser capazes de:

- manter automaticamente o equilíbrio robótico;
- manter o equilíbrio numa trajetória em terrenos com alguma irregularidade.

O objetivo final é comparar o desempenho das propostas híbridas apresentadas no controle de acionamentos robóticos com técnicas de controle clássicas, em especial o controle PID, e desenvolver ambientes que viabilizem a avaliação dos novos algoritmos de acionamento robótico para robôs auto-equilibrantes. Tais ambientes poderão ser usados em pesquisas e no ensino de pós-graduação em Engenharia, bem como encorajar estudantes e profissionais a explorar a utilização de técnicas inteligentes para alcançar resultados melhores em suas aplicações.

Estrutura da dissertação

O capítulo 2 apresenta os principais requisitos de projeto e os recursos necessários. No capítulo 3 é detalhado o desenvolvimento do modelo matemático, os parâmetros e a estratégia de controle do robô estudado. No capítulo 4 são apresentadas a característica do sensor utilizado, a montagem mecânica do auto-equilibrante, o código de programação e os testes e validação experimental. No capítulo 5, a modelagem computacional e sua validação são abordadas. O foco do capítulo 6 são os controles inteligentes e estudos de casos são apresentados e discutidos. Finalmente, no capítulo 7 temos as conclusões e proposta de trabalhos futuros são apresentadas.

1 SISTEMA AUTO-EQUILIBRANTE

Muitos sistemas robóticos são considerados como referência na bibliografia e frequentemente são usados na pesquisa e ensino da teoria de controle. Eles são tipicamente utilizados para realizar modelos experimentais, validar a eficiência de novas técnicas de controle e verificar sua implementação. Os sistemas robóticos mais comuns são o Acrobot (SPONG, 1995), o Pendubot (FANTONI et al., 2000), o Pêndulo Furuta (ACOSTA, 2010), o pêndulo invertido (FURUTA et al., 1992), o Pêndulo Roda de Reação (BLOCK et al., 2007), o bicicleta (ÅSTRÖM, et al., 2005), o sistema Beam-and-Ball (ANDREEV et al., 2002), e o Tora (JANKOVIC, 1996).

Apesar de sua estrutura simples, o pêndulo invertido é considerado, entre os últimos exemplos, uma das referências mais importantes na área de controle e robótica (BOUBAKER, 2012). Diferentes versões de modelos de pêndulo invertido existem oferecendo uma variedade de desafios interessantes para a teoria de controle. Os tipos mais familiares são os pêndulos rotacionais de um único link (ÅSTRÖM et al., 2000), o carrinho de pêndulo invertido (YOSHIDA, 1999), e o pêndulo invertido duplo (FURUTA et al., 1978), conforme mostra a figura 5.

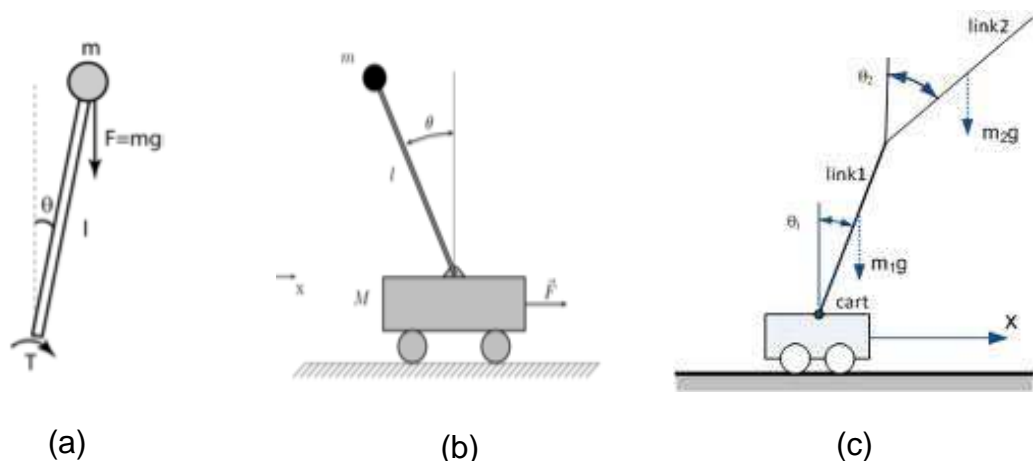


Figura 5 –Pêndulos Invertidos. (a) Pêndulo rotacional de um único link (ÅSTRÖM et al., 2000), (b) Carrinho de pêndulo invertido (YOSHIDA, 1999), (c) Pêndulo invertido duplo (FURUTA et al., 1978).

As versões menos comuns são o pêndulo rotacional de 2-links (BORTOFF, 1996), o pêndulo invertido duplo tipo paralelo (TSAI et al., 2007), o pêndulo invertido triplo (FURUTA et al., 1984), o pêndulo invertido quádruplo (LI et al., 2002) e o pêndulo esférico ou 3D (SHEN et al., 2004).

1.1 O mecanismo de auto-equilíbrio

O mecanismo de auto-equilíbrio é uma das características mais importantes dos sistemas dinâmicos, pois requer uma postura de equilíbrio estável para desenvolver determinadas tarefas. O corpo humano é um bom exemplo de uma estrutura que usa mecanismos de auto-equilíbrio em todos os dias da vida para ficar em pé, caminhar e correr.

Recentemente, habilidades de auto-equilíbrio humano foram adotadas para sistemas dinâmicos. Um robô que imita um mecanismo de auto-equilíbrio humano é chamado de um robô humanóide (SEKIGUCHI et al., 2007). Se o mecanismo de auto-equilíbrio falha o sistema torna-se instável. Portanto, mecanismos robustos de auto-equilíbrio são extremamente importantes em sistemas dinâmicos e requerem um sofisticado mecanismo de controle baseado em técnicas de sensoriamento adequado.

O conceito de auto-equilíbrio de um robô é baseado no modelo do pêndulo invertido. Este modelo tem sido mundialmente usado por pesquisadores de todo o mundo no projeto e controle de rodas, pernas robóticas, etc. (GOHER et al., 2010).

1.2 O problema do pêndulo invertido

O modelo do pêndulo invertido pode ser considerado um sistema robótico mais simples, com somente um corpo rígido e somente uma junta rotacional.

O objetivo do sistema de pêndulo invertido é controlar tanto o ângulo de equilíbrio quanto a posição desejada. O controle do sistema de pêndulo invertido é bastante desafiador, já que uma única força de entrada tem que controlar ângulo e posição, como numa configuração de uma única entrada com múltiplas saídas (LEE et al., 2013).

O pêndulo invertido é um típico sistema dinâmico não linear incluindo um ponto de equilíbrio estável, que se refere ao pêndulo quando se encontra na posição de repouso para baixo, e um ponto de equilíbrio instável quando o pêndulo encontra-se na posição vertical para cima.

A estrutura do pêndulo invertido é bem simples, o que explica o desenvolvimento de muitos modelos virtuais e aplicações de dispositivos reais (BOUBAKER, 2012).

1.3 Aplicações

A grande possibilidade de aplicação da tecnologia derivada deste sistema instável gerou interesse de muitos pesquisadores e entusiastas robóticos em todo o mundo. Nos últimos anos, pesquisadores aplicaram a ideia de um modelo de pêndulo invertido móvel para vários problemas, tais como: projetos de caminhada por marcha para robôs humanóides, cadeiras de rodas robóticas e sistemas de transporte pessoais (KIM et al., 2005).

Recentemente, o conceito de um sistema de pêndulo invertido móvel foi aplicado para robôs de serviço comercial tal como o veículo de transporte pessoal chamado *Segway*. Uma das razões de usar um sistema pêndulo invertido móvel como um veículo de transporte deve-se ao fato de serem melhores para manobras em espaços estreitos (AMBROSE et al., 2004).

O sucesso comercial do *Segway* atraiu a atenção de muitos pesquisadores para desenvolver e controlar sistemas de pêndulo invertido móveis. O movimento do *Segway* é controlado baseado na mudança do centro de gravidade pelo deslocamento do corpo e através da manipulação pelas mãos. Portanto, o controle de direção do *Segway* não é fácil para iniciantes que desejam estabilizar o veículo.

O pequeno robô JOE também demonstrou sucesso no desempenho do controle de auto-equilíbrio (GRASSER et al., 2002). O controle do sistema de pêndulo invertido móvel baseado no modelo do sistema foi apresentado em (PATHAK et al., 2005) e um observador de perturbação foi projetado para controlar o sistema pêndulo invertido móvel (JEONG et al., 2007). O controle de interação de

um sistema de pêndulo invertido móvel com outros objetos foi apresentado em (SASAKI et al., 2008).

Vários robôs auto-equilibrantes chamados BallBots tem sido desenvolvidos e controlados com vários algoritmos de controle. O BallBot I e II da série BallBot, têm uma simples estrutura de uma haste delgada para o pêndulo com um mecanismo de equilíbrio de duas rodas. O controle do BallBot I e II foi alcançado com controladores lineares através da ajuda de um controlador por rede neural. Uma rede *perceptron* multicamada é usada e implementada num DSP série 6000 para o BallBot I (KIM et al, 2008) e uma rede de função de base radial é usada e implementada no DSP 2812 para BallBot II (NOH et al., 2008).

1.4 Próximos desafios

Nesta sessão, são apresentados possíveis trabalhos futuros na área de auto-equilíbrio robótico que poderão ser desenvolvidos e os principais problemas ainda não solucionados (BOUBAKER, 2013). Um dos problemas mais complexos é a estabilização global assintótica da origem por uma realimentação contínua para sistemas híbridos. Para o caso do pêndulo invertido, o problema pode ser aproximado pelo projeto de um único controlador para realizar tanto a subida do pêndulo quanto o controle de estabilização. Uma solução completa para este problema crucial foi recentemente proposto em (GORDILLO et al., 2008).

Para sistemas não lineares, a manipulação com atrasos, a dinâmica interna instável e a saturação do atuador são também considerados como problemas desafiadores. Para o pêndulo de um único braço rotacional, o problema de regulação da saída sobre certas condições na presença de efeitos de *backlash* é resolvido em (GORDILLO et al., 2008) usando uma arquitetura híbrida, que combina sistemas de lógica *fuzzy* Tipo-1 e Tipo-2 e algoritmos genéticos. Uma solução para o mesmo problema para o carrinho de pêndulo invertido é proposto em (CÁZAREZ & CASTRO et al., 2010) usando uma linearização de entrada e saída e controle de energia.

O atrito é um fenômeno muito complexo e que está presente em todos os sistemas de controle. Quando os efeitos de atrito não são modelados, controladores

realimentados podem falhar. De fato, atritos são usualmente associadas com comportamentos oscilatórios ou em ciclos limites. Dessa forma, existem poucos resultados, como por exemplo, (CAMPBELL et al., 2008).

Em relação à questão da ausência de um método matemático da abordagem do controle *fuzzy*, provendo estabilidade e robustez, uma solução para este problema foi recentemente proposta em (CASTILLO et al., 2008), onde os controladores lógicos *fuzzy* Tipo-1 e Tipo-2 são baseados numa teoria de Lyapunov. Primeiro, o problema de controle é resolvido para o pêndulo de um link rotacional sem o efeito gravitacional para provar as condições de estabilidade e então condições robustas são provadas para sistemas não lineares com o efeito gravitacional. Uma extensão desta abordagem para sistemas mecânicos não suaves é encontrada em (CÁZAREZ & CASTRO et al., 2012).

Baseado no princípio da estabilização do pêndulo invertido, muitas estratégias de controles robóticos estão sendo desenvolvidas. As mais importantes e recentes realizações incluem o projeto de controle de sistemas subatuados (ORTEGA et al., 2002), pêndulos invertidos sobre rodas (YOUNIS et al., 2002) e geração do modelo de marcha para robôs humanóides e bípedes (AOYAMA et al., 2011). Para tais problemas de controle complexo, novas abordagens de controle híbrido, tais como abordagens de controle *neuro-fuzzy* (CARDENAS & MACIEL et al., 2011), estão crescendo. A aplicação de rede neural tem aumentado nos últimos anos na teoria de controle, uma vez que reduz o custo de desenvolvimento de controladores para sistemas complexos, particularmente sistemas não lineares (PEZESHKI et al., 2012).

Para sintonia de parâmetros e solução de problemas de otimização de controle restritos de sistemas robóticos, métodos bio-inspirados, tais como algoritmos genéticos (CÁZAREZ & CASTRO et al., 2010), otimização por enxame de partículas (MEHDI et al., 2011) e otimização por colônia de formigas (GARCÍA et al., 2009) são algoritmos de otimização promissores. A recente revisão concisa em (CASTILLO, 2012) explica e discute, por exemplo, o projeto de sistemas *fuzzy* tipo-2 usando métodos de otimização. É esperado que essas abordagens e outras similares possam emergir futuramente na área da teoria de controle não linear e especialmente para aplicações robóticas.

Atualmente, a teoria do caos é uma área de pesquisa extensivamente investigada para a solução de problemas de controle não linear. Dentre os campos

mais promissores, pode-se citar: análise do caos, controle de caos, e sincronização do caos. Recentemente foi apresentado que, se o pêndulo é colocado em determinados pontos, o movimento correspondente tornar-se-á caótico (SAKTHIVEL et al., 2012). Novos resultados na sincronização do caos do movimento do pêndulo com outros sistemas podem ser também encontrados em (LUO et al., 2011).

2 REQUISITOS DE PROJETO E RECURSOS NECESSÁRIOS

Diversas plataformas de desenvolvimento foram necessárias para este projeto: uma plataforma para o desenvolvimento de hardware, outra plataforma para o desenvolvimento de software e a plataforma para simulação (figura 6).

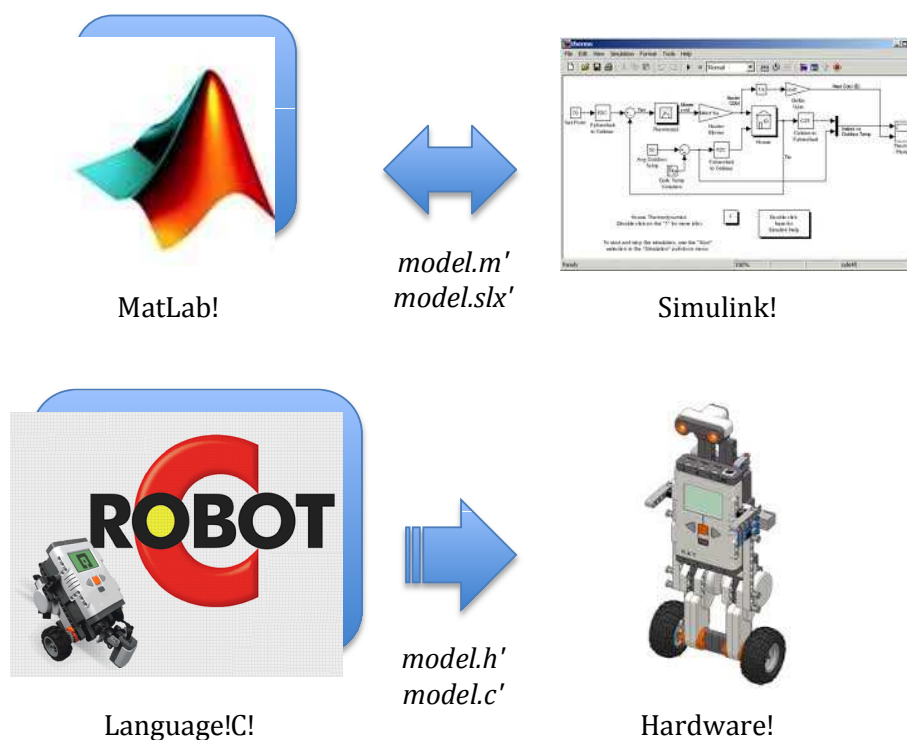


Figura 6 – Plataformas de desenvolvimento e simulação.

2.1 PLATAFORMA DE DESENVOLVIMENTO DE HARDWARE

A plataforma de desenvolvimento de hardware escolhida foi o kit de robótica Mindstorms NXT, modelo 8527, produzido pela LEGO Group (<http://www.lego.com>). Este produto é perfeitamente adequado ao uso didático, devido à sua complexidade e sofisticação, permitindo o desenvolvimento de diversos sistemas.

O controle dos robôs construídos com o kit NXT é feito usando um computador embarcado denominado NXT, com dimensões de 10,7 cm x 7,3 cm x 4,1 cm (altura, largura e profundidade, respectivamente), alimentado por 6 pilhas de 1,5 V.



Figura 7 – Kit lego.

O diagrama de bloco do NXT encontra-se na figura 08. O controle principal é realizado por um microprocessador de 32 bits modelo AT91SAM7S256 com *clock* de 48 MHz, fabricado pela Atmel Corporation (<http://www.atmel.com>). Este microprocessador é uma CPU ARM versão 4T, com core ARM7TDMI, 256 KB de memória flash (onde ficam armazenados códigos executáveis e arquivos de imagem e som) e 64 KB de SRAM de alta velocidade. Os ARMs utilizam conjuntos de instruções RISC e dominam 80% do mercado de processadores embarcados de 32 bits.

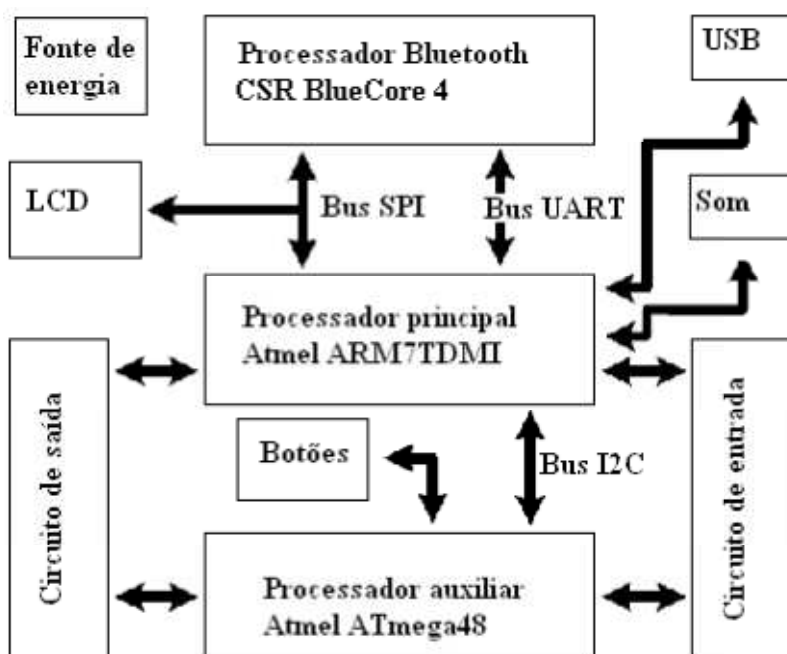


Figura 8 – Diagrama de blocos do NXT.

O NXT contém um microprocessador auxiliar para controlar os botões localizados abaixo do LCD, servomotores e sensores do NXT: uma CPU AVR modelo ATmega48, de 8 bits e *clock* de 8 MHz, também fabricada pela Atmel Corporation. Este chip, bem mais simples que o processador principal, tem 4 KB de memória flash, 512 Bytes de SRAM e 256 Bytes de EEPROM. Presentes estão 14 conversores analógico-digitais (conhecidos pela sigla em inglês ADC) de 10 bits, usados para converter as leituras de sensores analógicos e 6 geradores de sinais modulados por largura de pulso (“*Pulse-width modulation*” - PWM) usados para controlar os servomotores. Um terceiro processador compõe o NXT: um *BlueCore 4* da CSR plc (<http://www.csr.com>), responsável por implementar a funcionalidade de comunicação sem fio por Bluetooth 2.0. O NXT dispõe de portas para conexão de 4 sensores e 3 servomotores. Estes são conectados por cabos de 6 vias com conectores muito semelhantes, mas não totalmente idênticos, ao do padrão RJ12 (a única diferença é a localização da trava do conector).

O M-NXT é acompanhado de 03 servomotores e encoder com 1 grau de precisão (figura 09). Os servomotores funcionam a uma tensão nominal de 9 V e têm velocidade de rotação máxima de 170 rpm na ausência de carga.

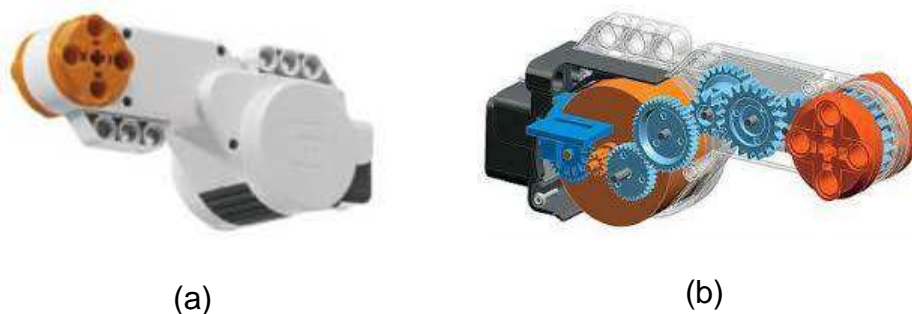


Figura 9 – Servomotor NXT. (a) Vista lateral; (b) Vista Interior.

Diversos sensores acompanham o NXT. Destes, apenas o sensor ultrassônico de distância (figura 10) foi também considerado na construção do robô auto-equilibrante. Este sensor emite pulsos de ultrassom (som com frequência acima de 20 kHz, inaudível para ser humanos) e mede o tempo que estes pulsos demoram a retornar, após reflexão em algum objeto à sua frente. O sensor permite medir a distância a objetos que se encontrem entre 0 cm e 255 cm com precisão de 3 cm e

taxa de 333 amostras/s (fixada pelo processador ATmega48, que controla as entradas e saídas, e que amostra os sensores a cada 3 ms).

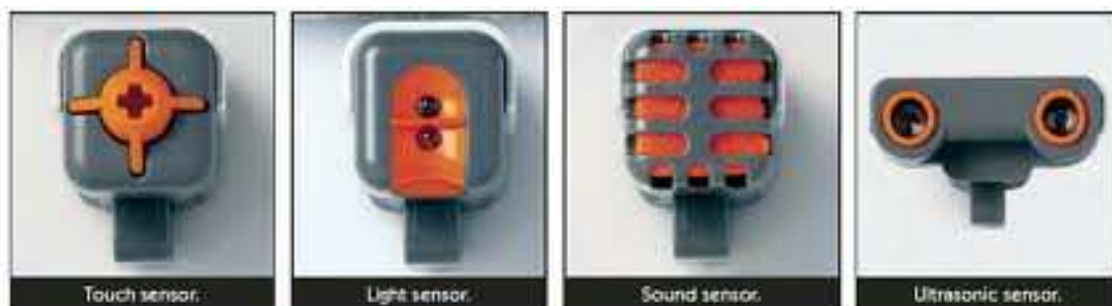


Figura 10 – Sensores que acompanham o kit: sensor de toque, sensor de luz, sensor som e sensor ultrassônico.

o **Sensor Gyro**

O sensor de velocidade de rotação é um componente essencial no desenvolvimento de um robô auto-equilibrante de duas rodas. É ele que gera a entrada necessária ao algoritmo de controle que manterá o robô equilibrado em pé, sem tombar. O sensor escolhido foi o NXT Gyro Sensor, modelo NGY1044, da *HiTechnic Products* (<http://www.hitechnic.com>). Este sensor (também chamado Gyro) tem faixa de operação entre $-360^{\circ}/s$ e $+360^{\circ}/s$ com precisão de $1^{\circ}/s$, sendo sensível a rotações em torno de um único eixo (representado pela seta preta na figura 11). Apesar de sua saída ser analógica e, portanto contínua, ele é amostrado a uma taxa fixa de 333 amostras/s pelo microprocessador ATmega48 do NXT.



Figura 11 – NXT Gyro Sensor, da *HiTechnic Products*.

O elemento sensor presente no interior do Gyro é um giroscópio de estado sólido de um eixo modelo XV-3500CB, da *Epson Toyocom Corporation* (<http://www.epsontoyocom.co.jp/english/>). Este componente tem dimensões extremamente reduzidas (5,0 mm x 3,2 mm x 1,3 mm). Seu funcionamento se baseia

em dois princípios físicos: a força de *Coriolis* e o efeito piezoelétrico (tanto direto quanto inverso). A estrutura principal dentro do chip é composta de dois ressonadores a quartzo com braços em T e dois braços sensores, montados perpendicularmente (vide figura 12). A aplicação de uma tensão alternada com frequência de aproximadamente 50 kHz gera uma força oscilante nos 6 ressonadores, pelo efeito piezoelétrico inverso (figura 12, painel 'a'). Quando o componente se sujeita a uma rotação no mesmo plano dos braços em T, estes passam a sofrer o efeito de uma força de *Coriolis* (figura 12, painel 'b'). Esta força distorce mecanicamente os braços sensores, gerando uma tensão pelo efeito piezoelétrico direto (figura 12, painel 'c'). A tensão de saída do componente é linearmente proporcional à velocidade de rotação.

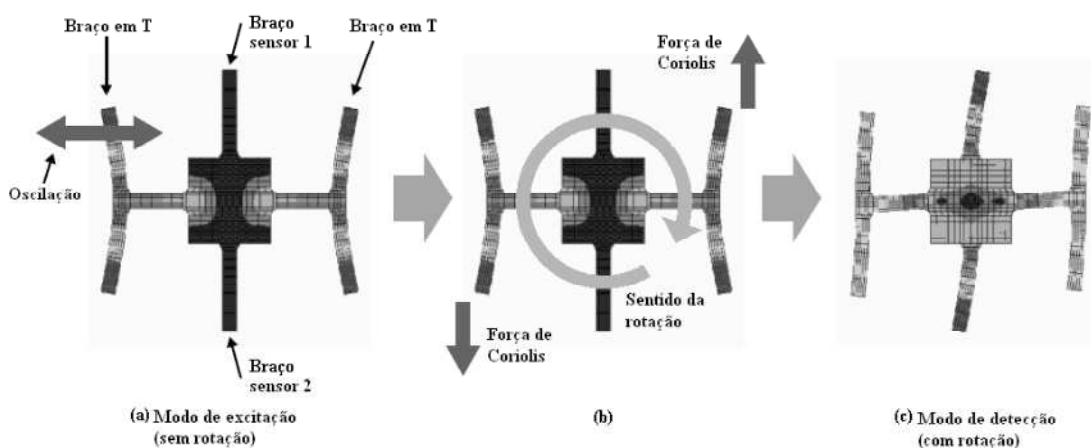


Figura 12 – Estrutura do giroscópio de estado sólido XV-3500CB, da *Epson Toyocom Corp.* e seu princípio de funcionamento.

2.2 PLATAFORMA DE DESENVOLVIMENTO DE SOFTWARE: ROBOTC

Para o desenvolvimento do código de programação que será embarcado no auto-equilibrante foi escolhido o software RobotC versão 3.61 da *Carnegie Mellon University*.



Figura 13 – Software RobotC da Carnegie Mellon University.

A seguir estão relacionadas algumas características que tornam o RobotC uma excelente linguagem de programação para o NXT e a mais adequada ao projeto:

- Alta compatibilidade com o ANSI C. Implementa todas as formas de loop (*for*, *while*, *do-while*) e desvio condicional (*if-else*, *switch*) do C.
- Baixo custo e facilidade de obtenção (pode ser baixado pela Internet).
- Ambiente integrado de desenvolvimento (*Integrated Development Environment* – IDE) com editor e *debugger*, permitindo o uso de breakpoints, execução passo a passo e visualização dos valores das variáveis (*watches*) – vide figura 14.
- Matemática de ponto flutuante, com implementação de funções matemáticas avançadas (por ex., exponenciais e funções trigonométricas).
- Arrays de variáveis.
- Gerenciamento de sistema de arquivos, permitindo gravar e ler arquivos de dados da memória flash.
- Gera códigos executáveis muito rápidos: 130 vezes mais rápidos que os gerados usando NXT-G e 86% mais rápidos que os gerados usando RoboLab, que é a segunda linguagem mais rápida para o NXT.
- Gera códigos executáveis muito menores que os da maioria das outras linguagens: 20 vezes menores, em média, que os executáveis gerados em NXT-G. Isto permite desenvolver algoritmos bastante complexos capazes de caber na memória flash do processador principal do NXT (256 KB).

```

11
12 /*SELECT SENSOR PORT (01, 02, 03 or 04), and WHEEL DIAMETER (millimeters).*/
13 const sensors Gyro = 03;
14 const float your_wheel_diameter = 50;
15
16 /*These are driver files to read gyro sensors. Make sure the path is correct.
17 You may delete them and link if you use the RIT-Chess Gyro*/
18
19 /*This is the Reguay Driver code. Place in same directory as this program*/
20 #include "segway-driver-1v3-PIDNN.h"
21
22 void main()
23 {
24   SerialDebugStream();
25   //Start balancing and wait for configuration to finish
26   StartTask(balancing);
27   while(stating_balancing_task())
28   {
29     //distance desired in cm:
30     int dist=200;//cm
31     int ang=dist/0.04884;
32
33
34     while(!task)
35     {
36       speed = -1;
37       while(mMotorEncoder[motorB] < ang) (waitMsec(100));
38       //#!: Polling 10 times/sec is good enough -- Save processor power for the balancing task.
39       //#!: Going forwards means that the motors run backwards, because of their orientation.
40       // Go to go forward for 1000 degrees, the encoder must reach -1000 degrees.
41       speed = 0;
42       while(mMotorEncoder[motorB] > 0) (waitMsec(100));

```

Errors

```

19:196 Warning: Unreferenced variable 'a'
19:196 Warning: Unreferenced variable 'd'
20:196 Warning: Unreferenced variable 'gma'
13:195 Warning: Unreferenced variable 'j'
17:195 **Info**: 'p1_prev' is written but has no read references
18:200 **Info**: 'p2_prev' is written but has no read references
18:201 **Info**: 'p3_prev' is written but has no read references

```

Figura 14 – Ambiente de desenvolvimento integrado do RobotC.

2.3 PLATAFORMA DE SIMULAÇÃO: MATLAB-SIMULINK

MATLAB é um dos mais populares ambientes de computação numérica. É usado tanto na indústria quanto no meio acadêmico. MATLAB tem várias extensões numéricas, conhecidas por *toolboxes*. Neste trabalho usaremos o software MATLAB como o ambiente fundamental para desenvolvimento dos controladores a serem implementados.

Simulink é uma ferramenta bastante flexível e intuitiva de fazer projetos de sistemas, pois utiliza interfaces gráficas (figura 15). É integrado ao MATLAB e as funções do MATLAB podem rodar dentro do Simulink, ou seja, é possível importar e exportar para o *workspace* do MATLAB. Isto o torna ideal para simulações de controle.



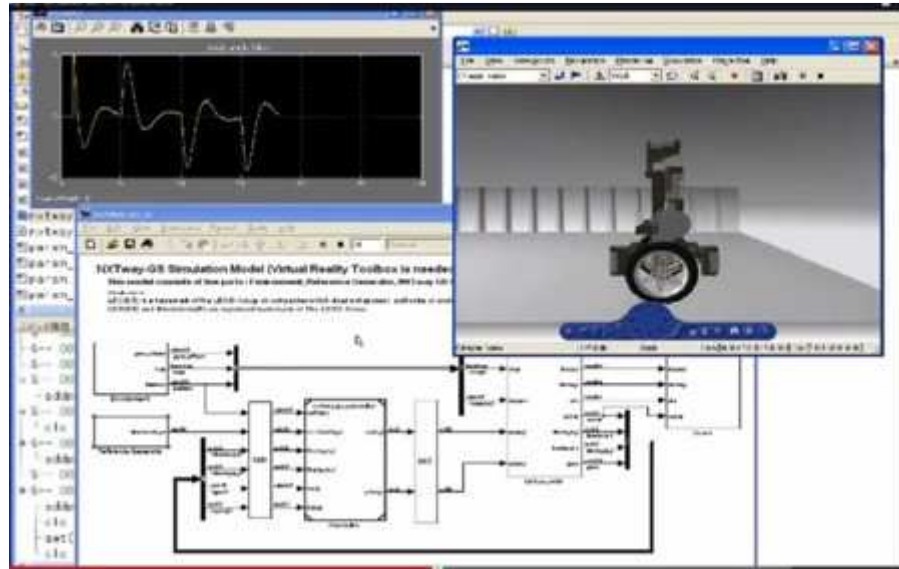


Figura 15 – Ambiente de desenvolvimento Simulink-MATLAB.

3 MODELAGEM MATEMÁTICA E ESTRATÉGIA DE CONTROLE

3.1 MODELAGEM MATEMÁTICA

A modelagem de um sistema dinâmico é um dos principais requisitos na área de Controle e Automação. O auto-equilibrante sobre duas rodas pertence a um classe de sistemas dinâmicos não lineares, multivariáveis, instáveis, de alta ordem, fortemente acoplados e subatuados, tornando sua modelagem física não trivial e sua dinâmica complexa. Nos últimos anos, diversos trabalhos sugerem a modelagem matemática para esse sistema (SUNDIN, 2012; GÖÇMEN, 2012; BAGEANT, 2011; OOI, 2003). A correta modelagem física oferece diversos elementos para a análise de sua dinâmica e para a realização de um projeto de controlador adequado. Neste trabalho, descrevem-se detalhadamente as equações de movimento do auto-equilibrante, as forças atuantes sobre cada parte do sistema, a possibilidade de linearização, as equações de estado e a função de transferência do sistema linearizado, bem como é apresentada a implementação no Simulink do diagrama de blocos do sistema linearizado.

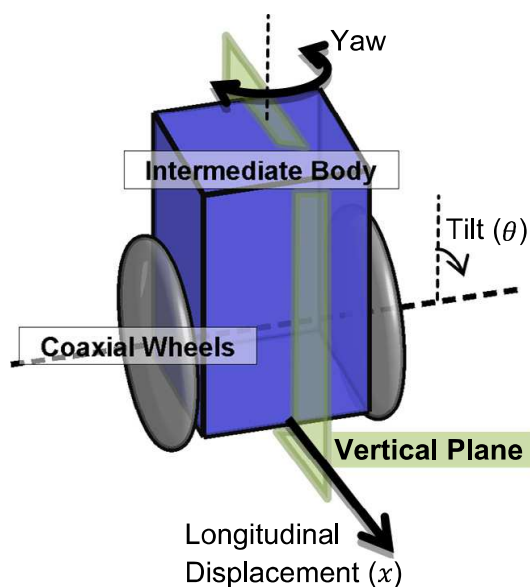


Figura 16 – Coordenadas do robô (CHAN, 2013).

3.1.1 Considerações iniciais

O robô auto-equilibrante proposto consiste de um longo corpo com duas rodas montadas cada uma no fim. Por simplicidade desta derivação, as duas rodas serão tratadas como um única roda, e será assumido que o robô percorre somente em linha reta. O modelo pode ser simplificado, considerando o corpo como um ponto de massa, rodando sobre o eixo das rodas. O conjunto de considerações feitas para simplificar o modelo é apresentado abaixo:

- As rodas estão sempre em contato com o chão e não sofrem deslizamento;
- As perdas mecânicas e elétricas podem ser aproximadas para zero;
- A resposta do sistema elétrico é significativamente mais rápida do que a do sistema mecânico, então a dinâmica do sistema elétrico pode ser desprezada;
- O movimento do robô é restrito a linha reta então o sistema pode ser analisado como um sistema bidimensional somente com movimento planar.
- Todos os corpos são rígidos;
- O ângulo de inclinação da vertical do corpo levantado é suficientemente pequeno para permitir a linearização do sistema;

3.1.2 Modelagem física

Sob essas condições, o robô pode ser modelado como mostrado na Figura 17.

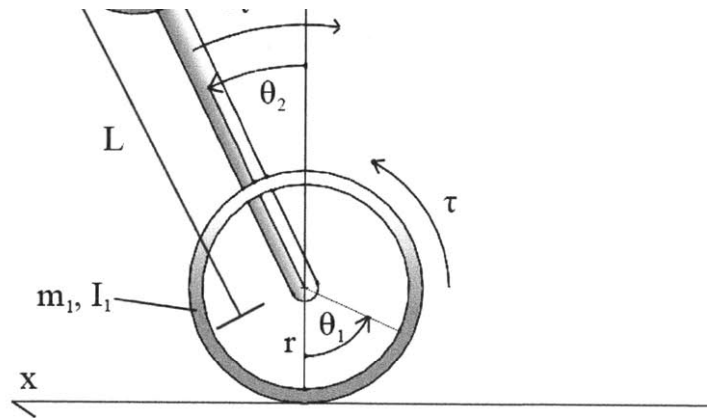


Figure 2-1: Diagram showing basic two-wheeled robot, with dimension

$$\begin{bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_1 \end{bmatrix} = \begin{bmatrix} -\frac{m_1 g L \cos \theta_2}{I_1 + m_1 L^2} \\ \frac{m_1 g r \sin \theta_1}{I_1 + m_1 r^2} \end{bmatrix}$$

Figura 17 – Diagrama do auto-equilibrante (BAGEANT, 2011).

(x_G, y_G) : As coordenadas (x,y) do centro de gravidade do corpo

(x, y) : As coordenadas (x,y) do centro de gravidade da roda

θ_c : Ângulo do corpo a partir da linha vertical

θ_r : Ângulo da roda a partir da linha vertical

P_r : Peso da roda

P_c : Peso do corpo

I_r : Momento de Inércia da roda em relação ao centro de gravidade

I_c : Momento de Inércia do corpo em relação ao centro de gravidade

N : Força Normal

V : Força Vertical

H e H' : Forças Horizontais

r : Raio da Roda

L : Distância do Centro de Massa do corpo ao eixo de rotação da roda.

τ_c : Torque do corpo

τ_r : Torque da roda

F_{at} : Força de atrito

Para deduzir as equações de movimento do sistema, considere o diagrama de corpo livre mostrado na figura 18.

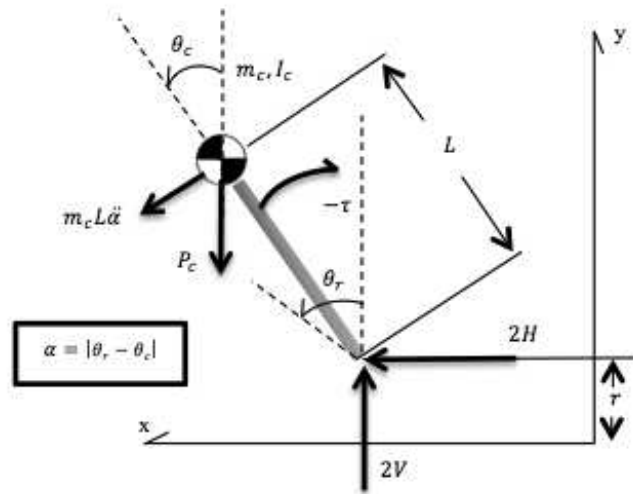


Figura 18 – Diagrama de corpo livre do corpo do auto-equilibrante.

O movimento rotacional (rotação) do corpo em torno de seu centro de gravidade pode ser descrito por:

$$\sum \tau_c = I_c \ddot{\theta}_r(t)$$

$$-2\tau_c + [2V]L \sin \theta_c + [-2H]L \cos \theta_c + m_c L (\ddot{\theta}_c - \ddot{\theta}_r) L = I_c \ddot{\theta}_r(t) \quad (1)$$

A aplicação das leis de Newton para um instante genérico t para o movimento horizontal (translação) do centro de gravidade do corpo é dado por:

$$\sum F_x = m_c \ddot{x}_c(t)$$

$$\ddot{x}_c(t) = \ddot{x}_{Gc}(t)$$

$$\sum F_x = 2H(t) = m_c \ddot{x}_{Gc}(t)$$

$$H(t) = \frac{1}{2} m_c [\ddot{x}_{Gc}(t)]$$

$$H(t) = \frac{1}{2} m_c [r \ddot{\theta}_c - (L \sin \theta_c) \dot{\theta}_c^2 + (L \cos \theta_c) \ddot{\theta}_c] \quad (2)$$

A aplicação das leis de Newton para um instante genérico t para o movimento vertical do centro de gravidade do corpo é dado por:

$$\begin{aligned}\sum F_y &= m_c \ddot{y}_{Gc}(t) \\ \sum F_y &= 2V(t) - P_c = m_c \ddot{y}_{Gc}(t) \\ V(t) &= \frac{1}{2} m_c [g + \ddot{y}_{Gc}(t)] \\ V(t) &= \frac{1}{2} m_c [g - (L \cos \theta_c) \dot{\theta}_c^2 - (L \sin \theta_c) \ddot{\theta}_c] \quad (3)\end{aligned}$$

Para deduzir as equações de movimento do sistema, considere o diagrama de corpo livre da roda, mostrado na figura 19.

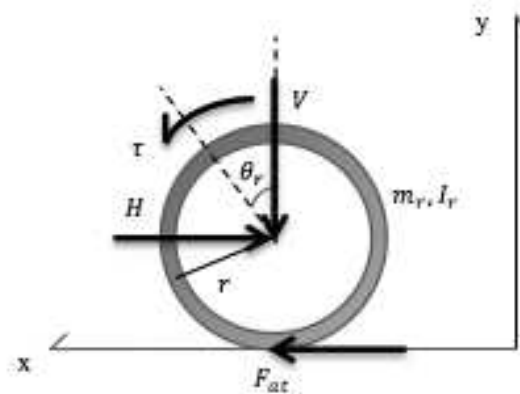


Figura 19 – Diagrama de corpo livre da roda.

O centro de gravidade da roda é dado por:

$$\begin{cases} x_{Gr} = r\theta_r \\ y_{Gr} = r \end{cases}$$

Derivando estas últimas equações com relação ao tempo, teremos:

$$\dot{x}_{Gr} = r\dot{\theta}_r \quad \text{e} \quad \dot{y}_{Gr} = 0$$

Derivando novamente obtemos as acelerações lineares da roda:

$$\begin{cases} \ddot{x}_{Gr} = r\ddot{\theta}_r \\ \ddot{y}_{Gr} = 0 \end{cases}$$

O movimento rotacional da roda em torno de seu centro de gravidade pode ser descrito por:

$$\begin{aligned} \sum \tau_r &= I_r \ddot{\theta}_r(t) \\ \tau_r - F_{at}r - m_c L^2 \ddot{\theta}_c &= I_r \ddot{\theta}_r + I_c \ddot{\theta}_c \\ \tau_r &= F_{at}r + m_c L^2 \ddot{\theta}_c + I_r \ddot{\theta}_r + I_c \ddot{\theta}_c \end{aligned} \quad (4)$$

A aplicação das leis de Newton para um instante genérico t para o movimento horizontal (translação) do centro de gravidade da roda é dado por:

$$\begin{aligned} \sum F_x &= m_r \ddot{x}_{Gr}(t) \\ \sum F_x &= -H(t) + F_{at} + \frac{1}{2} m_c \ddot{\theta}_c (r + L \cos \theta_c) = (m_r + m_c) r \ddot{\theta}_r \\ F_{at} &= -\left(\frac{1}{2} m_c L \sin \theta_c\right) \dot{\theta}_c^2 + m_r r \ddot{\theta}_r + \frac{1}{2} m_c r \ddot{\theta}_r \end{aligned} \quad (5)$$

3.1.3 Desenvolvimento das equações

As forças H e V podem ser eliminadas das equações (1) a (3), resultando:

De (2) e (3) \rightarrow (1) :

$$-2\tau_c = -m_c g L \sin \theta_c + (m_c r L \cos \theta_c) \ddot{\theta}_c + (m_c L^2 + I_c) \ddot{\theta}_r \quad (6)$$

De (5) \rightarrow (4) :

$$\tau_r = (m_c L^2 + I_c) \ddot{\theta}_c - \left(\frac{1}{2} m_c L r \sin \theta_c\right) \dot{\theta}_c^2 + \left[r^2 \left(m_r + \frac{1}{2} m_c\right) + I_r\right] \ddot{\theta}_r \quad (7)$$

Portanto, as equações não-lineares do sistema são:

$$\begin{cases} \tau_c = \left(I_r + r^2 \left(m_r + \frac{1}{2} m_c \right) \right) \ddot{\theta}_r + (m_c L^2 + I_c) \ddot{\theta}_c - \left(\frac{1}{2} m_c r L \sin \theta_c \right) \dot{\theta}_c^2 \\ \tau_r = \frac{1}{2} [(m_c L^2 + I_c) \ddot{\theta}_r + (m_c r L \cos \theta_c) \ddot{\theta}_c - m_c g L \sin \theta_c] \end{cases} \quad (8)$$

$$\begin{cases} \tau_c = (H_1) \ddot{\theta}_r + (H_3) \ddot{\theta}_c - \left(\frac{1}{2} m_c r L \sin \theta_c \right) \dot{\theta}_c^2 \\ \tau_r = \frac{1}{2} (H_3) \ddot{\theta}_r + (H_2) \ddot{\theta}_c - m_c g L \sin \theta_c \end{cases}$$

Tal que:

$$\begin{bmatrix} H_1 & H_2 \\ H_2 & H_3 \end{bmatrix} = \begin{bmatrix} I_r + r^2 \left(m_r + \frac{1}{2} m_c \right) & m_c r L \cos \theta_c \\ m_c r L \cos \theta_c & m_c L^2 + I_c \end{bmatrix}$$

E lembrando que :

$$\tau_r = -\tau_c$$

3.2 PARÂMETROS DA PLANTA

A tabela 1 e a figura 20 apresentam os parâmetros físicos do NXT.

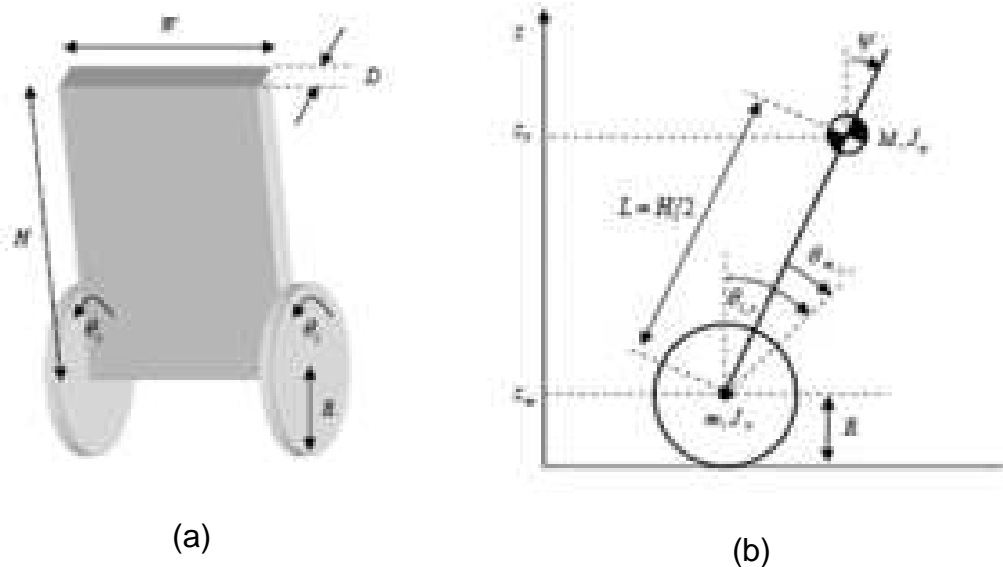


Figura 20 –Parâmetros do NXT (DUNG, 2010). (a) vista 3D do robô; (b) visão lateral.

Os valores indicados na tabela 1 foram obtidos em (DUNG, 2010).

Tabela 1 – Parâmetros do NXT.

Parâmetro (valor)	Unidade	Descrição
$g = 9,81$	[m/s ²]	Aceleração da Gravidade
$m = 0,0013$	[kg]	Massa Roda
$R = 0,056$	[m]	Raio Roda
$J_W = mR^2/2$	[kgm ²]	Momento de Inércia da Roda
$M = 0,6$	[kg]	Massa Corpo
$W = 0,14$	[m]	Largura Corpo
$D = 0,04$	[m]	Profundidade Corpo
$H = 0,144$	[m]	Altura do Corpo
$L = H/2$	[m]	Distância do centro de massa do eixo da roda
$J_\psi = ML^2/3$	[kgm ²]	Momento de Inércia pitch Corpo
$J_\theta = M(W^2 + D^2)/12$	[kgm ²]	Momento de Inércia yaw Corpo
$J_m = 1 \times 10^{-5}$	[kgm ²]	Momento de Inércia motor DC
$R_m = 6,69$	[Ω]	Resistencia motor DC
$K_b = 0,468$	[V s/rad]	Constante de velocidade do motor DC
$K_t = 0,317$	[Nm/A]	Torque constante do motor DC
$n = 1$	Relação de engrenagens	
$f_m = 0,0022$	Coeficiente de atrito entre corpo e motor DC	
$f_w = 0$	Coeficiente de atrito entre roda e motor DC	

Calculando-se os valores indicados na tabela 2, obtêm-se os parâmetros da equação do sistema:

Tabela 2 – Parâmetros e equações do NXT.

Equações do Sistema	Parâmetros
$\ddot{\theta}_2 = \frac{1}{(\alpha_1 \cos \theta_2)} \left[-\beta_1 \ddot{\theta}_1 + (\alpha_2 \sin \theta_2) \dot{\theta}_2^2 + \tau \right]$ $\ddot{\theta}_1 = \frac{1}{(\beta_2 \cos \theta_2)} \left[-\alpha_3 \ddot{\theta}_2 + \alpha_4 \sin \theta_2 - \tau \right]$	$\alpha_1 = \frac{1}{2} m_2 r L = \alpha_2 = 0,0012096$ $\alpha_3 = m_2 L^2 + I_2 = 0,0031468$ $\alpha_4 = m_2 g L = 0,42336$ $\beta_1 = \left(I_1 - \frac{1}{2} r^2 m_2 - r^2 m_1 \right) = 0,1990$ $\beta_2 = m_2 L = 0,432$

3.3 SISTEMA DE CONTROLE

O auto-equilibrante deste trabalho foi desenvolvido para executar dois tipos de controles:

- Controle de equilíbrio;
- Controle de posição.

A estratégia de controle completa é apresentada no diagrama de controle da figura 21.

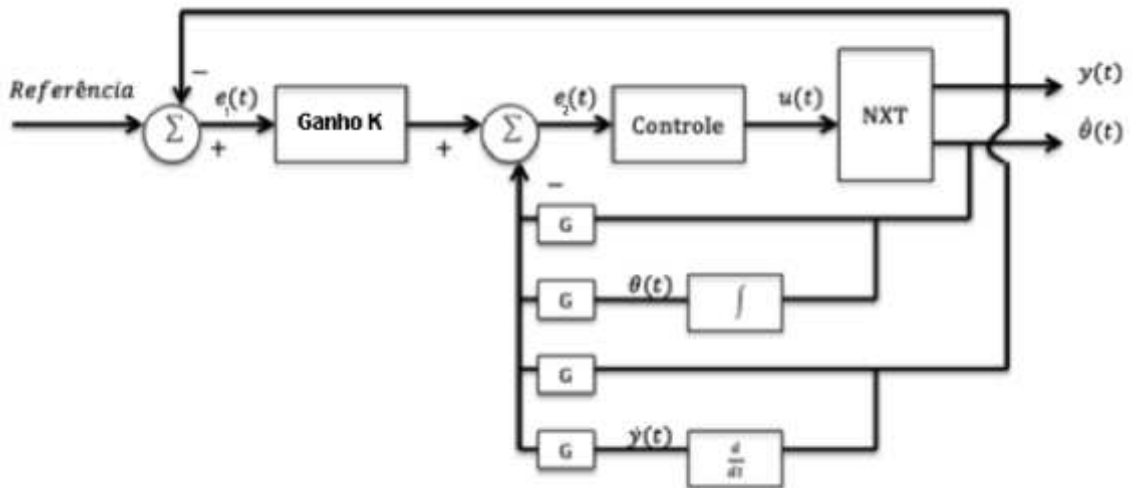


Figura 21 – Diagrama de controle do Auto-equilibrante.

Um sistema perfeitamente estável não tem deslocamento angular em relação ao eixo vertical, não tem velocidade angular, não tem deslocamento linear nem velocidade linear. Logo, o vetor de referência $[\theta_d, \dot{\theta}_d, y_d, \dot{y}_d]$ ideal deve ser nulo. O erro total no sistema pode então ser descrito como uma soma de todos os desvios no sistema, como visto na equação (9). A influência do peso de cada desvio pode ser determinado pelas constantes $G_\theta, G_{\dot{\theta}}, G_y, G_{\dot{y}}$. Neste projeto, as constantes foram consideradas as mesmas do projeto de controle PID da *RoboSquare* (WITZAND, 2009), visto que esta estratégia obteve bons resultados para o robô LEGO auto-equilibrante de duas rodas.

$$e(t) = G_\theta(\theta - \theta_d) + G_{\dot{\theta}}(\dot{\theta} - \dot{\theta}_d) + G_y(y - y_d) + G_{\dot{y}}(\dot{y} - \dot{y}_d) \quad (9)$$

Tabela 3 – Ganho das variáveis

	G_θ	$G_{\dot{\theta}}$	G_y	$G_{\dot{y}}$
Ganho	25	0.23	272,8	24,6

Conforme o diagrama de controle apresentado na figura 21, o erro é uma composição da realimentação das quatro variáveis de controle: ângulo inclinação (θ), velocidade angular ($\dot{\theta}$), deslocamento linear (y) e velocidade linear (\dot{y}). As

variáveis $\dot{\theta}$ e y são obtidas através do sensor gyro e do encoder do motor, respectivamente. As variáveis θ e \dot{y} são obtidas por cálculo de integração e derivação respectivamente.

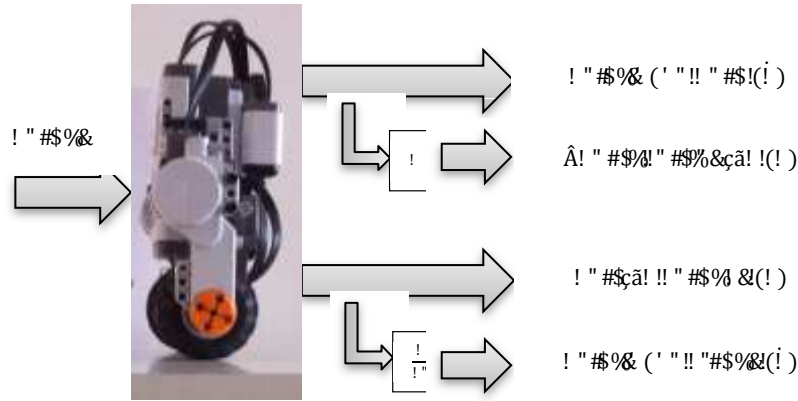


Figura 22 – Variáveis de entrada e saída do Auto-equilibrante.

Jin (2013) prova conceitualmente que a estratégia de realimentação utilizada é a mesma caso cada variável realimentada fosse controlada por um controle PID individual e então combinadas posteriormente. Isto pode ser provado pela fatoração:

$$u_1(t) = k_p G_\theta \theta(t) + k_i G_\theta \int_0^t \theta(\tau) d\tau + k_d G_\theta \frac{d\theta(t)}{dt} \quad (10)$$

$$u_2(t) = k_p G_{\dot{\theta}} \dot{\theta}(t) + k_i G_{\dot{\theta}} \int_0^t \dot{\theta}(\tau) d\tau + k_d G_{\dot{\theta}} \frac{d\dot{\theta}(t)}{dt} \quad (11)$$

$$u_3(t) = k_p G_y y(t) + k_i G_y \int_0^t y(\tau) d\tau + k_d G_y \frac{dy(t)}{dt} \quad (12)$$

$$u_4(t) = k_p G_{\dot{y}} \dot{y}(t) + k_i G_{\dot{y}} \int_0^t \dot{y}(\tau) d\tau + k_d G_{\dot{y}} \frac{d\dot{y}(t)}{dt} \quad (13)$$

$$u_1 + u_2 + u_3 + u_4 = k_p e(t) + k_i \int_0^t e(\tau) d\tau + k_d \frac{de(t)}{dt} \quad (14)$$

Onde θ pode ser numericamente aproximado por:

$$\theta(t) = \int_0^t \dot{\theta}(\tau) d\tau \approx \theta(t-1) + t\dot{\theta}(t) \quad (15)$$

Onde \dot{y} pode ser numericamente aproximado por:

$$\dot{y}(t) \approx \frac{y(t) - y(t-1)}{\Delta t} \quad (16)$$

Portanto, pela prova de (JIN, 2013) a resposta de um controlador PID representa a mesma resposta se fosse utilizado quatro controladores PID em malha fechada. Portanto, a estratégia de controle adotada nesse trabalho utilizará apenas um controlador PID em malha fechada para o controle de equilíbrio do robô auto-equilibrante de duas rodas.

3.3.1 Controle de equilíbrio

Todas as variáveis são somadas e subtraídas do valor de referência desejado, ou seja, o vetor nulo na condição de auto-equilíbrio: $[\theta, \dot{\theta}, y, \dot{y}] = [0, 0, 0, 0]$. O diagrama de controle de equilíbrio é dado pelo diagrama em destaque da figura 23.

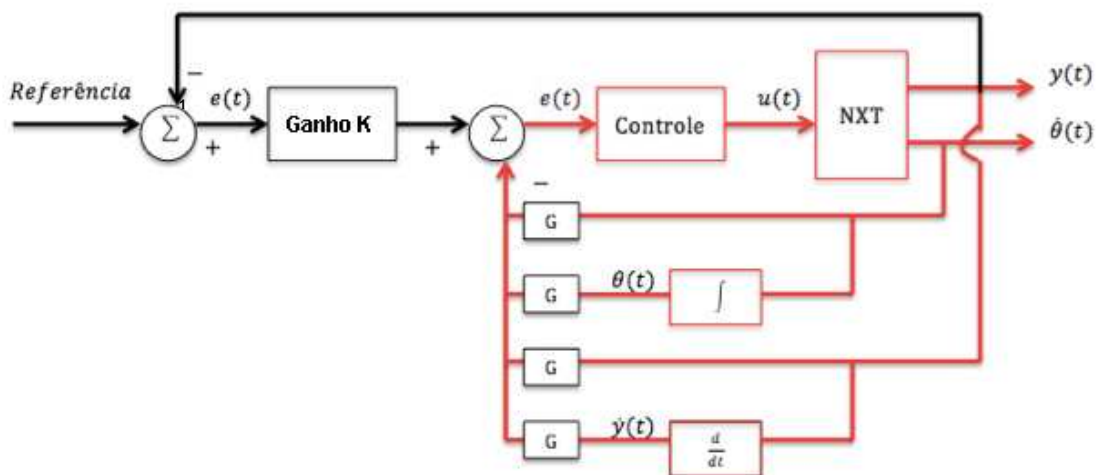


Figura 23 – Diagrama de controle de equilíbrio em destaque.

Utilizaremos inicialmente o controlador PID para executar o auto-equilíbrio. O diagrama do controle PID é dado conforme a figura 24.

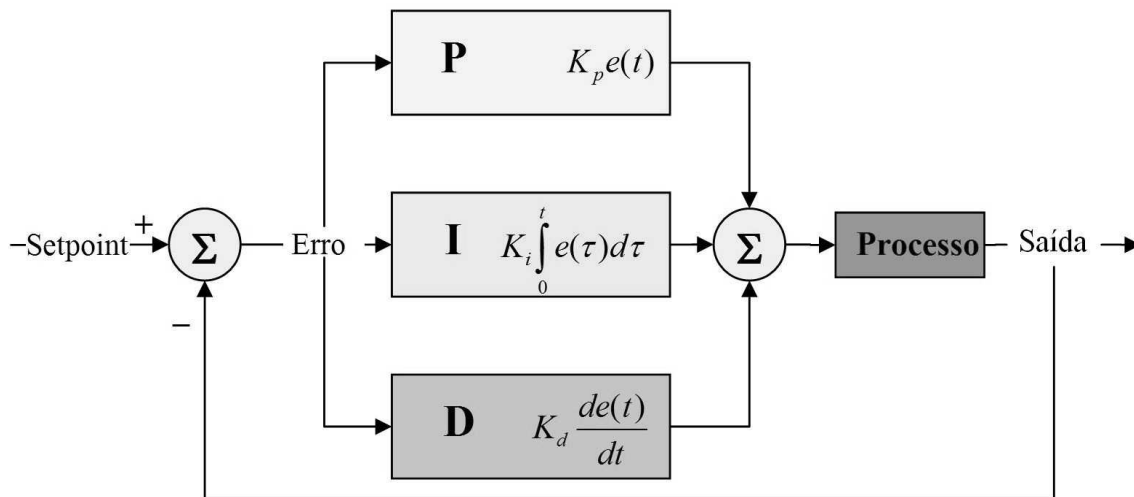


Figura 24 – Diagrama do controle PID.

Para determinação dos ganhos do controle PID, os valores de K_p , K_i e K_d foram ajustados de acordo com o método de Ziegler-Nichols (OGATA, 2003). Esse método consiste na determinação dos ganhos aproximados para um sistema de controle. A sintonia começa determinando todos os valores dos ganhos como zero. O valor de K_p é aumentado até o sinal de controle $u(t)$ oscilar numa amplitude constante. Este valor é chamado como ganho crítico K_c . O período de oscilação foi medido e chamado de P_c , conforme figura 25. Os valores obtidos estão apresentados na tabela 4.

Tabela 4 – Ganho crítico e período crítico.

K_c	P_c (s)
0.056	0.25

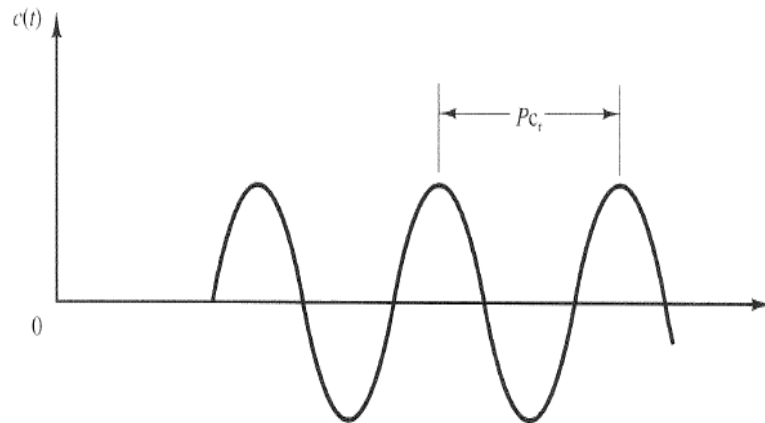


Figura 25 – Oscilação sustentada com período crítico P_c .

Usando as equações do método de Ziegler-Nichols (17), (18) e (19), os ganhos puderam ser aproximados conforme tabela 5.

$$K_p = 0.60K_c \quad (17)$$

$$K_i = 2 \frac{K_p}{P_c} \quad (18)$$

$$K_d = \frac{K_p P_c}{8} \quad (19)$$

Tabela 5 – Ganho PID inicial pelo método de Ziegler-Nichols.

K_p	K_i	K_d
0.0336	0.26888	0.00105

Com os valores da tabela 6 os ganhos foram mudados um por um para verificar se a estabilidade foi influenciada positivamente. Verificou-se

experimentalmente que K_d deveria ser diminuído para 0.000504 para alcançar a estabilidade do sistema.

Tabela 6 – Ganho PID final.

K_p	K_i	K_d
0.0336	0.26888	0.000504

No código de controle de equilíbrio do software RobotC, que será detalhado no capítulo 4, foram implementados os ganhos da tabela 6. O resultado experimental demonstra a estabilidade do controle de auto-equilíbrio do robô conforme observado na figura 26.



Figura 26 – Operação de Auto-equilíbrio.

3.3.2 Controle de posição

Além da realimentação das quatro variáveis mencionadas, observa-se mais um ramo de realimentação que é responsável pelo controle de posição. Nesse ramo, a variável de posição do encoder (y) é realimentada e subtraída do valor de referência desejado. Para executar o controle de posição, utiliza-se o controlador proporcional (k_p). A variável y é multiplicada por k_p e subtraída do valor de referência desejado y_{ref} .

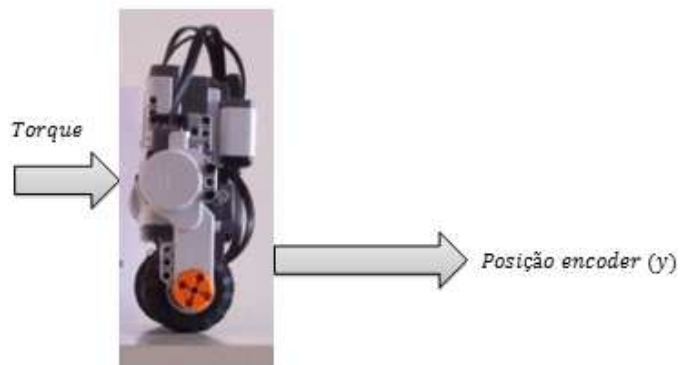


Figura 27 – Variáveis de entrada e saída do Auto-equilibrante para controle de posição.

A escolha do controlador proporcional atende ao requisito do projeto em que o NXT deve deslocar-se rapidamente se estiver distante e deslocar-se lentamente se estiver próximo da posição desejada. O diagrama de controle de posição é apresentado pelo diagrama da figura 28.

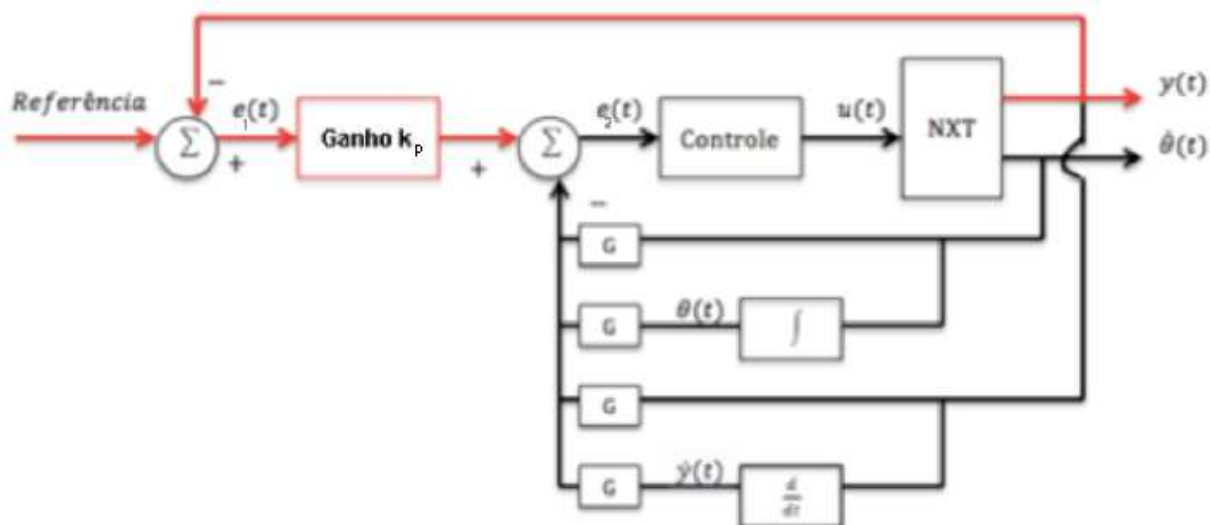


Figura 28 – Diagrama de controle de posição em destaque.

O erro de posição é multiplicado por um ganho (k) e somado com a realimentação negativa das quatro variáveis ponderadas. O resultado dessa soma determina o erro de entrada do controle PID. A equação do erro pode ser descrita por:

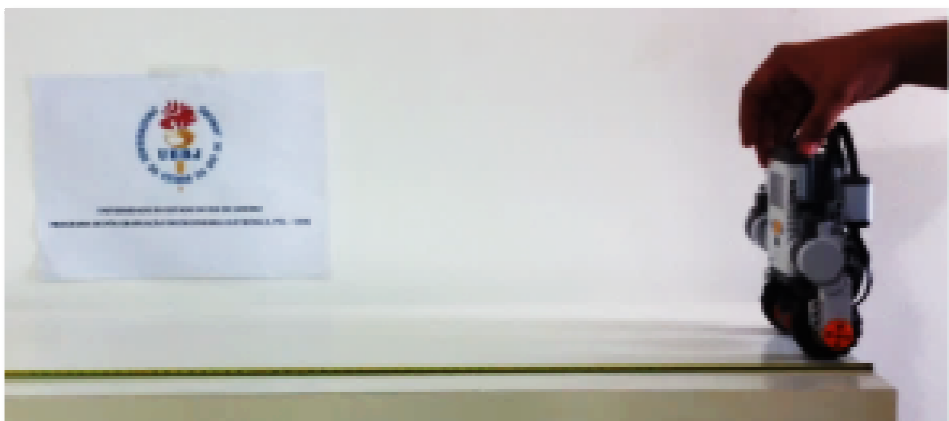
$$e(t) = y_{\text{ref}} - y$$

A lei de controle pode ser definida por:

$$u(t) = k_p e(t)$$

No RobotC foi implementado o código de controle de posição e a performance do NXT pode ser observada conforme figura 29.

(A) Posição Inicial



(B) Posição Final

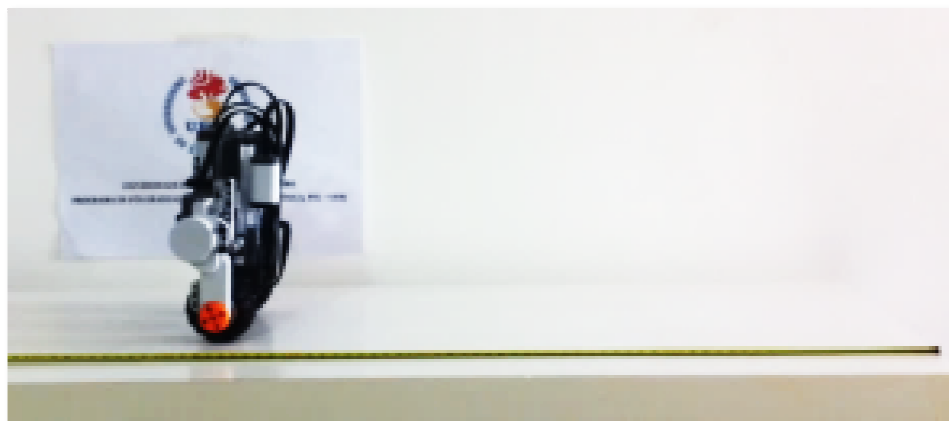


Figura 29 –Operação do controle de posição. (A) Posição inicial; (B) Posição final.

O valor do ganho proporcional (k) no Simulink foi obtido após diversos experimentos. O melhor valor foi $k = 0.495$, pois a posição final obtida atingiu o valor desejado de posição, conforme figura 30.

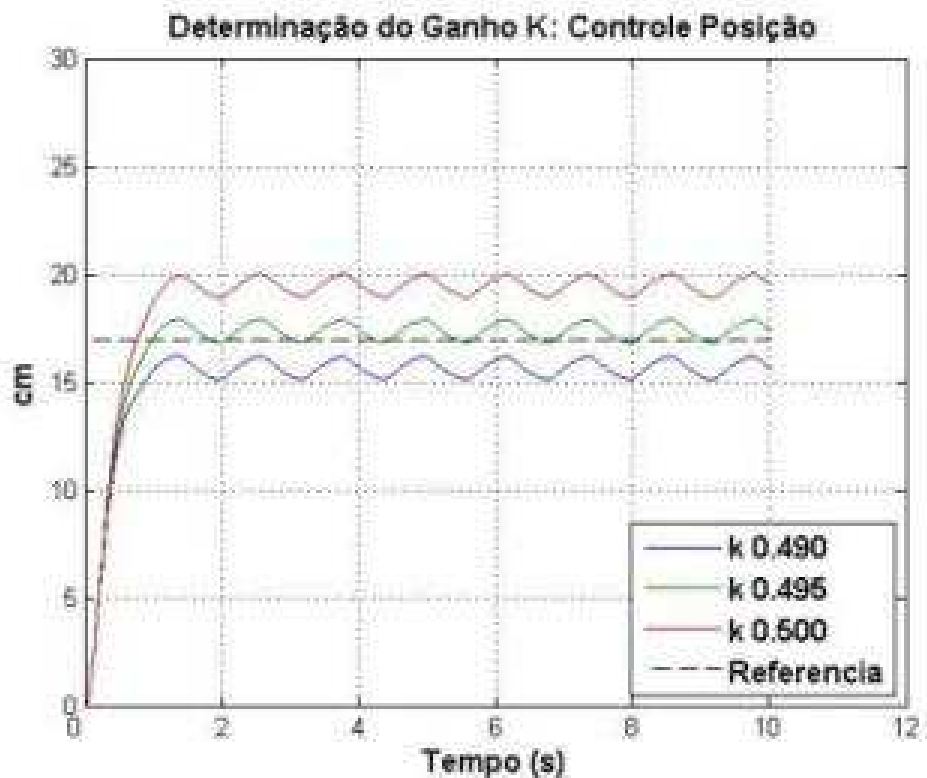


Figura 30 –Determinação do ganho K para controle de posição.

O comportamento oscilatório em torno do ponto desejado de posição em regime estacionário indica a característica do controlador proporcional. Ou seja, o controle proporcional de um sistema sem integrador ocasiona um erro estacionário (OGATA, 2003).

4 DESENVOLVIMENTO, PROGRAMAÇÃO E VALIDAÇÃO

4.1 CARACTERIZAÇÃO DO SENSOR GYRO

Todo giroscópio tem alguma deriva sistemática, também conhecida como *bias*, que é uma leitura da velocidade angular quando o giroscópio não está em movimento. Sensores desse tipo também tem uma variação ao longo do tempo em sua deriva sistemática, conhecida por *bias drift*, conforme figura 31. Esse problema não pode ser compensado porque depende de fatores externos como a temperatura. Ao longo do tempo, o problema de *bias* do gyro pode causar problemas com o cálculo do valor do ângulo, que é resultado da integração da velocidade angular. O ângulo calculado pode continuamente aumentar ou diminuir, mesmo quando o auto-equilibrante estiver estacionário na posição de equilíbrio. Este escorregamento angular também causa o deslocamento linear, pois para compensar o erro angular o robô afasta-se da posição de referência criando um erro de posição.

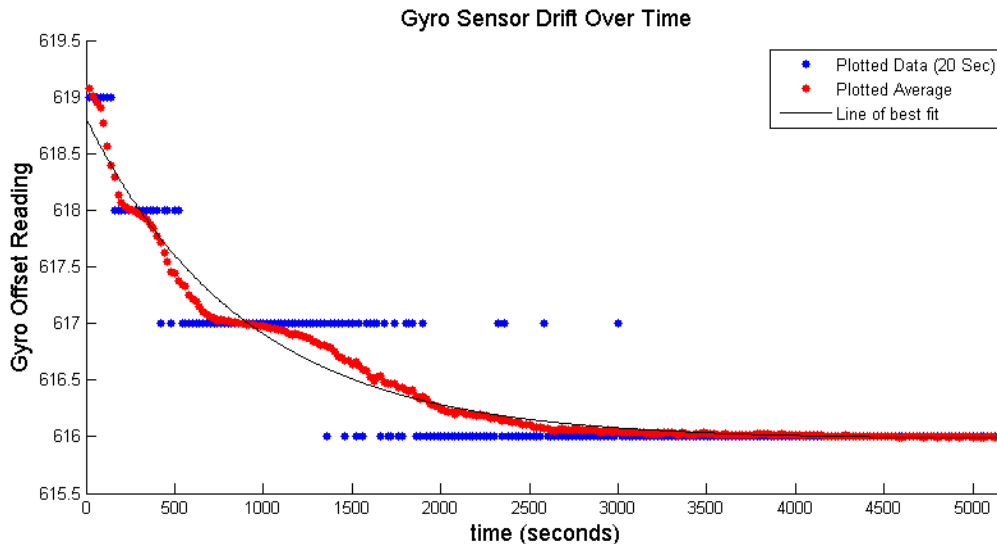


Figura 31 – Exemplo do problema de *drift* do sensor durante o tempo (Witzand, 2009).

Outro problema associado com o uso de um sensor inercial é a necessidade de um duplo integrador. Ou seja, pequenos ruídos e desequilíbrios podem resultar num erro quadrático de posição (figura 32). Portanto, esse tipo de sensor exige o

uso de modelos de planta bem detalhados, cuidados na calibração e um bom conhecimento sobre as suas características.

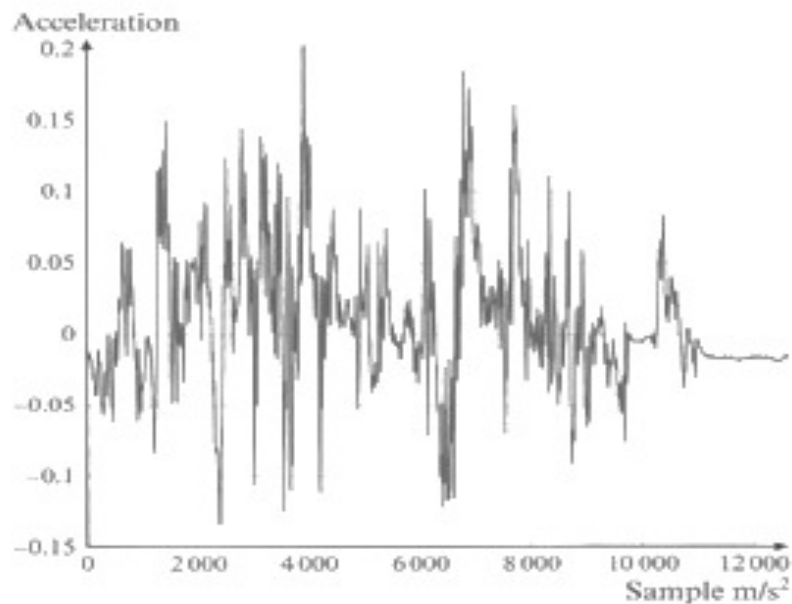
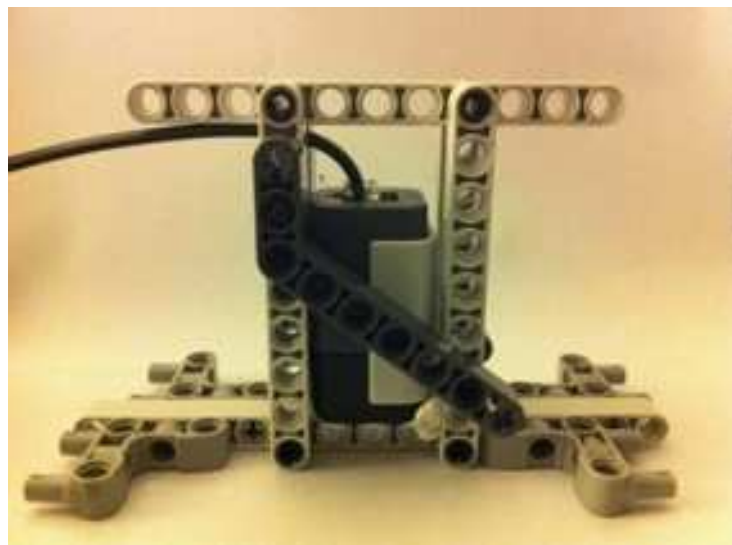


Figura 32 – Exemplo da medida de uma Unidade de Medida Inercial (Siciliano, 2008).

O sensor *Hitechnic*, usado no robô experimental, tem saída com valor inteiro entre 0 e 1023, sendo esta faixa fixada pelos ADCs do ATmega48, que são de 10 bits. A unidade usada é graus por segundo. Quando o sensor está na posição estacionária, ou seja, sem rotação, ele apresenta um valor de saída diferente do centro de escala (512). Este valor é chamado de offset. Como este valor varia para cada sensor, é necessário determiná-lo experimentalmente.

(A)



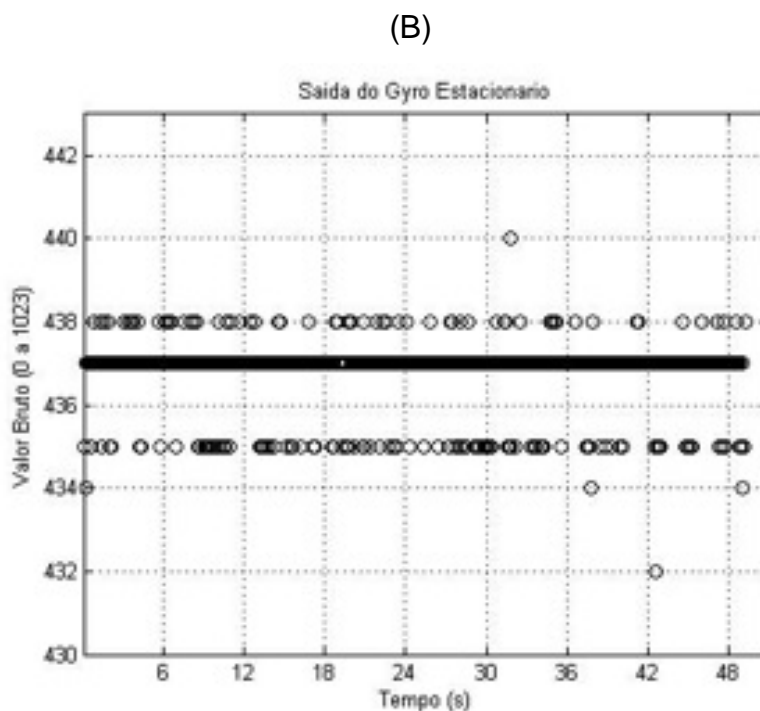


Figura 33 – Gyro estacionário: (A) Giga de teste; (B) Saída do gyro, amostrada a cada 3 milissegundos.

Através da giga de teste, montada para garantir a posição estacionária do gyro (figura 33-A), obteve-se a saída do gyro em regime estacionário. Verificou-se que o offset do sensor usado é de 437 (figura 33-B). No entanto, observou-se uma variação significativa desse valor no tempo (432 a 440). Essa variação é chamada de *drift*. Esse tipo de problema é comum em sensores de velocidade de rotação e de baixo custo.

Além da característica apresentada, alguns sensores gyro como o *Hitechnic* gyro, por exemplo, são diretamente afetados por mudanças na entrada de tensão. Como o NXT é alimentado por bateria, essa condição representa um sério problema. O acionamento dos motores do NXT resulta numa queda de tensão e, portanto, numa mudança do offset (figura 24).

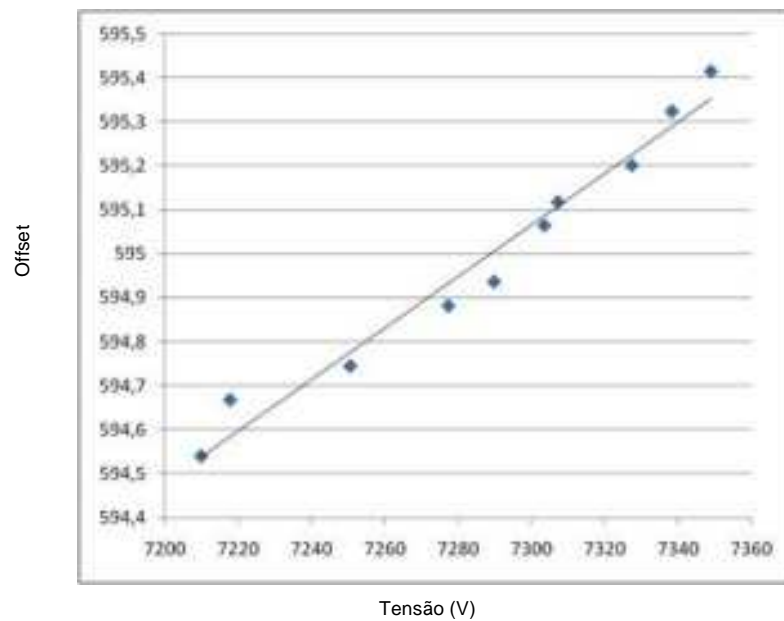


Figura 34 – Relação entre tensão e offset do gyro.

A relação entre tensão e offset do gyro é linear: a queda de tensão de 1 V fará o offset cair com 5 graus, conforme figura 34.

Para evitar os problemas apresentados do gyro é necessário haver algum tipo de compensação no algoritmo de controle. Caso contrário, o cálculo do ângulo de inclinação do robô irá acumular erro, provocando perda de equilíbrio após um determinado tempo.

4.2 MONTAGEM MECÂNICA DO ROBÔ

Dentre os componentes do kit NXT foram utilizados: um controlador NXT, dois servomotores, um sensor ultrassônico, quatro cabos de conexão, peças de suporte estrutural e um sensor Gyro. A seguir encontram-se fotos do robô já montado, com diversas vistas (frontal, traseira, esquerda, direita e por cima – figura 35).

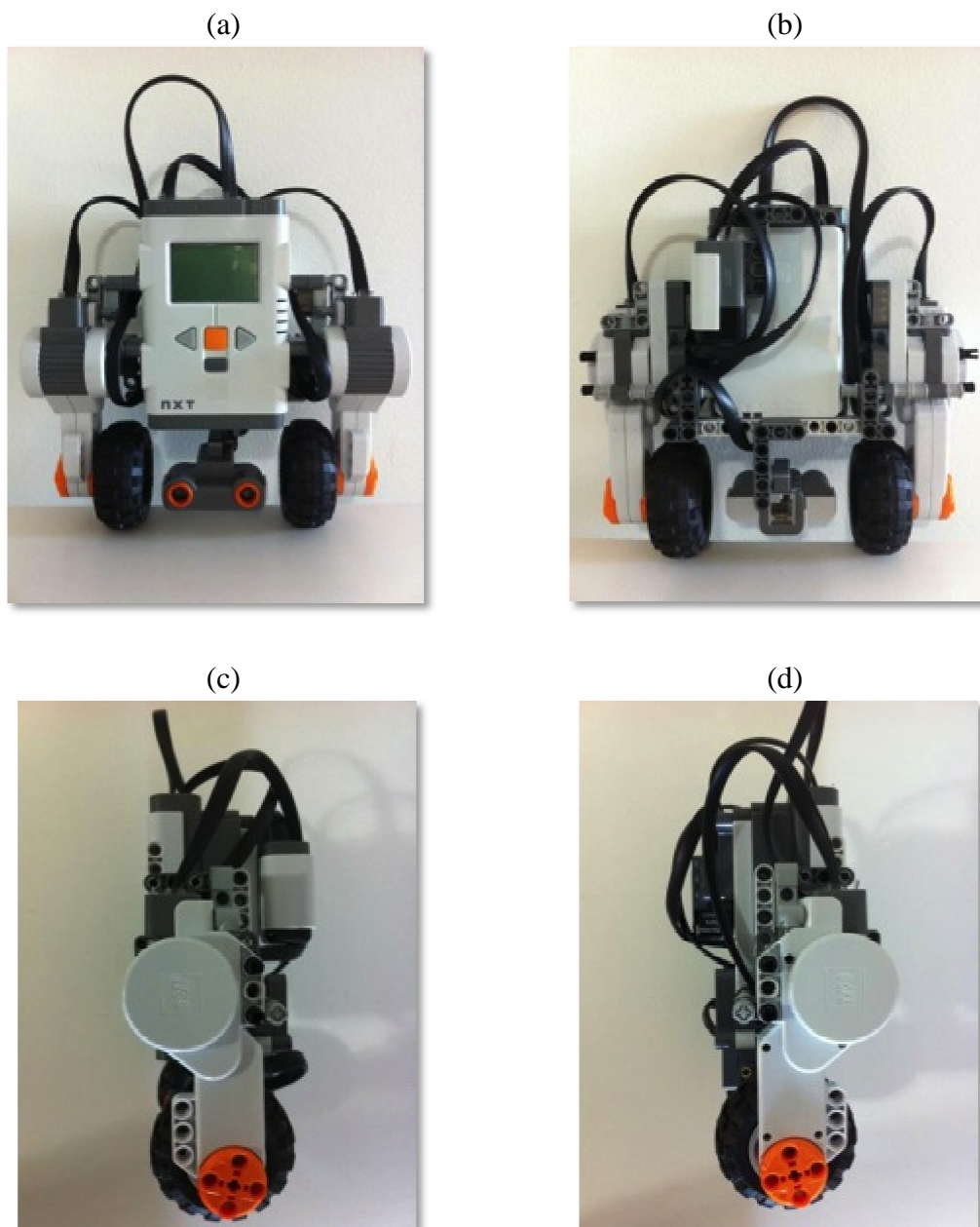


Figura 35 – Robô Auto-equilibrante de 02 rodas. (a) vista frontal; (b) vista traseira ; (c) vista direita; (d) vista esquerda.

A orientação escolhida para montagem do sensor foi fixar seu eixo paralelo aos eixos das rodas, uma vez que é sensível a rotações em torno de apenas um eixo (figura 36).



Figura 36 – Detalhe do sensor.

○ **Bateria**

O alto consumo de corrente dos servomotores (máximo de 2 A) indicaram a necessidade da aquisição de uma bateria recarregável para reduzir o custo com frequentes substituições de baterias alcalinas AA. A bateria recarregável possui tecnologia Lítio Íon Polímero com capacidade de 2100 mAh e uma fonte 120 Vac/ 10 Vdc - 7 W, para a bateria. Uma pequena modificação na montagem estrutural do robô é necessária com a inserção da bateria recarregável (figura 37).



(a)



(b)



(c)



(d)

Figura 37 –Bateria Recarregável NXT. (a) Montagem sem bateria recarregável; (b) Montagem com bateria recarregável; (c) Bateria recarregável; (d) Carregador bateria.

4.3 CÓDIGO DE CONTROLE E PROGRAMAÇÃO

Foi desenvolvido no software RobotC o código de programação que é embarcado no robô LEGO NXT. Este código foi desenvolvido em linguagem C++, conforme é apresentado no anexo. Os principais pontos implementados na programação serão apresentados nesta sessão.

○ Programação do sistema de controle

O mecanismo de auto-equilíbrio será apresentado passo a passo na mesma ordem que foi desenvolvido o programa no anexo.

1. As constantes do controle PID e a constante do erro ponderado são configuradas no início do programa.
2. O próximo passo é criar os objetos: sensor gyro e servo motor. Esses objetos contêm os métodos de leitura dos ângulos e velocidades dos sensores e motores.
3. Um *loop* infinito é inicializado e irá operar continuamente enquanto o auto-equilibrante opera em equilíbrio.

4. O primeiro passo nesse *loop* é verificar se o auto-equilibrante foi reinicializado. Caso positivo o *loop* é pausado, todos os parâmetros são reinicializados, e o programa espera até o processo equilíbrio ser reinicializado.
5. Caso o auto-equilibrante não tenha sido reiniciado, os valores dos sensores e motores são então lidos para a malha de controle.
6. O valor de erro do controlador é então calculado através dos valores do gyro e dos motores que são multiplicados pelos seus respectivos ganhos.
7. O valor de erro é passado através do controlador, que tentará corrigir o erro para o valor de referência desejado, que é a posição vertical de equilíbrio.
8. A saída de controle é enviada para cada motor, causando a rotação nos motores na velocidade desejada.
9. Como o sensor gyro opera na frequência de 333 Hz, o loop é atrasado para evitar que o gyro seja lido com frequência. O tamanho do atraso determina quão rápido o controlador pode responder a um distúrbio físico.
10. O loop do programa retorna ao passo 4, reiniciando o processo de equilíbrio novamente.

o **Método de Integração**

Durante cada tempo de amostragem, a velocidade angular do gyro é medida e multiplicada pelo tempo de amostragem (10 ms), fornecendo um valor aproximado para o deslocamento angular. Esse valor é somado ao ângulo anterior.

A razão para usar esse método ao invés de um cálculo numérico de integração mais preciso é a necessidade de um tempo de reação rápido. Usando um método de interpolação obtêm-se um atraso do tempo.

$$\theta(t) = \int_0^t \dot{\theta}(\tau) d\tau \approx \theta(t-1) + t\dot{\theta}(t) \quad (20)$$

Para investigar o método de integração, foi implementada a equação (20) no software RobotC e embarcado no NXT. Foi testada e verificada o cálculo correto do deslocamento angular, em graus, à medida que o NXT era girado, conforme figura 38.

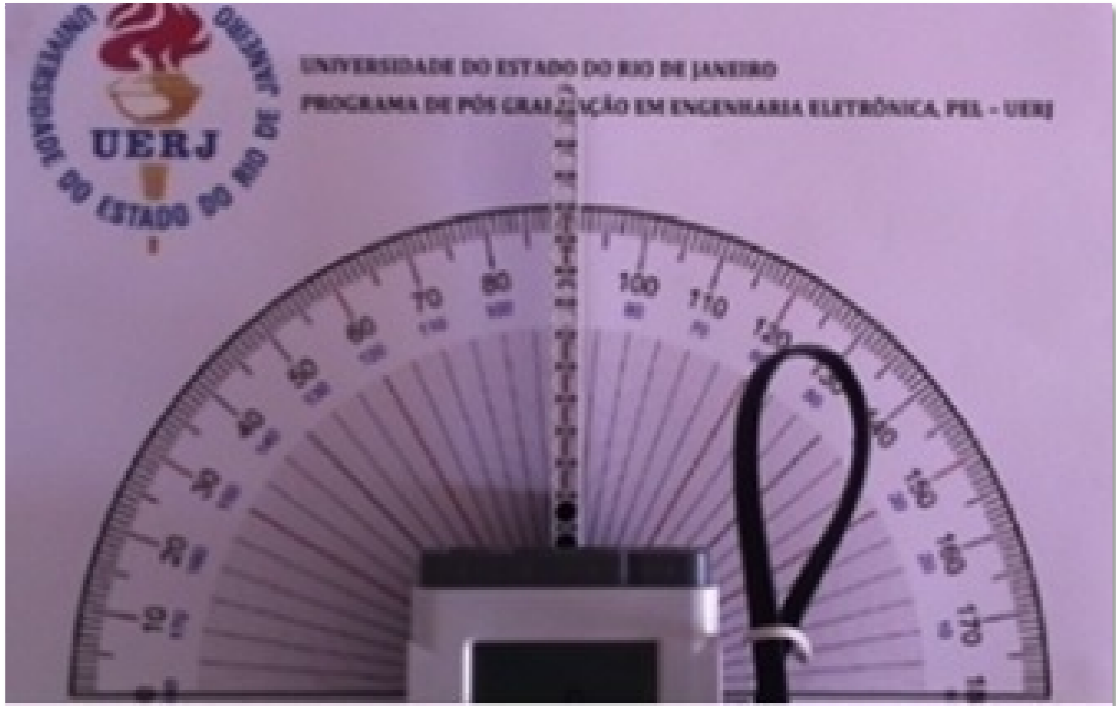


Figura 38 – Teste do método de integração.

○ Método de Derivação

A velocidade linear é calculada usando-se a posição medida de um dos *encoders*. Ela é aproximadamente:

$$\dot{y}(t) \approx \frac{y(t) - y(t-1)}{\Delta t} \quad (21)$$

○ Calibração do sensor gyro

Para compensar o problema de deriva sistêmica (*bias*) do gyro foi inserido no início do programa o cálculo de offset. Para se obter acurácia no cálculo foi criado um *loop* que faz 40 leituras do giroscópio na posição estacionária. Uma única leitura não seria suficiente visto que a leitura do sensor variar entre $\pm 1^\circ$,

sendo necessário o cálculo da média. O valor médio é o valor de offset do sensor e representa o valor zero do giroscópio. O método do processo é resumido na figura 39. Cabe ressaltar que esta calibração não é exata. Depois de zerar, o sensor gyro pode ter ainda uma pequena quantidade de *bias*.

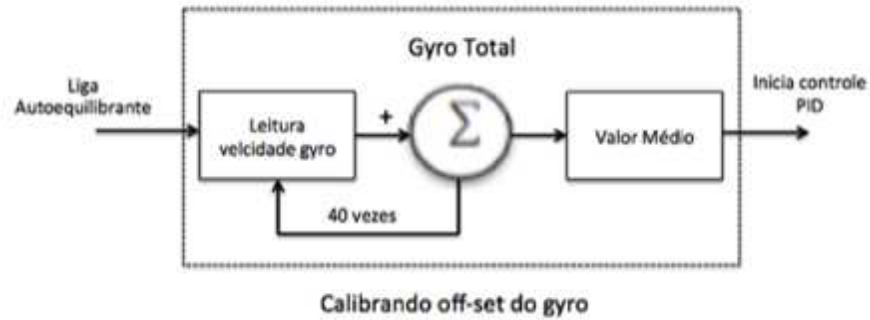


Figura 39 – Processo de cálculo do off-set do gyro.

Visto que o sensor gyro não pode ser recalibrado durante sua operação, foi desenvolvido um elemento de compensação do problema de *bias* ao longo do tempo, redirecionando o valor do verdadeiro ângulo para o ângulo de *off-set* θ_{offset} . Quando o auto-equilibrante está em equilíbrio vertical a média do ângulo ao longo do tempo permanece próximo de zero. Quando em movimento, o ângulo torna-se dependente da velocidade do robô. Para simplificar, foi considerada uma relação linear para este deslocamento, conforme equação (22).

$$\theta = \alpha * \theta_{gyro} + (1 - \alpha) * \theta_{offset} \quad (22)$$

O valor de α foi escolhido tal que o sensor gyro pode contribuir com a maioria das mudanças angulares durante curtos períodos de tempo, mas não tão perto de um que o deslocamento angular possa ocorrer.

Lindstrom (2013) determinou e validou experimentalmente a relação entre a velocidade e o ângulo de *off-set*, considerando o valor de 0,995 para α . A equação 22 e o valor de $\alpha = 0,995$ foram utilizados neste projeto para compensação do problema de bias no sensor gyro.

4.4 TESTES E VALIDAÇÃO EXPERIMENTAL

Foram conduzidos testes reais, com o robô montado e programado, para verificar seu correto funcionamento. O cenário de teste utilizado foi em superfície plana e nivelada, movimento reto, sem perturbações nem desvios. Os seguintes dados obtidos foram gravados na memória flash do NXT durante os testes, permitindo traçar as curvas para análise do comportamento do robô:

- Tempo, em segundos;
- Velocidade angular, em graus/s;
- Deslocamento angular, em graus;
- Velocidade linear, em cm/s;
- Deslocamento linear, em cm.

Cabe ressaltar que o método de *datalog* sugerido pelo RobotC é limitado em 5.000 dados. Como a taxa de amostragem do código é de 10 ms, isso corresponde ao tempo máximo de cinco segundos de log de uma variável. E caso seja necessário capturar mais de uma variável, o *datalog* de cada variável torna-se menor.

Para aumentar o *datalog* do NXT, foi verificada uma alternativa de captura dos dados através do *debugger* do RobotC. Com isso foi possível obter mais de 20.000 dados para cada variável sem comprometer o desempenho do robô. Este método também funcionou pela comunicação Bluetooth.

O sistema de controle foi testado (figura 40) sendo registrado o log das variáveis de controle durante sua operação figuras {41, 42, 43 e 44}. As curvas de cada variável ao longo do tempo foram geradas no MATLAB.



Figura 40 – Auto-equilibrante em operação.

A estabilidade do auto-equilibrante foi satisfatória. Uma pequena oscilação pode ser observada durante seu equilíbrio. No entanto, esse comportamento não representa baixo desempenho pois o deslocamento angular e linear é pequeno. Na condição de operação sem distúrbio, o deslocamento angular está em torno de $\pm 1^\circ$, (figura 41) que está dentro da faixa de acurácia do sensor gyro utilizado.

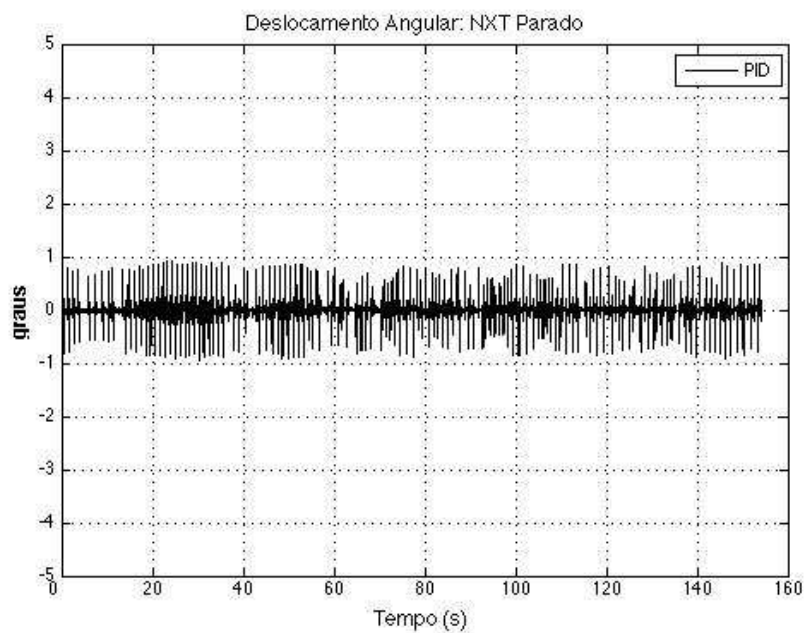


Figura 41 – Deslocamento angular do auto-equilibrante em operação.

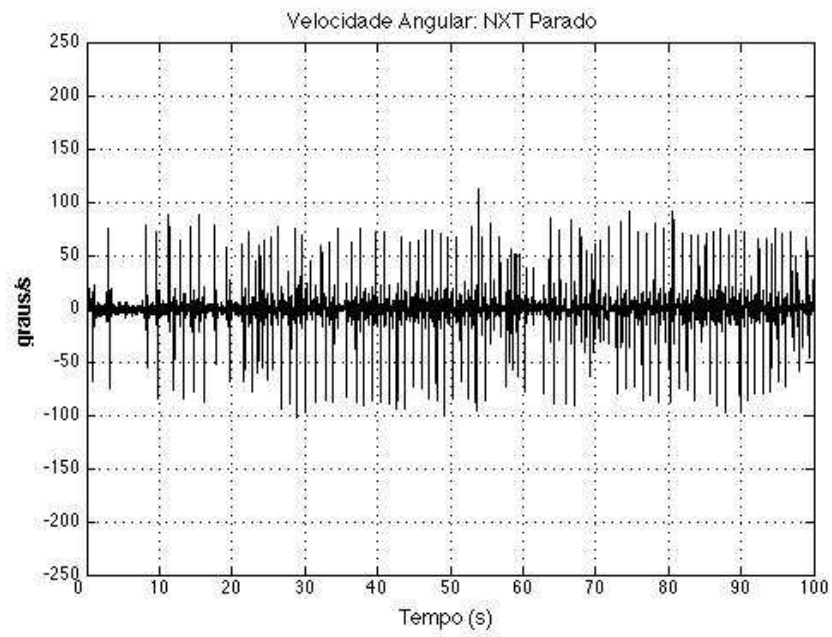


Figura 42 – Velocidade angular do auto-equilibrante em operação.

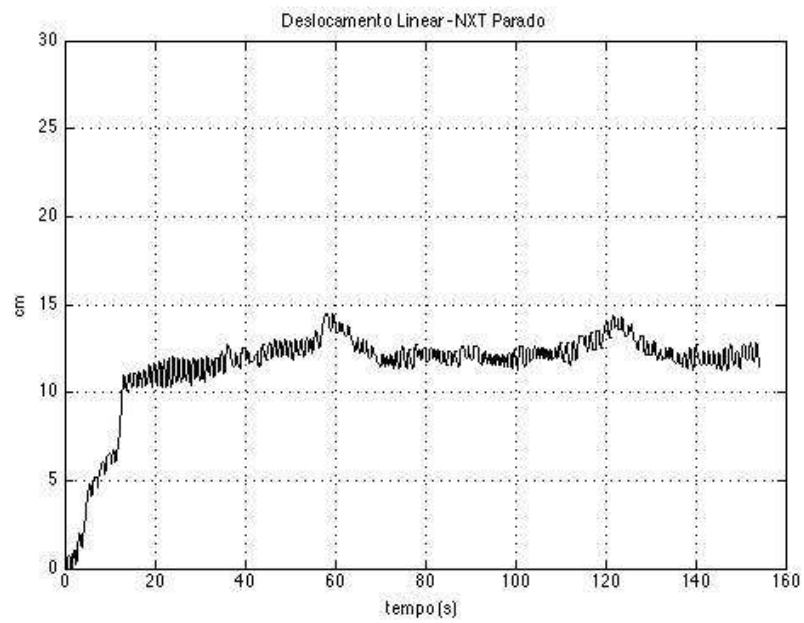


Figura 43 – Deslocamento linear do auto-equilibrante em operação.

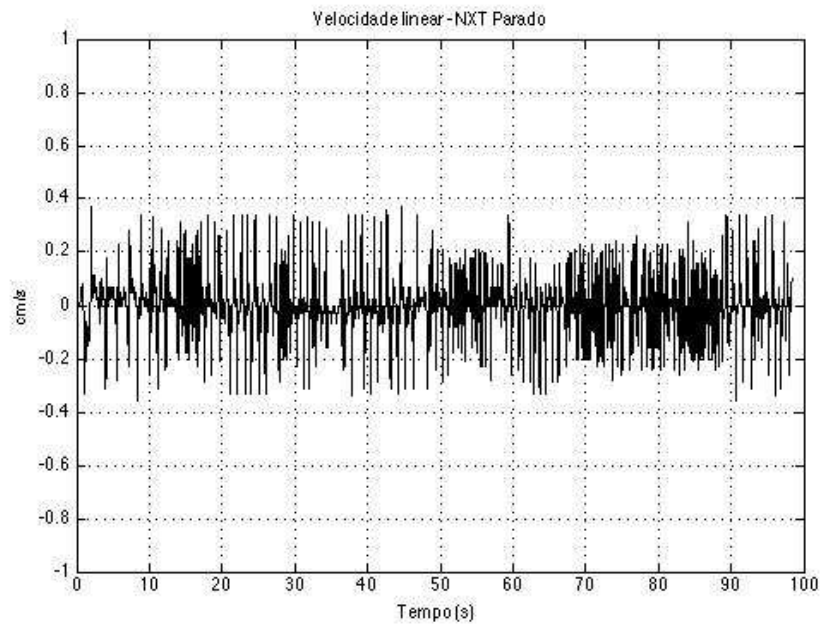


Figura 44 – Velocidade linear do auto-equilibrante em operação.

No caso de um distúrbio externo durante a execução do programa, o robô retorna rapidamente à posição de estabilidade, como visto na figura 45. Para o teste aplicado, o tempo de restabelecimento do auto-equilíbrio foi de aproximadamente quatro segundos, como mostra a figura 46.

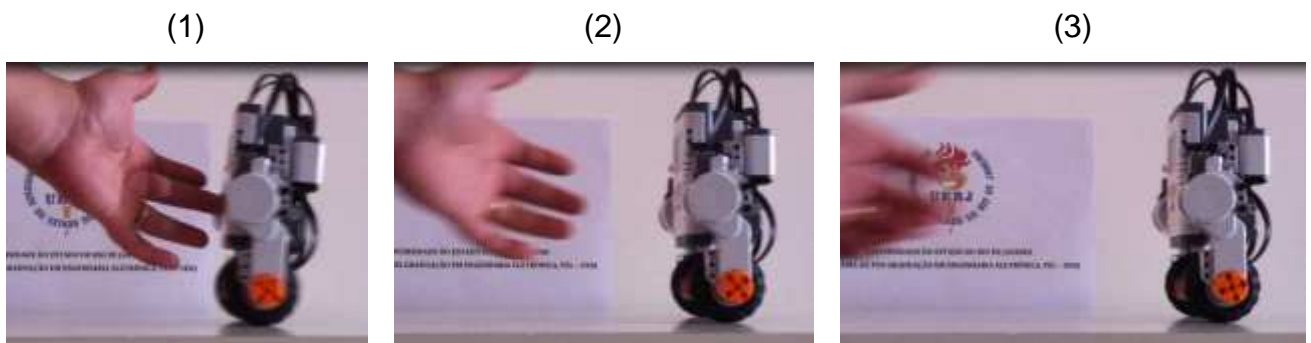


Figura 45 – Rejeição do robô para distúrbio externo.

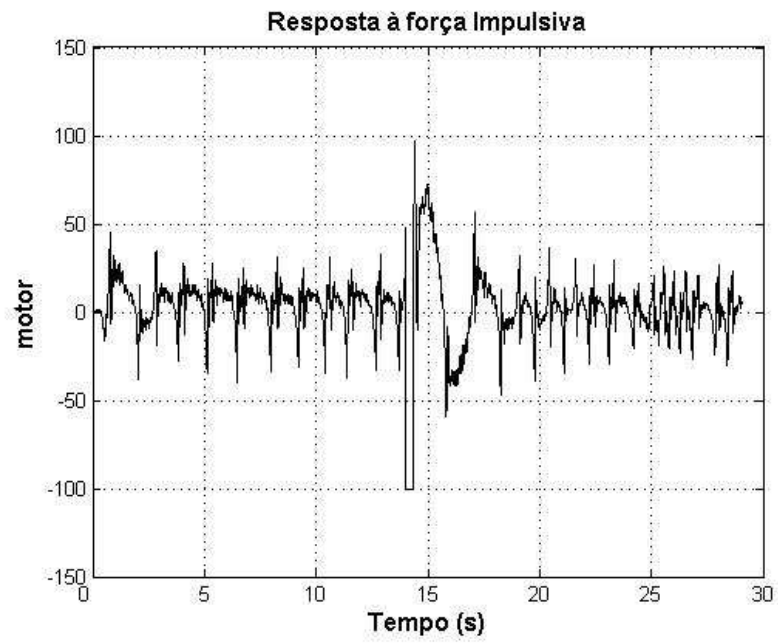


Figura 46 – Rejeição à distúrbio externo.

5 DESENVOLVIMENTO DE AMBIENTE DE SIMULAÇÃO

5.1 MODELAGEM COMPUTACIONAL

A plataforma de simulação do robô auto-equilibrante (figura 47) foi desenvolvida no ambiente MATLAB-Simulink. Essa plataforma simula a mesma resposta do auto-equilibrante NXT durante sua operação no mundo real. Para isso três principais blocos foram implementados: o bloco das variáveis de entrada (primeiro bloco – figura 47) , o bloco de controle de tarefas (segundo bloco – figura 47), e o bloco da malha de controle do robô NXT (terceiro bloco – figura 47).

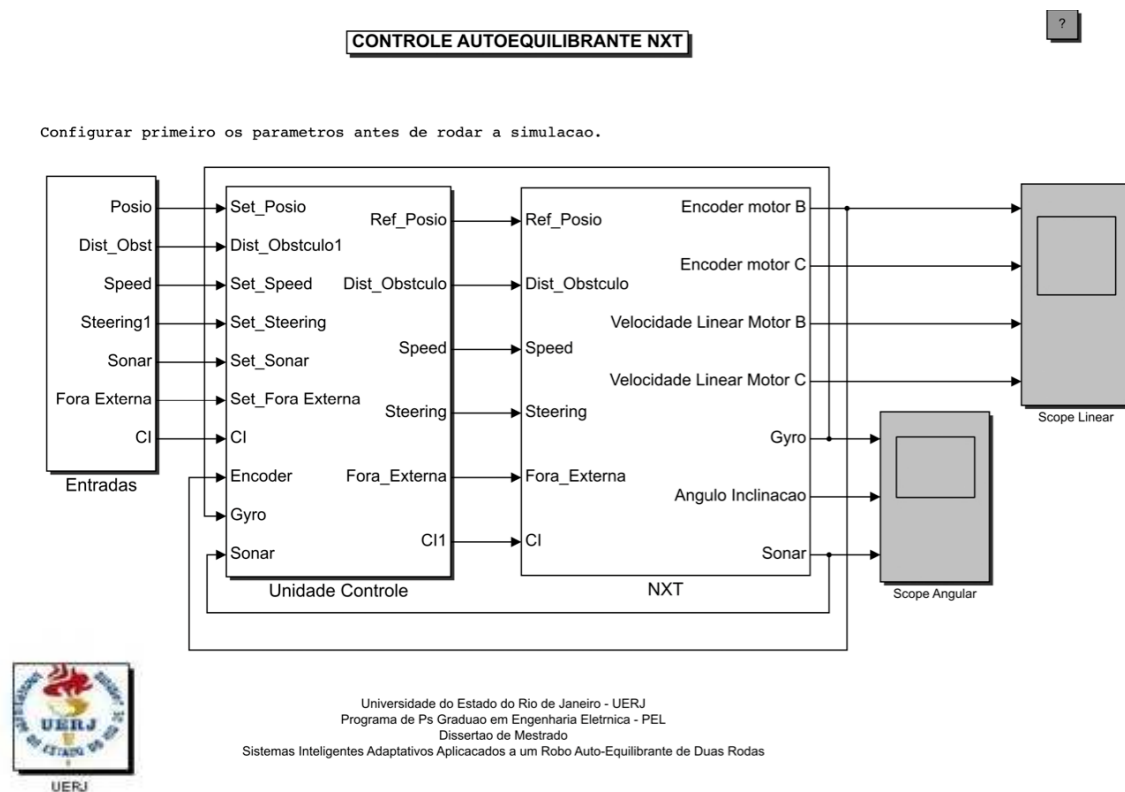


Figura 47 – Plataforma MATLAB-Simulink de simulação do auto-equilibrante NXT.

O bloco das variáveis de entrada é mostrado em detalhe na figura 48. Ao clicar sobre esse bloco é aberta a caixa de diálogo que permite introduzir os valores de cada variável de entrada do sistema:

- Velocidade;
- Posição;
- Condição Inicial (deslocamento angular);
- Força Externa.

A variável de velocidade ajusta o valor em percentual de velocidade desejada nos motores. A variável posição permite ajustar a posição final desejada em cm. A condição inicial define o ângulo inicial do robô para o caso da posição vertical inicial de teste ter alguma inclinação. A variável de força externa prevê a introdução de uma força externa no sistema durante a operação.

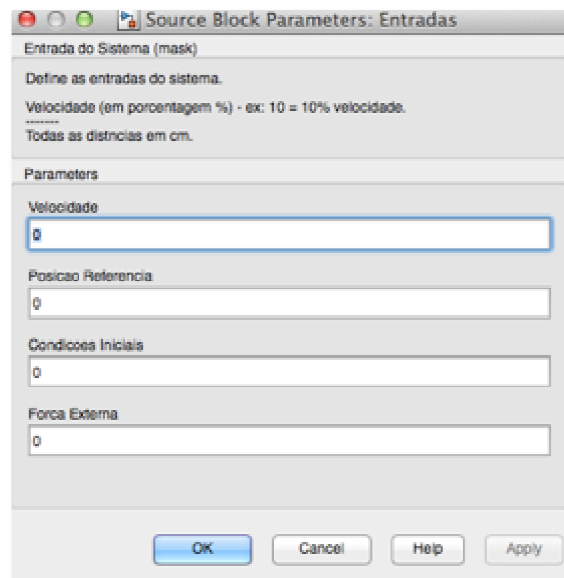


Figura 48 – Variáveis de entrada na plataforma de simulação.

O bloco do controle de tarefas permite selecionar a tarefa desejada do auto-equilibrante:

- Controle estacionário
- Controle de posição

O controle estacionário considera o vetor de referência nulo, ou seja, os deslocamentos angulares e lineares e as velocidades angulares e lineares são nulas. Portanto, o auto-equilibrante opera parado. No controle de posição o

valor da referência de posição determina o deslocamento que o robô deve executar.

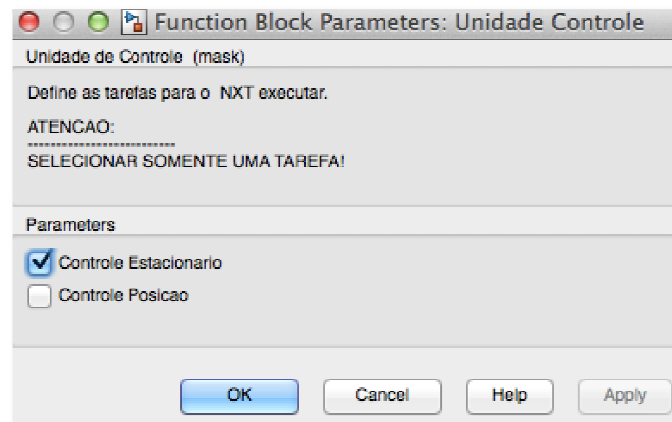


Figura 49 – Seleção do tipo de controle na plataforma de simulação.

O bloco da malha de controle do auto-equilibrante (figura 50) apresenta a malha fechada completa de controle. Permite acessar outros sub-blocos tais como:

- Modelo dinâmico do NXT;
- Controlador;
- Modelo do sensor Gyro;
- Ganhos de realimentação;
- Ajuste de velocidade;
- Ajuste de referência;
- Integrador;
- Derivador.

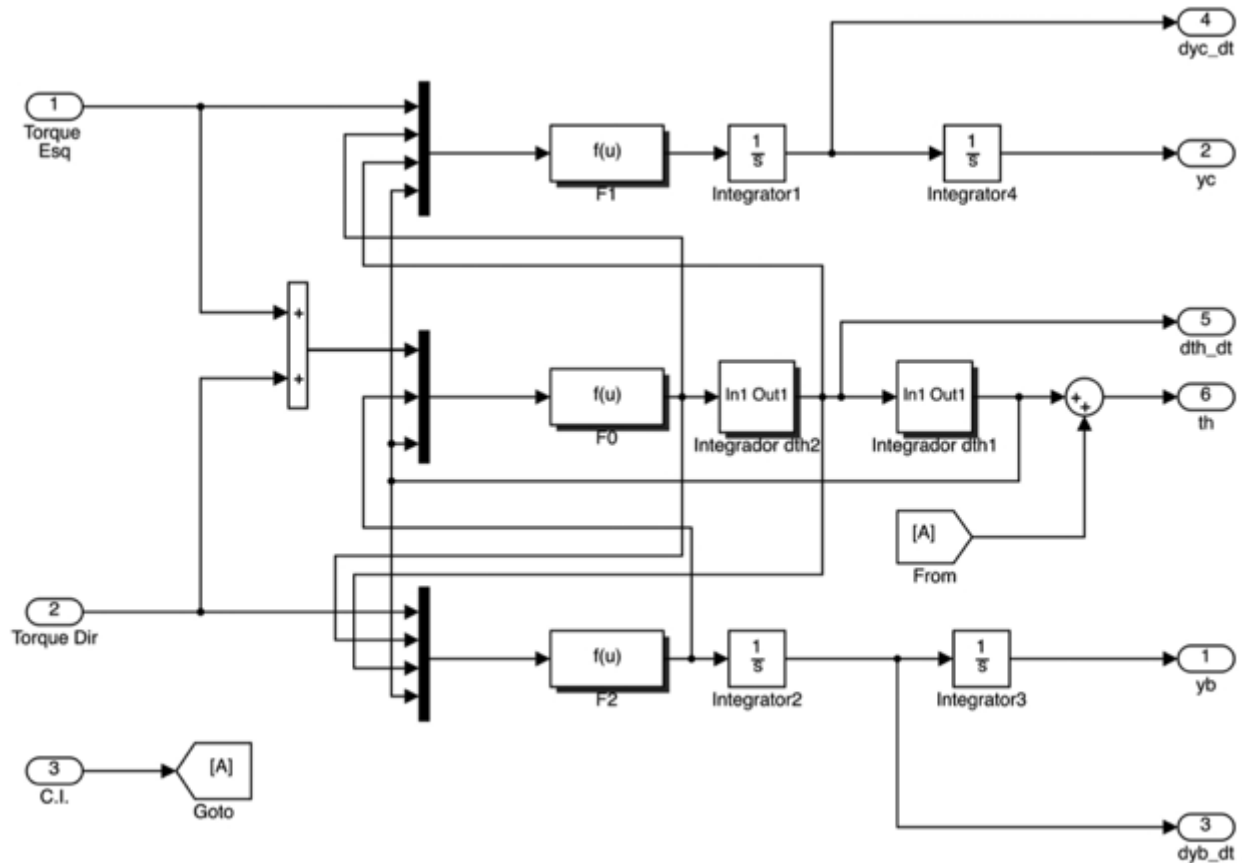
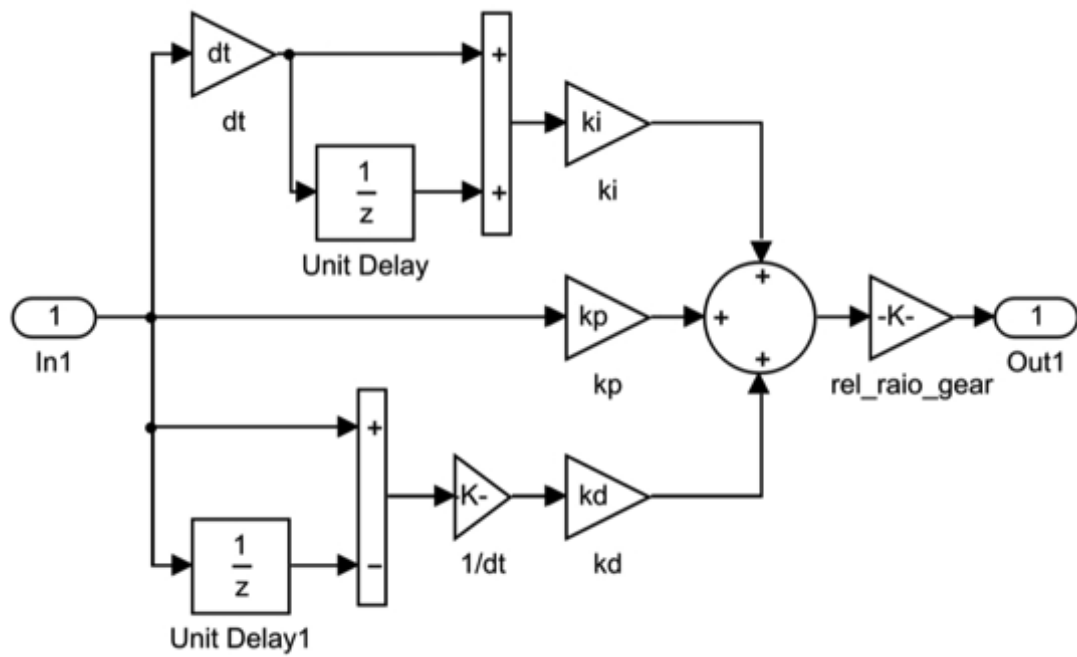


Figura 51 – Modelo dinâmico do auto-equilibrante NXT.

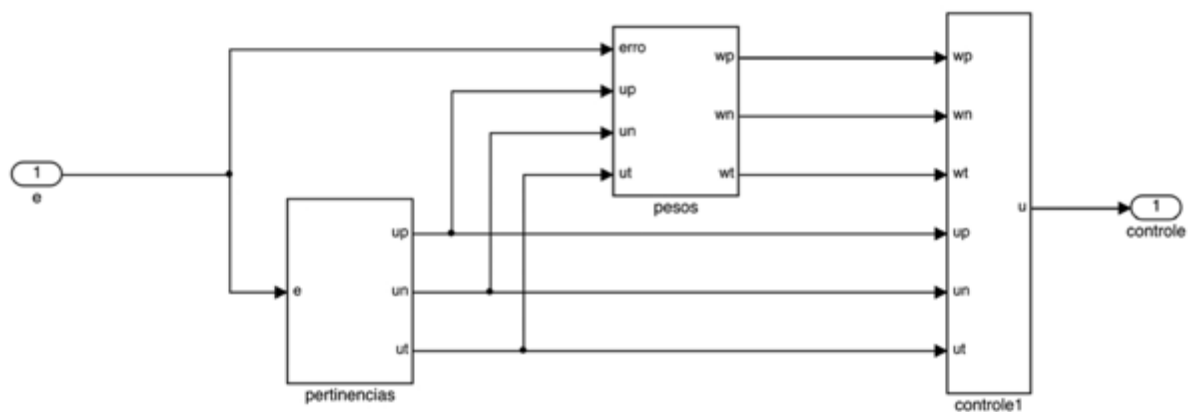
o Controlador

O bloco controlador é formado por três tipos de controladores: PID, ONFC e PIDNN; que podem ser seleccionados conforme desejado. O desenvolvimento dos controles ONFC e PIDNN será explicado no capítulo 6. Na figura 52 apresenta-se a implementação no Simulink de cada tipo de controle.

(A)



(B)



(C)

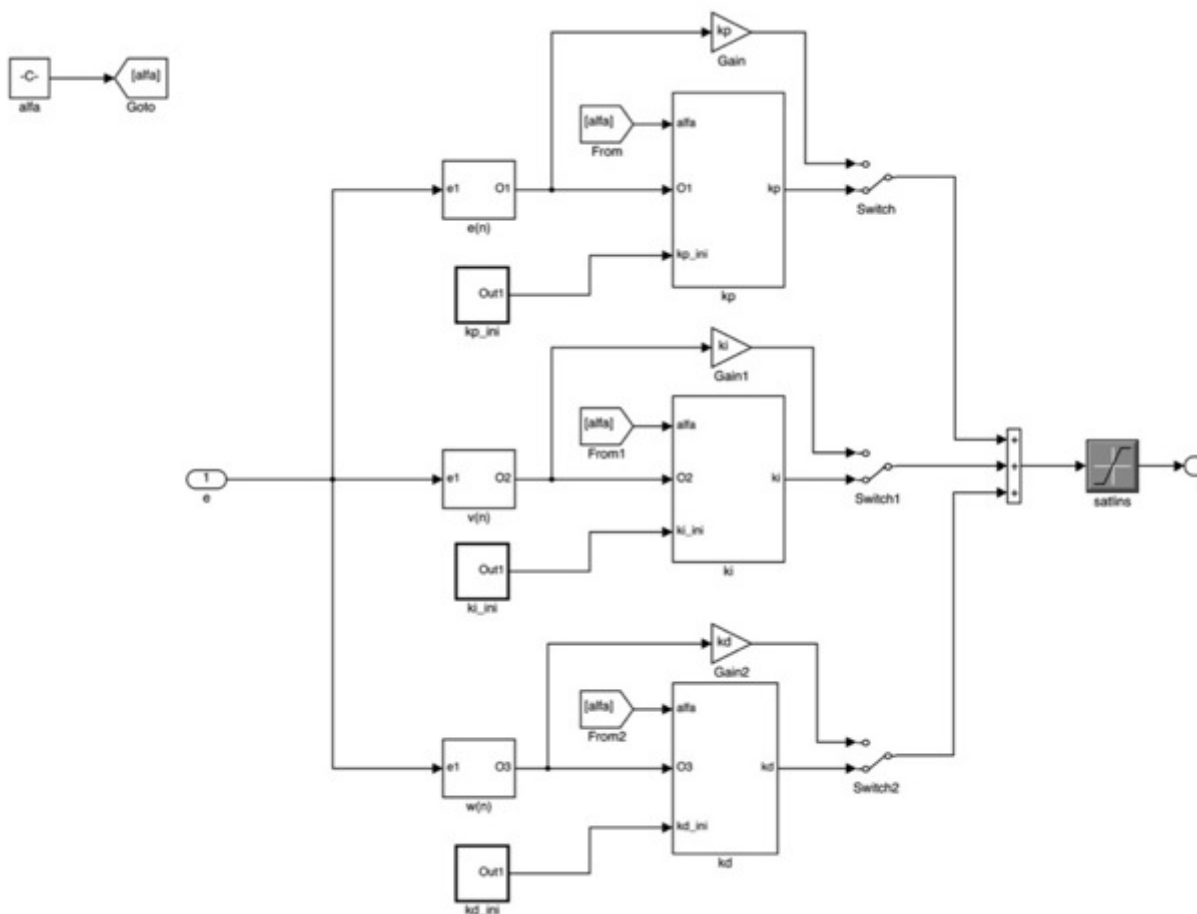


Figura 52 – Controlador: (A) PID; (B) ONFC; (C) PIDNN.

o Modelo do sensor gyro

Observa-se que o valor de saída do sensor gyro, durante operação, é periódico, ruidoso e com *spikes* positivos e negativos, conforme figura 53. Como o gyro também possui o problema de flutuação em seus valores de saída, é necessário considerar que parte de sua resposta deve-se à presença desse ruído dinâmico.

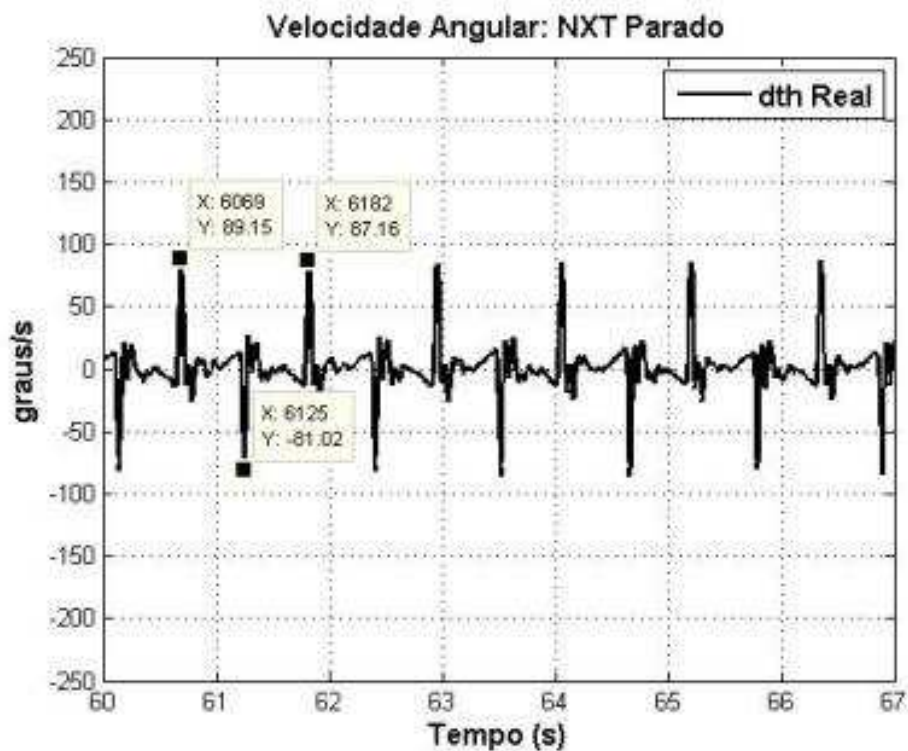


Figura 53 – Resposta dinâmica do sensor gyro.

Para simular o comportamento ruidoso do gyro foi inserido um bloco no simulink com dois geradores de pulso, defasados em 0,25 segundos; amplitude de 0,2; período de 0,5 segundos e *duty cycle* de 5%. Esses sinais são subtraídos gerando um trem de pulsos conforme figura 54 – B.

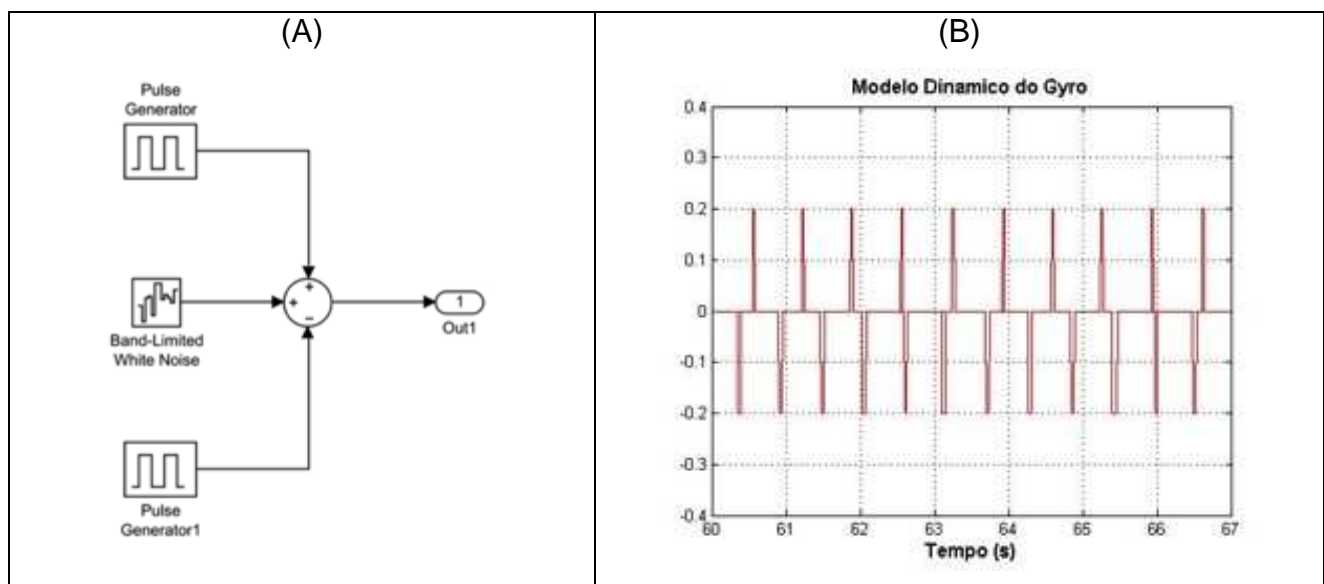


Figura 54 – Modelo do sensor gyro: (A) Bloco Simulink do ruído do sensor gyro. (B) Resposta dinâmica do bloco.

- **Ganhos de realimentação**

Os ganhos de realimentação podem ser ajustados pela janela da figura 55.

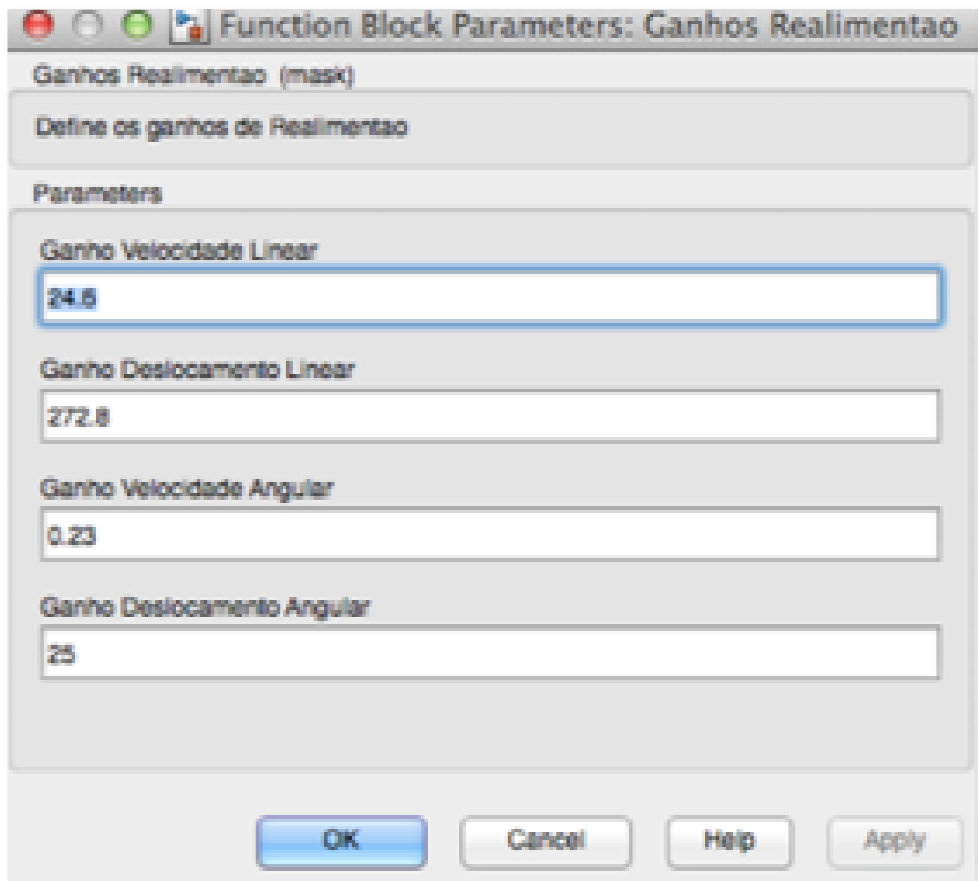


Figura 55 – Ganhos de realimentação.

- **Integração**

A implementação da equação de integração (20) no Simulink é dada pela figura 56.

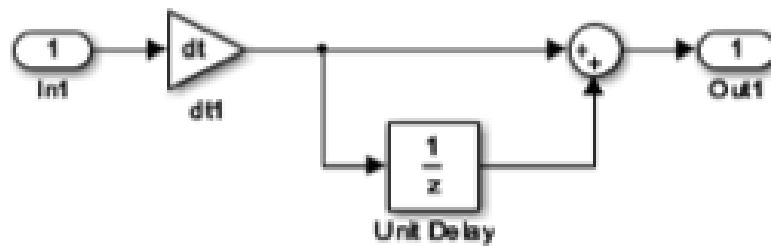


Figura 56 – integração.

o Derivação

A implementação da equação de derivação (21) no Simulink é dada pela figura 57.

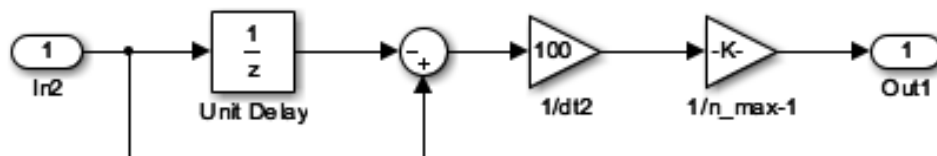


Figura 57 – Derivação.

o Compensação do problema de *bias* do gyro

Para minimizar o problema de bias do sensor gyro, foi implementado no Simulink um filtro (figura 58). Este filtro é o mesmo usado no código do RobotC e discutido na seção 4.3.

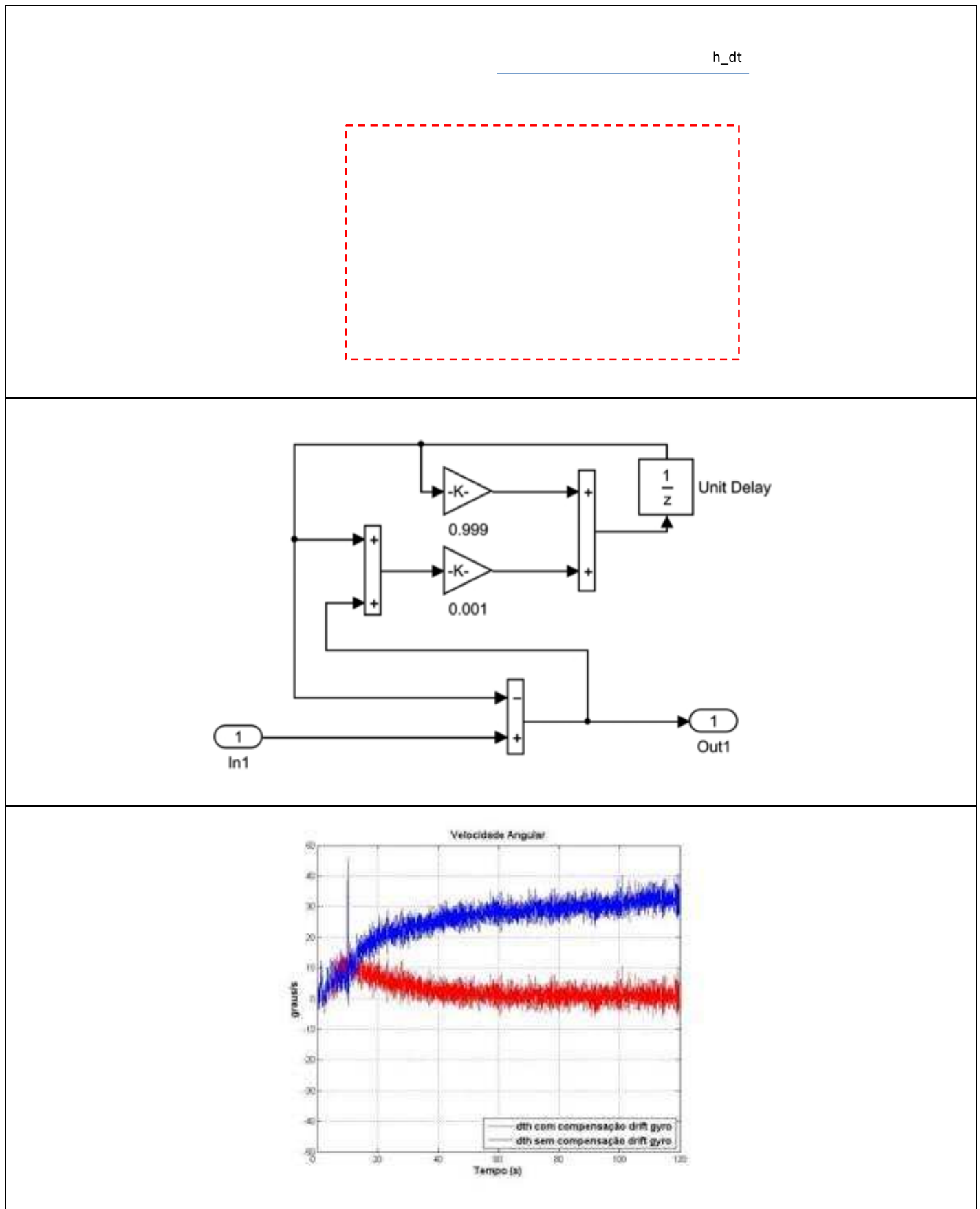


Figura 58 – Implementação do filtro de *bias* no Simulink e efeito na resposta da velocidade angular.

5.2 TESTES E VALIDAÇÃO DO MODELO COMPUTACIONAL

A metodologia utilizada para validar o modelo matemático do auto-equilibrante exige a comparação da resposta de controle em simulação e em hardware. Para esse fim, as equações matemáticas devem ser inseridas em um modelo computacional que simulará a resposta real.

A validação do modelo exige que os resultados dessa simulação sejam equivalentes à realidade de interesse, apresentando um valor mínimo de erro médio quadrático.

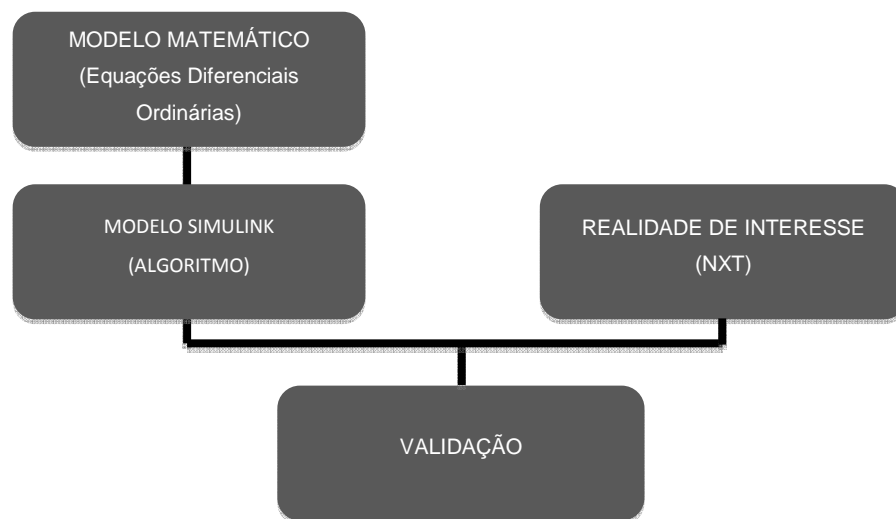


Figura 59 –Sistemática de validação do modelo.

Para validarmos o modelo computacional, desenvolvido no ambiente MATLAB-Simulink, realizamos um teste experimental de auto-equilíbrio do robô numa superfície plana, considerando apenas o controle estacionário. As mesmas condições de testes foram inseridas no ambiente de simulação.

As figuras 60(a), 61(a), 62(a) e 63(a) apresentam as curvas sobrepostas de simulação e operação real do auto-equilibrante para fins comparativos. Em todos os casos observa-se a equivalência entre a resposta real e simulada.

Também foram apresentadas as distribuições de frequência simulada e experimental de cada variável nas figuras 60(b), 61(b), 62(b) e 63(b). Verifica-se que em todos os casos a proximidade entre os resultados.

Por fim, verifica-se nas figuras 60(c), 61(c), 62(c) e 63(c) a correlação entre as variáveis simuladas e reais como forma de validarmos o modelo computacional.

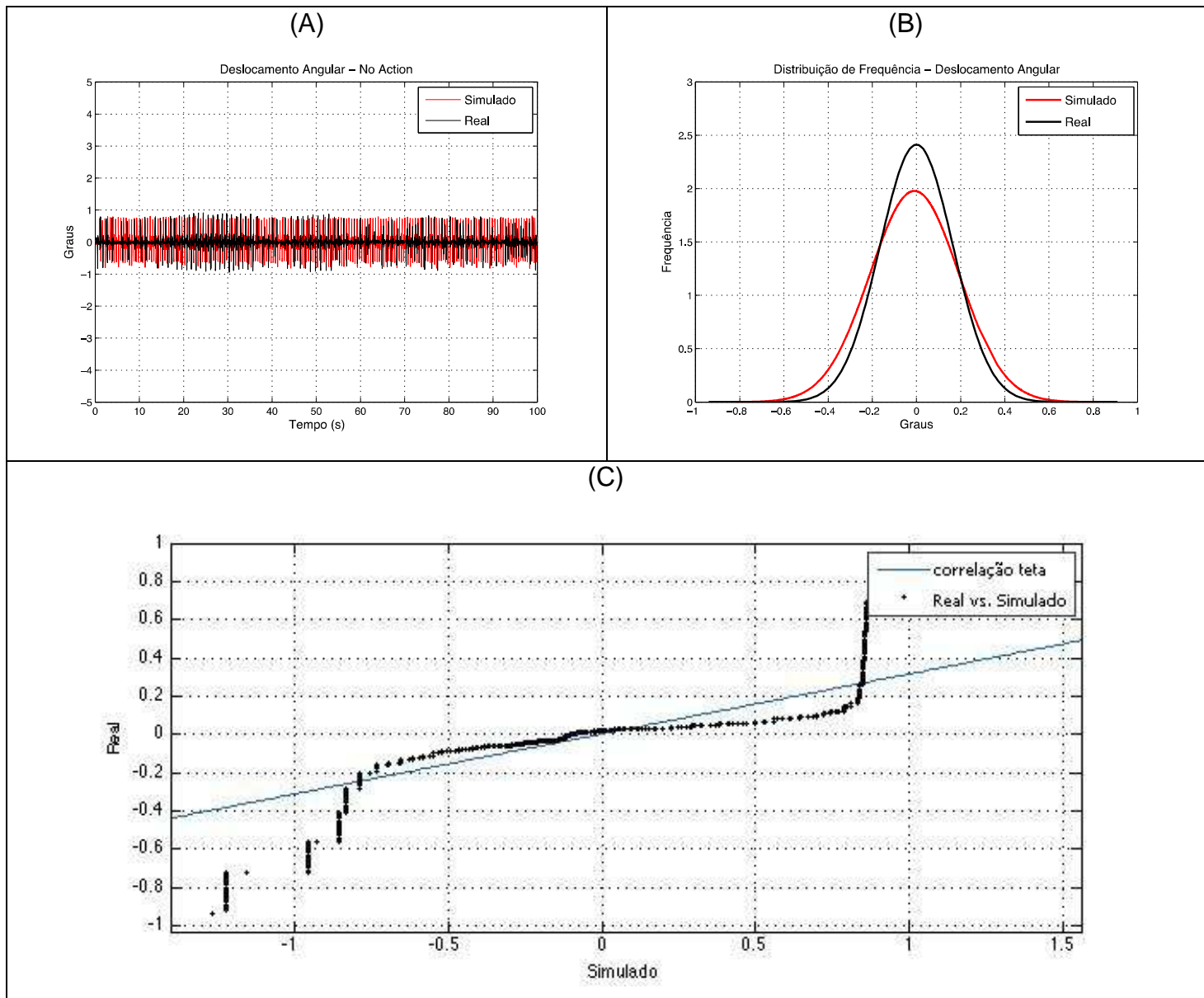


Figura 60 –Deslocamento angular: (A) real e simulado para operação de controle PID estacionário;(B) Distribuição de Frequência;(C) Correlação entre simulado e real.

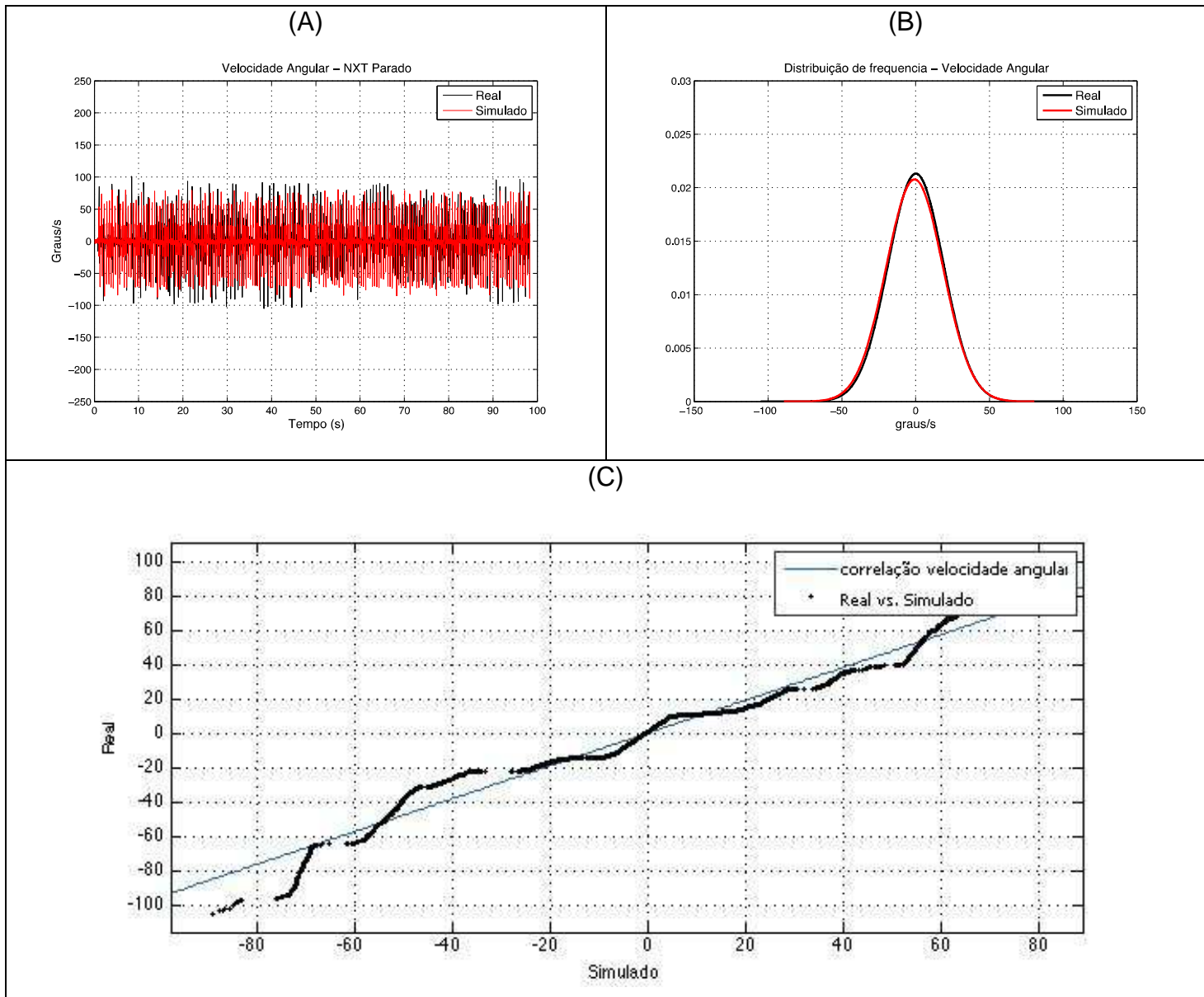


Figura 61 – (A) Velocidade angular real e simulado para operação de controle PID estacionário. (B) Distribuição de Frequência. (C) Correlação entre simulado e real.

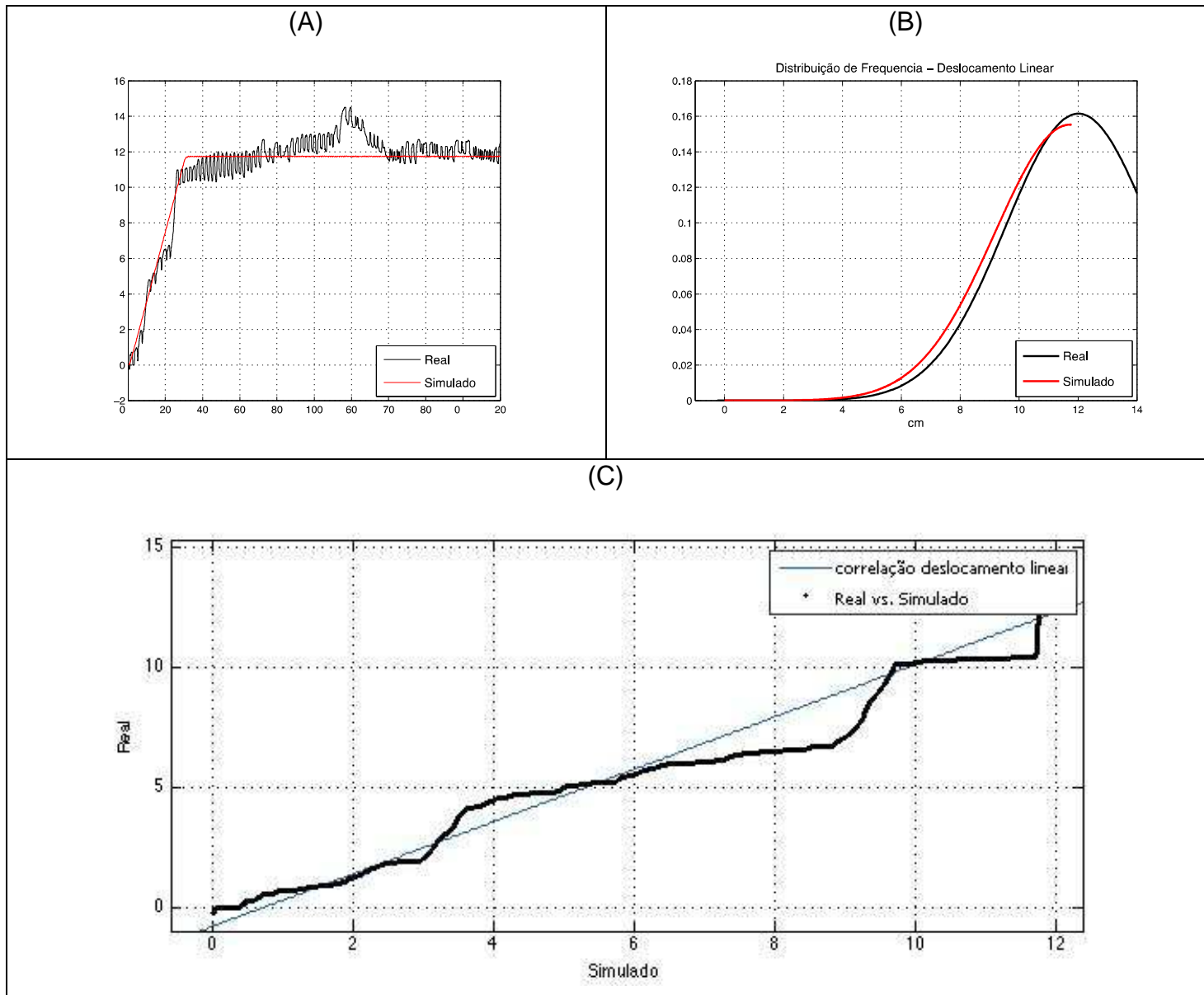


Figura 62 – (A) Deslocamento linear real e simulado para operação de controle PID estacionário. (B) Distribuição de Frequência. (C) Correlação entre simulado e real.

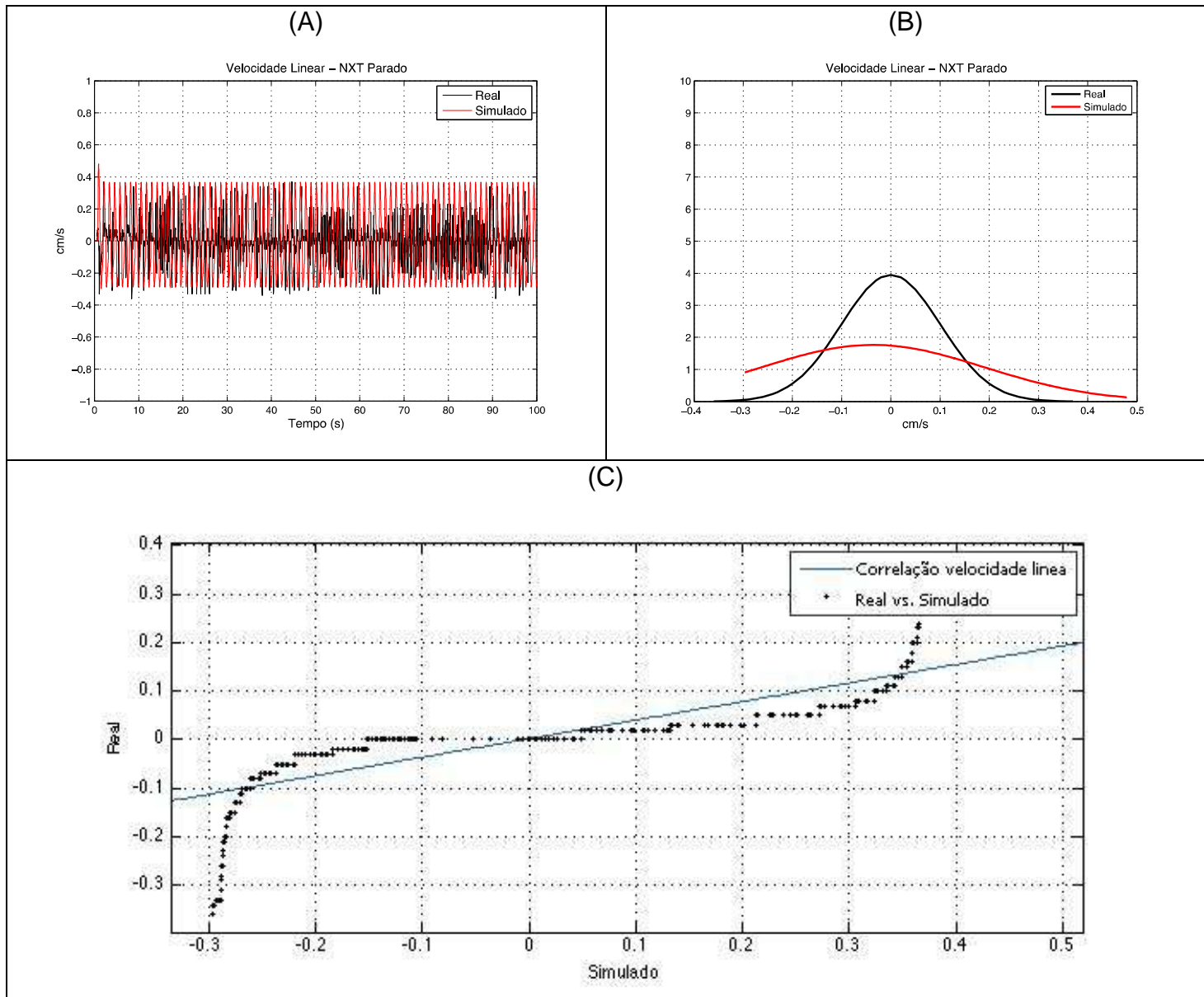


Figura 63 – (A) Velocidade linear real e simulada para operação de controle PID estacionário. (B) Distribuição de Frequência. (C) Correlação entre simulado e real.

A correlação entre os resultados reais e simulados mostram-se satisfatórias indicando que o modelo computacional é adequado para o desenvolvimento de controladores considerando o máximo de não linearidades intrínsecas do sistema.