PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

João Vitor Amarante Franco

Ziliani

Projeto, Simulação e Controle de um Robô Móvel Autoequilibrante

Projeto de Graduação

Projeto de Graduação apresentado ao Departamento de Engenharia Mecânica da PUC-Rio

Orientador: Marco Antonio Meggiolaro

Rio de Janeiro

Dezembro de 2019

Agradecimentos

Agradeço ao meus pais que puderam me dar essa oportunidade de estudo, aos meus amigos que me ajudaram quando foi necessário e meu orientador, Dr. Marco Antonio Meggiolaro, por dar suporte no desenvolvimento deste e de muitos outros trabalhos.

Resumo

Ziliani, João Vitor. Meggiolaro, Marco. **Projeto, Simulação e Controle de um Robô Móvel Autoequilibrante**. Rio de Janeiro, 2019. 81p. Projeto de Graduação - Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Este trabalho versa a aplicação de controle a um sistema de pêndulo invertido sobre duas rodas. O trabalho se inicia com a introdução ao pêndulo invertido e a motivação para a compreensão de sua dinâmica. Segue-se com a fundamentação teórica que permite a modelagem matemática, bem como as técnicas de controle utilizadas. Com a teoria esclarecida é mostrado o passo a passo para a obtenção do equacionamento da dinâmica do sistema para a realização das simulações. Na penúltima seção do trabalho, são discutidos os resultados obtidos para cada uma das simulações e condições iniciais. As principais conclusões do trabalho desenvolvido são que controles lineares são capazes de controlar sistemas altamente não-lineares mas para otimizar o tempo de correção à inclinação, que é o principal problema, é necessário aplicar técnicas de otimização de ganhos uma vez que o tempo de correção é essencial para o bom funcionamento do equipamento.

Palavras-chave:

Pêndulo invertido sobre duas rodas; LQR; PID; Lagrangiano.

Abstract

Ziliani, João Vitor. Meggiolaro, Marco. **Design, Simulation and Control of a Self-Balancing Mobile Robot**. Rio de Janeiro, 2019. 81p. Projeto de Graduação - Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

This work deals with the application of control to an inverted pendulum system under two wheels. The work begins with the introduction to the inverted pendulum and the motivation to understand its dynamics. It follows with the theoretical foundation that allows the mathematical modeling as well as the control techniques used. With the clarified theory is shown step by step to obtain the equation of the system dynamics to perform the simulations. In the penultimate section of the paper, the results obtained for each of the simulations and initial conditions are discussed. The main conclusions of the work developed are that linear controls are able to control highly non-linear systems, but to optimize the correction time for the slope, which is the main problem, it is necessary to apply gain optimization techniques since the correction time is essential for the proper functioning of the equipment.

Keywords:

Two wheeled inverted pendulum; LQR; PID; Lagrangian.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Variáveis

θ	Ângulo entre o eixo vertical z e a haste do pêndulo
θ_r	Ângulo entre o eixo vertical z e a haste do pêndulo de referência/desejado
φ	Posição angular de rotação do TWIP no eixo z
φ_r	Posição angular de rotação do TWIP no eixo z de referência/desejada
φ_f	Posição angular final rotação do TWIP no eixo z
τ_e	Toque na roda esquerda
τ_d	Toque na roda direita
ψ_e	Posição angular da roda esquerda
ψ_{er}	Posição angular da roda esquerda de referência/desejada
ψ_d	Posição angular da roda direita
ψ_{dr}	Posição angular da roda direita de referência/desejada
ω	Velocidade rotacional do pêndulo
d	Distância entre as duas rodas
$\vec{g}$	Aceleração da gravidade
h	Altura do centro de gravidade do TWIP
l	Comprimento do pêndulo até o centro de massa
m	Massa do sistema
r	Raio da roda
I	Tensor inercial
$T_{translação}$	Energia cinética de translação
$T_{rotação}$	Energia cinética de rotação
T	Energia cinética total
U_g	Energia potencial gravitacional
U	Energia potencial total
V_p	Velocidade translacional longitudinal do pêndulo

V_{pr} Velocidade translacional longitudinal do pêndulo de referência/desejada

1 Introdução

A sociedade está sempre em busca de maneiras novas, eficientes e práticas de locomoção urbana. As motocicletas e bicicletas são, sem dúvidas, as opções mais tradicionais e de baixo custo que encontramos, no entanto devido a velocidade que estes desempenham, não podem ser usadas entre pedestres e por isso novas abordagens ao problema da mobilidade urbana são aventadas.

Dentre diversas soluções ao problema de transporte pessoal, existe o chamado transportador pessoal robótico autoequilibrante (TPRE). Este equipamento que consiste em uma plataforma com duas rodas paralelas lado a lado que atuam individualmente, assemelhando-se ao funcionamento do clássico pêndulo invertido Choquehuanca *et al.* (2010).

Em contraste com as tradicionais soluções de transporte pessoal como motocicletas e bicicletas, o TPRE se destaca pela capacidade de rotação em torno do próprio eixo, tamanho compacto, baixa emissão de ruído e fonte de alimentação elétrica que permite que o veículo seja carregado em qualquer ambiente com tomadas de energia convencionais assim como carregamos nossos notebooks.

O transportador pessoal robótico autoequilibrante mais famoso do mundo foi revelado ao público em dezembro de 2001 vide Figura 1. O empreendedor novaiorquino Dean Kamen demonstrou o produto na emissora americana de televisão ABC News em um programa diário matinal de grande audiência chamado *“Good Morning America”.* Originalmente o *Segway,* como é chamado, apresentava três configurações e velocidade que chegavam até 19km/h.

Figura 1 – Segway desenvolvido pela Segway Inc.

Devido sua versatilidade, robôs autoequilibrantes podem ser usados como transporte pessoal e - aliado a manipuladores e visão por câmeras - pode também ser usado em ambientes exploratórios cujas limitações espaciais podem jogar a favor do design compacto do TPRE.

A agência espacial norte americana (NASA) criou o *Robonaut RMP* vide Figura 2 que combina a tecnologia autoequilibrante do *Segway*, a capacidade de manipulação de objetos e visão por câmeras Ambrose *et al.* (2004).

Figura 2 – Robonaut desenvolvido pela NASA

1.1 Objetivo

O objetivo deste trabalho é desenvolver um modelo 3D da dinâmica de um robô móvel autoequilibrante sobre duas rodas, permitindo a consideração do acoplamento entre as dinâmicas longitudinal e lateral.

O trabalho também procura propor duas estratégias de controle (PID e LQR) capazes de estabilizar o veículo em torno de $\theta = 0^o$, ou seja, verticalmente ao mesmo tempo em que o robô executa uma trajetória curva em 3D.

1.2 Pêndulo Invertido sobre Carro

O pêndulo invertido em sua forma mais simples de um pêndulo invertido pode ser representado por uma massa ligada através de uma haste sem massa a uma base de massa M. O nome dado a este sistema é pêndulo invertido sobre carro conforme mostrado na Figura 3. A concepção mais simples do problema contempla o movimento no plano lateral xy para que seja possível equilibrar a haste em $\theta = 0^o$.

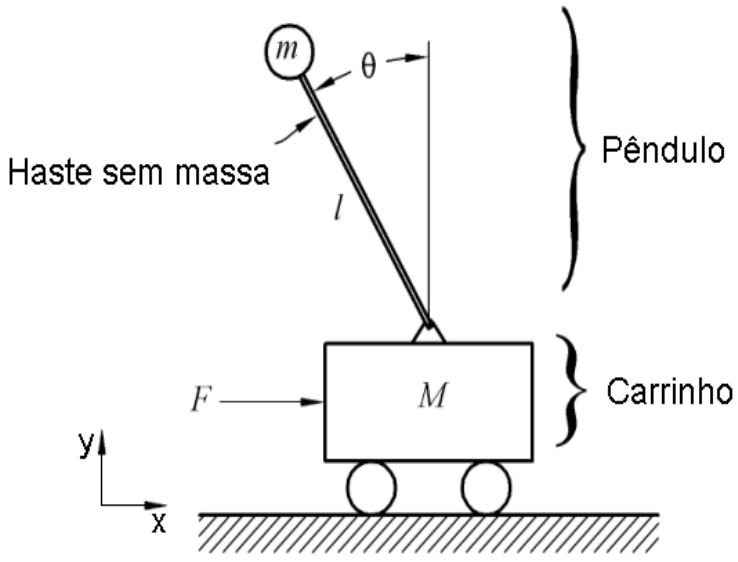

Figura 3 – Pêndulo invertido sobre carro

Apesar da dinâmica do pêndulo invertido sobre carro ser instável, o sistema conserva o equilíbrio caso não haja nenhuma perturbação, todos os componentes estejam perfeitamente balanceados e a inclinação da haste seja nula, ou seja, $\theta_{t=0} =$

0°. É sabido que tais condições não podem ser atingidas uma vez que o alinhamento e balanceamento dos componentes nunca será perfeito, existem perturbações externas ao sistema e a inclinação do sistema dificilmente será nula, por conseguinte é necessária a atuação da força $\boldsymbol{F}$ no sistema mostrado na Figura 3 para equilibrar o pêndulo.

Diversos trabalhos foram publicados para compreender a modelagem e o controle de sistemas contendo o pêndulo invertido simples. No início do século 19, Stephenson *et al.* (1908) analisou este problema e teorizou maneiras de estabilização do mesmo e empiricamente concluiu que o pêndulo pode ser estabilizado mediante aplicação de movimentos verticais oscilatórios de alta frequência na base articulada. No ano seguinte, Stephenson chegou às mesmas conclusões para os pêndulos invertidos duplos e triplos.

Com as conclusões obtidas por Stephenson *et al.* (1908), Lowenstern *et al.* (1932) desenvolveu com sucesso a equação geral de movimento para pêndulos invertidos.

Ao longo da segunda metade do século 19, pesquisadores como Bagddanoff *et al.* (1965), Blitzer *et al.* (1965), e Ness *et al.* (1967), conduziram experimentos para atingir a estabilização do pêndulo invertido partindo de condições iniciais distintas.

O escopo de alguns artigos a partir dos anos 80, como o de Sahba *et al.* (1983) e Yamakawa *et al.* (1989), se concentraram em técnicas de controle para sistema não linear como o descrito pelo pêndulo duplo invertido sobre carro.

O pêndulo invertido sobre carro é tido como o caso de pêndulo invertido mais estudado na literatura e têm função comparativa a outros pêndulos.

1.3 Pêndulo Invertido sobre Duas Rodas

Assim como já descrito, o pêndulo invertido sob duas rodas (TWIP), alvo do presente estudo, consiste em uma plataforma com duas rodas paralelas lado a lado que atuam individualmente a partir do desequilíbrio do sistema, assemelhando-se ao funcionamento do clássico pêndulo invertido Choquehuanca *et al.* (2010), vide Figura 4.

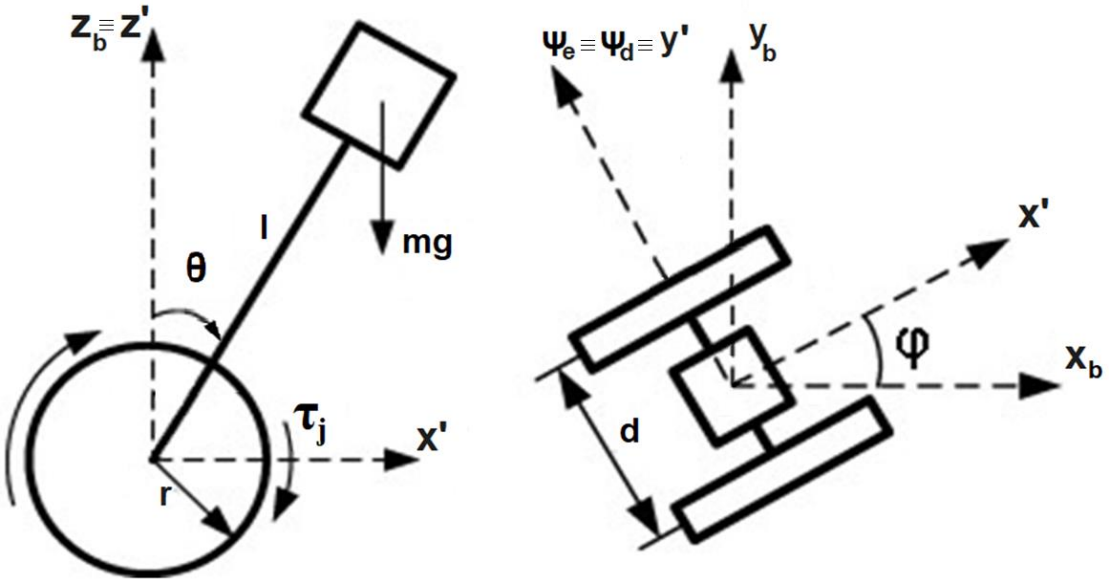

Figura 4 – Pêndulo invertido sobre duas rodas

Um dos primeiros trabalhos envolvendo o TWIP foi o de Kanamura *et al.* (1988) que anos mais tarde foi aperfeiçoado por Yuta *et al*. (1996). Este trabalho contemplou a dinâmica 2D do sistema bem como a implementação de um algoritmo para o controle autônomo do robô.

Um ano após o lançamento comercial do primeiro pêndulo invertido para transporte pessoal, o *Segway*, Kim *et al.* (2003) desenvolveu um robô soldador não tripulado, capaz de realizar soldas em ambientes hostis e com limitações espaciais vide Figura 5.

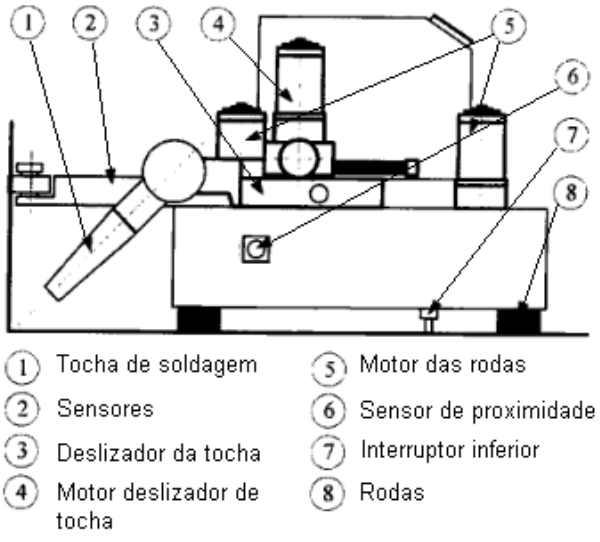

Figura 5 – Robô móvel de soldagem desenvolvido por Kim

Algumas abordagens para a estabilização do pêndulo invertido sobre duas rodas foram usadas ao longo da última década. Kim *et al.* (2008) desenvolveu um controle inteligente aplicando algoritmo de aprendizado de rede neural sem conhecer a dinâmica do sistema.

Dois anos após Kim *et al.* (2008), Sekiyama *et al.* (2010) desenvolveu um modelo tridimensional do TWIP e aplicou, devido à forte não linearidade do sistema, um esquema de controle sliding (SMC) para garantir a convergência das variáveis de estado para os valores desejados. No mesmo ano, Li *et al.* (2010) modelou o atrito entre as rodas e a superfície para desenvolver um controle robusto fuzzy adaptativo para o TWIP.

Apesar do sistema TWIP ser um problema relativamente bem estudado, o modelo dinâmico 3D é raramente explorada pois reserva severas complicações matemáticas devido a não linearidade, exigindo desse modo grande poder computacional para simulações.

1.4 Organização do Trabalho

Este trabalho está organizado em cinco capítulos e oito apêndices:

- No Capítulo 1 é apresentado o tema do trabalho, uma breve descrição sobre os dois pêndulos invertidos mais comuns na literatura e são apresentados os trabalhos mais relevantes referente a eles.

- No Capítulo 2 é feita uma revisão dos fundamentos teóricos para modelar as equações da dinâmica, linearizar o sistema e das técnicas de controle PID e LQR para a estabilização do sistema.

- No Capítulo 3 é descrito em detalhes o desenvolvimento matemático da dinâmica do pêndulo invertido, a obtenção dos ganhos PID e LQR e a prototipagem do pêndulo invertido sobre duas rodas.

- No Capítulo 4 são relatados os resultados obtidos e uma análise comparativa entre as respostas do sistema aos ganhos PID e LQR.

- No Capítulo 5 é apresenta a discussão dos resultados, as conclusões do trabalho e as propostas de trabalhos futuros.

- No Apêndice A é apresentado o código desenvolvido em ambiente MATLAB com o objetivo de obter as equações lagrangianas, acelerações, linearização, discretização e finalmente os ganhos do LQR.

- No Apêndice B é apresentado o código desenvolvido em ambiente MATLAB com o objetivo de simular o comportamento do sistema baseado no equacionamento não linear.

- No Apêndice C é apresentada uma função auxiliar para simular o comportamento do sistema TWIP.

- No Apêndice D é apresentado o código desenvolvido em ambiente Arduino que tem por objetivo carregar pacotes para o desenvolvimento do código para o protótipo do sistema TWIP.

- No Apêndice E é apresentado o código desenvolvido em ambiente Arduino que tem por objetivo ler os sensores encoders.

- No Apêndice F é apresentado o código desenvolvido em ambiente Arduino que executa a atuação dos motores.

- No Apêndice G é apresentado o código desenvolvido em ambiente Arduino que tem por objetivo ler os sensores giroscópio.

- No Apêndice H é apresentado o código desenvolvido em ambiente Arduino que tem por objetivo acelerar a leitura múltipla dos sensores.

2 Fundamentos Teóricos

Neste capítulo, é feita uma breve revisão dos conceitos de modelagem da dinâmica do sistema, linearização Jacobiana e técnicas de controle tais como o Proporcional, Integral e Derivativo (PID) e o Regulador linear quadrático (LQR). Todos os conceitos apresentados nas subseções seguintes foram utilizados para desenvolver o problema do pêndulo invertido sobre rodas.

Importante ressaltar que não há um método geral para o desenvolvimento de controladores não lineares segundo Slotine e Li *et al.* (1991), logo técnicas alternativas são aplicadas aos problemas não-lineares como controle robusto, adaptativo, linearização por realimentação*, trial-and-error* e *gain-scheduling*, entretanto este trabalho concentra os esforços em linearizar o sistema em torno do ponto de operação e aplicação de duas técnicas de controle a título de comparação.

2.1 Linearização Jacobiana

Grande parte da teoria de controle foi desenvolvida para sistemas lineares, contudo sistemas reais são, em sua maioria, não-lineares, logo é necessário desenvolver uma representação linear do modelo não-linear em torno do ponto de operação.

Como a linearização é uma aproximação em torno do ponto de operação, logo o comportamento do sistema linearizado só pode ser considerado uma boa reprodução do sistema real para simulações nas vizinhanças do ponto de linearização.

Apesar de ser feita a linearização das equações da dinâmica, o comportamento do sistema foi simulado utilizando o equacionamento não-linear, ou seja, a linearização foi utilizada aqui apenas para a obtenção dos ganhos LQR do sistema para então, ser aplicada a lei de controle.

Segundo Amba *et al.* (2015) para analisar a natureza do comportamento do sistema, as funções que descrevem a dinâmica do sistema devem ser linearizadas localmente em $x(k)$ por expansão em Série de Taylor.

A linearização Jacobiana resulta em um sistema linear não existente que se aproxime do sistema real no ponto de operação fornecido $x(k)$.

Dado o sistema 1:

$$\begin{aligned} \Delta\dot{x} &= A\Delta x + B\Delta u \\ \Delta y &= C\Delta x + D\Delta u \end{aligned} \tag{1}$$

Onde $\Delta x = x - x_0$, $\Delta u = u - u_0$, x representa os estados definidos para o sistema — que no sistema analisado são os ângulos da das rodas e do pêndulo em relação a vertical e suas respectivas velocidades — , x_0 representa os valores dos estados x no ponto de equilíbrio e u_0 a ação de controle no ponto de equilíbrio.

As matrizes A (2) e B (4) são definidas da seguinte maneira:

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} & \frac{\partial f_1}{\partial x_5} & \frac{\partial f_1}{\partial x_6} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} & \frac{\partial f_2}{\partial x_5} & \frac{\partial f_2}{\partial x_6} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} & \frac{\partial f_3}{\partial x_5} & \frac{\partial f_3}{\partial x_6} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} & \frac{\partial f_4}{\partial x_5} & \frac{\partial f_4}{\partial x_6} \\ \frac{\partial f_5}{\partial x_1} & \frac{\partial f_5}{\partial x_2} & \frac{\partial f_5}{\partial x_3} & \frac{\partial f_5}{\partial x_4} & \frac{\partial f_5}{\partial x_5} & \frac{\partial f_5}{\partial x_6} \\ \frac{\partial f_6}{\partial x_1} & \frac{\partial f_6}{\partial x_2} & \frac{\partial f_6}{\partial x_3} & \frac{\partial f_6}{\partial x_4} & \frac{\partial f_6}{\partial x_5} & \frac{\partial f_6}{\partial x_6} \end{bmatrix} \tag{2}$$

Onde a matriz $\boldsymbol{f}$ é dada por:

$$f = \begin{bmatrix} 0 \\ \dot{\psi}_e \\ 0 \\ \dot{\psi}_d \\ 0 \\ \dot{\theta} \end{bmatrix} \tag{3}$$

$$B = \begin{bmatrix} 0 \\ \Omega \\ 0 \\ \Sigma \\ 0 \\ \Lambda \end{bmatrix} \tag{4}$$

Onde Ω, Σ e Λ são os coeficientes dos termos contendo torque nas equações das acelerações $\dot{\psi}_e$, $\dot{\psi}_d$ e $\dot{\theta}$ respectivamente.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

2.2 Regulador Linear Quadrático

A teoria do controle ótimo, desenvolvida por Kalman *et al.* (1960) lida a minimização da operação de um sistema dinâmico. Uma das principais conclusões do trabalho de Kalman *et al.* (1960) foi a técnica chamada de regulador linear quadrático (LQR, do inglês *Linear-Quadratic Regulator*) pois ela otimiza os ganhos do controle linear.

Dado um sistema genérico de múltiplas entradas e múltiplas saídas (MIMO) representado pelo sistema 7:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{7}$$

Segundo Portela *et al.* (2016) o controle do regulador linear quadrático encontra o vetor $u(t)$ utilizando de um estado inicial genérico, que seja capaz de transferir um estado para outra região do espaço de estados desejado. A lei de controle que estabiliza o sistema mostrado em 7 é:

$$u(t) = -Kx(t) \tag{8}$$

Desse modo temos que:

$$\dot{x}(t) = (A - BK)x(t) \tag{9}$$

onde $\boldsymbol{K}$ é a matriz de realimentação de estados dada pela equação 10.

$$K = R^T B^T S \tag{10}$$

Segundo Ogata *et al*. (2011), $\boldsymbol{K}$ é uma matriz simétrica definida positiva e deve satisfazer a equação reduzida de Ricciati:

$$A^T S + SA - SB^T R^{-1} B^T S + Q = 0 \tag{11}$$

onde $\boldsymbol{Q}$ e $\boldsymbol{R}$ são as matrizes de ponderação simétrica semidefinida positiva.

A função custo $\boldsymbol{J}$ a ser minimizada é dada por:

$$J = \int_0^\infty (x(t)^T Q x(t) \; + \; u(t)^T R u(t)) dt \tag{12}$$

2.3 Controle PID

O Controle PID (proporcional, integral e derivativo) é a técnica mais tradicional encontrada na indústria e por isso todos os livros acadêmicos de introdução ao controle de sistemas dedicam uma seção para discutir essa técnica.

A larga aplicabilidade e popularidade na indústria é devida sua simplicidade frente a outros modelos e aplicação genérica em plantas onde o modelo matemático é desconhecido.

A técnica de controle PID, assim como o nome sugere, possui três ganhos, o proporcional K_p, integral T_i e o derivativo T_d e é definido pela equação 13:

$$u(t) = K_p e(t) + T_i \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \tag{13}$$

Onde, $e(t) = SP - PV(t)$ é o erro (SP é o ponto de ajuste e PV é a variável do processo no tempo).

O método de obtenção mais popular para os ganhos proporcional, integral e derivativo é o de Ziegler-Nichols *et al.* (1942). O método de Ziegler-Nichols possui duas versões:

Método 1: Os parâmetros ***L*** e ***T*****,** *delay* e constante respectivamente, são obtidos por simulação ou experimentalmente como mostrado na Figura 6.

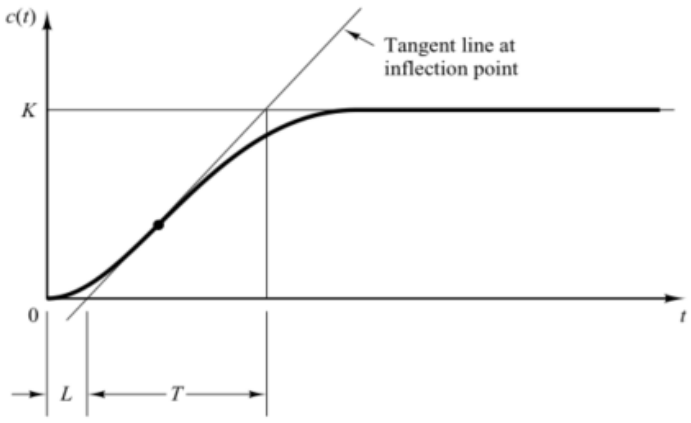

Figura 6 – Identificação de L e T

Após a determinação dos valores de L e T, o valor dos ganhos são obtidos pela Tabela 1.

Tabela 1 – Ganhos PID método 1 Ziegler-Nichols

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Método 2: O ganho integral é configurado como $T_i = \infty$ e o derivativo $T_d = 0$.

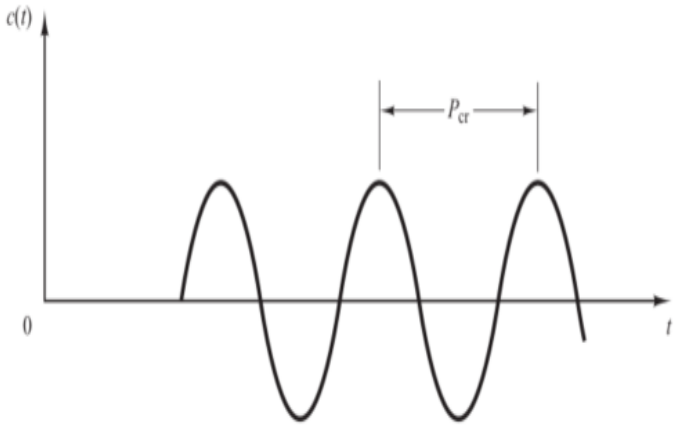

Figura 7 – Identificação do período crítico

Usando a planta da Figura 7, aumenta-se o valor de K_p de 0 a um valor crítico, ou seja, onde as saídas sustentem oscilações.

Quando isso ocorrer, o período é identificado e então o valor dos ganhos são obtidos com base nos dados da Tabela 2.

Tabela 2 – Ganhos PID método 2

Type of Controller	K_p	T_i	T_d
P	$0.5K_{\text{cr}}$	∞	0
PI	$0.45K_{\text{cr}}$	$\frac{1}{1.2}P_{\text{cr}}$	0
PID	$0.6K_{\text{cr}}$	$0.5P_{\text{cr}}$	$0.125P_{\text{cr}}$

Apesar de funcional este método, ele não é aplicável a sistemas de natureza instável, logo o método de Ziegler-Nichols não pode ser aplicado no TWIP.

A alternativa mais recorrente ao método mostrado é ajustar os ganhos de modo maneira empírica, ou seja, ajustar os parâmetros e verificar se os resultado da simulação são “satisfatórios” para o sistema.

2.4 Formalismo Lagrangiano

Para modelar a dinâmica do sistema, o tratamento Lagrangiano foi selecionado pois além de reduzir substancialmente o esforço analítico devido à abordagem energética e não vetorial como a Newtoniana, o lagrangiano nos permite obter o número mínimo de equações necessárias, uma vez que as coordenadas generalizadas sejam escolhidas corretamente.

Para obter a função lagrangiana (14) é necessário determinar a energia cinética e a energia potencial total do conjunto base e pêndulo.

$$\mathcal{L} = T - U \tag{14}$$

A energia cinética total T é dada pela soma das energias cinéticas de translação $T_{translação}$ e de rotação $T_{rotação}$, vide equação 15.

$$T = T_{translação} + T_{rotação} \tag{15}$$

A contribuição da energia potencial para o equacionamento é dada na forma de energia potencial gravitacional, vide equação 16.

$$U = U_g = mgh_{CG} \tag{16}$$

A obtenção da equação lagrangiana é dada pela equação diferencial 17.

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = Q_i \tag{17}$$

Onde $i = [1, 2, 3]$ e $q_1 = \psi_e$ e $q_2 = \psi_d$ são as posições angulares da roda esquerda e direita respectivamente e $q_3 = \theta$ é a posição angular da haste do pêndulo em relação a vertical (eixo z). O termo Q_i são perturbações externas ao sistema.

3 Modelagem do Sistema

Neste capítulo, é descrito em detalhes o desenvolvimento matemático da dinâmica do pêndulo invertido pela forma lagrangiana, a obtenção dos ganhos LQR, valores para os ganhos PID por tentativa e erro e a prototipagem do pêndulo invertido sobre duas rodas bem como a descrição dos componentes utilizados

3.1 Equacionamento

Assim como descrito no capítulo 2, a abordagem utilizada para a obtenção da dinâmica do sistema foi a abordagem lagrangiana devido a complexidade do desenvolvimento vetorial.

Para a obtenção das equações foram feitas três considerações:

1. Atrito viscoso negligenciável devido às baixas velocidades;
2. Atrito estático infinito;
3. A base do robô não possui massa significativa, ou seja, a única massa considerada foi a do pêndulo.

Para a obtenção da função lagrangiana (14), é necessário encontrar as energias cinéticas e potenciais.

A energia cinética do sistema é descrita por:

$$T = \underbrace{\frac{1}{2} m V_p^2}_{\text{Translacional-longitudinal}} + \underbrace{\frac{1}{2} \omega^T I \omega}_{\text{Rotacional}} \tag{18}$$

Para obter a energia cinética total do sistema, é necessário encontrar a velocidade do pêndulo V_p, o tensor de inércia I e a velocidade de rotação ω do sistema.

Iniciando o equacionamento pelo termo translacional-longitudinal, temos a relação 19 para V_p.

$$\begin{pmatrix}\dot{x}_p\\ \dot{y}_p\\ \dot{z}_p\end{pmatrix} = \begin{pmatrix}\dot{x}_b\\ \dot{y}_b\\ \dot{z}_b\end{pmatrix} + l\begin{bmatrix}\cos\varphi\cos\theta & \sin\theta\sin\varphi\\ \cos\theta\sin\varphi & \sin\theta\cos\varphi\\ -\sin\theta & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ \dot{\varphi}\end{bmatrix} \tag{19}$$

Onde,

$$\begin{pmatrix}\dot{x}_b\\ \dot{y}_b\\ \dot{z}_b\end{pmatrix} = r\begin{bmatrix}\frac{\dot{\psi}_e+\dot{\psi}_d}{2}\begin{bmatrix}\cos\varphi\\ \sin\varphi\end{bmatrix}\\ 0\end{bmatrix} \tag{20}$$

Logo,

$$\begin{pmatrix}\dot{x}_p\\ \dot{y}_p\\ \dot{z}_p\end{pmatrix} = \begin{bmatrix}\frac{\dot{\psi}_e+\dot{\psi}_d}{2}\begin{bmatrix}\cos\varphi\\ \sin\varphi\end{bmatrix}\\ 0\end{bmatrix} + l\begin{bmatrix}\cos\varphi\cos\theta & \sin\theta\sin\varphi\\ \cos\theta\sin\varphi & \sin\theta\cos\varphi\\ -\sin\theta & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ \dot{\varphi}\end{bmatrix} \tag{21}$$

$$\dot{x}_p = \frac{\dot{\psi}_e+\dot{\psi}_d}{2} r\cos\varphi + l\dot{\theta}\cos\varphi\cos\theta - l\dot{\varphi}\sin\theta\sin\varphi \tag{22}$$

$$\dot{y}_p = \frac{\dot{\psi}_e+\dot{\psi}_d}{2} r\sin\varphi + l\dot{\theta}\cos\theta\sin\varphi - l\dot{\varphi}\sin\theta\cos\varphi \tag{23}$$

$$\dot{z}_p = -l\dot{\theta}\sin\theta \tag{24}$$

Por tanto o termo de velocidade ao quadrado V_p^2 da energia cinética é dado por:

$$V_p^2 = \left(\frac{\dot{\psi}_e+\dot{\psi}_d}{2} r\right)^2 + l^2\dot{\theta}^2 + l^2\dot{\varphi}^2\sin^2\theta + \left(\dot{\psi}_e+\dot{\psi}_d\right) rl\dot{\theta}\cos\theta \tag{25}$$

Então temos energia cinética da dinâmica translacional-longitudinal descrita pela equação 27.

$$T_{translação} = \frac{1}{2} m V_p^2 \tag{26}$$

$$T_{translação} = \frac{1}{2} m \left[\left(\frac{\dot{\psi}_e + \dot{\psi}_d}{2} r \right)^2 + l^2 \dot{\theta}^2 + l^2 \dot{\varphi}^2 \sin^2 \theta + \left(\dot{\psi}_e + \dot{\psi}_d \right) r l \dot{\theta} \cos \theta \right] \tag{27}$$

Para o termo rotacional, temos a velocidade rotação dada por:

$$\omega = \begin{bmatrix} -\dot{\theta} \sin \varphi \\ \dot{\theta} \cos \varphi \\ \dot{\varphi} \end{bmatrix} \tag{28}$$

Onde,

$$\dot{\varphi} = \left(\dot{\psi}_d + \dot{\psi}_e \right) \frac{r}{d} \tag{29}$$

Onde $\dot{\varphi}$ nada mais é que a velocidade de rotação do TWIP em torno do eixo z, conforme mostrado na Figura 4.

$$I^{"} = \begin{bmatrix} I_r & 0 & 0 \\ 0 & I_\theta & 0 \\ 0 & 0 & I_\varphi \end{bmatrix} \tag{30}$$

Como foram utilizados dois sistemas cartesianos distintos para a obtenção das velocidades conforme ilustrado na Figura 4, é necessário rotacionar o tensor inercial, e para isto foi encontrada a matriz de rotação $\boldsymbol{R}$:

$$R^{xyz}_{x^{"}y^{"}z^{"}} = \begin{bmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \tag{31}$$

$$R = \begin{bmatrix} \cos \theta \cos \varphi & -\sin \varphi & \cos \varphi \sin \theta \\ \sin \varphi \cos \theta & \cos \varphi & \sin \varphi \sin \theta \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \tag{32}$$

Para encontrar o tensor inercial $\boldsymbol{I}$, temos:

$$I = RI^{"}R^{T} \tag{33}$$

$$I = \begin{bmatrix} \cos\theta\cos\varphi & -\sin\varphi & \cos\varphi\sin\theta \\ \sin\varphi\cos\theta & \cos\varphi & \sin\varphi\sin\theta \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} I_r & 0 & 0 \\ 0 & I_\theta & 0 \\ 0 & 0 & I_\varphi \end{bmatrix} \begin{bmatrix} \cos\theta\cos\varphi & \sin\varphi\cos\theta & -\sin\theta \\ -\sin\varphi & \cos\varphi & 0 \\ \cos\varphi\sin\theta & \sin\varphi\sin\theta & \cos\theta \end{bmatrix} \tag{34}$$

$$I = \begin{bmatrix} \xi\cos^2\varphi + I_\theta\sin^2\varphi & \cos\varphi\sin\varphi\,(\xi - I_\theta) & \cos\varphi\sin\theta\cos\theta\big(I_\varphi - I_r\big) \\ \cos\varphi\sin\varphi\,(\xi - I_\theta) & \cos\varphi & \sin\varphi\sin\theta\cos\theta\,\big(I_\varphi - I_r\big) \\ \cos\varphi\sin\theta\cos\theta\big(I_\varphi - I_r\big) & \sin\varphi\sin\theta\cos\theta\,\big(I_\varphi - I_r\big) & I_r\sin^2\theta + I_\varphi\cos^2\theta \end{bmatrix} \tag{35}$$

Onde $\xi = I_r\cos^2\theta + I_\varphi\sin^2\theta$. Logo a energia cinética de rotação é dada pelo desenvolvimento da equação 35.

$$T_R = \frac{1}{2}I_{11}\omega_1^2 + \frac{1}{2}I_{22}\omega_2^2 + \frac{1}{2}I_{33}\omega_3^2 + I_{12}\omega_1\omega_2 + I_{13}\omega_1\omega_3 + I_{23}\omega_2\omega_3 \tag{36}$$

$$T_R = \frac{1}{2}I_\theta\dot{\theta}^2 + \frac{1}{2}\big(I_r\sin^2\theta + I_\varphi\cos^2\theta\big)\dot{\varphi}^2 \tag{37}$$

Por tanto a energia cinética total do sistema é descrita pela equação 18 é:

$$T = \frac{1}{2}m\left[\left(\frac{\dot{\psi}_e + \dot{\psi}_d}{2}r\right)^2 + l^2\dot{\theta}^2 + l^2\dot{\varphi}^2\sin^2\theta + \big(\dot{\psi}_e + \dot{\psi}_d\big)rl\dot{\theta}\cos\theta\right] + \frac{1}{2}I_\theta\dot{\theta}^2 + \frac{1}{2}\big(I_r\sin^2\theta + I_\varphi\cos^2\theta\big)\dot{\varphi}^2 \tag{38}$$

No sistema, temos a energia potencial descrita pelo potencial gravitacional:

$$U = mgl\cos\theta \tag{39}$$

Logo a função lagrangiana (14) é dada pela equação 40:

$$\mathcal{L} = \frac{1}{2}m\left[\left(\frac{\dot{\psi}_e + \dot{\psi}_d}{2}r\right)^2 + l^2\dot{\theta}^2 + l^2\dot{\varphi}^2\sin^2\theta + \left(\dot{\psi}_e + \dot{\psi}_d\right)rl\dot{\theta}\cos\theta\right] + \frac{1}{2}I_\theta\dot{\theta}^2 + \frac{1}{2}\left(I_r\sin^2\theta + I_\varphi\cos^2\theta\right)\dot{\varphi}^2 - mgl\cos\theta \tag{40}$$

Para encontrar a primeira equação da dinâmica do sistema, temos a expressão 41:

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\psi}_e}\right) - \frac{\partial\mathcal{L}}{\partial\psi_e} = Q_1 \tag{41}$$

Onde Q_1é o torque produzido pela roda esquerda, logo temos:

$$-\frac{m}{2}\left(lr\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right) - \left(\frac{r^2}{2}\right)\left(\ddot{\psi}_d + \ddot{\psi}_e\right) + 2\left(\frac{lr}{d}\right)^2\left(\sin^2\theta\left(\ddot{\psi}_d - \ddot{\psi}_e\right) + \dot{\theta}\sin 2\theta\left(\dot{\psi}_d - \dot{\psi}_e\right)\right)\right) - \left(\frac{r}{d}\right)^2\left(\left(I_r - I_\varphi\right)\left(\dot{\theta}\sin 2\theta\right)\left(\dot{\psi}_d - \dot{\psi}_e\right)\right) - \left(\frac{r}{d}\right)^2\left(I_r\sin^2\theta + I_\varphi\cos^2\theta\right)\left(\ddot{\psi}_d - \ddot{\psi}_e\right) = \tau_e \tag{42}$$

Para a segunda equação da dinâmica, temos expressão 42:

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\psi}_d}\right) - \frac{\partial\mathcal{L}}{\partial\psi_d} = Q_2 \tag{43}$$

Onde Q_2 é o torque produzido pela roda direita, logo temos:

$$\frac{m}{2}\left(lr\left(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right) + \left(\frac{r^2}{2}\right)\left(\ddot{\psi}_d + \ddot{\psi}_e\right) + 2\left(\frac{lr}{d}\right)^2\left(\sin^2\theta\left(\ddot{\psi}_d - \ddot{\psi}_e\right) + \dot{\theta}\sin 2\theta\left(\dot{\psi}_d - \dot{\psi}_e\right)\right)\right) + \left(\frac{r}{d}\right)^2\left(\left(I_r - I_\varphi\right)\left(\dot{\theta}\sin 2\theta\right)\left(\dot{\psi}_d - \dot{\psi}_e\right)\right) + \left(\frac{r}{d}\right)^2\left(I_r\sin^2\theta + I_\varphi\cos^2\theta\right)\left(\ddot{\psi}_d - \ddot{\psi}_e\right) = \tau_d \tag{44}$$

Finalmente a terceira equação da dinâmica é dada pela expressão 44

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} = Q_3 \tag{45}$$

Onde Q_3 é a soma dos torques produzidos pelas rodas direita e esquerda, então:

$$\begin{aligned} I_\theta \ddot{\theta} + \left(\frac{mlr}{2d}\right)\left(\dot{\theta} d \sin\theta\left(\dot{\psi}_d + \dot{\psi}_e\right) - \frac{lr}{2d}\sin\theta\left(\dot{\psi}_d - \dot{\psi}_e\right)^2\right) \\ + \left(\frac{m}{2}\right)\left(2l^2\ddot{\theta} + (lr)\left(\cos\theta\left(\ddot{\psi}_d + \ddot{\psi}_e\right) - \dot{\theta}\sin\theta\left(\dot{\psi}_d + \dot{\psi}_e\right)\right)\right) \\ - \left(\frac{r}{d\sqrt{2}}\right)^2\left((I_r - I_\varphi)(\sin 2\theta)\left(\dot{\psi}_d + \dot{\psi}_e\right)^2\right) - mgl\sin\theta = -(\tau_d + \tau_e) \end{aligned} \tag{46}$$

Resumindo, as equações de movimento que representam o sistema do pêndulo invertido sobre duas rodas são:

$$\begin{aligned} -\frac{m}{2}\left(lr\left(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta\right) - \left(\frac{r^2}{2}\right)\left(\ddot{\psi}_d + \ddot{\psi}_e\right) + 2\left(\frac{lr}{d}\right)^2\left(\sin^2\theta\left(\ddot{\psi}_d - \ddot{\psi}_e\right) + \dot{\theta}\sin 2\theta\left(\dot{\psi}_d - \dot{\psi}_e\right)\right)\right) \\ - \left(\frac{r}{d}\right)^2\left((I_r - I_\varphi)(\dot{\theta}\sin 2\theta)\left(\dot{\psi}_d - \dot{\psi}_e\right)\right) - \left(\frac{r}{d}\right)^2\left(I_r\sin^2\theta + I_\varphi\cos^2\theta\right)\left(\ddot{\psi}_d - \ddot{\psi}_e\right) = \tau_e \end{aligned} \tag{47}$$

$$\begin{aligned} \frac{m}{2}\left(lr\left(\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta\right) + \left(\frac{r^2}{2}\right)\left(\ddot{\psi}_d + \ddot{\psi}_e\right) + 2\left(\frac{lr}{d}\right)^2\left(\sin^2\theta\left(\ddot{\psi}_d - \ddot{\psi}_e\right) + \dot{\theta}\sin 2\theta\left(\dot{\psi}_d - \dot{\psi}_e\right)\right)\right) \\ + \left(\frac{r}{d}\right)^2\left((I_r - I_\varphi)(\dot{\theta}\sin 2\theta)\left(\dot{\psi}_d - \dot{\psi}_e\right)\right) + \left(\frac{r}{d}\right)^2\left(I_r\sin^2\theta + I_\varphi\cos^2\theta\right)\left(\ddot{\psi}_d - \ddot{\psi}_e\right) = \tau_d \end{aligned} \tag{48}$$

$$\begin{aligned} I_\theta \ddot{\theta} + \left(\frac{mlr}{2d}\right)\left(\dot{\theta} d \sin\theta\left(\dot{\psi}_d + \dot{\psi}_e\right) - \frac{lr}{2d}\sin\theta\left(\dot{\psi}_d - \dot{\psi}_e\right)^2\right) \\ + \left(\frac{m}{2}\right)\left(2l^2\ddot{\theta} + (lr)\left(\cos\theta\left(\ddot{\psi}_d + \ddot{\psi}_e\right) - \dot{\theta}\sin\theta\left(\dot{\psi}_d + \dot{\psi}_e\right)\right)\right) \\ - \left(\frac{r}{d\sqrt{2}}\right)^2\left((I_r - I_\varphi)(\sin 2\theta)\left(\dot{\psi}_d + \dot{\psi}_e\right)^2\right) - mgl\sin\theta = -(\tau_d + \tau_e) \end{aligned} \tag{49}$$

3.2 Linearização Jacobiana

O processo de linearização, assim como já discutido na subseção 2.1, deve ser aplicado a sistemas não-lineares para que seja possível aplicar e calibrar o controle

linear. Este processo foi realizado em torno do ponto de operação que, para o sistema do TWIP, é $\theta = 0°$, ou seja, quando o ângulo de inclinação do pêndulo é nulo.

As demais variáveis do sistema possuem infinitos outros pontos de operação e equilíbrio, mas por conveniência o ponto de linearização escolhido foi a origem, onde todas as variáveis estão zeradas (x_0)

Considerando o sistema não-linear 50:

$$\dot{x}(t) = Ax(t) + Bu \tag{50}$$

Onde $\boldsymbol{A}$ é a matiz linearizada da dinâmica, $\boldsymbol{x}$ é o vetor que possui seis coordenadas, sendo elas as posições e velocidade das rodas esquerda e direita ($\psi_e, \dot{\psi}_e, \psi_d$ e $\dot{\psi}_d$) e posição e velocidade angular do pêndulo em relação a vertical (θ e $\dot{\theta}$), $\boldsymbol{B}$ é a matriz dos coeficientes dos torques aplicados nas rodas esquerda e direita e $\boldsymbol{u}$ são os torques aplicados nas rodas esquerda e direita.

A matriz linearizada $\boldsymbol{A}$ (2) e $\boldsymbol{B}$ (4), por questões estéticas, não foram incluída neste documento na sua forma literal mas podem ser encontradas no código do Apêndice A pelas variáveis simbólicas *jac* e *Bx*.

A substituição dos valores de massa, gravidade, comprimento da haste do pêndulo, tensor de inércia, diâmetro da roda e gravidade, resultam em:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -824.37 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -824.37 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 288.53 & 0 \end{bmatrix} \tag{51}$$

(52)

$$B = \begin{bmatrix} 0 & 0 \\ 4023 & 12207 \\ 0 & 0 \\ 12207 & 4023 \\ 0 & 0 \\ -2269 & -2269 \end{bmatrix}$$

Importante ressaltar que a linearização só deve ser considerada nas vizinhanças do ponto de operação, ou seja, os ganhos que foram obtidos na subseção seguinte 3.3 só são otimizados na região em torno de $\theta = 0°$.

3.3 Regulador Linear Quadrático

O regulador linear quadrático é uma das técnicas de controle escolhidas para controlar o sistema. O algoritimo de obtenção da matriz de ganho, já descrito na subseção 2.2, parte da determinação da matriz de peso dos estados $\boldsymbol{Q}$ e da matriz de atuação $\boldsymbol{R}$.

O processo de determinação destas matrizes é empírico, contudo quanto maior o peso escolhido para uma variável em $\boldsymbol{Q}$, mais enérgico será o controle na correção do sistema para a variável escolhida. A mesma lógica se aplica a matriz $\boldsymbol{R}$ de atuações.

Os valores escolhidos para $\boldsymbol{Q}$ e $\boldsymbol{R}$ foram:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{53}$$

$$R = 1 \tag{54}$$

A escolha dos valores foram feitas com o objetivo de dar a mesma importância para todas as variáveis.

O controle LQR foi implementado no *software* comercial MATLAB como mostrado no Apêndice A por meio de "*KLQR=lqr(A,B,Q,R)"*, onde obteve-se:

$$K_{LQR} = \begin{bmatrix} -1 & -1{,}1320 & 0 & -0{,}1319 & -62{,}0664 & -8{,}1699 \\ 0 & -0{,}1319 & -1 & -1{,}1320 & -62{,}0664 & -8{,}1699 \end{bmatrix} \quad (55)$$

Importante frisar que os ganhos obtidos são otimizados apenas na vizinhança da linearização

3.4 PID

Na subseção 2.3 foram discutidos os métodos de obtenção de ganhos para o controle PID. Embora a técnica desenvolvida por Ziegler-Nichols *et al.* (1942) ser amplamente utilizada na indústria, sua aplicabilidade é limitada a sistemas estáveis cuja posição inicial é retomada com o tempo. A título de exemplificação, podemos aplicar o método de Ziegler-Nichols em um sistema real de pêndulo simples, cuja posição de estabilidade sempre é retomada após a introdução de uma perturbação.

Como o método não pode ser aplicado, os ganhos do PID foram obtidos por tentativa e erro. Os valores do ganhos para θ foram:

- Proporcional: $K_p = 5$;
- Integral: $T_i = 0$;
- Derivativo:$T_d = 1$.

Como o ganho integral foi escolhido nulo, podemos chamar este controle de PD.

3.5 Modelagem do Sistema Experimental

Com o objetivo de aplicar a modelagem matemática já descrita neste capítulo, foi feita a prototipagem do pêndulo invertido sobre rodas. O software comercial escolhido para a modelagem 3D foi o *SOLIDWORKS* devido a sua versatilidade de implementação de sistemas mecânicos.

O modelo 3D desenvolvido, que foi base para a construção do modelo real, é mostrado na Figura 8:

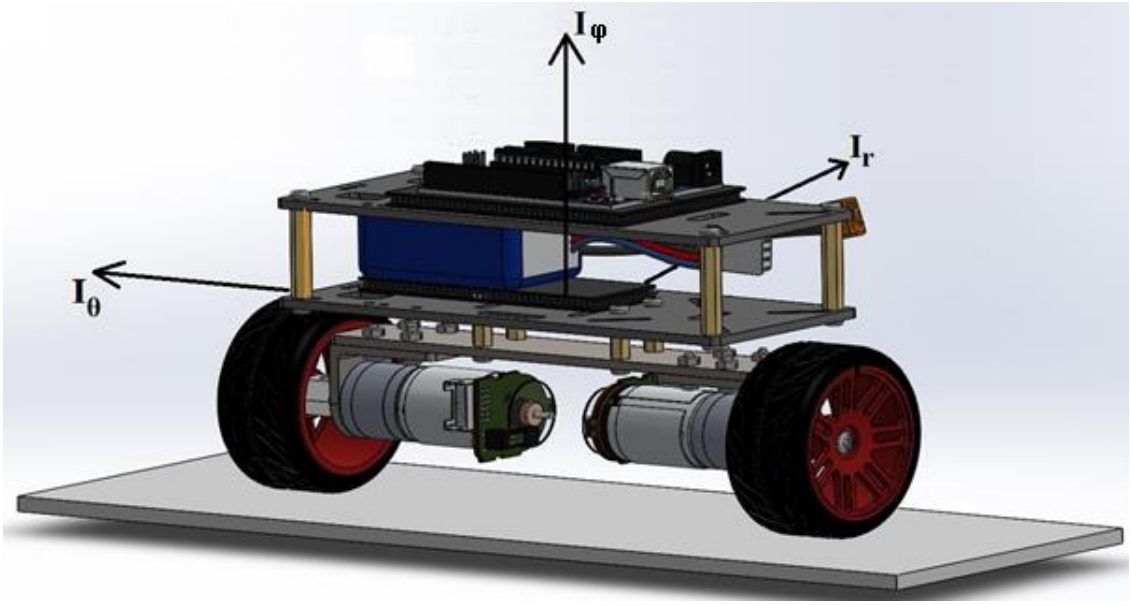

Figura 8 – Modelagem 3D do pêndulo invertido sobre duas rodas

O desenvolvido do protótipo físico do robô autoequilibrante foi realizado no laboratório de robótica da PUC RIO com auxílio do professor orientador. O robô é mostrado na Figura 9.

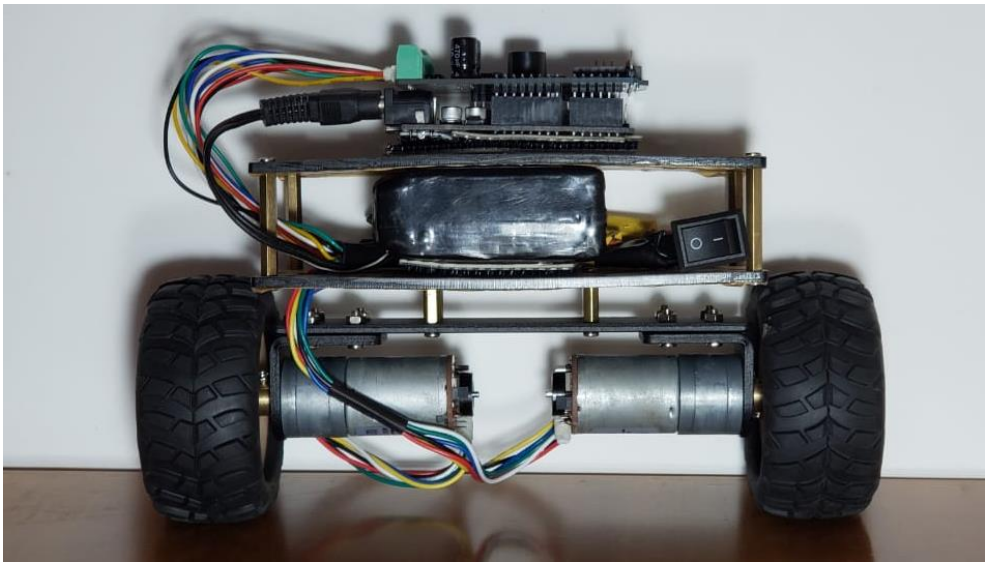

Figura 9 – Modelo real do pêndulo invertido sobre duas rodas

A lista de componentes utilizados para a o desenvolvimento deste protótipo físico estão descritos na Tabela 3:

Tabela 3 - Componentes utilizados

#	Descrição	Quantidade
1	Arduino Uno 7-12V	1
2	Motores JGA25-371 DC gearmotor com encoder	2
3	Chapa separadora de material plástico	2
4	Separadores de aço	4
5	Roda de borracha	2
6	L298P Motor Drive Controller Expansion Board 7~18V	1
7	Bateria Maxamps LiPo 3S 1800mAh	1
8	Interruptor mecânico	1

Dentre os componentes selecionados na Tabela 3, os que merecem atenção especial são os itens 1, 2 e 7.

1. Arduino Uno: A plataforma embarcada escolhida para interagir com o ambiente externo por meio de *software* e *hardware.* Tensão operacional de 5V e tensão de alimentação recomendada de 7-12V, possui 6 pinos analógicos de entrada, 14 pinos pinos de entrada e saída digitais e conexão USB como mostrado na Figura 10. Este microcontrolador foi escolhido pela facilidade de reprodução do experimento e baixo custo;

Figura 10 – Arduino Uno

2. A placa controladora de motores foi utilizada pela facilidade dos terminais dos motores e por possuir um giroscópio integrado;

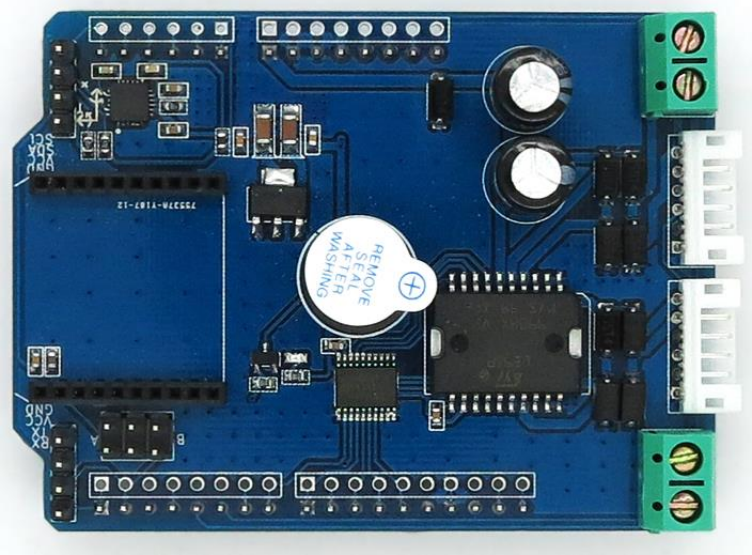

Figura 11 – Shield para arduino com controlador de motor integrado

3. A bateria especificada foi a tipo LiPo 3S 1800mAh para tempo de operação mínimo de 4,5h.

Os códigos para o protótipo físico foram desenvolvidos em linguagem C/C++ e encontram-se nos Apêndices D, E, F, G e H para consulta.

4 Resultados

Com a obtenção das acelerações e da linearização do sistema descrito pelas equações 47, 48 e 49, foi possível implementar o código mostrado nos Apêndices B e C em ambiente *MATLAB* para realizar a simulação do comportamento do sistema para diversas situações.

Todas as simulações foram realizadas utilizando a dinâmica não-linear uma vez que os ganhos para o controle LQR só são otimizados para condições de operação próximas às condições de linearização, por conseguinte a inclinação $\boldsymbol{\theta}$ não foi demasiadamente grande.

Nesta seção foram simulados 6 casos conforme mostrado na Tabela 4:

Tabela 4 – Casos analisados

# Caso	$\boldsymbol{\theta_0}$	$\dot{\boldsymbol{\varphi}}_{\boldsymbol{r}}$
1	$0{,}2$	0
2	$0{,}2$	$\pi/10$
3	$0{,}2$	$7\sin(\pi t)$
4	$0{,}4$	0
5	$0{,}4$	$\pi/10$
6	$0{,}4$	$7\sin(\pi t)$

4.1 Comportamento do Sistema PD

Nesta subseção vamos discutir os seis casos simulados utilizando os ganhos PD obtidos empiricamente. Para todos os casos o sistema simulado por $5s$.

As estimativas das velocidades de rotação (*yaw)*, translação longitudinal da base, e posição no espaço são dadas por:

$$V_{tl} = \frac{r}{2}(\dot{\psi}_e + \dot{\psi}_d),\ \dot{\varphi} = \frac{r}{d}(\dot{\psi}_d - \dot{\psi}_e) \tag{55}$$

$$x_b = V_{tl} \cos \varphi \, , x_b = V_{tl} \sin \varphi \tag{56}$$

1. Caso 1

No primeiro caso, temos o sistema foi simulado utilizando $\theta_0 = 0{,}2$, $\dot{\varphi}_r = 0/s$ e $\theta_r = 0°$. O comportamento resultante do controle PD utilizado pode ser verificado na Figura 12 e Figura 13.

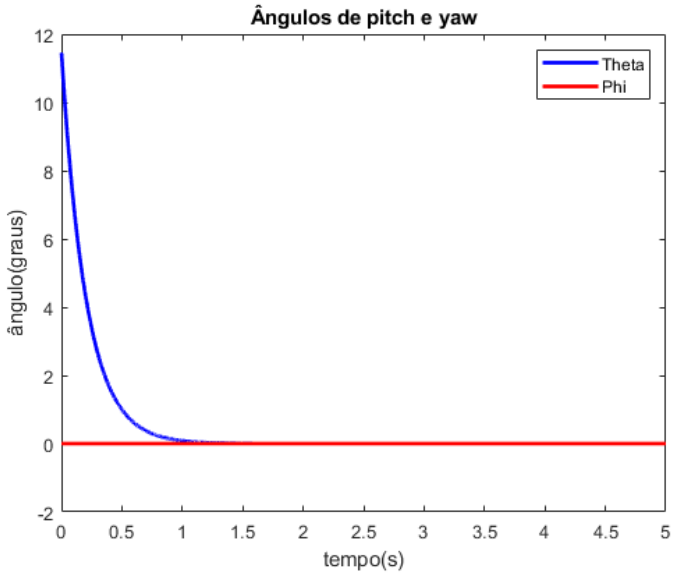

Figura 12 – Comportamento das variáveis θ e φ – PD Caso 1

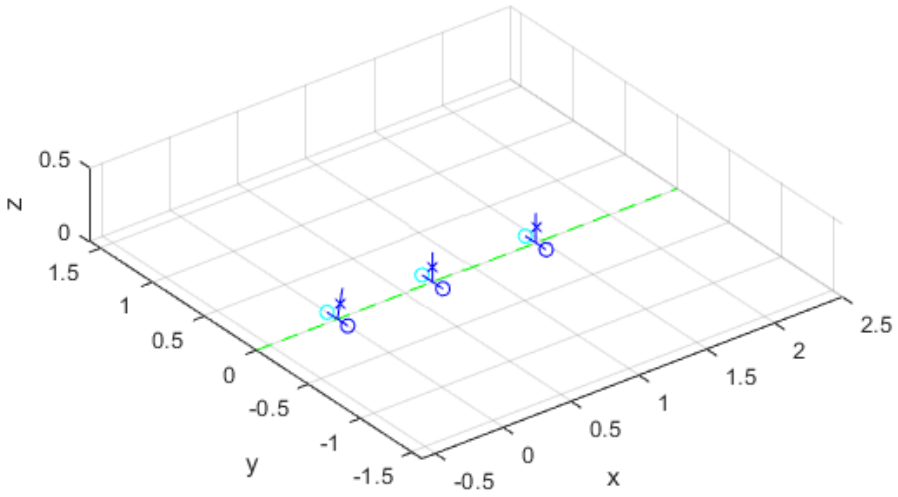

Figura 13 – Trajetória executada – PD Caso 1

O sistema governado pelo controle PD obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 1{,}716°$ e para a variável φ obteve erro RMS de $\varepsilon_{\varphi RMS} = 0°$ pois a trajetória retilínea foi mantida. O tempo de estabilização do sistema para $\theta < 0{,}50^{o}$ foi de $t_{estabilização} = 0{,}645s$.

2. Caso 2

No segundo caso, temos o sistema foi simulado utilizando $\theta_0 = 0{,}2$, $\dot{\varphi}_r = {}^{\pi}/_{10}\,/s$ e $\theta_r = 0^{o}$. O comportamento resultante do controle PD utilizado pode ser verificado na Figura 14 e Figura 15.

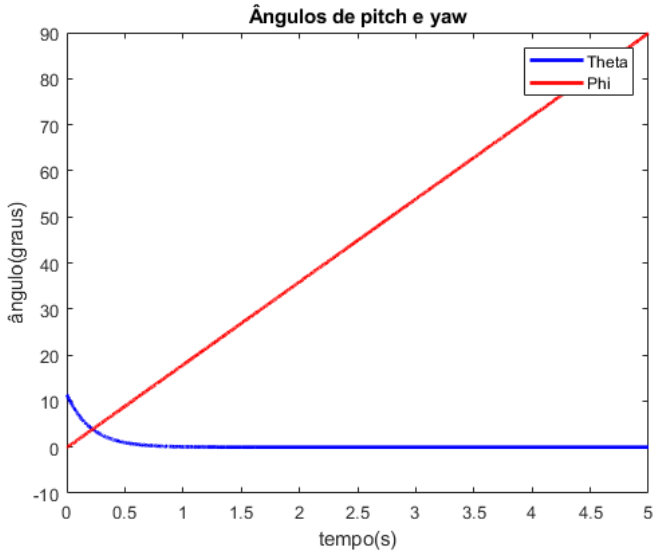

Figura 14 – Comportamento das variáveis θ e φ – PD Caso 2

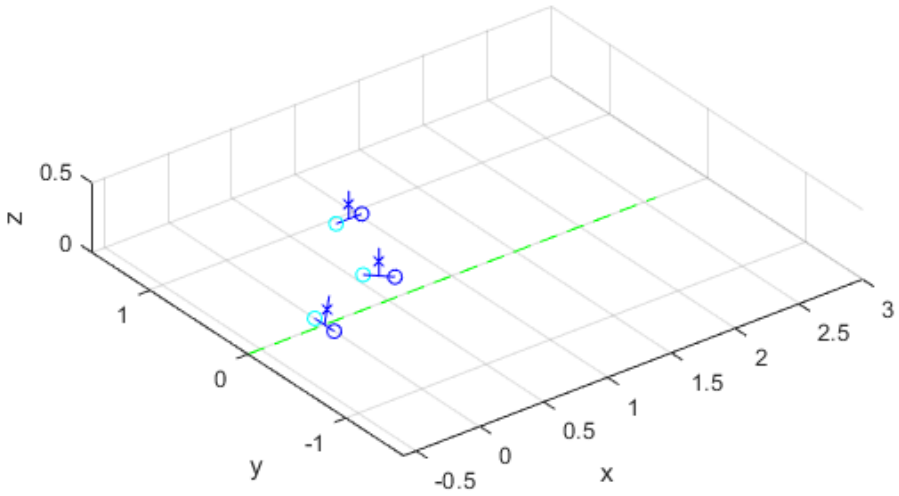

Figura 15 – Trajetória executada – PD Caso 2

O sistema governado pelo controle PD obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 1{,}716°$ e para a variável φ obteve erro RMS de $\varepsilon_{\varphi RMS} = 52{,}11^{o}$. O tempo de estabilização do sistema para $\theta < 0{,}50^{o}$ foi de $t_{estabilização} = 0{,}645s$.

3. Caso 3

Para o terceiro caso , temos o sistema simulado utilizando $\theta_0 = 0{,}2$, $\dot{\varphi}_r = 7\,\mathrm{sin}(\pi t)\ /s$ $e\ \theta_r = 0\,^{\circ}$. O comportamento resultante do controle PD utilizado pode ser verificado na Figura 16 e Figura 17, onde nesta a numeração representa o desenvolvimento da trajetória executada.

O sistema governado pelo controle PD obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 1{,}708°$ e para a variável φ obteve erro RMS de $\varepsilon_{\varphi RMS} = 299{,}26°$. O tempo de estabilização do sistema para $\theta < 0{,}50\,^{\circ}$ foi de $t_{estabilização} = 0{,}648s$.

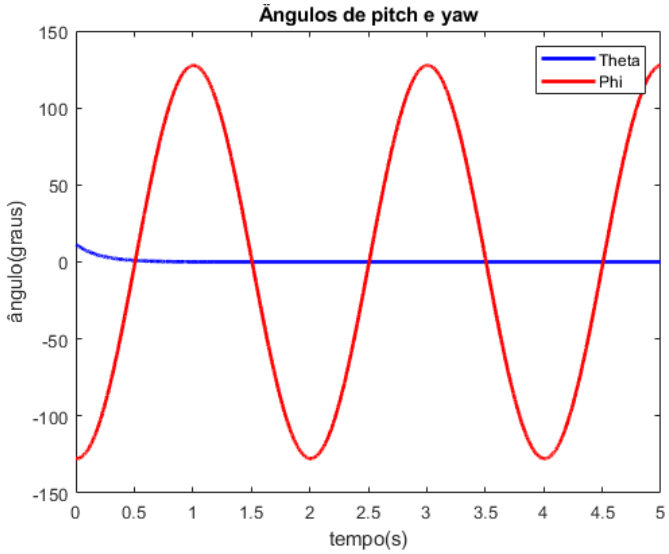

Figura 16 – Comportamento das variáveis θ e φ – PD Caso 3

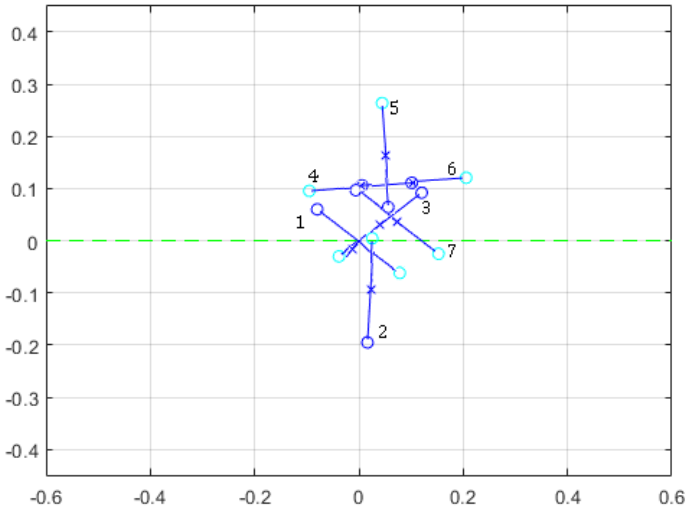

Figura 17 – Trajetória executada – PD Caso 3

4. Caso 4

Neste caso, temos o sistema simulado utilizando $\theta_0 = 0{,}4$, $\dot{\varphi}_r = 0\ /s$ e $\theta_r = 0\,^{\circ}$. O comportamento resultante do controle PD utilizado pode ser verificado na Figura 18 e Figura 19.

O sistema governado pelo controle PD obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 3{,}233°$ e para a variável φ o erro RMS foi de $\varepsilon_{\varphi RMS} = 0°$ pois a trajetória retilínea foi mantida. O tempo de estabilização do sistema $\theta < 0{,}50\,^{\circ}$ foi de $t_{estabilização} = 0{,}762s$.

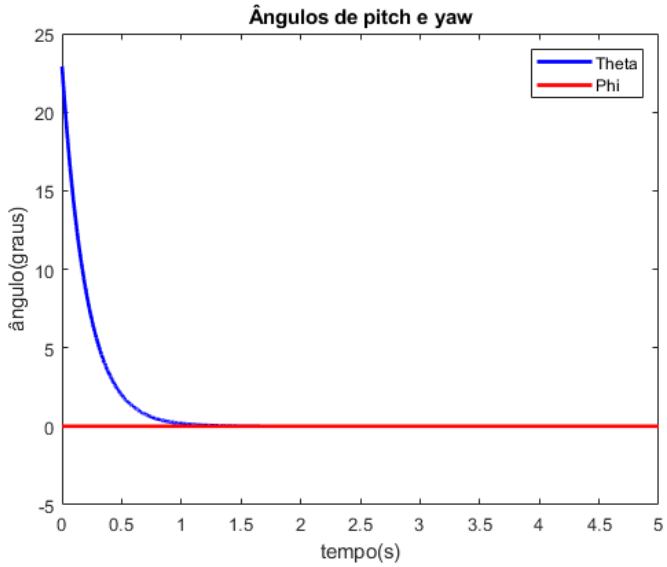

Figura 18 – Comportamento das variáveis θ e φ – PD Caso 4

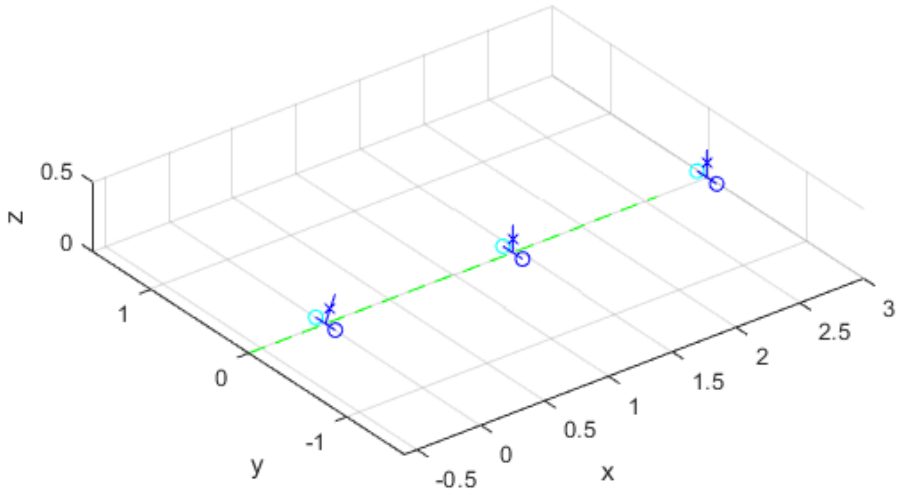

Figura 19 – Trajetória executada – PD Caso 4

5. Caso 5

Para o quinto, temos o sistema simulado utilizando $\theta_0 = 0{,}4$, $\dot{\varphi}_r = {}^{\pi}/_{10}\,/s$ e $\theta_r = 0°$. O comportamento resultante do controle PD utilizado pode ser verificado na Figura 20 e Figura 21.

O sistema governado pelo controle PD obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 3{,}233°$ e para a variável φ o erro RMS foi de $\varepsilon_{\varphi RMS} = 52{,}00°$. O tempo de estabilização do sistema para $\theta < 0{,}50°$ foi de $t_{estabilização} = 0{,}762s$.

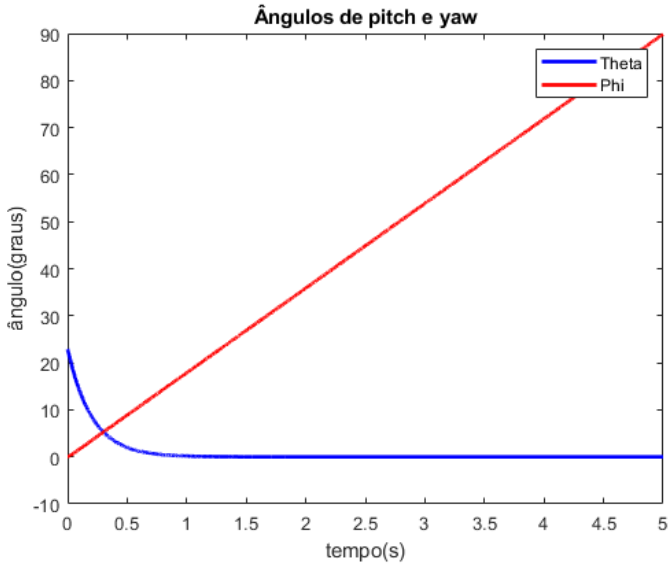

Figura 20 – Comportamento das variáveis θ e φ – PD Caso 5

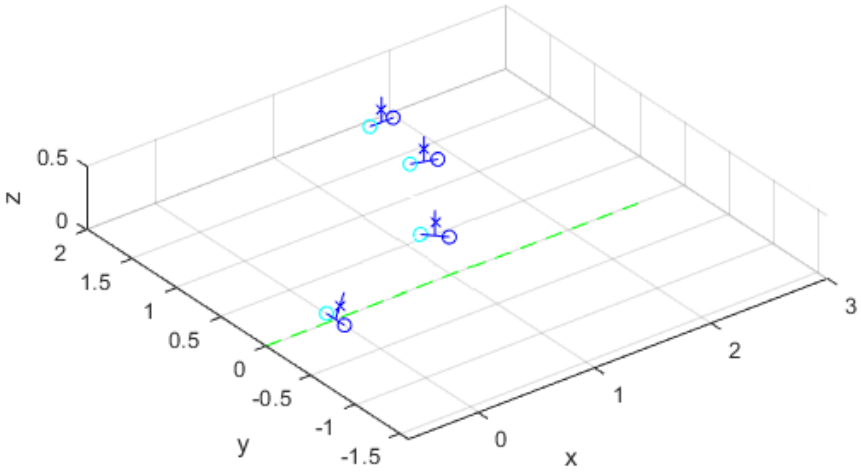

Figura 21 – Trajetória executada – PD Caso 5

6. Caso 6

Para o sexto, temos o sistema simulado utilizando $\theta_0 = 0{,}4$, $\dot{\varphi}_r = 7\,\mathrm{sin}(\pi t)\ /s$ $e\ \theta_r = 0\,°$. O comportamento resultante do controle PD utilizado pode ser verificado na Figura 22 e Figura 23, onde nesta a numeração representa o desenvolvimento da trajetória executada.

O sistema governado pelo controle PD obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 3{,}213°$ e para a variável φ o erro RMS foi de $\varepsilon_{\varphi RMS} = 299{,}45°$. O tempo de estabilização do sistema para $\theta < 0{,}50\,°$ foi de $t_{estabilização} = 0{,}765s$.

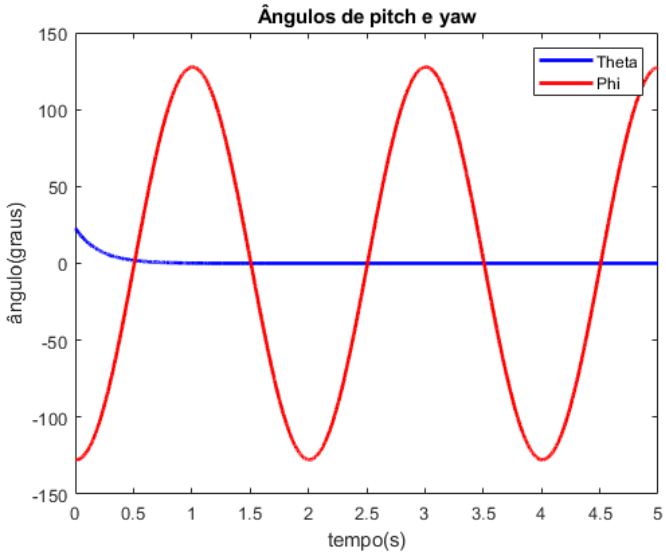

Figura 22 – Comportamento das variáveis θ e φ – PD Caso 6

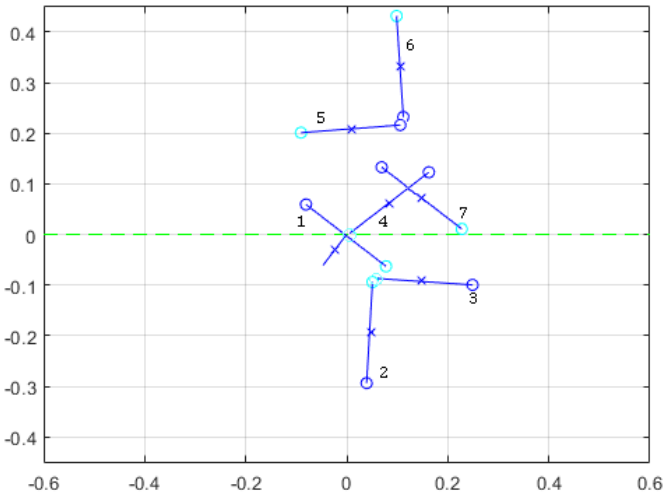

Figura 23 – Trajetória executada – PD Caso 6

4.2 Comportamento do Sistema LQR

Assim como na subseção anterior, aqui são apresentados os resultados simulados.

1. Caso 1

No primeiro caso, temos o sistema foi simulado utilizando $\theta_0 = 0{,}2$, $\dot{\varphi}_r = 0\,°/s$ e $\theta_r = 0\,°$. O comportamento resultante do controle LQR utilizado pode ser verificado na Figura 24 e Figura 25.

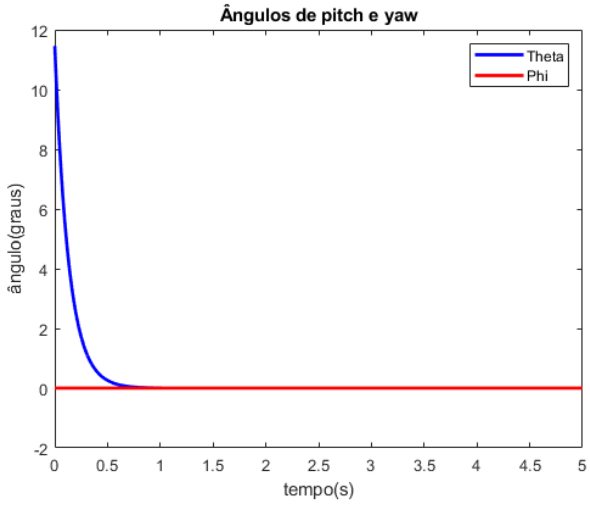

Figura 24 – Comportamento das variáveis θ e φ – LQR Caso 1

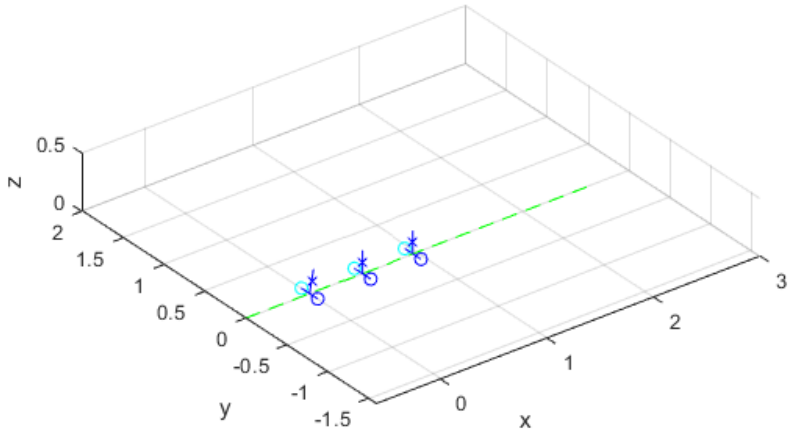

Figura 25 – Trajetória executada – LQR Caso 1

O sistema governado pelo controle LQR obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 1{,}296°$ e para a variável φ o erro RMS foi de $\varepsilon_{\varphi RMS} = 0°$

pois a trajetória retilínea foi mantida. O tempo de estabilização do sistema para $\theta < 0{,}50^{o}$ foi de $t_{estabilização} = 0{,}407s$.

2. Caso 2

No segundo caso, temos o sistema foi simulado utilizando $\theta_0 = 0{,}2$, $\dot{\varphi}_r = {}^{\pi}/_{10}\,/s$ e $\theta_r = 0^{o}$. O comportamento resultante do controle LQR utilizado pode ser verificado na Figura 26 e Figura 27.

O sistema governado pelo controle LQR obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 1{,}296°$ e para a variável φ o erro RMS foi de $\varepsilon_{\varphi RMS} = 51{,}88°$ pois a trajetória retilínea foi mantida. O tempo de estabilização do sistema para θ<0,50° foi de $t_{estabilização} = 0{,}407s$.

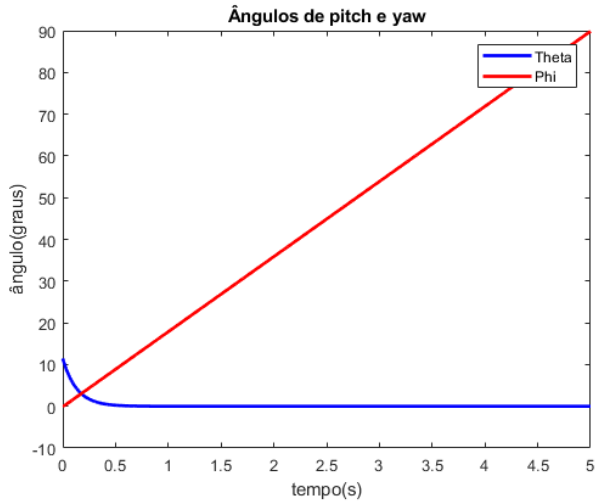

Figura 26 – Comportamento das variáveis θ e φ – LQR Caso 2

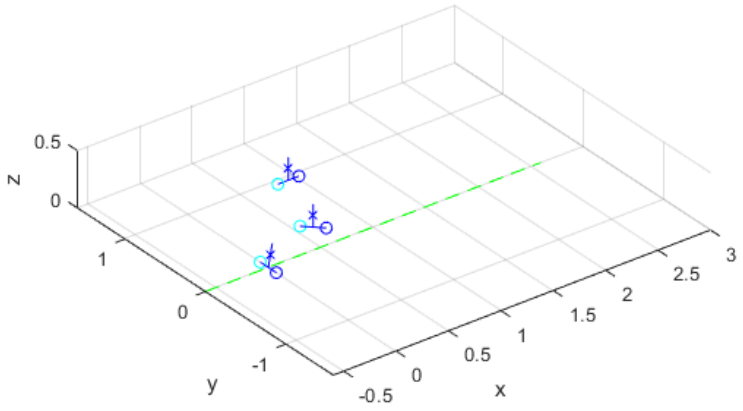

Figura 27 – Trajetória executada – LQR Caso 2

3. Caso 3

Para o terceiro caso , temos o sistema simulado utilizando $\theta_0 = 0{,}2$, $\dot{\varphi}_r = 7\sin(\pi t)\ /s$ e $\theta_r = 0\,°$. O comportamento resultante do controle LQR utilizado pode ser verificado na Figura 28 e Figura 29, onde nesta a numeração representa o desenvolvimento da trajetória executada.

O sistema governado pelo controle LQR obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 1{,}295°$ e para a variável φ obteve erro RMS de $\varepsilon_{\varphi RMS} = 299{,}56°$. O tempo de estabilização do sistema para $\theta < 0{,}50\,°$ foi de $t_{estabilização} = 0{,}407s$.

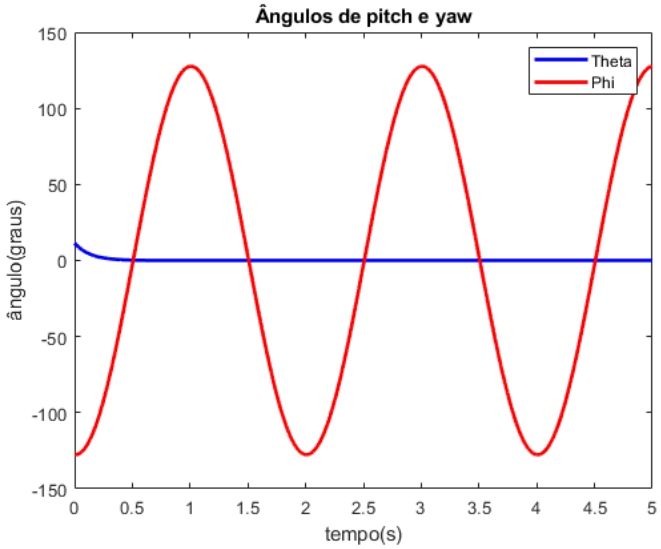

Figura 28 – Comportamento das variáveis θ e φ – LQR Caso 3

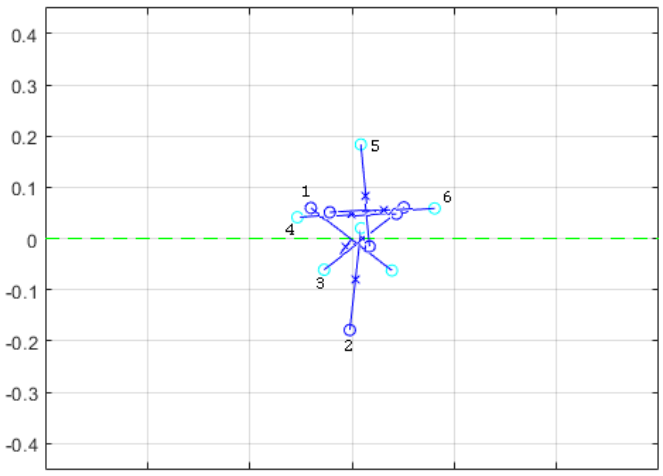

Figura 29 – Trajetória executada – LQR Caso 3

4. Caso 4

Neste caso, temos o sistema simulado utilizando $\theta_0 = 0{,}4$, $\dot{\varphi}_r = 0\,^{o}/s$ e $\theta_r = 0\,^{o}$. O comportamento resultante do controle LQR utilizado pode ser verificado na Figura 30 e Figura 31.

O sistema governado pelo controle LQR obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 2{,}408°$ e para a variável φ obteve erro RMS de $\varepsilon_{\varphi RMS} = 0°$ pois a trajetória retilínea foi mantida. O tempo de estabilização do sistema para $\theta \leq 0{,}50\,^{o}$ foi de $t_{estabilização} = 0{,}401s$

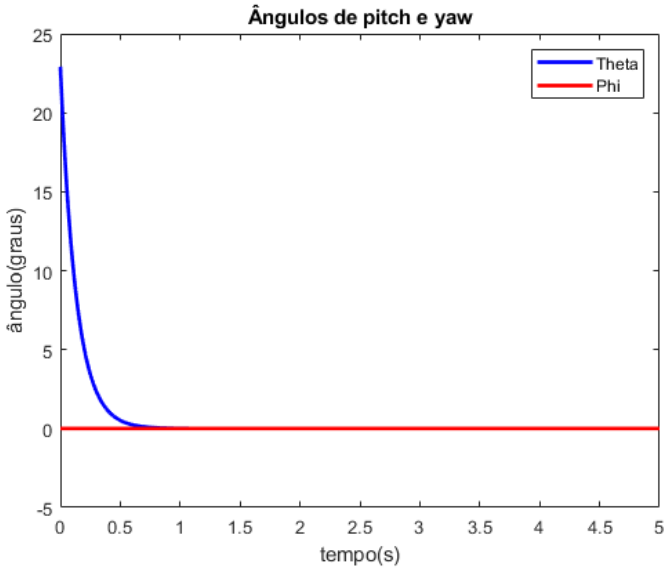

Figura 30 – Comportamento das variáveis θ e φ – LQR Caso 4

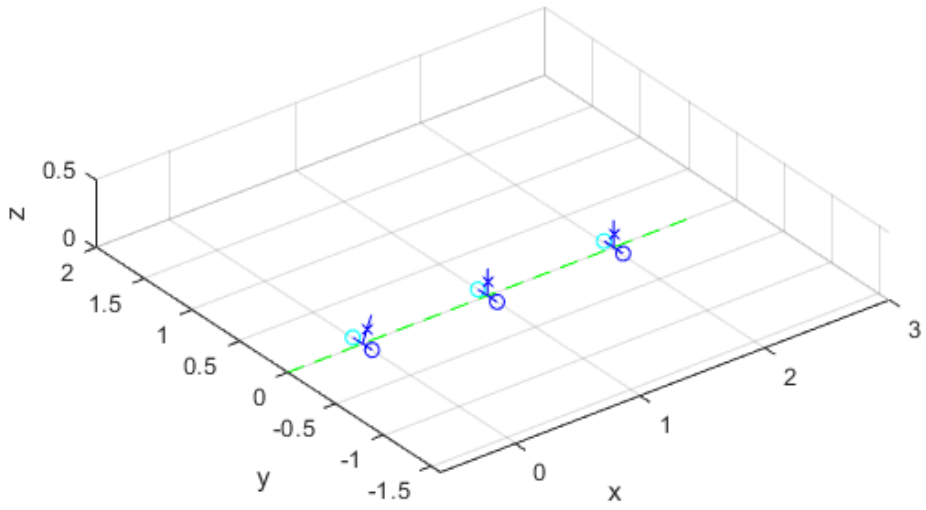

Figura 31 – Trajetória executada – LQR Caso 4

5. Caso 5

Para o quinto caso, temos o sistema simulado utilizando $\theta_0 = 0{,}4$, $\dot{\varphi}_r = {}^{\pi}/_{10}\,/s$ e $\theta_r = 0°$. O comportamento resultante do controle LQR utilizado pode ser verificado na Figura 32 e Figura 33.

O sistema governado pelo controle LQR obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 2{,}408°$ e para a variável φ o erro RMS foi de $\varepsilon_{\varphi RMS} = 51{,}99°$. O tempo de estabilização do sistema para $\theta < 0{,}50°$ foi de $t_{estabilização} = 0{,}481s$.

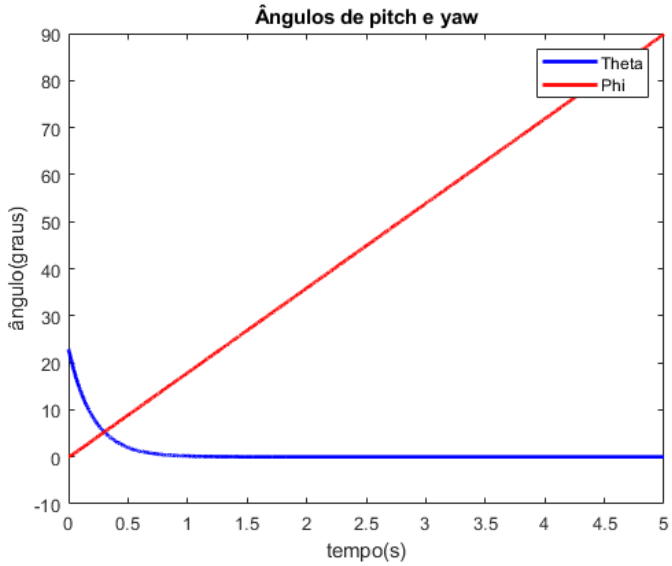

Figura 32 – Comportamento das variáveis θ e φ – LQR Caso 5

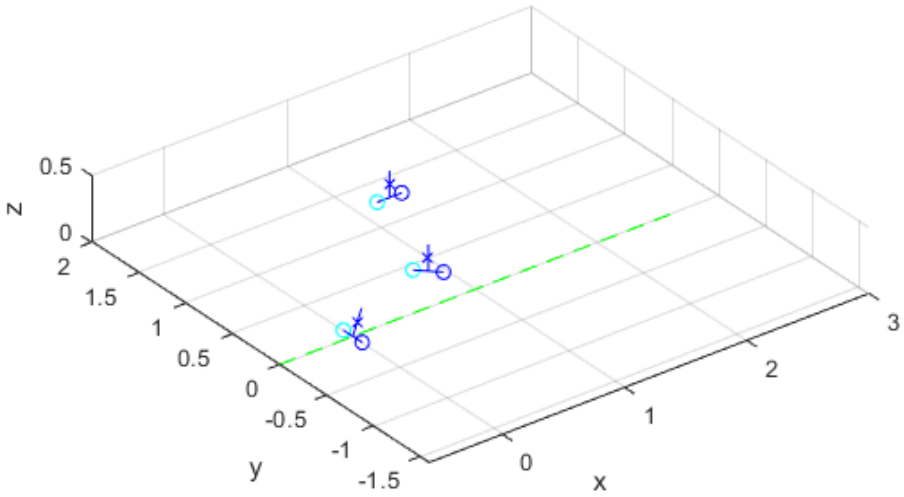

Figura 33 – Trajetória executada – LQR Caso 5

6. Caso 6

Para o sexto caso, temos o sistema simulado utilizando $\theta_0 = 0{,}4$, $\dot{\varphi}_r = 7\,\mathrm{sin}(\pi t)\ /s\ \ e\ \theta_r = 0\,°$. O comportamento resultante do controle LQR utilizado pode ser verificado na Figura 34 e Figura 35, onde nesta a numeração representa o desenvolvimento da trajetória executada.

O sistema governado pelo controle LQR obteve, para a variável θ, erro RMS associado de $\varepsilon_{\theta RMS} = 2{,}405°$ e para a variável φ o erro RMS foi de $\varepsilon_{\varphi RMS} = 299{,}83°$. O tempo de estabilização do sistema para $\theta < 0{,}50\,°$ foi de $t_{estabilização} = 0{,}481s$.

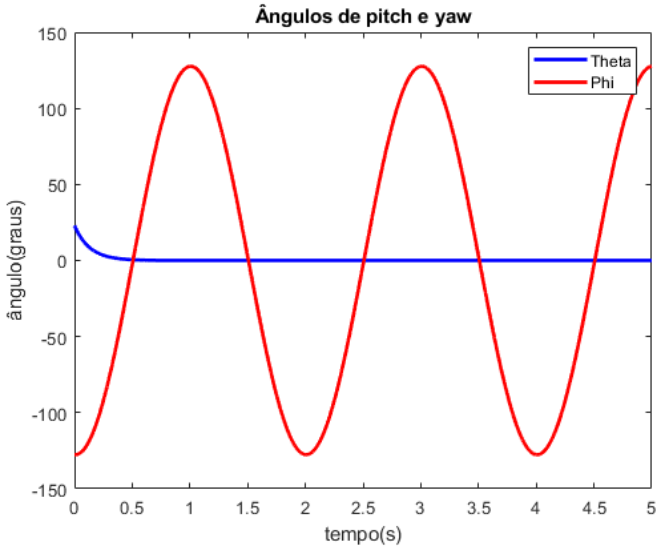

Figura 34 – Comportamento das variáveis θ e φ – LQR Caso 6

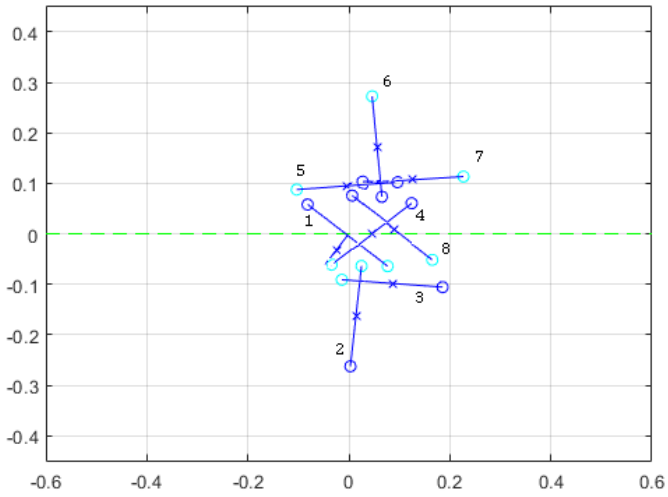

Figura 35 – Trajetória executada – LQR Caso 6

4.3 Análise dos Resultados

Esta subseção é dedicada para a análise dos resultados comparativos dos sistemas apresentados em 4.1 e 4.2 e para o detalhamento dos problemas enfrentados com o controle.

Inicialmente, o controle desenvolvido tinha o objetivo de controlar as variáveis $\theta, \dot{\theta}, \psi_e, \dot{\psi}_e, \psi_d$ e $\dot{\psi}_d$, contudo as simulações que fizeram o uso dos ganhos para $\psi_e, \dot{\psi}_e, \psi_d$ e $\dot{\psi}_d$ se tornam instáveis. Algumas medidas, como a variação dos valores dos pesos nas matrizes de peso $\boldsymbol{Q}$ e $\boldsymbol{R}$, se mostraram ineficazes, logo não foi possível controlar a trajetória utilizando os ganhos obtidos pelo LQR, mas sim ganhos obtidos por tentativa e erro.

Embora o controle de $\psi_e, \dot{\psi}_e, \psi_d$ e $\dot{\psi}_d$ tenha sido executado via LQR o trabalho, que tinha por principal objetivo controlar a inclinação do pêndulo, foi bem sucedido ao empregar os ganhos para as variáveis θ e $\dot{\theta}$.

A Tabela 5 compila as informações obtidas das simulações e expõe uma análise breve dos erros e tempos de correção da inclinação do pêndulo baseado em trajetórias e inclinações iniciais distintas.

Tabela 5 – Análise de performance

# Caso	$\varepsilon_{\theta RMS}$	$\varepsilon_{\varphi RMS}$	$t_{estabilização}(s)$
PD – 1	1,716°	0°	0,645
LQR – 1	1,296°	0°	0,407
PD – 2	1,716°	52,11°	0,645
LQR – 2	1,296°	51,88°	0,407
PD – 3	1,708°	299,26°	0,648
LQR – 3	1,295°	299,56°	0,407
PD – 4	3,233°	0°	0,762
LQR – 4	2,408°	0°	0,401
PD – 5	3,233°	52,00°	0,762
LQR – 5	2,408	51,99°	0,481
PD – 6	3,213	299,45°	0,765
LQR – 6	2,405	299,83°	0,481

Pela Tabela 5 é possível observar que para trajetórias constantes, a correção do ângulo θ não é afetada, ao passo que, para trajetórias variáveis e com curvas mais acentuadas, a correção do ângulo θ pode ser sutilmente favorecida. O beneficiamento sobre o erro médio quadrático para a trajetória variável foi calculado conforme a Tabela 6.

Tabela 6 – Befecifiamento de correção por trajetória

# Caso	Beneficiamento (%)
PD 1 – 3	0,47%
PD 2 – 3	0,47%
LQR 1 – 3	0,08%
LQR 2 – 3	0,08%
PD 4 – 6	0,62%
PD 5 – 6	0,62%
LQR 4 – 6	0,12%
LQR 5 – 6	0,12%

Outro ponto relevante mostrado pelos dados da Tabela 5 é a evidente superioridade do controle LQR frente ao PD, uma vez que tempo de correção da inclinação θ para o caso LQR ocorreu, em média, 37% mais veloz.

5 Conclusões

O estudo apresentado neste trabalho alcançou o objetivo que era desenvolver um modelo 3D da dinâmica de um robô móvel autoequilibrante, admitindo a consideração do acoplamento entre as dinâmicas longitudinal e lateral.

Os controladores propostos obtiveram diferentes performances para diferentes condições iniciais e ficou a clara a superioridade da aplicação do controle LQR frente ao PID. Apesar do melhor desempenho do controle LQR, ambos controladores conseguiram atingir o equilíbrio do pêndulo dentro da marca de $1s$, que para competições de robôs autoequilibras, é uma valor razoável.

A trajetória executada pelo robô foi exerceu papel fundamental nas simulações de equilíbro pois, a introdução de uma rotação constante não influencia em nada o controle ao passo que uma trajetória variável com curvas mais acentuadas gerou um beneficiamento na correção do ângulo θ.

Os trabalhos futuros sugeridos incluem:

1) Levar em consideração as forças de atrito e a massa dos componentes da base e analisar as limitações;

2) Comparar os esforços computacionais e resultados das técnicas de controle não-linear e linear.

Referências Bibliográficas

AMBROSE, O. R.; SAVELY, R. T.; MICHAEL, S.; DIFTLER, M. A.; SPAIN I.; RADFORD, N.; MARTIN, L.; **"Mobile Manipulation using NASA's Robonaut".** IEEE International Conference Robotics and Automation Vol 2, pp. 2104-2109, 2004.

BAGDANOFF, J. L.; Citron, S. J., **"Experiments with an inverted pendulum subject to random parametric excitation".** pp. 447-452, 1965.

BLITZER, L.; **Inverted pendulum.** vol. 33, no. 12, pp. 1076-1078, 1965.

CASTRO, A.; **"Modeling and dynamics analysis of a two-wheeled inverted-pendulum".** A thesis presented to the academic faculty Georgia Institute of Technology, 2012.

CHOQUEHUANCA, C. R. M.; **"Projeto e Controle Robusto de um Transportador Pessoal Robótico Autoequilibrante".** Tese (Mestrado) — PUC-Rio, 2010.

DING, F.; Huang, J.; Wang, Y.; Matsuno, T.; Fukuda, T.; and Sekiyama, K.; **"Modeling and control of a novel narrow vehicle".** International Conference on Robotics and Biomimetics, 2010.

KALMAN, R. E.; **"Contributions to the Theory of Optimal Control".** 1960.

KAWAMURA, T.; Yamafuji, K.; **"Robots driven by parallel bicycles".** International Journal of Mechanics and Control, Vol. 09, No. 02, 2008

KIM, S. B.; Bui, T. H.; Nguyen, T. T.; Chung, T. L.; **"A simple nonlinear control of a two-wheeled welding mobile robot".** International Journal of Control, Automation, and Systems, vol. 1, no. 1, pp. 35-42, 2003

KIM, S. S.; JUNG, S.; **"Control Experiment of a Wheel-Driven Mobile Inverted Pendulum Using Neural Network"**, IEE Transactions on Control System Technology, VOL. 16, NO. 2, 2008.

LOWENSTERN, E. R.; **“Stabilizing effect of imposed oscillations on a dynamical system”.** Philosophical Magazine, vol. 13, pp. 458-486, 1932.

NESS, D. J.; **“Small oscillations of a stabilized, inverted pendulum”.** vol. 35, no. 10, pp. 964-967, 1967.

PORTELLA, K. M.; Moraes, D. D.; Mantovani, Q. L.; Gatelli, M. L. C.; Venturini, M. S.; Bellinaso, L. V.; Paglione, P.; **“Controle por regulador linear quadrático e controle por mapeamento exponencial aplicado a um sistema de pêndulo invertido”.** Universidade Federal de Santa Maria, 2016.

SAHBA, M.; **“Computer-aided design of feedback controllers for nonlinear systems with applications to control of a double-inverted pendulum”.** IEE proceedings. D, Control theory and applications, vol. 130, no. 6, pp. 350-358, 1983.

Segway Robotics, **“PUMA Project”**, Acessado em 01 de dezembro de 2019. Disponível em: https://pt.wikipedia.org/wiki/Personal_Urban_Mobility_and_Accessibility .

STEPHENSON, A.; **“A new type of dynamic stability”.** Proc. of the Manchester Literary and Philosophical Society, 1908.

YAMAKAWA, T.;**“Stabilization of an inverted pendulum by a high-speed fuzzy logic controller hardware system”**,Department of Electrical Engineering and Computer Science, Kumamoto University, Kumamoto 860.

YUTA, S.; Ha, Y. S.; **“Trajectory tracking control for navigation of inverse pendulum type self-contained mobile robot”.** Robotics and Autonomous Systems 17, 1996.

ZIEGLER, J. G.; NICHOLS, N. B.; **“Optimum setting for automatic controllers”**, Trans, ASME 64 1942

Apêndice A – Equações & Ganhos

Este código foi desenvolvido no *software* comercial *MATLAB* e tem por objetivo obter:

1. As três equações lagrangianas, descritas no código como eq1, eq2 e eq3;
2. As equações das acelerações $\ddot{\psi}_e, \ddot{\psi}_d$ e $\ddot{\theta}$, descritas no código como *PSI_PPE, PSI_PPD e THETA_PP*, resolvendo o sistema contendo as equações 46,47 e 48 usando o toolbox simbólico do *MATLAB*. As acelerações foram obtidas em função de $\psi_e, \dot{\psi}_e, \psi_d, \dot{\psi}_d, \theta$ e $\dot{\theta}$ e foram usadas no código do Apêndice C;
3. Linearização Jacobiana da dinâmica;
4. A discretização;
5. Os ganhos LQR.

```
clear
clc

syms  psi_e psi_pe psi_ppe psi_d psi_pd psi_ppd theta theta_p theta_pp
syms  psi_et(t) psi_pet(t) psi_ppet(t) psi_dt(t) psi_pdt(t) psi_ppdt(t)
theta_t(t) theta_pt(t) theta_ppt(t)
qpp=sym([psi_ppe psi_ppd theta_pp]);
qp= sym([psi_pe psi_pd theta_p]);
q= sym([psi_e psi_d theta]);
syms a b c tau_e tau_d r m d g l x1 x2 x3 x4 x5 x6

% A: Descrever os termos T e U

T=(m/2)*(((psi_pe+psi_pd)*(r/2))^2+(psi_pe+psi_pd)*r*l*theta_p*cos(theta)+(
l^2)*(theta_p^2)+((l^2)*(sin(theta))^2)*((psi_pd-
psi_pe)*(r/d))^2)+(b*theta_p^2)/2+(a*(sin(theta))^2+c*(cos(theta))^2)*(((ps
i_pd-psi_pe)*(r/d))^2)/2;
```

```
U=m*g*l*cos(theta);

L=T-U; %lagrangeano
%% B: Aplicar a equação de Lagrange e encontrar os coeficientes.

L1=diff(L,q(1));%derivada do lagrangeano em psie
L2=diff(L,q(2));%derivada do lagrangeano em psid
L3=diff(L,q(3));%derivada do lagrangeano em theta

L21=diff(L,qp(1));%derivada do lagrangeano em psi_pe
L22=diff(L,qp(2));%derivada do lagrangeano em psi_pd
L23=diff(L,qp(3));%derivada do lagrangeano em thetap

%Substituindo variável no tempo
auxL1=subs(L21,[psi_e psi_pe psi_d psi_pd theta theta_p],[psi_et(t)
psi_pet(t) psi_dt(t) psi_pdt(t) theta_t(t) theta_pt(t)]);
auxL2=subs(L22,[psi_e psi_pe psi_d psi_pd theta theta_p],[psi_et(t)
psi_pet(t) psi_dt(t) psi_pdt(t) theta_t(t) theta_pt(t)]);
auxL3=subs(L23,[psi_e psi_pe psi_d psi_pd theta theta_p],[psi_et(t)
psi_pet(t) psi_dt(t) psi_pdt(t) theta_t(t) theta_pt(t)]);

% auxL1=subs(L21,[psi_pe psi_pd theta_p],[psi_pet(t) psi_pdt(t)
theta_pt(t)]);
% auxL2=subs(L22,[psi_pe psi_pd theta_p],[psi_pet(t) psi_pdt(t)
theta_pt(t)]);
% auxL3=subs(L23,[psi_pe psi_pd theta_p],[psi_pet(t) psi_pdt(t)
theta_pt(t)]);



L31=diff(auxL1,t);%derivada do lagrangeano em t
L32=diff(auxL2,t);%derivada do lagrangeano em t
L33=diff(auxL3,t);%derivada do lagrangeano em t

%Resubstituindo variáveis de estado
L41=subs(L31,[diff(psi_et(t),t) diff(psi_pet(t),t) diff(psi_dt(t),t)
diff(psi_pdt(t),t) diff(theta_t(t),t) diff(theta_pt(t),t) psi_et(t)
psi_pet(t) psi_dt(t) psi_pdt(t) theta_t(t) theta_pt(t)],[psi_pe psi_ppe
psi_pd psi_ppd theta_p theta_pp psi_e psi_pe psi_d psi_pd theta theta_p]);
```

```
L42=subs(L32,[diff(psi_et(t),t) diff(psi_pet(t),t) diff(psi_dt(t),t)
diff(psi_pdt(t),t) diff(theta_t(t),t) diff(theta_pt(t),t) psi_et(t)
psi_pet(t) psi_dt(t) psi_pdt(t) theta_t(t) theta_pt(t)],[psi_pe psi_ppe
psi_pd psi_ppd theta_p theta_pp psi_e psi_pe psi_d psi_pd theta theta_p]);
L43=subs(L33,[diff(psi_et(t),t) diff(psi_pet(t),t) diff(psi_dt(t),t)
diff(psi_pdt(t),t) diff(theta_t(t),t) diff(theta_pt(t),t) psi_et(t)
psi_pet(t) psi_dt(t) psi_pdt(t) theta_t(t) theta_pt(t)],[psi_pe psi_ppe
psi_pd psi_ppd theta_p theta_pp psi_e psi_pe psi_d psi_pd theta theta_p]);

eq1=L41-L1-tau_e;
eq2=L42-L2-tau_d;
eq3=L43-L3+tau_e+tau_d;

 %% Isolando

 % Determinação da equação para a aceleração angular Theta
aux1=isolate(eq1==0,qpp(1));%isolado psi_ppe
aux2=isolate(eq2==0,qpp(1));%isolado psi_ppe
aux22=isolate(eq3==0,qpp(1));%isolado psi_ppe
eq4=rhs(aux1)==rhs(aux2);%substituindo psi_ppe
eq44=rhs(aux22)==rhs(aux1);%substituindo psi_ppe
aux3=isolate(eq4,qpp(2));%isolado psi_ppd
aux33=isolate(eq44,qpp(2));%isolado psi_ppd
eq5=rhs(aux3)==rhs(aux33);%substituindo psi_ppd
THETA_PP=isolate(eq5,qpp(3));%equação para theta_pp dependente das
derivadas de primeira ordem e ordem zero

 % Determinação da equação para a aceleração angular Psi_e
aux1=isolate(eq2==0,qpp(2));%isolado psi_ppd
aux2=isolate(eq3==0,qpp(2));%isolado psi_ppd
aux22=isolate(eq1==0,qpp(2));%isolado psi_ppd
eq4=rhs(aux1)==rhs(aux2);%substituindo psi_ppd
eq44=rhs(aux22)==rhs(aux1);%substituindo psi_ppd
aux3=isolate(eq4,qpp(3));%isolado theta_pp
aux33=isolate(eq44,qpp(3));%isolado theta_pp
eq5=rhs(aux3)==rhs(aux33);%substituindo theta_pp
PSI_PPE=isolate(eq5,qpp(1));%equação para psi_ppe dependente das derivadas
de primeira ordem e ordem zero

 % Determinação da equação para a aceleração angular Psi_d
```

```
 aux1=isolate(eq3==0,qpp(3));%isolado theta_pp
aux2=isolate(eq1==0,qpp(3));%isolado theta_pp
aux22=isolate(eq2==0,qpp(3));%isolado theta_pp
eq4=rhs(aux1)==rhs(aux2);%substituindo theta_pp
eq44=rhs(aux22)==rhs(aux1);%substituindo theta_pp
aux3=isolate(eq4,qpp(1));%isolado psi_ppde
aux33=isolate(eq44,qpp(1));%isolado theta_pp
eq5=rhs(aux3)==rhs(aux33);%substituindo psi_ppe
PSI_PPD=isolate(eq5,qpp(2));%equação para psi_ppd dependente das derivadas
de primeira ordem e ordem zero

%substituição de variáveis auxiliares
PPE=subs(PSI_PPE,[psi_e psi_pe psi_d psi_pd theta theta_p],[x1 x2 x3 x4 x5
x6]);
PPD=subs(PSI_PPD,[psi_e psi_pe psi_d psi_pd theta theta_p],[x1 x2 x3 x4 x5
x6]);
PP=subs(THETA_PP,[psi_e psi_pe psi_d psi_pd theta theta_p],[x1 x2 x3 x4 x5
x6]);

coeftau1=collect(PPE,tau_e);
coeftau2=collect(PPE,tau_d);
coeftau3=collect(PPD,tau_e);
coeftau4=collect(PPD,tau_d);
coeftau5=collect(PP,tau_e);
coeftau6=collect(PP,tau_d);

F=[x2,
    rhs(PPE),
    x4,
    rhs(PPD),
    x6
    rhs(PP)]
G=[0,
    rhs(coeftau1)+rhs(coeftau2),
    0,
    rhs(coeftau3)+rhs(coeftau4),
    0,
    rhs(coeftau5)+rhs(coeftau6)]
jac=jacobian(F,[x1,x2,x3,x4,x5,x6])
```

```
%O pêndulo equilibra em X=[0;0;0;0;0;0], logo a linearização é dada por:


x1=0;x1=0;x2=0;x3=0;x4=0;x5=0;x6=0;
jac_linear=vpa(subs(jac),3)
Bx=vpa(subs(G),3)
B= [0 0,
    4023.38 12207.10,
    0 0,
    12207.11 4023.38,
    0 0,
    -2268.91 -2268.91]


%% LQR

m=1;
r=0.035;
a=1.20e-3;
b=3.40e-3;
c=3.99e-3;
l=0.1;
d=0.2;
g=9.81;

A=eval(jac_linear);
C= eye(6);
D=zeros(6,2);
t=1/100;
[AD, BD, CD, DD]=c2dm(A, B, C, D, t, 'zoh')

Q=[1 0 0 0 0 0%psi_e
    0 1 0 0 0 0%psi_pe
    0 0 1 0 0 0%psi_d
    0 0 0 1 0 0%psi_pd
    0 0 0 0 1 0%theta
    0 0 0 0 0 1]%theta_pd
R=[1 0;
    0 0.5];
[KLQR]=lqr(AD,BD,Q/1000000000,R) %matriz K de ganhos para ser add ao modelo
C=eye(6)
D=[0 0;0 0;0 0;0 0;0 0;0 0]
```

Apêndice B - Simulação

Este código foi desenvolvido no *software* comercial *MATLAB* e tem por objetivo obter:

1. O comportamento dinâmico do sistema por meio da solução das equações diferenciais contidas na função auxiliar “*axufunc*” (Apêndice C) pelo método de Runge-Kutta (ode45);
2. O comportamento gráfico bidimensional x-z do sistema ao longo do tempo;
3. O comportamento gráfico bidimensional x-y do sistema ao longo do tempo;
4. O comportamento gráfico tridimensional do sistema ao longo do tempo;
5. O comportamento gráfico dos ângulos de *pitch* θ e *yaw* φ do sistema ao longo do tempo;
6. O comportamento gráfico da energia do sistema em *joule* ao longo do tempo;
7. Erro RMS dos ângulos *pitch* θ e *yaw* φ para o tempo simulado.

```
clear
close all
global m r a b c l d g I;


m = 1; %kg
r = 0.035; %m
l = 0.1; %m
d = 0.2;%m
g = 9.81; %m/s^2
%momentos de inércia de roll(x"), pitch(y") e yaw(z") [kg*m^2]
a=1.20e-3; b=3.40e-3; c=3.99e-3;
I = b + m*l^2;
```

```
q0 = [0 0 0.4 0 0 0 0 0 0]'; %estado inicial em radianos: [psi_e psi_d
theta psi_pe psi_pd theta_p xb yb phi]
tf = 2; %tempo final de simulação
[t,q] = ode45('auxfunc',[0 tf],q0);

psi_e = q(:,1); psi_d = q(:,2); theta = q(:,3);
psi_pe = q(:,4); psi_pd = q(:,5); theta_p = q(:,6);
xb = q(:,7); yb = q(:,8); phi = q(:,9); %centro da base

%cinemática direta:
xc = xb + l*sin(theta).*cos(phi); %centro de massa total
yc = yb + l*sin(theta).*sin(phi);
zc = l*cos(theta); %assume z = 0 no centro das rodas
xee = xb + 2*l*sin(theta).*cos(phi); %extremidade do pendulo, assumida em
2*l
yee = yb + 2*l*sin(theta).*sin(phi);
zee = 2*l*cos(theta); %assume z = 0 no centro das rodas

xe = xb - sin(phi)*d/2; %centro da roda esquerda
ye = yb + cos(phi)*d/2;
xd = xb + sin(phi)*d/2; %centro da roda direita
yd = yb - cos(phi)*d/2;

passo = 0.01; %passo das animações em segundos

f = figure(1); %vista lateral x-z
set(f, 'doublebuffer', 'on'); % for smoothness in the animation
tempo = -passo;
for i = 1:size(t,1)
 if t(i) >= (tempo+passo)
  tempo = t(i);
  hold off
  plot([-6*l 6*l],[0 0],'g--'); %floor
  axis equal
  axis([-6*l 6*l -4.5*l, 4.5*l]);
  hold on
  plot(xe(i),0,'co') %desenha roda esquerda
  plot(xd(i),0,'bo') %desenha roda direita
  plot(xc(i),zc(i),'bx') %desenha centro de massa
  plot([xe(i) xd(i)],[0 0],'b') %desenha base
```

```
  plot([xb(i) xee(i)],[0 zee(i)],'b') %desenha pêndulo
  drawnow;
  pause(passo) % T sec per frame
 end
end

f = figure(2); %vista superior x-y
set(f, 'doublebuffer', 'on'); % for smoothness in the animation
tempo = -passo;
for i = 1:size(t,1)
 if t(i) >= (tempo+passo)
  tempo = t(i);
  hold off
  plot([-6*l 6*l],[0 0],'g--'); %floor
  axis equal
  axis([-6*l 6*l -4.5*l, 4.5*l]);
  hold on
  plot(xe(i),ye(i),'co') %desenha roda esquerda
  plot(xd(i),yd(i),'bo') %desenha roda direita
  plot(xc(i),yc(i),'bx') %desenha centro de massa
  plot([xe(i) xd(i)],[ye(i) yd(i)],'b') %desenha base
  plot([xb(i) xee(i)],[yb(i) yee(i)],'b') %desenha pêndulo
  drawnow;
  pause(passo) % T sec per frame
 end
end

f = figure(3); %vista 3D
set(f, 'doublebuffer', 'on'); % for smoothness in the animation
tempo = -passo;
for i = 1:size(t,1)
 if t(i) >= (tempo+passo)
  tempo = t(i);
  hold off
  plot3([-6*l 6*l],[0 0],[0 0],'g--'); %floor
  axis equal
  grid on
  view(3)
  axis([-6*l 6*l -6*l 6*l 0 2*l]);
  hold on
```

```
  plot3(xe(i),ye(i),0,'co') %desenha roda esquerda
  plot3(xd(i),yd(i),0,'bo') %desenha roda direita
  plot3(xc(i),yc(i),zc(i),'bx') %desenha centro de massa
  plot3([xe(i) xd(i)],[ye(i) yd(i)],[0 0],'b') %desenha base
  plot3([xb(i) xee(i)],[yb(i) yee(i)],[0 zee(i)],'b') %desenha pêndulo
  drawnow;
  pause(passo) % T sec per frame
 end
end

figure(4)
plot(t,theta*180/pi,'b')
hold on
plot(t,phi*180/pi,'r')
title('ângulos de pitch e yaw')
xlabel('tempo(s)')
ylabel('ângulo(graus)')
legend('theta','phi')

figure(5)
phi_p = (psi_pd - psi_pe)*r/d; %velocidade angular de yaw da base
v = r*(psi_pe + psi_pd)/2; %velocidade de translação da base
vcxy = v + l*cos(theta).*theta_p; %velocidade horizontal do centro de massa
vcz = -l*sin(theta).*theta_p; %velocidade vertical do centro de massa
vct = l*sin(theta).*phi_p; %velocidade transversal do centro de massa
T = 0.5*m*(vcxy.^2 + vcz.^2 + vct.^2) + 0.5*b*theta_p.^2 +
0.5*(a*sin(theta).^2+c*cos(theta).^2).*phi_p.^2;
U = m*g*l*cos(theta);
plot(0,0,'b.')
hold on
plot(t,T+U,'r')
title('energia')
xlabel('tempo(s)')
ylabel('T+U(J)')

disp('Erro RSM do theta')
s=ones(size(theta,1),1)*0;
ermsv=sqrt(sum((abs(s-theta)).^2)/size(theta,1))

disp('Erro RSM do phi')
s=ones(size(phi,1),1)*0.4*tf;
ermsv=sqrt(sum((abs(s-phi)).^2)/size(phi,1))
```

Apêndice C – Função Auxiliar

Este código foi desenvolvido no *software* comercial *MATLAB* e tem por objetivo obter, dado uma condição inicial, o comportamento dinâmico do sistema por meio da aplicação dos ganhos PID ou LQR no cálculo dos torques das rodas direita e esquerda do robô para então calcular $\dot{\psi}_e, \dot{\psi}_d, \dot{\theta}, \ddot{\psi}_e, \ddot{\psi}_d, \ddot{\theta}, \dot{x}_b, \dot{y}_b$ *e* $\dot{\varphi}$.

Para resolver as equações diferenciais $\ddot{\psi}_e$, $\ddot{\psi}_d$ e $\ddot{\theta}$ contidas neste Apêndice, foi usada a função pronta do *MATLAB* ode45 que utiliza o método de Runge-Kutta.

A função ode45 necessita de uma função que retorne a derivada do estado ("*auxfunc*"), e portanto precisa das acelerações $\ddot{\psi}_e$, $\ddot{\psi}_d$ e $\ddot{\theta}$.

O equacionamento das acelerações $\ddot{\psi}_e, \ddot{\psi}_d$ e $\ddot{\theta}$, aqui chamadas de *psi_ppe, psi_ppd e theta_pp,* foi obtido simbolicamente usando o toolbox simbólico do *MATLAB* (rotina descrita no Apêndice A) e por esse motivo as equações na sua forma explícita são extensas.

```
function psi_ppe = auxfunc(t, q)
%equações para simular um Segway em 3D
global m r a b c l d g I;
t

psi_e = q(1); %giro da roda esquerda
psi_d = q(2); %giro da roda direita
theta = q(3); %ângulo de pitch (sentido horário)
psi_pe = q(4); %derivada de psi_e
psi_pd = q(5); %derivada de psi_d
theta_p = q(6); %derivada de theta
xb = q(7); %posição do ponto médio entre as rodas
yb = q(8);
phi = q(9); %ângulo de yaw

%valores de referência de posição e velocidade
theta_r = 0;
theta_pr = 0;
psi_er = 0;
psi_per = 0;
psi_dr = 0.1;
psi_pdr = 0;
%v_r = 0.2; %velocidade linear longitudinal desejada
v_r = r*(psi_pe + psi_pd)/2+2*cos(4*pi*t*0.5); %velocidade linear
longitudinal desejada
%phi_pr = pi/2; %velocidade angular de yaw da base desejada
%phi_pr = 0;
```

```
phi_pr = 7*sin(2*pi*t*0.5);
mu=0.8;%coeficiente de atrito

%estimativas das variáveis não medidas
v = r*(psi_pe + psi_pd)/2; %velocidade de translação longitudinal
phi_p = (psi_pd - psi_pe)*r/d; %velocidade angular de yaw da base

au=2; %2 = LQR; 1 = PID

if au ==2 %LQR
    tau_e = -62.0664*(theta_r - theta) - 8.1699*(theta_pr - theta_p)  -
0.1*(phi_pr - phi_p) + 0*(v_r - v);
    tau_d = -62.0664*(theta_r - theta) - 8.1699*(theta_pr - theta_p)  +
0.1*(phi_pr - phi_p) - 0*(v_r - v);
end
if au==1 %PID
    tau_e = -5*(theta_r - theta) - 1*(theta_pr - theta_p) - 0.1*(phi_pr -
phi_p) + 0*(v_r - v);
    tau_d = -5*(theta_r - theta) - 1*(theta_pr - theta_p) + 0.1*(phi_pr -
phi_p) + 0*(v_r - v);
end

%equação da aceleração psi_ppe
psi_ppe= -(((tau_d + (m*(l*r*theta_p^2*sin(theta) -
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 -
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2))/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) - (tau_e +
(m*(l*r*theta_p^2*sin(theta) +
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 +
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2))/((m*(r^2/2 - (2*l^2*r^2*sin(theta)^2)/d^2))/2 -
(r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))/((l*m*r*cos(theta))/(2*((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)) - (l*m*r*cos(theta))/(2*((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))) + ((tau_d + (m*(l*r*theta_p^2*sin(theta) -
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 -
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2))/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) + (2*(tau_d + tau_e +
(m*(l*r*theta_p*sin(theta)*(psi_pd + psi_pe) -
(2*l^2*r^2*cos(theta)*sin(theta)*(psi_pd - psi_pe)^2)/d^2))/2 -
(r^2*(2*a*cos(theta)*sin(theta) - 2*c*cos(theta)*sin(theta))*(psi_pd -
psi_pe)^2)/(2*d^2) - g*l*m*sin(theta) - (l*m*r*theta_p*sin(theta)*(psi_pd +
psi_pe))/2))/(l*m*r*cos(theta)))/((2*(m*l^2 + b))/(l*m*r*cos(theta)) -
(l*m*r*cos(theta))/(2*((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2))))/((((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)/((m*(r^2/2 - (2*l^2*r^2*sin(theta)^2)/d^2))/2 -
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) - ((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))/((l*m*r*cos(theta))/(2*((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
```

```
c*cos(theta)^2))/d^2)) - (l*m*r*cos(theta))/(2*((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))) - (((m*(r^2/2 - (2*l^2*r^2*sin(theta)^2)/d^2))/2 -
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2)/((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2) - 1)/((2*(m*l^2 + b))/(l*m*r*cos(theta)) -
(l*m*r*cos(theta))/(2*((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2))));

%equação da aceleração psi_ppd
psi_ppd=-(((tau_d + tau_e + (m*(l*r*theta_p*sin(theta)*(psi_pd + psi_pe) -
(2*l^2*r^2*cos(theta)*sin(theta)*(psi_pd - psi_pe)^2)/d^2))/2 -
(r^2*(2*a*cos(theta)*sin(theta) - 2*c*cos(theta)*sin(theta))*(psi_pd -
psi_pe)^2)/(2*d^2) - g*l*m*sin(theta) - (l*m*r*theta_p*sin(theta)*(psi_pd +
psi_pe))/2)/(m*l^2 + b) + (2*(tau_d + (m*(l*r*theta_p^2*sin(theta) -
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 -
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2)))/(l*m*r*cos(theta)))/((2*((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))/(l*m*r*cos(theta)) - (l*m*r*cos(theta))/(2*(m*l^2 +
b))) - ((tau_d + tau_e + (m*(l*r*theta_p*sin(theta)*(psi_pd + psi_pe) -
(2*l^2*r^2*cos(theta)*sin(theta)*(psi_pd - psi_pe)^2)/d^2))/2 -
(r^2*(2*a*cos(theta)*sin(theta) - 2*c*cos(theta)*sin(theta))*(psi_pd -
psi_pe)^2)/(2*d^2) - g*l*m*sin(theta) - (l*m*r*theta_p*sin(theta)*(psi_pd +
psi_pe))/2)/(m*l^2 + b) + (2*(tau_e + (m*(l*r*theta_p^2*sin(theta) +
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 +
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2)))/(l*m*r*cos(theta)))/((2*((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))/(l*m*r*cos(theta)) - (l*m*r*cos(theta))/(2*(m*l^2 +
b))))/(((2*((m*(r^2/2 - (2*l^2*r^2*sin(theta)^2)/d^2))/2 -
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2))/(l*m*r*cos(theta)) -
(l*m*r*cos(theta))/(2*(m*l^2 + b)))/((2*((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))/(l*m*r*cos(theta)) - (l*m*r*cos(theta))/(2*(m*l^2 +
b))) - ((2*((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2))/(l*m*r*cos(theta)) -
(l*m*r*cos(theta))/(2*(m*l^2 + b)))/((2*((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2))/(l*m*r*cos(theta)) - (l*m*r*cos(theta))/(2*(m*l^2 +
b))));

%equação da aceleração theta_pp
theta_pp=(((tau_e + (m*(l*r*theta_p^2*sin(theta) +
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 +
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2)))/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) + (2*(tau_d + tau_e +
(m*(l*r*theta_p*sin(theta)*(psi_pd + psi_pe) -
(2*l^2*r^2*cos(theta)*sin(theta)*(psi_pd - psi_pe)^2)/d^2))/2 -
(r^2*(2*a*cos(theta)*sin(theta) - 2*c*cos(theta)*sin(theta))*(psi_pd -
psi_pe)^2)/(2*d^2) - g*l*m*sin(theta) - (l*m*r*theta_p*sin(theta)*(psi_pd +
psi_pe))/2))/(l*m*r*cos(theta)))/(((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) - 1) + ((tau_e +
(m*(l*r*theta_p^2*sin(theta) +
```

```
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 +
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2))/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) - (tau_d +
(m*(l*r*theta_p^2*sin(theta) -
(2*l^2*r^2*theta_p*cos(theta)*sin(theta)*(2*psi_pd - 2*psi_pe))/d^2))/2 -
(r^2*(2*a*theta_p*cos(theta)*sin(theta) -
2*c*theta_p*cos(theta)*sin(theta))*(2*psi_pd -
2*psi_pe))/(2*d^2))/((m*(r^2/2 - (2*l^2*r^2*sin(theta)^2)/d^2))/2 -
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2))/(((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)/((m*(r^2/2 - (2*l^2*r^2*sin(theta)^2)/d^2))/2 -
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) - ((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)))/(((l*m*r*cos(theta))/(2*((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)) - (l*m*r*cos(theta))/(2*((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)))/(((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2)/((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2) - ((m*(r^2/2 - (2*l^2*r^2*sin(theta)^2)/d^2))/2 -
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2)/((m*(r^2/2 +
(2*l^2*r^2*sin(theta)^2)/d^2))/2 + (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)) - ((2*(m*l^2 + b))/(l*m*r*cos(theta)) -
(l*m*r*cos(theta))/(2*((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2)))/(((m*(r^2/2 -
(2*l^2*r^2*sin(theta)^2)/d^2))/2 - (r^2*(a*sin(theta)^2 +
c*cos(theta)^2))/d^2)/((m*(r^2/2 + (2*l^2*r^2*sin(theta)^2)/d^2))/2 +
(r^2*(a*sin(theta)^2 + c*cos(theta)^2))/d^2) - 1));

v = r*(psi_pe + psi_pd)/2;
xb_p = v*cos(phi);
yb_p = v*sin(phi);
phi_p = (psi_pd - psi_pe)*r/d;

%return
psi_ppe = [psi_pe psi_pd theta_p psi_ppe psi_ppd theta_pp xb_p yb_p
phi_p]'; %precisa ser vetor coluna
```

Apêndice D – Função Arduino

Projetado com um microcontrolador Atmel AVR (8-bit) com suporte de entrada/saída embutido, o Arduino é uma plataforma de prototipagem eletrônica de hardware livre. A linguagem de programação é essencialmente C/C++.

Este código foi desenvolvido no *software* livre *Arduino* e tem por objetivo carregar bibliotecas de filtro de sinal, giroscópio, encoder e thread. As threads estão descritas no Apêndice H.

As funções contidas nos Apêndices D, E, F, G e H foram utilizadas no protótipo da Figura 8 descrito na seção 3.5 com intuito de replicar a dinâmica observada nas simulações da seção 4.

```
//=======libraries========
#include "I2Cdev.h"
#include "MPU6050.h"
#include <Kalman.h>
//========================
#include <Encoder.h>
#include "Wire.h"
#include "Thread.h"
#include "ThreadController.h"
```

Apêndice E – Função Encoder

Este código foi desenvolvido no *software* livre *Arduino* e tem por objetivo ler os sensores encoders.

```
//Encoder object definition
Encoder Enc_right(2,5);
Encoder Enc_left(3,4);

#define RESOLUTION 225 // ppr
#define WHEEL_DIAMETER 75 //mm

double encoder_left=0.0;
double encoder_right=0.0;
double X=0.0; // Control variable

void encoders_read()
{
  encoder_left=Enc_left.read();
  encoder_right=Enc_right.read();
}
float encoder_2_mm(double in)
{
  float movement_pulses=0;
  //for(int i=0;i<4;i++){movement_pulses+=abs(encoder_pos[i]);}
  //movement_pulses/=4;

  movement_pulses=abs(in);
  return {(PI*WHEEL_DIAMETER*movement_pulses)/RESOLUTION};
}
//=====================================================
void reset_encoder() //reset encoder memory
{
  Enc_left.write(0);
  Enc_right.write(0);
}
```

Apêndice F – Função Motor

Este código foi desenvolvido no *software* livre *Arduino* e tem por objetivo acionar os motores.

```
//motor 1 port definition left
#define IN1 7
#define IN2 8
#define ENA 9 //pwm port

//motor 2 port definition right
#define IN3 10
#define IN4 12
#define ENB 11  //pwm port

void MOTOR_SETUP()
{
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(ENA,OUTPUT);
  pinMode(ENB,OUTPUT);
}
void motor_cmd(int l,int r)
{
  if(l>255){l=255;}
  if(r>255){r=255;}
  if(l>0) //motor forward
  {
    digitalWrite(IN1,HIGH);
    digitalWrite(IN2,LOW);
    analogWrite(ENA,abs(l));
  }
  if(l<0) // motor backwards
  {
```

```
    digitalWrite(IN1,LOW);
    digitalWrite(IN2,HIGH);
    analogWrite(ENA,abs(l));
  }

  if(r>0) // motor backwards
  {
    digitalWrite(IN3,HIGH);
    digitalWrite(IN4,LOW);
    analogWrite(ENB,abs(r));
  }
  if(r<0) //motor forward
  {
    digitalWrite(IN3,LOW);
    digitalWrite(IN4,HIGH);
    analogWrite(ENB,abs(r));
  }
  if(r==0)
  {
    digitalWrite(IN3,HIGH);
    digitalWrite(IN4,HIGH);
    analogWrite(ENB,0);
  }
  if(l==0)
  {
    digitalWrite(IN1,LOW);
    digitalWrite(IN2,LOW);
    analogWrite(ENA,0);
  }
}
```

Apêndice G – Função IMU

Este código foi desenvolvido no *software* livre *Arduino* por um desenvolvedor anônimo e adaptado. O objetivo é ler o giroscópio IMU.

```
/*
    Arduino UNO    MPU6050
 A4           SDA
 A5           SCL
 VCC          5V
 GND          GND
 */
 //=======libraries========
#include "I2Cdev.h"
#include "MPU6050.h"
#include <Kalman.h>
//========================

//====kalman_initialization========
 Kalman kalmanY,kalmanX,kalmanZ;
 double kalAngleX,kalAngleY=0,kalAngleZ;
 int32_t timer;
 float Theta=0.0; // control variable theta
//================================

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

// ====class default I2C address is 0x68===
MPU6050 accelgyro;
//=======================================

//===raw values variables initialization====
int16_t ax, ay, az;
int16_t gx, gy, gz;
```

```
    int offset[6]={-2300,-1859,1311,71,21,-29}; // offset values for MPU6050
[ax,ay,az,gx,gy,gz]

    //=================================

    //==filtered variables initialization==
    unsigned long last_read_time;
    float         last_x_angle;  // These are the filtered angles
    float         last_y_angle;
    float         last_z_angle;
    float          last_gyro_x_angle;  // Store the gyro angles to compare
drift
    float         last_gyro_y_angle;
    float         last_gyro_z_angle;
    //=================================

    void set_last_read_angle_data(unsigned long time, float x, float y,
float z, float x_gyro, float y_gyro, float z_gyro) {
      last_read_time = time;
      last_x_angle = x;
      last_y_angle = y;
      last_z_angle = z;
      last_gyro_x_angle = x_gyro;
      last_gyro_y_angle = y_gyro;
      last_gyro_z_angle = z_gyro;
    }

    inline unsigned long get_last_time() {return last_read_time;}
    inline float get_last_x_angle() {return last_x_angle;}
    inline float get_last_y_angle() {return last_y_angle;}
    inline float get_last_z_angle() {return last_z_angle;}
    inline float get_last_gyro_x_angle() {return last_gyro_x_angle;}
    inline float get_last_gyro_y_angle() {return last_gyro_y_angle;}
    inline float get_last_gyro_z_angle() {return last_gyro_z_angle;}
    //=================================================================
    void IMU_SETUP()
    {
      #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
      Wire.begin();
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
      Fastwire::setup(400, true);
```

```
#endif

 accelgyro.initialize();

 // use the code below to change accel/gyro offset values from the calibration sketch
  accelgyro.setXAccelOffset(offset[0]);
  accelgyro.setYAccelOffset(offset[1]);
  accelgyro.setZAccelOffset(offset[2]);
  accelgyro.setXGyroOffset(offset[3]);
  accelgyro.setYGyroOffset(offset[4]);
  accelgyro.setZGyroOffset(offset[5]);

  set_last_read_angle_data(millis(), 0, 0, 0, 0, 0, 0);
  timer=millis();


}
//============================================================
void IMU_READ()
{
   accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
   unsigned long t_now = millis();     // Get the time of reading for rotation computations

  // Convert gyro values to degrees/sec
  float FS_SEL=131;
  float gyro_x=gx/FS_SEL;
  float gyro_y=gy/FS_SEL;
  float gyro_z=gz/FS_SEL;

  // Compute the (filtered) gyro angles
  float dt =(t_now - get_last_time())/1000.0;
  float gyro_angle_x = gyro_x*dt + get_last_x_angle();
  float gyro_angle_y = gyro_y*dt + get_last_y_angle();
  float gyro_angle_z = gyro_z*dt + get_last_z_angle();

  // Compute the drifting gyro angles
  float unfiltered_gyro_angle_x = gyro_x*dt + get_last_gyro_x_angle();
  float unfiltered_gyro_angle_y = gyro_y*dt + get_last_gyro_y_angle();
  float unfiltered_gyro_angle_z = gyro_z*dt + get_last_gyro_z_angle();
```

```
      // Get angle values from accelerometer, expression taken from MPU6050
datasheet
      float RADIANS_TO_DEGREES = 180/3.14159;
      float      accel_angle_y      =      atan(1*ax/sqrt(pow(ay,2)      +
pow(az,2)))*RADIANS_TO_DEGREES;
      float      accel_angle_x      =      atan(ay/sqrt(pow(ax,2)      +
pow(az,2)))*RADIANS_TO_DEGREES;
      float accel_angle_z = 0;

      //calculating absolute angle values
      float alpha = 0.04; // parameter taken from MPU6050 datasheet
      double angle_x = alpha*gyro_angle_x + (1.0 - alpha)*accel_angle_x;
      double angle_y = (alpha*gyro_angle_y + (1.0 - alpha)*accel_angle_y);
      double angle_z = gyro_angle_z;  //Accelerometer doesn't give z-angle

     //kalman's filer
      kalmanX.setAngle(angle_x);
      kalmanY.setAngle(angle_y);
      kalmanZ.setAngle(angle_z);

      //calculating angles from kalman's filter
      kalAngleX=(float)(kalmanX.getAngle(angle_x,gyro_x,(double)(micros()-
timer)/1000000));
      kalAngleY=(float)(kalmanY.getAngle(angle_y,gyro_y,(double)(micros()-
timer)/1000000));
      kalAngleZ=-
(float)(kalmanZ.getAngle(angle_z,gyro_z,(double)(micros()-timer)/1000000));
      kalAngleZ=constrain(kalAngleZ ,-90,90);
      timer=micros();

      Theta=kalAngleY; // imu output

      set_last_read_angle_data(t_now,   angle_x,   angle_y   ,   angle_z,
unfiltered_gyro_angle_x, unfiltered_gyro_angle_y, unfiltered_gyro_angle_z);
    }
    //=======================================================
    void display() {
      // read raw accel/gyro measurements from device
      // display
```

```
  Serial.print("accelRawX:");
  Serial.print(ax);
  Serial.print("\t");
  Serial.print("accelRawY:");
  Serial.print(ay);
  Serial.print("\t");
  Serial.print("accelRawZ:");
  Serial.print(az);
  Serial.print("\t");
  Serial.print("GyroRawX:");
  Serial.print(gx);
  Serial.print("\t");
   Serial.print("GyroRawY:");
  Serial.print(gy);
  Serial.print("\t");
  Serial.print("GyroRawZ:");
  Serial.print(gz);
  Serial.print("\t");
  Serial.print("kalmanY:");
  Serial.print( kalAngleY);
  Serial.print("\t");
  Serial.print("kalmanX:");
  Serial.print( kalAngleX);
  Serial.print("\t");
  Serial.print("kalmanZ:");
  Serial.println( kalAngleZ);
}
```

Apêndice H – Função Thread

Este código foi desenvolvido no *software* livre *Arduino* e tem por objetivo acelerar a leitura múltipla dos sensores.

```
Thread sensors;
Thread control;
ThreadController cpu;

float K[6]={-1252, -4.3, -1247, -4.2, -8958, -30.5 }; // LQR Gains

float U=0.0;
void read_all_sensor()
{
  //reading sensors
  IMU_READ(); //get Theta
  encoders_read(); //raw encoder data
  //convert ppr to mm
  float X1=encoder_2_mm(encoder_left);
  float X2=encoder_2_mm(encoder_right);
  X=(X1+X2)/2;  // take the mean value of both measurements
}

void Control_system()
{
  float Space_State[6]={X,0,Theta,0};

  //Control
  float aux=0.0; //summer
  for(int i=0;i<6;i++){aux+=-K[i]*Space_State[i];}
  U=aux;
  if(U>255){U=255;}
  if(U<-255){U=-255;}

  //Motor Actuation
  motor_cmd(U,-U);
```

```
  reset_encoder();
}

void Thread_setup()
{
  sensors.setInterval(3); // read all sensors every 3 ms
  sensors.onRun(read_all_sensor);

  control.setInterval(5);
  control.onRun(Control_system);

  cpu.add(&control);
  cpu.add(&sensors);
}
```