UNIVERSIDADE ESTADUAL PAULISTA

"JÚLIO DE MESQUITA FILHO"

Faculdade de Engenharia

Campus de Ilha Solteira

RAFAEL MÁXIMO DA SILVA

ESTRATÉGIA DE CONTROLE DISTRIBUÍDO APLICADO EM PLANTAS *BALL BALANCER*

Ilha Solteira

2021

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RAFAEL MÁXIMO DA SILVA

ESTRATÉGIA DE CONTROLE DISTRIBUÍDO APLICADO EM PLANTAS *BALL BALANCER*

Dissertação apresentada à Faculdade de Engenharia – UNESP – Campus de Ilha Solteira, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de Conhecimento: Automação.

Orientador: Prof. Dr. Jean Marcos de Souza Ribeiro

Ilha Solteira

2021

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

S586e

Silva, Rafael Máximo da.
Estratégia de controle distribuído aplicado em plantas ball balancer / Rafael Máximo da Silva. -- Ilha Solteira: [s.n.], 2021
138 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2021

Orientador: Jean Marcos de Souza Ribeiro
Inclui bibliografia

1. Controle distribuído. 2. Ball balancer. 3. Controle proporcional, integral e derivativo (PID). 4. Software SCADA. 5. Arduino.

João José Barbosa
Serviço Técnico de Biblioteca e Documentação
Diretor Técnico
CRB 8-5642

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Estratégia de Controle Distribuído aplicado em plantas Ball Balancer

AUTOR: RAFAEL MAXIMO DA SILVA
ORIENTADOR: JEAN MARCOS DE SOUZA RIBEIRO

Aprovado como parte das exigências para obtenção do Título de Mestre em ENGENHARIA ELÉTRICA, área: Automação pela Comissão Examinadora:

Prof. Dr. JEAN MARCOS DE SOUZA RIBEIRO (Participaçao Virtual)
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

Prof. Dr. RODRIGO CARDIM (Participaçao Virtual)
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira - UNESP

Prof. Dr. DIOGO RAMALHO DE OLIVEIRA (Participaçao Virtual)
Departamento de Eletroeletrônica / Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso do Sul - IFMS

Ilha Solteira, 04 de maio de 2021

À minha família, mas especialmente aos meus pais:
Edilson Alfredo e Maria Evanil
pelo exemplo, dedicação, incentivo
e amor que recebi.

À minha irmã,
Naara Carolina
Pelo apoio e carinho.
Ofereço

AGRADECIMENTOS

Primeiramente, agradeço a Deus, pela saúde, força concedida e pelo direcionamento, pois tem me ajudado em cada dia a vencer todos os obstáculos da vida. Em especial, dedico meus agradecimentos:

- Aos meus pais, Maria Evanil e Edilson Alfredo, e minha irmã Naara, pela ajuda, compreensão e amor que recebi; mesmo na distância eles nunca deixaram de me socorrer e apoiar. A todos da minha família, que nunca se esqueceram de mim, sempre realizando orações e sempre me apoiando;
- Ao meu orientador, Prof. Dr. Jean Marcos de Souza Ribeiro, pela oportunidade oferecida, pela orientação, pelo tempo dedicado e ensinamentos, os quais foram fundamentais no desenvolvimento deste trabalho;
- Aos professores e colegas do laboratório LPC, pelo apoio e pela colaboração durante a pesquisa;
- Aos técnicos Wendel Jordão e Valdemir Chaves do DEE-FEIS (Departamento de Engenharia Elétrica - Faculdade de Engenharia de Ilha Solteira) pela confecção do projeto mecânico dos módulos *Ball Balancer*;
- Agradeço a todos os professores e funcionários da FEIS-UNESP, que diretamente ou indiretamente contribuíram para a realização deste trabalho;
- Aos colegas do PPGEE - Programa de Pós-Graduação em Engenharia Elétrica, pela convivência e pela ajuda de forma direta ou indireta.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

Nesta dissertação foi realizado um estudo sobre aplicação de controle distribuído em plantas *Ball Balancer* no Matlab® e Simulink®, bem como a implementação em bancada com dois *Ball Balancer*, usando técnicas clássicas de controle. Para obter o controle cooperativo das plantas, foi considerada a possibilidade de chaveamento entre controladores em cada instante de amostragem, para o sistema que está em pior condição em relação ao ponto de operação desejado. Para avaliar isso, considerou-se que o pior caso é a planta que tem a maior norma no sinal de controle, ou seja, a maior resposta ao controle Proporcional, Integral e Derivativo (PID). Também, foi utilizada a estratégia de obter o índice de desempenho com menor esforço computacional, calculando o valor absoluto do erro do sistema. Inicialmente, as plantas foram analisadas, modeladas e identificadas, tendo como referência o módulo didático 2D *Ball Balancer* da QUANSER®. Posteriormente, o controle individual da planta em nível de simulação foi realizado no Matlab® e Simulink®. Por fim, foram comparados os resultados entre o controle considerando o seletor de chaveamento automático de controle e sem o seletor atuando em período determinado, mostrando uma melhor resposta com a estratégia do seletor. Na implementação foi utilizado o Arduino Mega com *shield* Ethernet, para o controle das duas plantas e comunicação com supervisório Elipse E3®. Para visualização dos gráficos, e ajustes de ganhos do controlador PID, foi confeccionada uma interface gráfica em Elipse E3®, que se comunica com o Arduino através do protocolo Modbus TCP. O vídeo do sistema de controle com dois *Ball Balancer* implementado em laboratório, sem levar em conta o índice de desempenho, pode ser visto no *link* https://youtu.be/MuFVLBfAST0, enquanto que o controle que leva em consideração o índice de desempenho se encontra no *link* https://youtu.be/7EdA0D4ycvY.

Palavras-chave: controle distribuído. *ball balancer*. controle proporcional integral e derivativo (PID). *software* SCADA. arduino.

ABSTRACT

In this dissertation, a study was carried out on the application of distributed control in Ball Balancer systems in Matlab® and Simulink®, as well as the implementation in workbench with two Ball Balancer, using classical control techniques. To obtain cooperative control of the plants, the possibility of switching between controllers at each sampling instant was considered, for the system that is in a worse condition concerning the desired point of operation. To evaluate this, it was considered that the worst case is the plant that has the highest modulus in the control signal, that is, the highest response to Proportional, Integral, and Derivative (PID) control. Also, the strategy of obtaining the performance index with less computational effort was used, calculating the absolute value of the system error. Initially, the plants were analyzed, modeled, and identified, using the QUANSER® 2D Ball Balancer didactic module as a reference. Subsequently, the individual control of the plant at the simulation level was carried out in Matlab® and Simulink®. Finally, the results were compared between the control considering the automatic control switch selector and without the selector operating in a determined period, showing a better response with the selector strategy. In the implementation, the Arduino Mega with Ethernet shield was used to control the two plants and communicate with the Elipse E3® supervisory. In order to visualize the graphs and adjust the gains of the PID controller, a graphic interface was made in Elipse E3®, which communicates with the Arduino through the Modbus TCP protocol. The video of the control system with two Ball Balancers implemented in the laboratory, without taking into account the performance index, can be seen at the link https://youtu.be/MuFVLBfAST0, while the control that takes into account the performance index is found at the link https://youtu.be/7EdA0D4ycvY.

Keywords: distributed control. ball balancer. proportional integral and derivative control (PID). SCADA software. arduino.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AD	Analógico Digital
DEE-FEIS	Departamento de Engenharia Elétrica - Faculdade de Engenharia de Ilha Solteira
ICSP	*In-Circuit Serial Programming* (Programação de Circuito Serial)
IDE	*Integrated Development Environment* (Ambiente Desenvolvimento Integrado)
IP	*Internet Protocol* (Protocolo de Internet)
ITO	*Indium Tin Oxide* (Óxido de Índio e Estanho)
LAN	*Local Area Network* (Rede de Área Local)
LPC	Laboratório de Pesquisa em Controle
NCS	*Networked Control System* (Sistema de Controle em Rede)
PD	Proporcional e Derivativo
PID	Proporcional, Integral e Derivativo
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
SCADA	*Supervisory Control And Data Acquisiton* (Controle Supervisório e Aquisição de Dados)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
UART	*Universal Asynchronous Receiver/Transmitter* (Receptor/transmissor assíncrono universal)
USB	*Universal Serial Bus* (Porta Serial Universal)
2 DOF	2 *Degrees of Freedom* (2 Graus de Liberdade)

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento do uso de redes de comunicação, várias áreas da engenharia estão sendo integradas a um ambiente conectado através de uma rede. Isso se deve a maior flexibilidade de tráfego de informação entre os componentes do sistema e ao seu menor custo de implantação, principalmente no caso de redes sem fio (WANG; LIU, 2008). O controle via rede, ou também referenciado pela sigla em inglês NCS (*Networked Control System*) é muito utilizado atualmente para projetos de automação industrial, que compreende sistemas de controle no espaço distribuído onde a comunicação entre sensores, atuadores e controladores ocorre através de uma rede de múltiplo propósito e compartilhada (GUPTA; CHOW, 2010; HESPANHA; NAGHSHTABRIZI; XU, 2007).

Apesar das vantagens que a rede industrial apresenta, a utilização da mesma, se resulta em vários problemas induzidos pela rede, com isso, há uma motivação na implementação e melhoria dos sistemas de controle em rede. Devido a extensa troca de dados pela rede, existe uma grande possibilidade de que ocorra perda de pacotes de dados (CHENG *et al*., 2013; LI; SHI, 2014), e geralmente uma rede compartilhada possui limitações de velocidade de transmissão, da quantidade de informação enviada em cada transmissão e a possibilidade da informação não ser entregue ao destino no tempo estabelecido, entre outros efeitos induzidos pela rede (CHEN *et al*., 2018; QU; YANG; HAN, 2018; SOUZA, 2012). Essa estrutura difere daquela estudada na teoria clássica de controle, onde os canais de comunicação de dados são considerados ideais e não são compartilhados.

A comunicação sobre canais perfeitos considerada na teoria de controle é válida, geralmente, para aplicações em sistemas onde o canal é dedicado somente a uma malha de controle. Portanto, o uso de redes compartilhadas pode comprometer a teoria de análise de sistemas de controle via redes com a teoria existente (MAESTRELLI, 2016; PIN; PARISINI, 2011). Desta forma, durante o projeto de controle é importante levar em conta fenômenos que podem ocorrer durante a transmissão e que são característicos da rede de comunicação, tais como: limitação da largura de faixa, perda de pacotes de dados, atraso e erros de quantização. Estes aspectos, que são tradicionalmente estudados em teoria de comunicação, se não forem considerados na malha de controle, podem piorar o desempenho do sistema controlado, levando-o inclusive à instabilidade (NETTO, 2018).

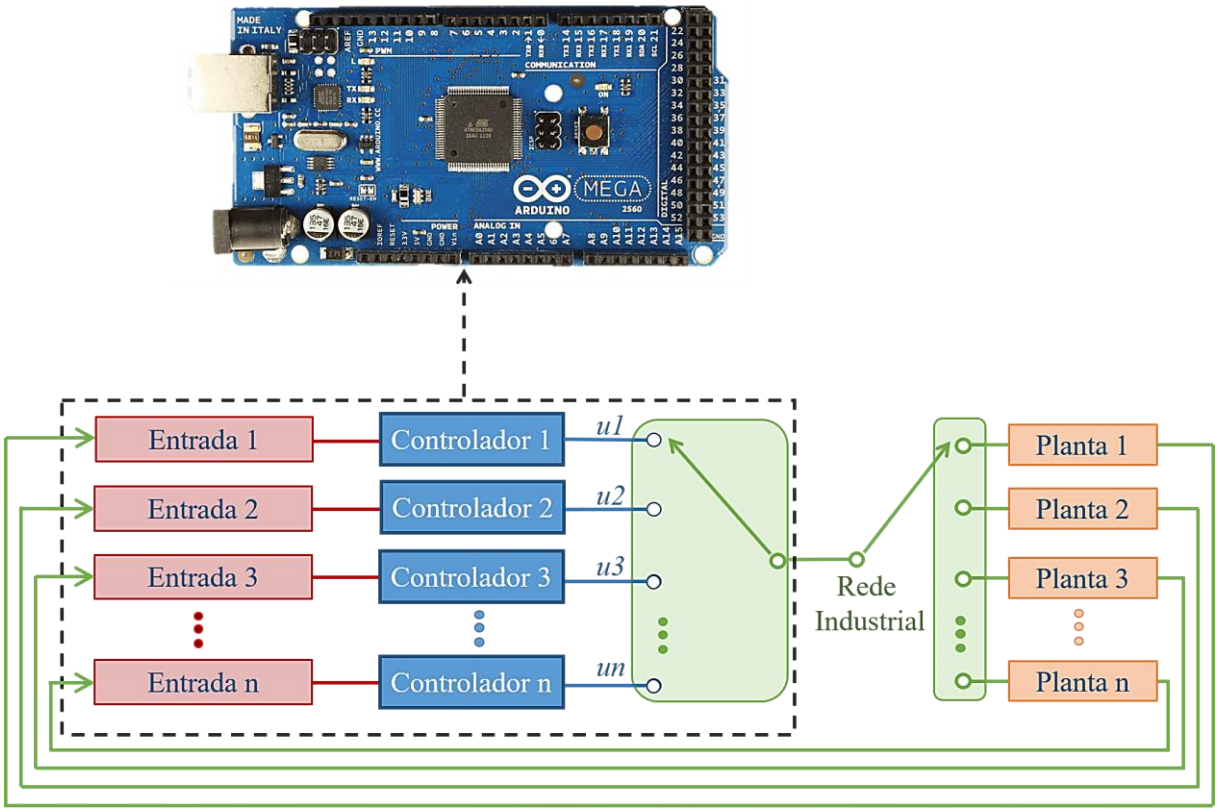
Observa-se que para diferentes níveis de imperfeições induzidas pela rede, estas podem ter efeitos indesejados no desempenho de controle de um sistema controlado via rede. Embora a eliminação destes fenômenos não possa ser separada das melhorias da própria infraestrutura de comunicação, o desenvolvimento de teorias e abordagens de controle para superá-las, é de grande necessidade e significativa (LI *et al*., 2019; ZHANG; GAO; KAYNAK, 2013). Portanto, como mitigar os efeitos dessas imperfeições no desempenho do sistema de controle em rede, tem-se recebido atenção considerável nos últimos anos por pesquisadores nas áreas de controle e comunicação (ZHANG *et al*., 2019).

Na literatura existem métodos de controle robusto que permitem assegurar a estabilidade de um sistema de controle em rede. Na referência (SOUSA; GEROMEL; DEAECTO, 2015), por exemplo, é determinado uma estratégia dinâmica de distribuição dos recursos da rede que permite assegurar estabilidade, bem como otimizar índices de desempenho H_2 e H_∞. Um dos benefícios desta política de controle é evitar o congestionamento da rede dificultando, portanto, a perda de pacotes no canal de comunicação.

Existem também, métodos que utilizam o controlador Proporcional, Integral e Derivativo (PID) para o controle de sistema em rede. Em (FADAEI; SALAHSHOOR, 2008), foi investigado o efeito do atraso na transmissão de dados, sobre o controle via rede, em seguida, uma abordagem de controle PID *Fuzzy*, foi proposta para melhorar o desempenho do controle em rede.

Deste modo, este trabalho tem por objetivo de tratar o controle cooperativo de vários sistemas, controlados através de uma rede de comunicação considerando uma estratégia para o seletor de controle. Na Figura 1 é apresentado um exemplo, em que há vários dispositivos conectados através da rede, e para diminuição do tráfico de informação na rede considerando que a cada período de amostragem apenas uma planta recebe o sinal através da rede, assim, não é necessário que todas as plantas recebam sinal de controle ao mesmo tempo. Levando em conta a capacidade de processamento do controlador central, nessa estrutura diminui-se a quantidade de pacotes enviados na rede.

Figura 1 – Arquitetura para diminuição do tráfico de informação na rede.



Fonte: próprio autor.

Assim, foi aplicado o controle distribuído, usando técnicas clássicas de controle em nível de simulação no Matlab® e Simulink®. Para obter o controle cooperativo das plantas, foi considerada uma estratégia para o índice de desempenho do chaveamento do controlador, com a ideia de chaveamento de controle, em cada instante de amostragem, para a planta que está em condição de maior necessidade de esforço do controlador em relação ao ponto de operação. Para avaliar isso, considerou-se a planta que tem o maior sinal de controle em módulo, ou seja, a maior resposta ao controle Proporcional, Integral e Derivativo (PID); em implementação em bancada foi verificado o pior caso através do cálculo do erro absoluto das variáveis controladas.

Para a simulação do controle distribuído, foi considerado inicialmente a modelagem de uma planta tendo como referência o modelo matemático do módulo didático *2D Ball Balancer* da QUANSER (2008). Os ganhos do controlador Proporcional e Derivativo (PD) e PID, foram encontrados levando em conta as especificações propostas pelo manual do fabricante Quanser®, e posteriormente a aplicação do controle individual da planta *Ball Balancer* em nível de simulação no Matlab® e Simulink®, foi realizada.

Para verificar esta estratégia proposta, foi realizado primeiramente a implementação com dois *Ball Balancer* sem considerar a estratégia de controle proposta, para análise do comportamento de dois sistemas sendo controlados com um Arduino Mega, comunicando com o supervisório Elipse E3®, através do protocolo Modbus TCP (*Transmission Control Protocol*). Posteriormente, foi testada a estratégia de decisão para escolha de qual planta receberá o sinal de controle em cada período de amostragem, verificou-se que para o controle de posição da esfera com a referência no centro, essa estratégia apresentou bom resultado para dois sistemas *Ball Balancer* controlados.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho foi investigar o controle distribuído em sistemas *Ball Balancer,* através de simulações e implementação em bancada, considerando uma estratégia para o seletor de controle. Os sistemas *Ball Balancer*, que foram projetados e construídos (pela equipe técnica do DEE-FEIS) durante o desenvolvimento do trabalho, demonstraram qualidade satisfatória para a implementação da estratégia proposta. A aplicação do método descrito neste trabalho, possibilita a diminuição do tráfego de informações em uma rede e, ao mesmo tempo, mantém as plantas estáveis no ponto de referência central.

1.2 ESTRUTURA DO TRABALHO

Este trabalho foi organizado da seguinte forma:

No Capítulo 2 é apresentada a modelagem matemática do sistema *Ball Balancer*, sua linearização e a estratégia de controle em cascata.

No Capítulo 3, as metodologias dos controles individuais da planta, utilizando o controlador Proporcional Derivativo (PD) e o controle Proporcional, Integral e Derivativo (PID), são apresentados.

Nos capítulos 4, 5 e 6 são apresentados os resultados de simulações.

No Capítulo 7, são apresentados os materiais e métodos utilizados para implementação do controle aplicado ao sistema *Ball Balancer*.

No Capítulo 8, são apresentados os resultados obtidos com duas plantas *Ball Balancer* sem considerar o índice de desempenho, com controle PID e com a estratégia de controle distribuído, usando o índice de desempenho, aplicado em plantas *Ball Balancer*.

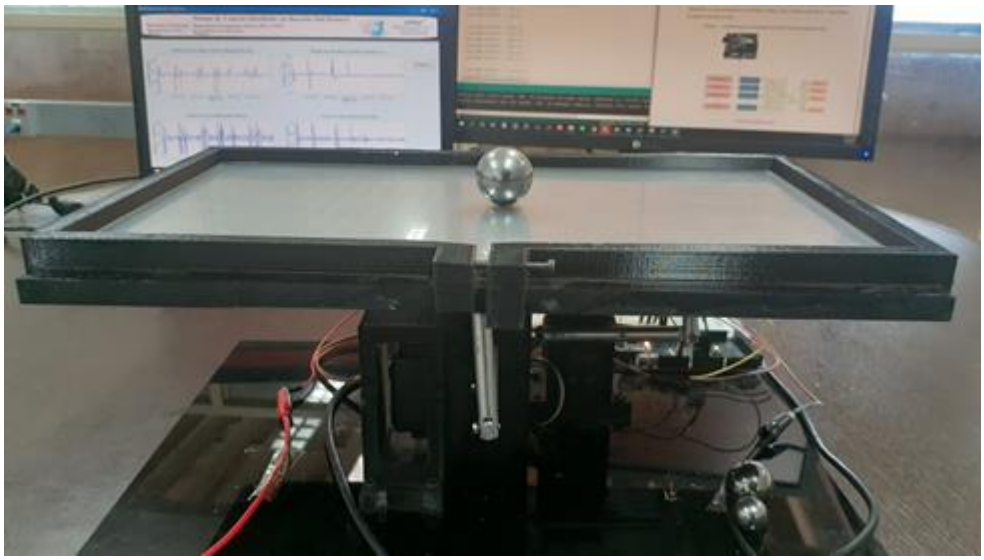
2 SISTEMA *BALL BALANCER*

Neste capítulo, o objetivo principal é a apresentação do sistema *Ball Balancer*, a modelagem matemática desse sistema, sua linearização e a estratégia de controle em cascata.

2.1 SISTEMA MECÂNICO

O sistema *Ball Balancer*, considerado neste trabalho, consiste em uma tela *touch screen* retangular sobre uma placa, que está acoplada a dois servomotores (um no eixo *x* e outro no eixo *y*), onde uma esfera pode ser colocada e movimenta-se com dois graus de liberdade (2 DOF - 2 *Degrees of Freedom*). A posição da esfera é medida através da tela *touch screen* resistiva 4 vias do fabricante Playtix®. O controle tem como objetivo de posicionar a esfera em um ponto de referência ou rastrear uma rota determinada. Na Figura 2, é apresentada uma foto do sistema *Ball Balancer* implementado no Laboratório de Pesquisa em Controle (LPC).

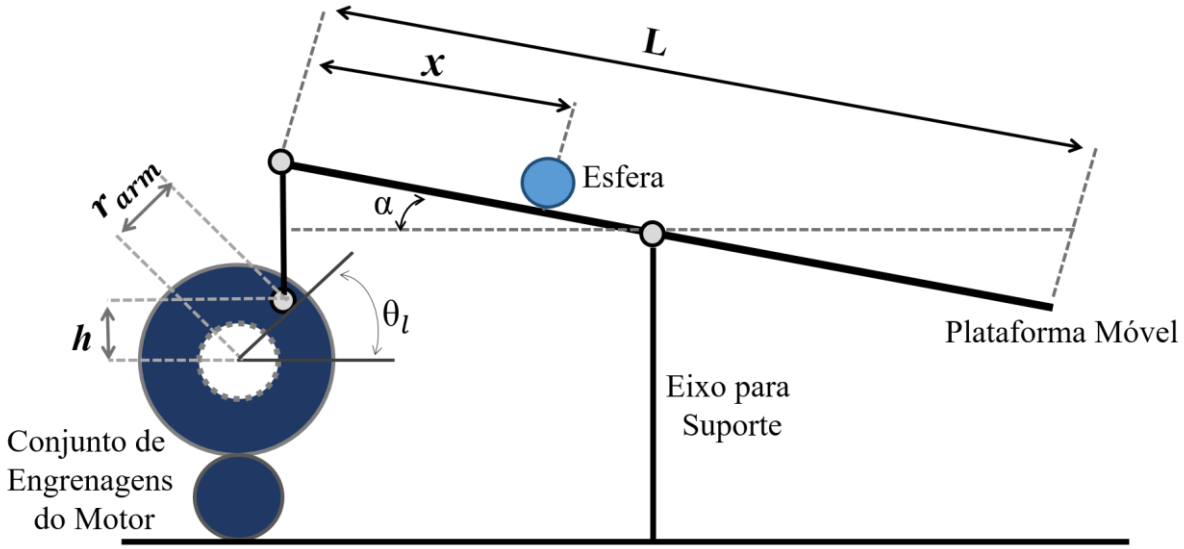
Figura 2 – Sistema *Ball Balancer* construído para a realização do trabalho.



Fonte: próprio autor.

O movimento em cada eixo do sistema *Ball Balancer* é independente do outro e, como há simetria no equipamento, é utilizado o mesmo modelo para os dois eixos. Por isto, será mostrado o modelo matemático de apenas um eixo. Na Figura 3 é apresentada a planta esquemática do movimento no eixo *x*, que ilustra o movimento em apenas um eixo do sistema *Ball Balancer*.

Figura 3 – Planta esquemática do movimento no eixo *x* e *y* do *Ball Balancer*.



Fonte: adaptado de (QUANSER, 2008).

Ao controlar a posição das engrenagens de carga do servomotor, o ângulo de inclinação da plataforma ($\alpha(t)$) pode ser ajustado para equilibrar a bola em uma posição plana deseja. O modelo matemático do sistema *Ball Balancer* será mostrado a seguir.

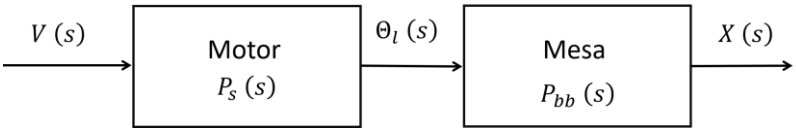
2.1.1 Modelagem do Sistema

O fabricante do servomotor HexTronik HX12K®, utilizado no módulo real, não disponibiliza sua função de transferência teórica, portanto, dadas as semelhanças entre projetos, utilizou-se como referência o modelo matemático do módulo didático 2D *Ball Balancer* da QUANSER (2008), o qual é parte integrante do Laboratório de Pesquisa em Controle.

Para a modelagem matemática deste sistema, é tomado que cada eixo é independente do outro e visto que ambos têm o mesmo funcionamento, pode-se então representar apenas um mesmo modelo matemático como definido em QUANSER (2008) para ambos.

Para se obter a função de transferência completa do sistema, primeiramente deve-se obter a função de transferência do motor e em seguida da mesa, a partir das equações de movimentos. Na Figura 4 é mostrado o diagrama de malha aberta do sistema *Ball Balancer* com o motor.

Figura 4 – Diagrama de malha aberta do sistema *Ball Balancer* e motor.



Fonte: adaptado de (QUANSER, 2008).

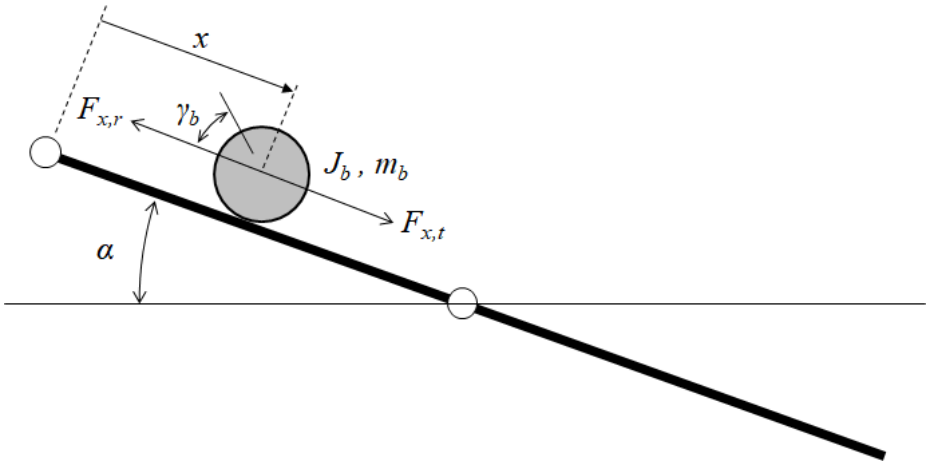
O fabricante fornece a função de transferência do motor e seus parâmetros são obtidos através do comportamento do ângulo de carga em função de uma tensão de entrada (QUANSER, 2008) conforme (1).

$$P_s(s) = \frac{K}{(\tau s + 1)s} = \frac{\Theta_l(s)}{V(s)}, \tag{1}$$

sendo *K* a constante do motor CC igual a $1{,}76\ rad/sV$ e τ igual a $0{,}0285s$.

Para obter a função de transferência da mesa estabilizadora deve-se conhecer a dinâmica do movimento da bola em função do ângulo da carga. Ao aplicar o diagrama de corpo livre na esfera, conforme Figura 5, encontra-se as equações de movimento.

Figura 5 – Dinâmica do movimento da bola.



Fonte: adaptado de (QUANSER, 2008).

Relacionando-se a segunda Lei de Newton com o somatório das forças no qual a bola está submetida chega-se em (2).

$$m_b\left(\frac{d^2}{dt^2}x(t)\right) = F_{x,t} - F_{x,r}, \tag{2}$$

sendo que m_b é a massa da bola, $F_{x,t}$ a força de translação gerada pela gravidade e $F_{x,r}$ a força inercial da bola que resiste ao giro, conforme apresentado na Figura 5.

A força causada pela gravidade é dada por (3).

$$F_{x,t} = m_b\ g\ sen(\alpha(t)). \tag{3}$$

O movimento angular da bola $\gamma_b(t)$ pode ser convertido em movimento linear, da seguinte forma, descrita em (4).

$$x(t) = \gamma_b(t)\ r_b, \tag{4}$$

sendo r_b o raio da bola.

A força inercial da bola, que resiste ao giro, pode ser escrita como (5).

$$F_{x,r} = \frac{\tau_b}{r_b}, \tag{5}$$

sendo τ_b o torque cuja equação é dada por (6).

$$\tau_b = J_b \left(\frac{d^2}{dt^2} \gamma_b(t) \right). \tag{6}$$

Assim, a força inercial da bola $F_{x,r}$ torna-se (7).

$$F_{x,r} = \frac{J_b \left(\frac{d^2}{dt^2} x(t) \right)}{r_b^2}. \tag{7}$$

Substituindo a força de translação e força inercial da bola, em (2), obtém-se a equação não linear de movimento da bola (8).

$$\frac{d^2}{dt^2} x(t) = \frac{m_b\ g\ sen(\alpha(t))\ r_b^2}{m_b\ r_b^2 + J_b}. \tag{8}$$

Equacionando os ângulos, $\theta_l(t)$ (ângulo de carga) e $\alpha(t)$(ângulo da mesa), a partir da Figura 3, pode-se relacioná-los conforme (9).

$$sen(\alpha(t)) = \frac{2\ \ sen(\theta_l(t))\ r_{arm}}{L}. \tag{9}$$

Sabendo-se que o seno de um ângulo é aproximadamente o próprio ângulo para ângulos próximos a zero, e substituindo (9) em (8), obtém-se uma nova equação não linear (10).

$$\frac{d^2}{dt^2}x(t) = \frac{2\, m_b\, g\, \theta_l(t)\, r_{arm}\, r_b^2}{L\, (m_b\, r_b^2 + J_b)}. \tag{10}$$

O momento de inércia de uma esfera sólida é dado por (11) (HALLIDAY; RESNICK; WALKER, 1996).

$$J = \frac{2\, m\, r_b^2}{5}. \tag{11}$$

Considerando para o eixo x e que o sistema (motor, mesa e bola) sempre será o mesmo, pode-se agrupar as constantes conforme (12).

$$K_{bb} = \frac{2\, m_b\, g\, r_{arm}\, r_b^2}{L\, (m_b\, r_b^2 + J_b)}. \tag{12}$$

A constante K_{bb} torna-se coeficiente de ganho do ângulo $\theta_l(t)$ e obtém-se uma equação linear do movimento (13).

$$\frac{d^2}{dt^2}x(t) = K_{bb}\theta_l(t). \tag{13}$$

Assim, a relação entre o ângulo da carga e a posição da bola no domínio da frequência é obtida aplicando-se a transformada de Laplace em (13), e considerando as condições iniciais zeros, resulta-se em (14).

$$s^2\, X(s) = K_{bb}\Theta_l(s). \tag{14}$$

Obtendo-se então, a função de transferência da mesa estabilizadora $P_{bb}(s)$, conforme (15).

$$P_{bb}(s) = \frac{K_{bb}}{s^2}. \tag{15}$$

Na Tabela 1 são mostrados os parâmetros do sistema, necessário para o cálculo da constante K_{bb}, para o projeto considerado.

Tabela 1 – Parâmetros do sistema.

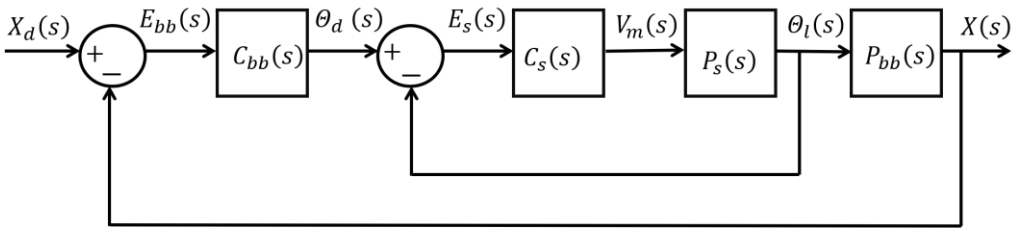
Constantes	**Descrição**	**Valor**
m_b	Massa da esfera (Kg)	0,110
r_{arm}	Distância do eixo do motor ao ponto de fixação da barra (cm)	6
r_b	Raio da bola (cm)	1,5
L	Comprimento da mesa (cm)	30

Fonte: próprio autor.

2.1.2 Controle em Cascata

Para controlar o sistema, o fabricante QUANSER® propõe o controle em cascata, conforme ilustrado na Figura 6, onde é inserido um controlador $C_s(s)$ para a posição angular do motor e outro controlador $C_{bb}(s)$ para a posição da mesa. Sendo $X(s)$ a medida real da posição da bola, $X_d(s)$ a posição desejada e $E_{bb}(s)$ o erro entre a posição desejada e real. Na malha interna tem-se $\Theta_l(s)$ o ângulo real do motor, $\Theta_d(s)$ o ângulo desejado, gerado pelo controlador da planta $C_{bb}(s)$, e $V_m(s)$ a tensão aplicada no motor para que alcance o ângulo desejado, gerado pelo erro entre o ângulo desejado e real $E_s(s)$.

Figura 6 – Diagrama do controle em cascata do *Ball Balancer*.



Fonte: adaptado de (QUANSER, 2008).

Para o projeto dos controladores deve-se determinar algumas especificações da resposta do motor e da planta. Nas Tabelas 2 e 3 encontram-se as especificações propostas pelo manual do instrutor da QUANSER (2008), sendo e_{ss} o erro de regime permanente, t_p o tempo de pico, M_P o máximo sobressinal, t_s o tempo de estabilização e C_{ts} a margem de erro do tempo de estabilização. As equações (16), (17) e (18) são utilizadas nas determinações de parâmetros do sistema, conforme consta em (DORF; BISHOP, 2001; NISE, 2013; OGATA, 1998).

Tabela 2 – Especificações do servomotor.

Sobressinal	e_{ss}	t_p
$M_P \leq 5\%$	$e_{ss} = 0(cm)$	$t_p \leq 0{,}15(s)$

Fonte: adaptado de (QUANSER, 2008).

Tabela 3 – Especificações para controle da posição da bola.

Sobressinal	$\mathbf{t_s}$	$\mathbf{c_{ts}}$	$\mathbf{e_{ss}}$
$M_P \leq 7{,}5\%$	$t_s \leq 2{,}5(s)$	$c_{ts} = 0{,}04$	$e_{ss} \leq 0{,}001(m)$

Fonte: adaptado de (QUANSER, 2008).

$$t_p = \frac{\pi}{\omega_n\sqrt{1-\xi^2}}, \tag{16}$$

$$M_P = 100e^{\left(-\frac{\pi\xi}{\sqrt{1-\xi^2}}\right)}, \tag{17}$$

$$t_s = -\frac{ln(c_{ts}\sqrt{1-\xi^2})}{\xi\omega_n}. \tag{18}$$

sendo t_p o tempo para que a resposta atinja o primeiro pico de sobressinal, M_P valor máximo de pico da curva de resposta, medido a partir da unidade, e t_s é definido como o tempo requerido para a resposta do sistema permanecer com valores no interior de uma certa faixa percentual (geralmente de 2 % ou 5 %) da amplitude de entrada (DORF; BISHOP, 2001; OGATA, 1998).

Neste capítulo foi apresentada a modelagem matemática do sistema em estudo, evidenciando que foi proposta a linearização do sistema para posterior projeto dos controladores. Contudo, é importante destacar que para a realização das simulações, o modelo real, não linear, foi utilizado, ou seja, a linearização foi usada apenas para o projeto dos controladores.

No próximo capítulo apresentam-se os projetos dos controladores PD e PID.

3 PROJETO DOS CONTROLADORES

Neste capítulo será apresentado o projeto dos controladores, primeiramente o projeto de controle do motor, e posteriormente o projeto da plataforma.

3.1 PROJETO DE CONTROLE DO MOTOR

Para o controle do motor será abordado o projeto do controlador PD, que visa ser implementado para gerar o sinal de tensão $V_m(s)$ a ser aplicada no motor. A função de transferência do controlador do motor dada em (19), foi obtida a partir do diagrama de blocos do controle do motor proposto pelo fabricante (QUANSER, 2008).

$$\frac{\Theta_l(s)}{\Theta_d(s)} = \frac{\frac{K_p K}{\tau}}{s^2 + \left(\frac{1 + K_v K}{\tau}\right) s + \frac{K_p K}{\tau}}. \tag{19}$$

Para obter os ganhos K_p e K_v do controlador, deve-se encontrar o coeficiente de amortecimento do sistema ξ, e a frequência natural do controlador ω_n, e comparar o denominador da equação (19) com a equação característica do segundo grau (20). O coeficiente de amortecimento (21) e a frequência natural (22), são obtidos manipulando a equação que representa o tempo de pico (16), equação de máximo sobressinal (17), e aplicando as especificações de projeto sugeridas pelo fabricante (QUANSER, 2008).

$$s^2 + 2\,\xi\omega_n\, s + \omega_n^2 = 0, \tag{20}$$

$$\xi = \sqrt{\frac{\ln^2(0{,}05)}{\pi^2 + \ln^2(0{,}05)}} = 0{,}690, \tag{21}$$

$$\omega_n = \frac{\pi}{0{,}15\sqrt{1 - (0{,}69)^2}} = 28{,}9\ (rad/s). \tag{22}$$

Assim, com os valores das equações (21) e (22) juntamente com as constantes K e τ substituídos nas equações (23) e (24), obtidas da comparação entre (19) e (20), encontra-se os valores dos ganhos proporcional e derivativo para controle do motor.

$$K_p = \frac{\omega_n^2 \tau}{K} = 13{,}562\ (V/rad), \tag{23}$$

$$K_v = \frac{\tau}{K}\left(2\xi\omega_n - \frac{1}{\tau}\right) = 0{,}078624\ (V.s/rad). \tag{24}$$

3.2 PROJETO DE CONTROLE DA PLATAFORMA *BALL BALANCER*

No controle da plataforma será abordado o projeto do controlador PID. A partir do diagrama de blocos do controlador proposto pelo fornecedor (QUANSER, 2008), obtém-se (25) de $\Theta_d(s)$ em função das posições desejada $X_d(s)$ e posição real da bola $X(s)$.

$$\begin{aligned}\Theta_d(s) = K_{p,bb}\big(X_d(s) - X(s)\big) + \frac{K_{i,bb}}{s}\big(X_d(s) - X(s)\big) \\ + K_{d,bb}\, s\, (b_{sd}X_d(s) - X(s)).\end{aligned} \tag{25}$$

Como o controle de posição do servomotor na malha interna já está completa, a dinâmica do motor pode ser considerada desprezível, assim, assume-se que o ângulo desejado θ_d é igual ao ângulo de carga θ_l. Substituindo $\Theta_d(s)$ em (14), obtém-se a equação (26).

$$\begin{aligned}s^2 X(s) = K_{bb}\Theta_l(s) = K_{bb}\Theta_d(s), \\ \Theta_d(s) = \frac{s^2\, X(s)}{K_{bb}}.\end{aligned} \tag{26}$$

Substituindo (26) em (25) e fazendo as devidas manipulações, obtém-se a função de transferência entre a entrada de referência $X_d(s)$, e o sinal de realimentação do sistema X(s).

$$\frac{X(s)}{X_d(s)} = \frac{s^2 b_{sd}K_{d,bb}K_{bb} + sK_{p,bb}K_{bb} + K_{i,bb}\, K_{bb}}{s^3 + s^2 K_{d,bb}K_{bb} + sK_{p,bb}K_{bb} + K_{i,b}K_{bb}}. \tag{27}$$

A equação característica (27) do sistema em malha fechada é de terceira ordem, e quando comparada com a equação característica de terceira ordem na forma expandida (28), pode ser encontrado os ganhos do controlador, conforme as equações (29) (30) e (31).

$$s^3 + (2\xi\omega_n + p_0)s^2 + (2\xi\omega_n p_0 + \omega_n^2)s + \omega_n^2 p_0 = 0, \tag{28}$$

$$K_{p,bb} = \frac{(2\xi\omega_n p_0 + \omega_n^2)}{K_{bb}}, \tag{29}$$

$$K_{d,bb} = \frac{(2\xi\omega_n + p_0)}{K_{bb}}, \tag{30}$$

$$K_{i,bb} = \frac{\omega_n^2 p_0}{K_{bb}}. \tag{31}$$

Com as especificações propostas pelo fabricante (QUANSER, 2008), aplicadas nas equações (17) e (18) devidamente manipuladas, obtém-se os valores da constante de amortecimento ξ e a frequência ω_n da mesa *Ball Balancer*.

$$\xi = \sqrt{\frac{\ln^2(0{,}075)}{\pi^2 + \ln^2(0{,}075)}} = 0{,}63615, \tag{32}$$

$$\omega_n = \frac{-\ln(c_{ts}\sqrt{1-\xi^2})}{\xi t_s} = 2{,}187037\ (rad/s). \tag{33}$$

Para o controle PD, o valor de p_0 é igual a zero, e os valores encontrados para as especificações propostas pelo manual da QUANSER® encontra-se a seguir nas equações de (34) à (36).

$$K_{p,bb} = \frac{(2\xi\omega_n p_0 + \omega_n^2)}{K_{bb}} = 1{,}7082\ (rad/m), \tag{34}$$

$$K_{i,bb} = \frac{\omega_n^2 p_0}{K_{bb}} = 0, \tag{35}$$

$$K_{d,bb} = \frac{(2\xi\omega_n + p_0)}{K_{bb}} = 0{,}99378\ (rad\,.s/m). \tag{36}$$

No controlador PID, o polo responsável pela constante de decaimento p_0 não pode ser zero, portanto, é necessário calcular seu valor. Sendo T_p a constante de decaimento do polo, e adotando-se T_p igual a um segundo em (37), tem-se (37).

$$p_0 = \frac{1}{T_p} = 1. \tag{37}$$

Portanto, com os valores das constantes ξ e ω_n disponibilizados nas equações (32) e (33), e considerando (37), substituídos nas equações (29), (30) e (31), encontra-se os valores dos ganhos do controlador PID.

$$K_{p,bb} = \frac{(2\xi\omega_n p_0 + \omega_n^2)}{K_{bb}} = 2{,}702\ (rad/m), \tag{38}$$

$$K_{i,bb} = \frac{\omega_n^2 p_0}{K_{bb}} = 1{,}7082(rad/m\ s), \tag{39}$$

$$K_{d,bb} = \frac{(2\xi\omega_n + p_0)}{K_{bb}} = 1{,}3509\ (rad.s/m). \tag{40}$$

Os ganhos obtidos nos projetos, destacados nesse capítulo, serão utilizados nas simulações do controle de apenas um sistema *Ball Balancer*, conforme será apresentado no Capítulo 4.

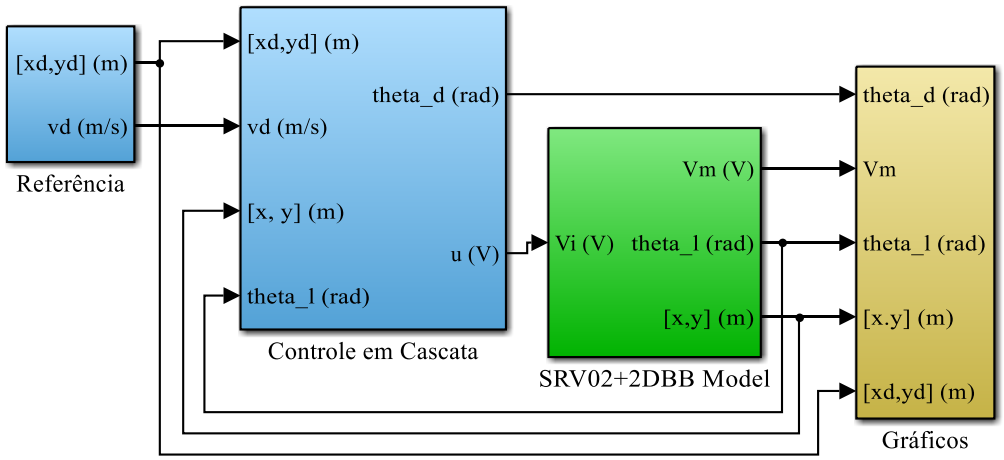
4 RESULTADOS DO CONTROLE INDIVIDUAL DA PLANTA *BALL BALANCER*

Neste capítulo, apresentam-se os resultados de simulação, obtidos para as técnicas de controle PD e PID do controle individual da planta. O objetivo principal deste estudo, em nível de simulação, é verificar as dificuldades e possibilidades de implementação de tais técnicas em um microcontrolador.

4.1 CONTROLE INDIVIDUAL DA PLANTA COM CONTROLADOR PD

Para simular o controlador PD, foi executado um programa no Matlab®, juntamente com o diagrama principal de blocos no Simulink®. Na Figura 7 é apresentado o diagrama principal com os subsistemas: referência, controle em cascata, modelo não linear da planta e gráficos.

Figura 7 – Diagrama de blocos principal para simulação dos controladores PD e PID para planta individual.



Fonte: próprio autor.

Os dados inseridos para simulação, encontram-se na Tabela 1, que são parâmetros referente a planta utilizada no projeto. Na Tabela 4 são mostrados o valor da constante K_{bb}, e os ganhos encontrados para o motor, e *Ball Balancer*. É importante salientar, que as simulações foram realizadas em condições ideais, ou seja, não foram levados em consideração, possíveis distúrbios e incertezas que possam existir na bancada.

Os resultados apresentados são referentes aos requerimentos de projeto sugeridos pelo manual da QUANSER®. Observa-se que na Figura 8, os resultados de simulação para

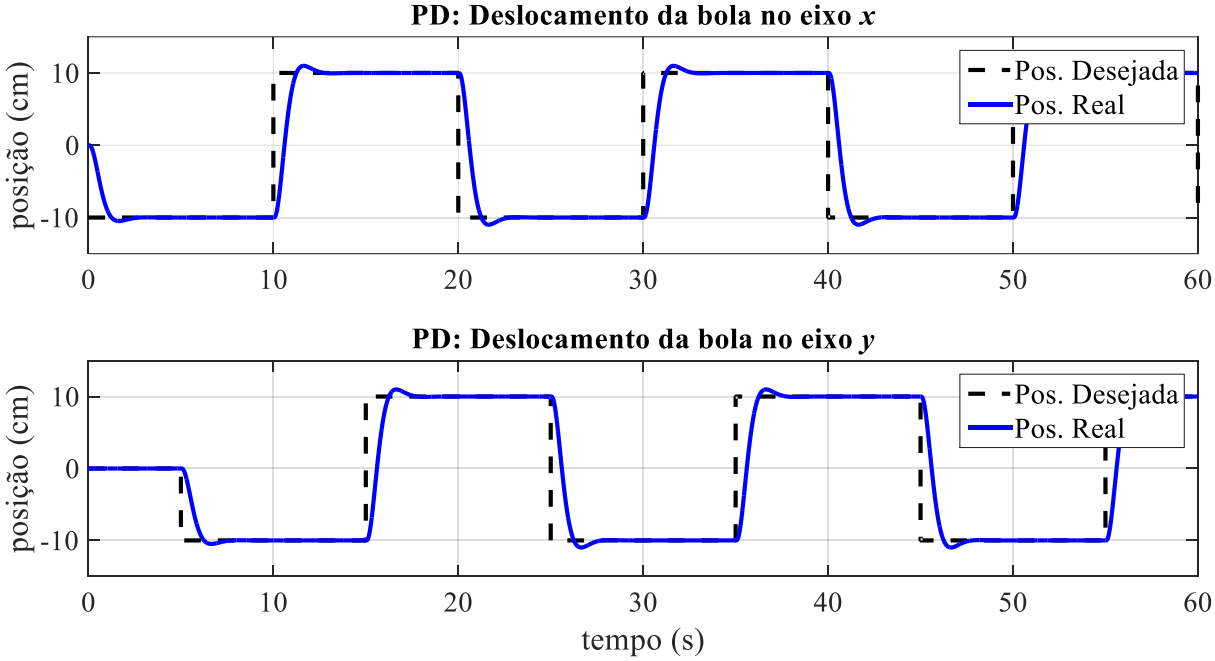
referência em onda quadrada, do controle de posição das coordenadas x e y apresentados, atendem as exigências de projeto.

Tabela 4 – Ganhos do controlador PD para uma planta.

Ganhos	**Descrição**	**Eixo (x, y)**
K_{bb}	Constante da planta $(m/s^2/rad)$	2,8
K_p	Ganho K_p do motor (V/rad)	13,562
K_v	Ganho K_v do motor $(V.s/rad)$	0,0786
$K_{p,bb}$	Ganho K_p do *Ball Balancer* (rad/m)	1,7082
$K_{i,bb}$	Ganho K_i do *Ball Balancer* $(rad/m/s)$	0
$K_{d,bb}$	Ganho K_d do *Ball Balancer* $(rad.s/m)$	0,9937

Fonte: próprio autor.

Figura 8 – Resultado de simulação do controlador PD, do rastreamento de posição, no eixo *x* e *y* para referência em preto.

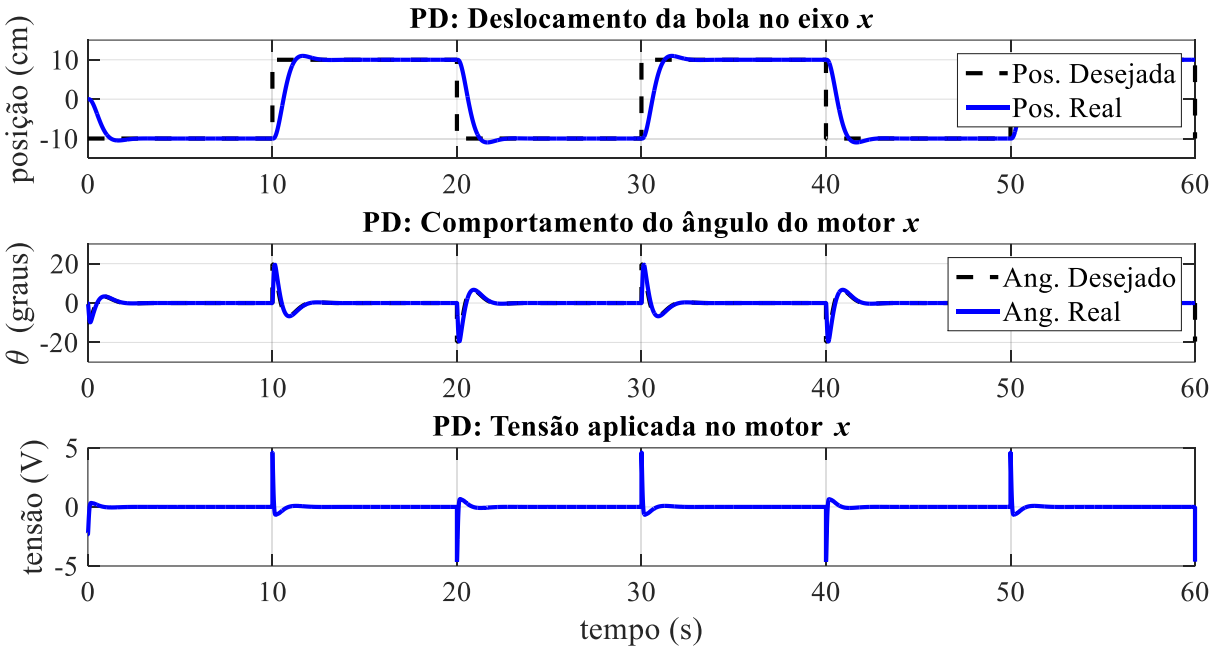


Fonte: próprio autor.

A malha externa de controle, gera o *setpoint* de posição angular do motor para a malha interna de controle. A malha interna é responsável pelo controle de posição angular do motor, e no projeto foram escolhidos ganhos que a deixasse mais rápida que a malha externa, responsável pelo controle de posição da bola sobre a mesa. Nota-se que a resposta da malha interna ficou realmente rápida, sendo que o ângulo real está praticamente sobreposto ao ângulo

desejado, como pode ser observado na Figura 9 (comportamento do ângulo relacionado com o eixo x); o mesmo acontece com o eixo y.

Figura 9 – Resultado de simulação do controlador PD, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência em preto.



Fonte: próprio autor.

Os sinais de controle, isto é, a tensão aplicada em cada motor, é apresentado na Figura 9. Observa-se que o nível de tensão, mesmo desconsiderando saturação, para o projeto do fabricante QUANSER®, não ultrapassou $5\ Volts$.

4.2 CONTROLE INDIVIDUAL DA PLANTA COM CONTROLADOR PID

Para simular o controlador PID, foi executado um programa no Matlab®, juntamente com o diagrama principal de blocos no Simulink®. Foi utilizado o mesmo bloco do Simulink® apresentado na Figura 6, com o diagrama principal com os subsistemas: referência, controle em cascata, modelo não linear da planta e gráficos. Assim como no controlador PD, foi executado o código no Matlab®, e obteve-se o resultado para o controle PID considerando as especificações de projeto do fabricante QUANSER®. Na Tabela 5 são mostrados o valor da constante K_{bb}, e os ganhos encontrados para o motor e *Ball Balancer*, para ambos os eixos.

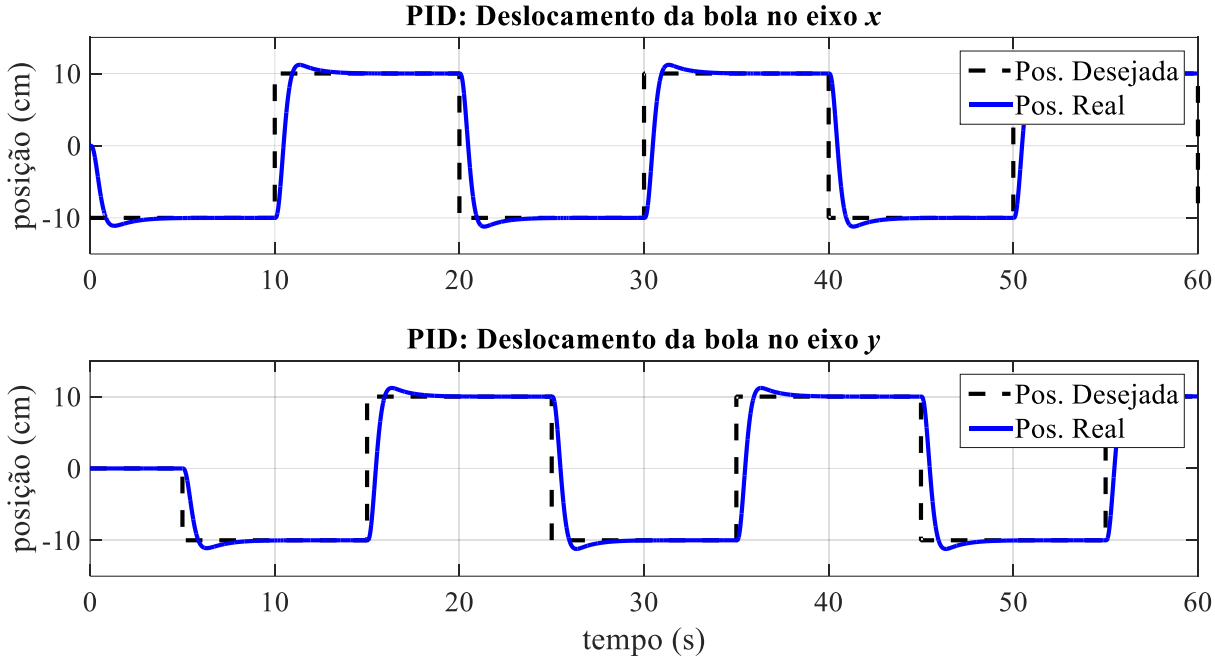
Na Figura 10 são mostradas as respostas da posição da bola em cada eixo, para o controle PID. Nota-se que o tempo de subida, está mais rápido, se comparado com o controlador PD.

Tabela 5 – Ganhos do controlador PID para uma planta.

Ganhos	**Descrição**	**Eixo (*x*, *y*)**
K_{bb}	Constante da planta $(m/s^2/rad)$	2,8
K_p	Ganho K_p do motor (V/rad)	13,562
K_v	Ganho K_v do motor $(V.s/rad)$	0,0786
$K_{p,bb}$	Ganho K_p do *Ball Balancer* (rad/m)	2,702
$K_{i,bb}$	Ganho K_i do *Ball Balancer* $(rad/m/s)$	1,7082
$K_{d,bb}$	Ganho K_d do *Ball Balancer* $(rad.s/m)$	1,3509

Fonte: próprio autor.

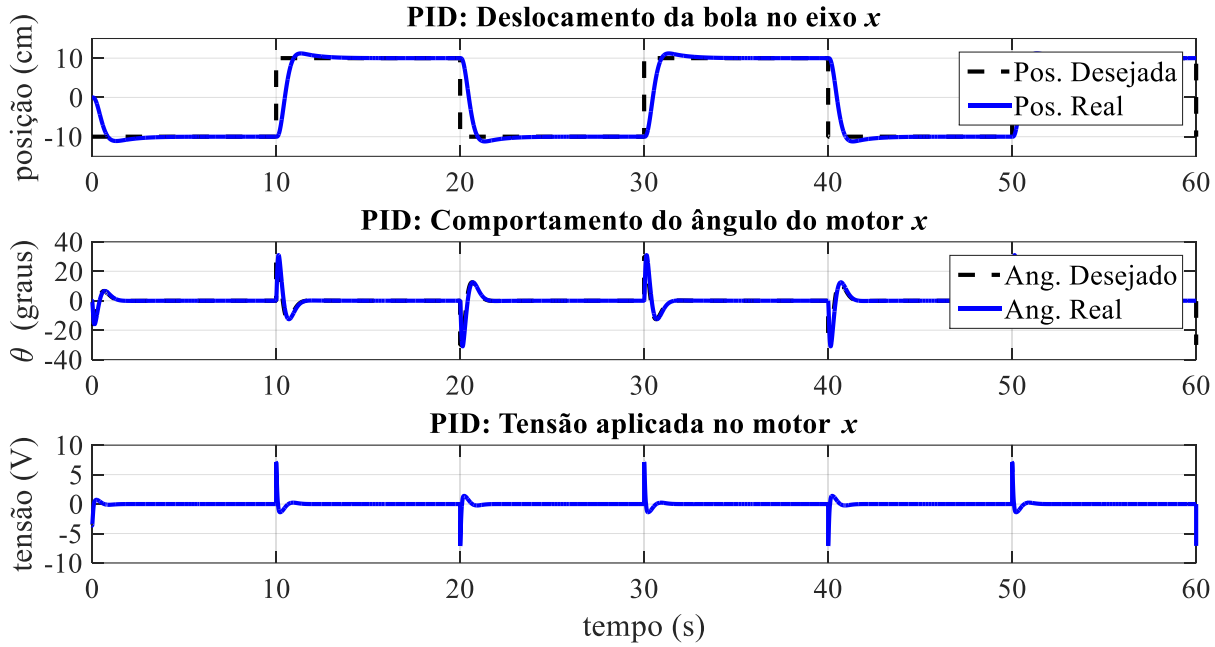
Figura 10 – Resultado de simulação do controlador PID, do rastreamento de posição, no eixo *x* e *y* para referência em preto.



Fonte: próprio autor.

Observa-se que a resposta de malha interna, apresentada na Figura 11, apresenta uma rápida resposta, mostrando que a especificação do projeto foi satisfeita, contudo, nota-se que o ângulo do motor, varia mais que no caso do controlador PD. É possível notar ainda, que existe um maior esforço de controle para técnica PID, se comparado com a técnica PD.

Figura 11 – Resultado de simulação do controlador PID, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência em preto.



Fonte: próprio autor.

Nesse capítulo, o controle individual de um sistema *Ball Balancer* foi estudado. Os resultados demonstraram que o controle individual está adequado aos requisitos exigidos em projeto. Na proposta deste trabalho, pretende-se realizar o controle de mais que um sistema, contudo, o sinal de controle deve estar presente em apenas um sistema por vez.

O sistema que receberá o sinal de controle deverá ser selecionado através de um índice de desempenho, que mostrará qual o sistema que se encontra em pior situação, e este receberá o sinal de controle. Com essa estratégia, pretende-se diminuir o tráfego de informações em uma rede e, ao mesmo tempo, manter toda a planta (com vários sistemas) estável. No Capítulo 5 apresenta-se a proposta e a simulação do controle distribuído com controle PID com duas plantas controladas.

5 CONTROLE DISTRIBUÍDO EM PLANTAS *BALL BALANCER*

A proposta deste estudo é a diminuição do tráfico de dados numa rede e, por isso, foi proposto a utilização de um controlador que envia dados (sinal de controle) para apenas um sistema por vez. A decisão do sistema que receberá o sinal de controle é a maior contribuição deste trabalho e ela pode ser observada nos quadros Desempenho e Decisão da Figura 12.

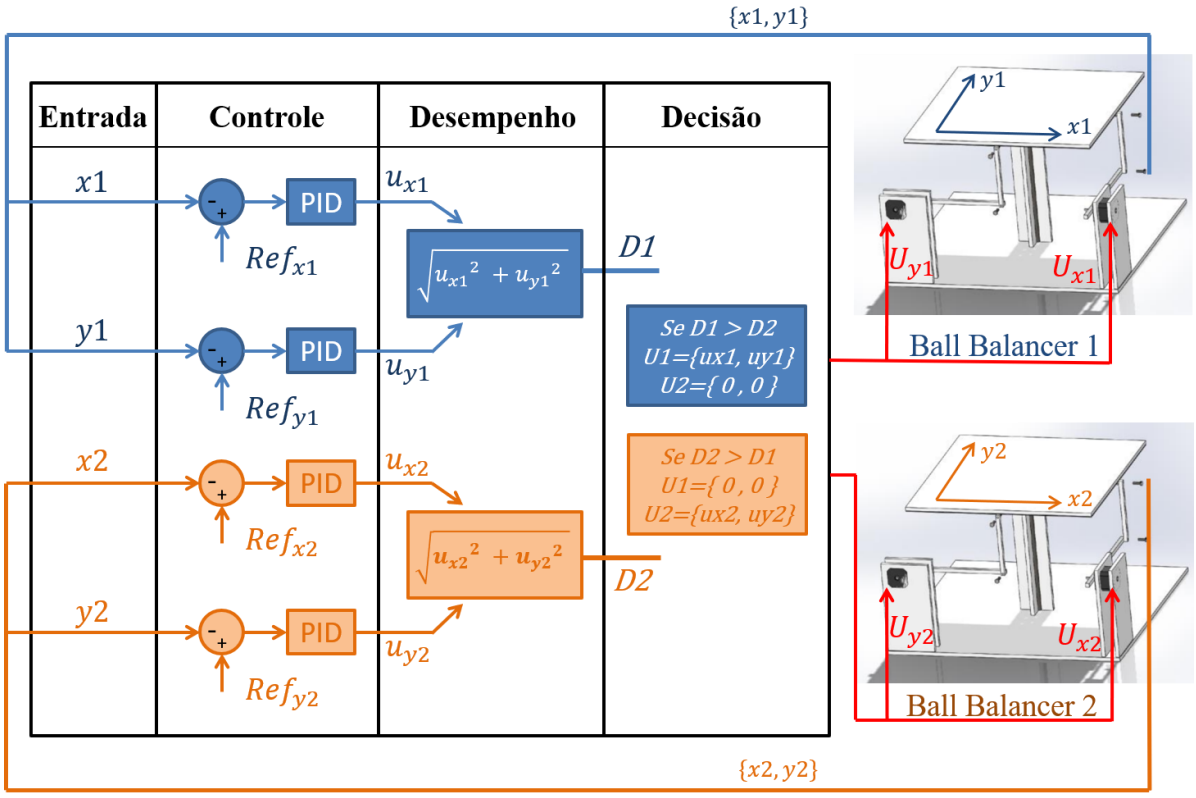
Neste capítulo, apresenta-se a estratégia de controle considerada, bem como os resultados de simulação, obtidos para as técnicas de controle PID para duas plantas. Também, para verificar, através de comparação de resultados, o bom desempenho da estratégia proposta, foram realizadas simulações em que o sinal de controle é entregue para um sistema por vez, contudo, sem a tomada de decisão baseada na proposta deste trabalho, ou seja, cada sistema receberá o sinal de controle dentro de um intervalo de tempo, que não leva em consideração o índice de desempenho proposto neste trabalho.

5.1 ESTRATÉGIA PARA O SELETOR DE CONTROLE

Para obter o controle cooperativo das plantas, foi considerado uma estratégia para o índice de desempenho do chaveamento do controlador, com a ideia de chaveamento de controle em cada instante de amostragem, para a planta que está em pior condição em relação ao ponto de equilíbrio dos estados. Para avaliar isso, considerou-se que o pior caso é a planta que tem o maior sinal de controle, ou seja, a maior resposta ao controle (PID). Na Figura 12 é apresentado um diagrama para o controle de duas plantas *Ball Balancer*, no qual o controle é aplicado de acordo com o ponto de decisão, nota-se que essa estratégia é para controlar o sistema que apresenta o pior desempenho, ou seja, que possui a maior norma $\sqrt{u_x^2 + u_y^2}$.

Naturalmente analisa-se o erro, porém, o sistema que tem o menor erro não significa que está em pior situação. Por exemplo, uma planta com erro zero de posição, mas que está passando sobre a referência (estática) com velocidade alta; outra planta com um erro pequeno, mas parada (estável) sobre esse erro, então, dependendo da velocidade da planta sobre a referência, o pior caso é a que está em movimento. Assim, a ideia foi chavear para o controlador que gera o maior sinal de controle.

Figura 12 – Estratégia de controle distribuído para dois sistemas *Ball Balancer*.



Fonte: próprio autor.

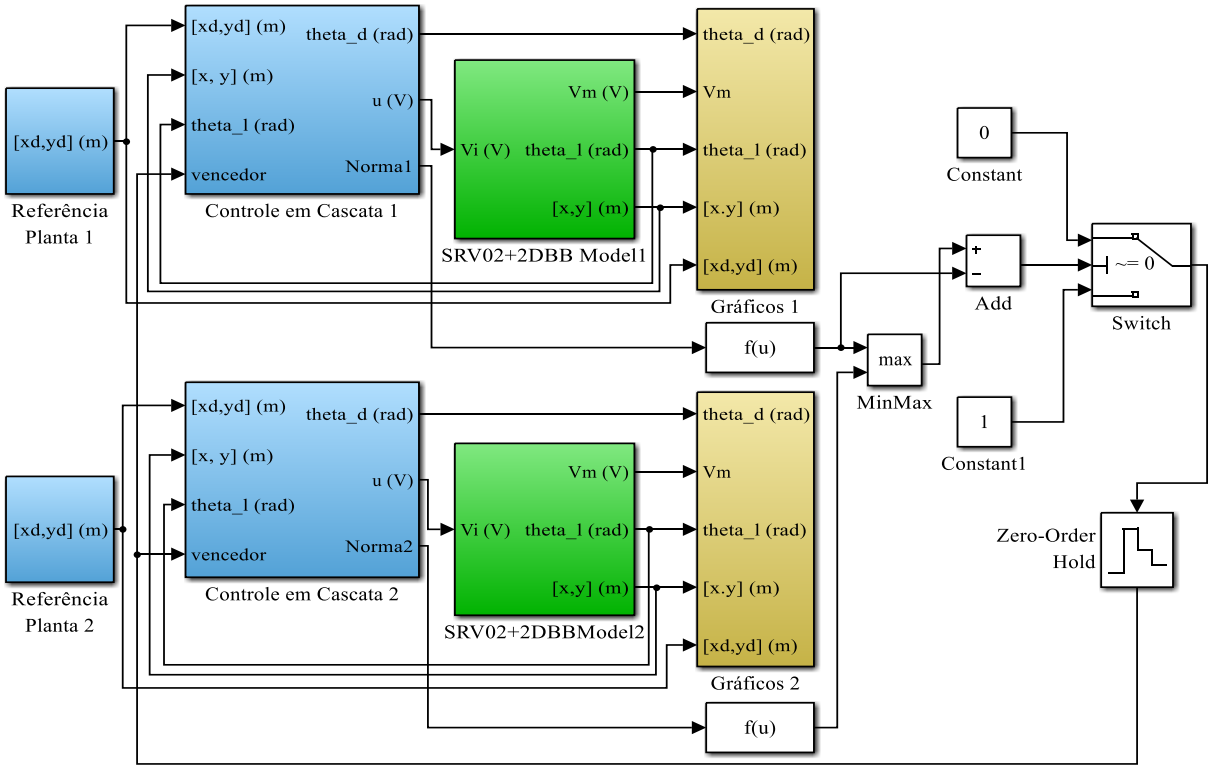
5.2 CONTROLE DISTRIBUÍDO DE DUAS PLANTAS COM CONTROLE PID

Para simular o controlador PID, para dois sistemas, foi executado um programa no Matlab®, juntamente com o diagrama principal de blocos no Simulink®. Na Figura 13 apresenta-se o diagrama principal com os subsistemas: referências, controle em cascata, modelo não linear da planta, gráficos de ambos os sistemas, e chaveamento para o controle.

Na Figura 13, utiliza-se os blocos de máximo (*MinMax*), subtrator (*Add*) e seletor (*Switch*) para a seleção do vencedor, caso o sinal na saída do subtrator for zero, então é escolhido pelo critério considerado no bloco de seleção (*Switch*), a constante zero, assim, o vencedor na entrada dos blocos de controle (bloco de controle em cascata), recebe zero, caso contrário recebe um. O bloco retenção de ordem zero (*Zero-Order Hold*), mantém sua entrada constante para o período de amostragem especificado (utilizou-se o padrão de $100\,ms$), e conforme a decisão de qual planta apresenta a maior norma em um determinado instante de amostragem, é encaminhado o sinal de decisão para as plantas, e apenas a planta de maior norma naquele período de amostragem entenderá a decisão de ser controlada. Na Figura 14 é apresentado o subsistema do controle em cascata referente ao primeiro sistema *Ball Balancer*,

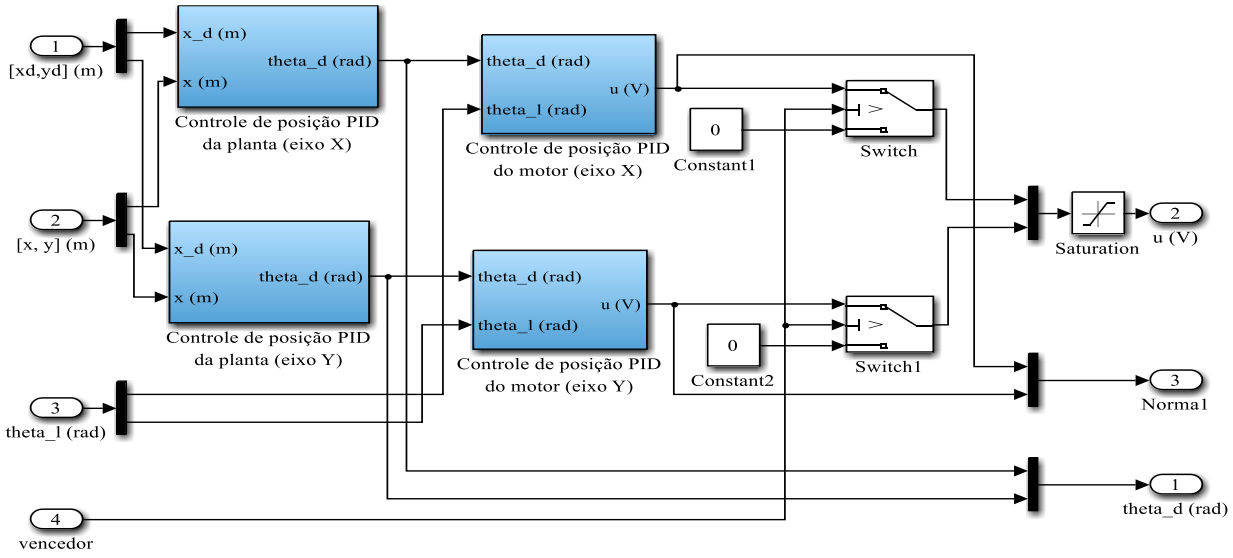
com a entrada do sinal vencedor para o sistema 1, se o vencedor receber sinal um, então o sistema 1 será controlado.

Figura 13 – Diagrama de blocos principal para simulação do controlador PID para dois sistemas *Ball Balancer*.



Fonte: próprio autor.

Figura 14 – Subsistema do controle em cascata da primeira planta, com a entrada do sinal vencedor para simulação do controlador PID.



Fonte: próprio autor.

O resultado para o controle PID, foi obtido considerando as especificações de projeto do fabricante QUANSER®, conforme consta nas Tabelas 2 e 3. Na Tabela 6 são mostrados os ganhos obtidos para o controle distribuído.

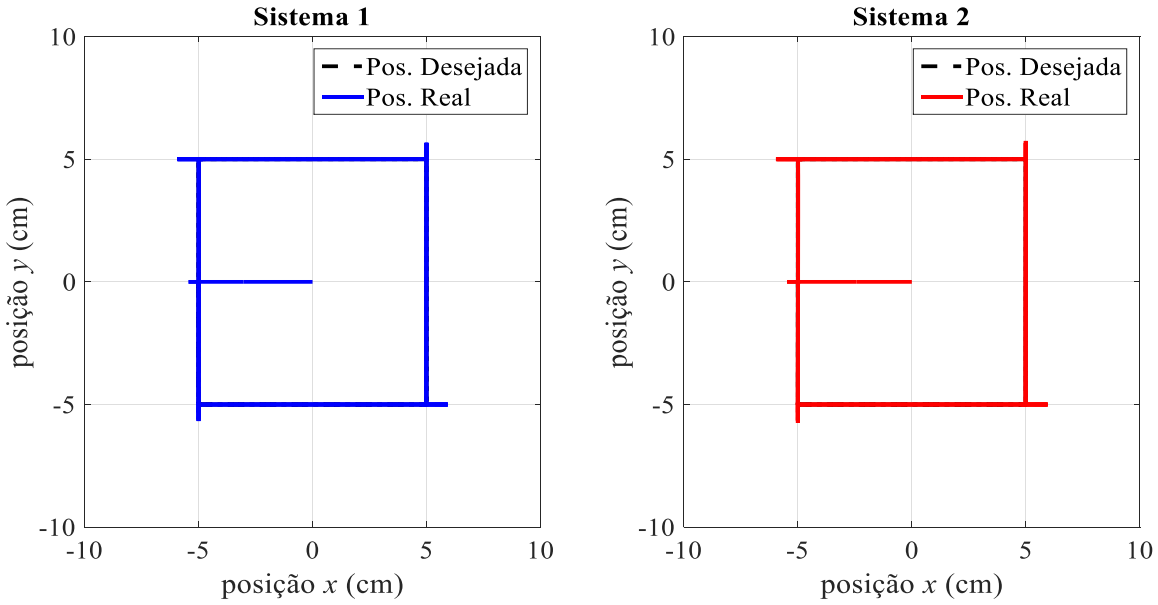
Tabela 6 – Ganhos para o controle PID para dois sistemas *Ball Balancer*.

Ganhos	**Descrição**	**Eixo (x, y)**
K_{bb}	Constante da planta $(m/s^2/rad)$	1,29
K_p	Ganho K_p do motor (V/rad)	13,6
K_v	Ganho K_v do motor $(V.s/rad)$	$-0{,}00612$
$K_{p,bb}$	Ganho K_p do *Ball Balancer* (rad/m)	3,7
$K_{i,bb}$	Ganho K_i do *Ball Balancer* $(rad/m/s)$	0
$K_{d,bb}$	Ganho K_d do *Ball Balancer* $(rad.s/m)$	2,15

Fonte: próprio autor.

Na Figura 15 é apresentado o resultado de simulação do controle distribuído de dois sistemas *Ball Balancer*, para o rastreamento em onda quadrada (referência em preto), apresenta-se o controle de posição das coordenadas x e y de ambos os sistemas.

Figura 15 – Resultado de simulação do controlador PID para rastreamento da posição da bola nos eixos *x* e *y* de dois sistemas *Ball Balancer*, para referência quadrada.

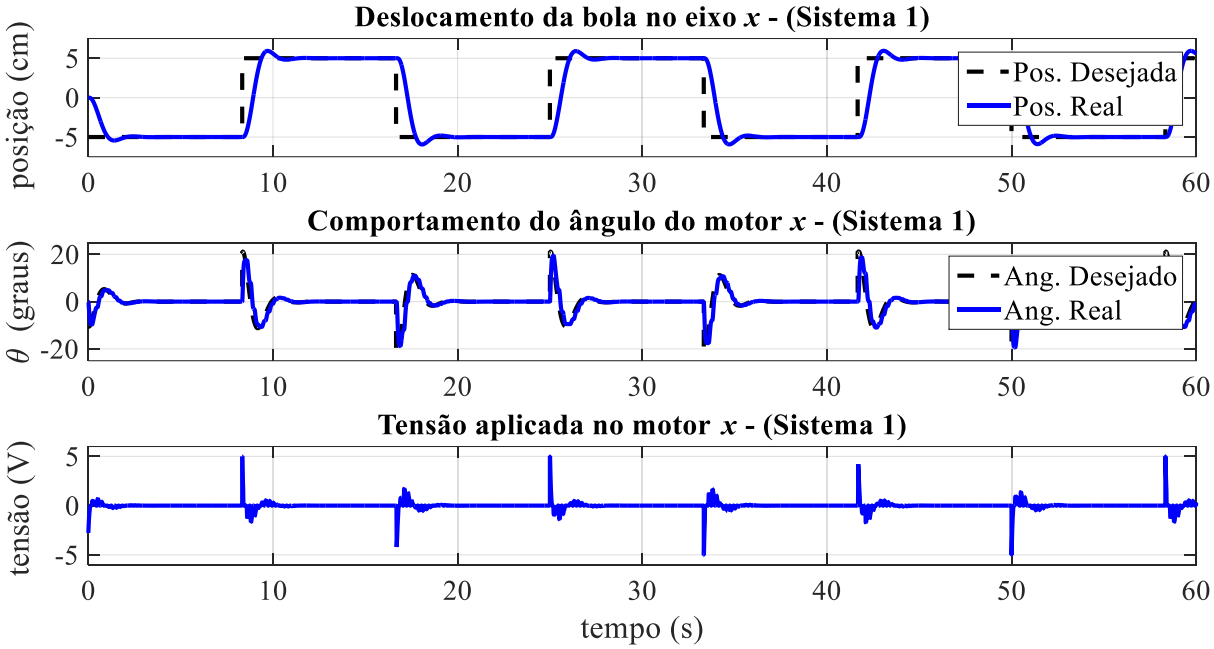


Fonte: próprio autor.

Observa-se que nas Figuras 16 a 17, os resultados de simulação do controle distribuído para dois sistemas *Ball Balancer*, para o rastreamento da referência quadrada, o controle de

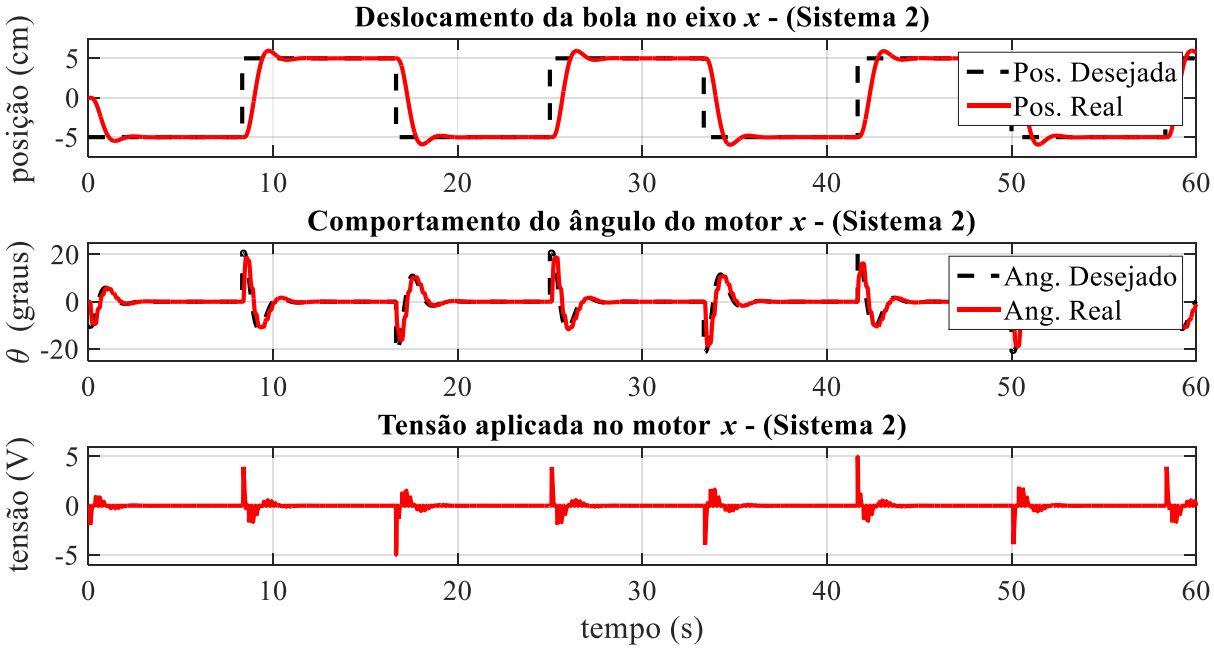
posição das coordenadas *x* e *y* de ambos os sistemas, está adequado aos requisitos exigidos em projeto. Nota-se também que a resposta de malha interna apresenta uma rápida resposta.

Figura 16 – Resultado de simulação do controlador PID do primeiro sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



Fonte: próprio autor.

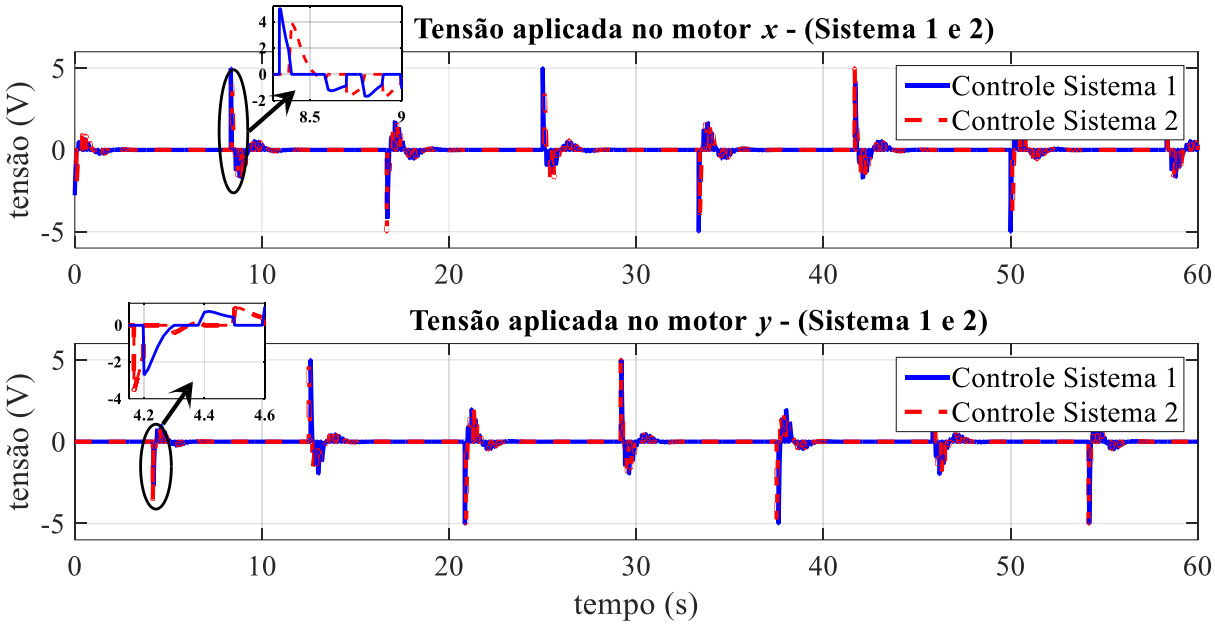
Figura 17 – Resultado de simulação do controlador PID do segundo sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



Fonte: próprio autor.

Pode-se observar a parte destacada na Figura 18 para a referência quadrada, a natureza cooperativa do controle, sendo que o esforço de controle da primeira e segunda planta, não sofrem alterações simultâneas, ou seja, quando o sinal de controle referente a primeira planta é atualizado, o sinal de controle da segunda planta, permanece constante em zero e vice-versa.

Figura 18 – Resultado de simulação do controlador PID para o controle distribuído de dois sistemas *Ball Balancer* para referência quadrada.



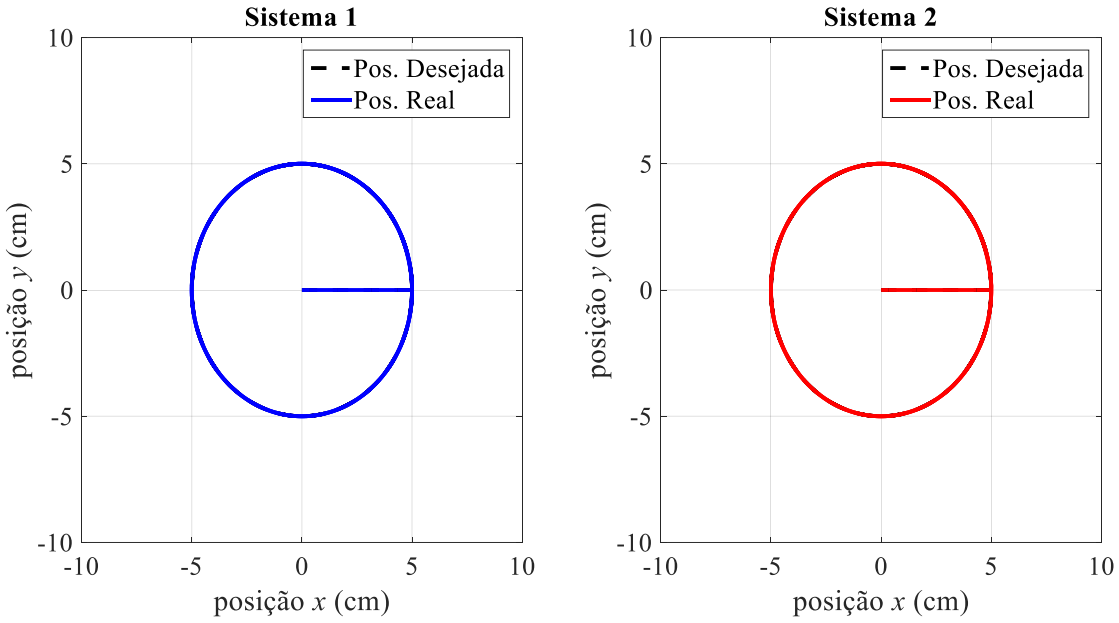
Fonte: próprio autor.

Na Figura 19 é apresentado o resultado de simulação do controle distribuído de duas plantas *Ball Balancer*, considerando o rastreamento em onda senoidal (referência em preto), apresenta-se o controle de posição das coordenadas x e y de ambos os sistemas.

Nota-se que nas Figuras 20 a 21, os resultados de simulação do controle distribuído para os dois sistemas, para o rastreamento da referência senoidal, o controle de posição das coordenadas x e y de ambos os sistemas, está adequado aos requisitos exigidos em projeto. Observa-se também que a resposta de malha interna apresenta uma rápida resposta confirmando que a especificação do projeto foi atendida.

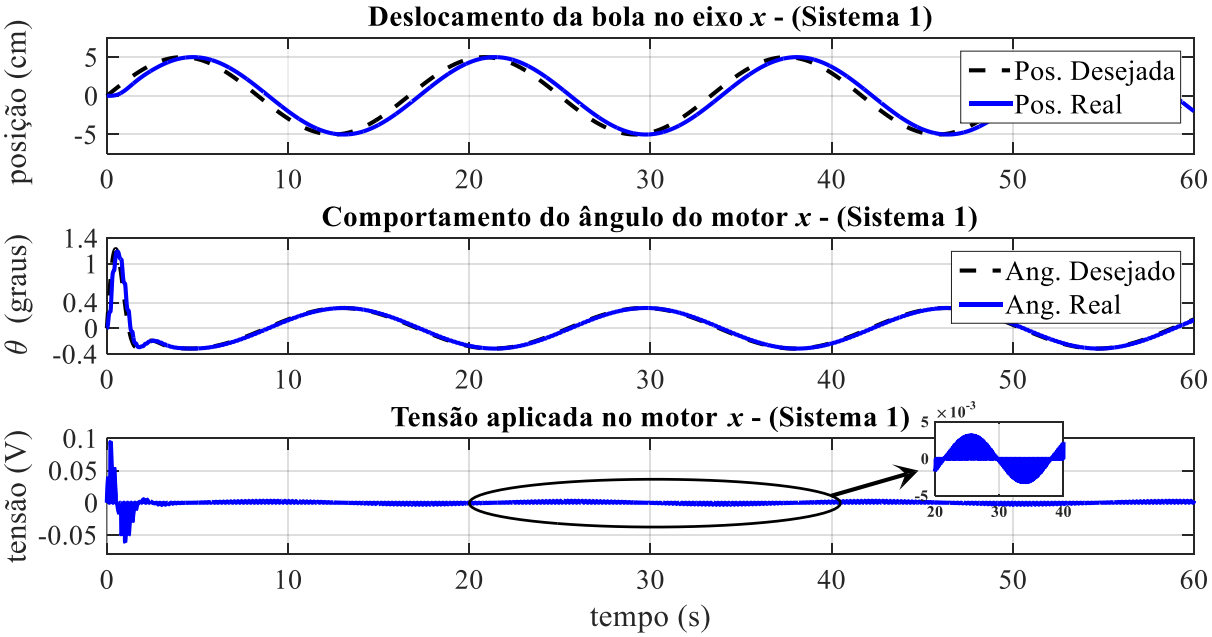
Pode-se observar também a natureza cooperativa do controle na parte destacada na Figura 22, sendo que o esforço de controle do primeiro e segundo *Ball Balancer*, não sofrem alterações ao mesmo tempo, ou seja, quando o sinal de controle referente ao primeiro sistema é atualizado, o sinal de controle do segundo sistema, permanece constante em zero e vice-versa.

Figura 19 – Resultado de simulação do controlador PID para rastreamento da posição da bola nos eixos *x* e *y* de dois sistemas *Ball Balancer*, para referência senoidal.



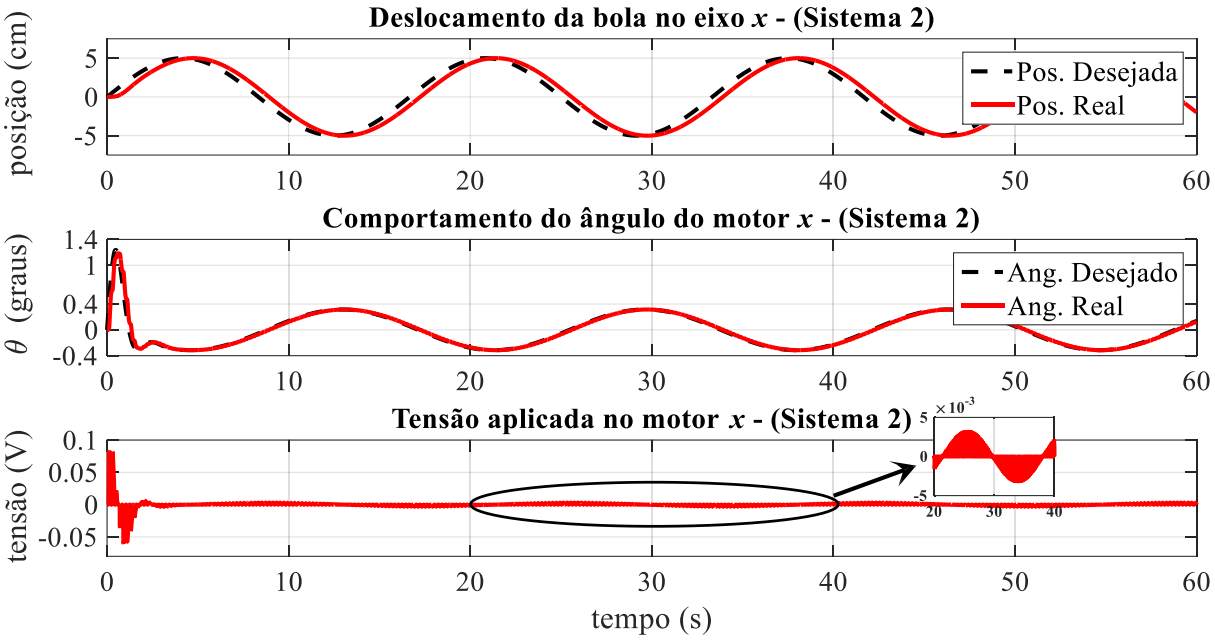
Fonte: próprio autor.

Figura 20 – Resultado de simulação do controlador PID do primeiro sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



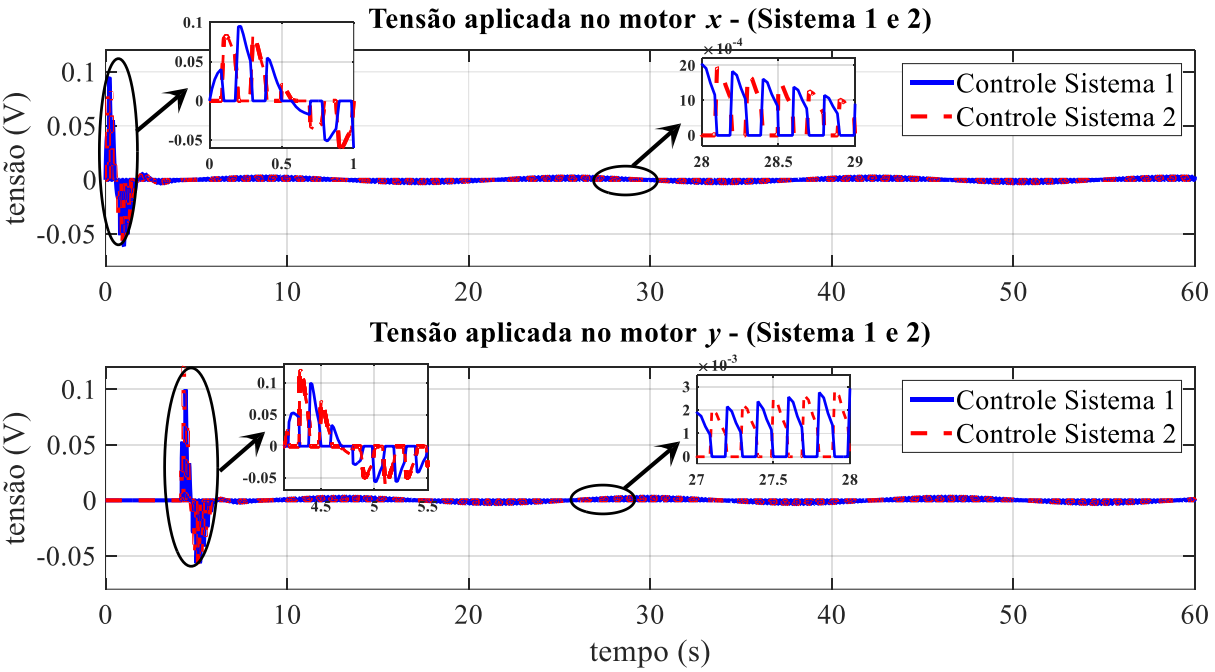
Fonte: próprio autor.

Figura 21 – Resultado de simulação do controlador PID do segundo sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



Fonte: próprio autor.

Figura 22 – Resultado de simulação do controlador PID para o controle distribuído de dois sistemas *Ball Balancer* para referência senoidal.



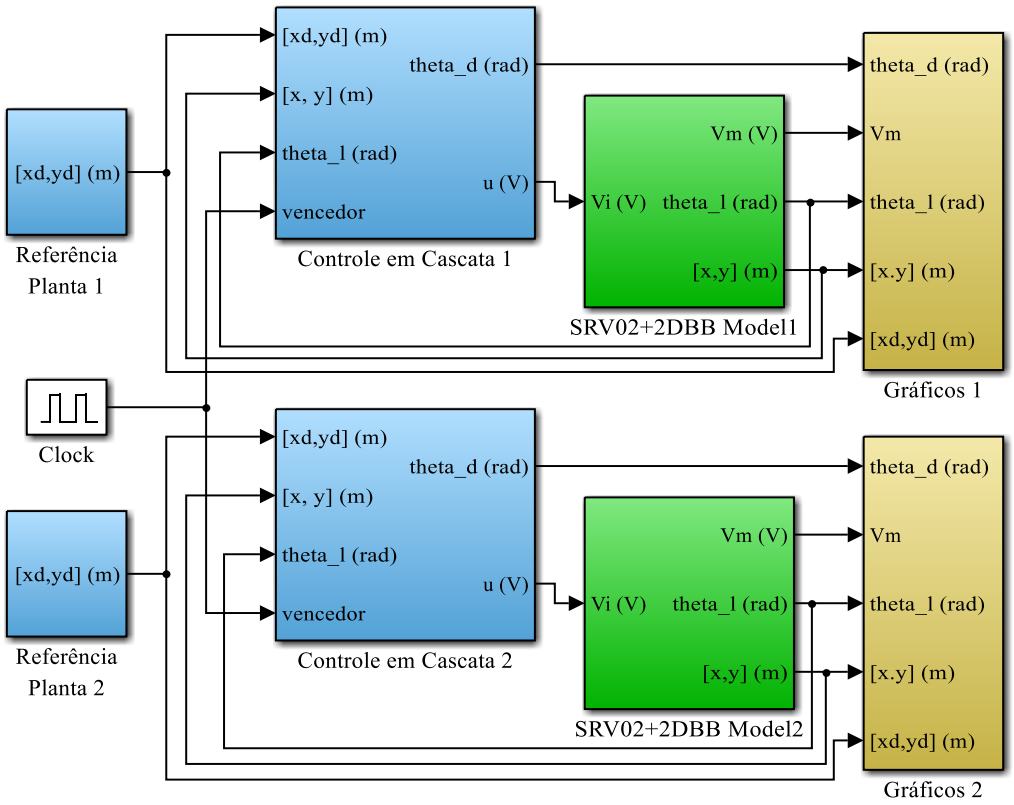
Fonte: próprio autor.

5.3 CONTROLE PID DE DUAS PLANTAS SEM O SELETOR DE CONTROLE

Para simular o controlador PID, para dois sistemas *Ball Balancer* sem o seletor de controle, foi executado um programa no Matlab®, juntamente com o diagrama principal de blocos no Simulink®. Na Figura 23 apresenta-se o diagrama principal com os subsistemas: referências, controle em cascata, modelo não linear da planta e gráficos das duas plantas. Nessa estrutura considerou-se o controle das duas plantas sem o seletor de controle, fazendo com que cada controle atue durante um determinado período, ou seja, é um chaveamento arbitrário. O resultado para o controle PID, foi obtido considerando as especificações de projeto do fabricante QUANSER®, conforme consta nas Tabelas 2 e 3.

Para a simulação com as duas plantas sem o seletor de controle, foi considerado primeiramente que um *Ball Balancer* será controlado por 400 ms (utilizou-se o bloco clock para gerar o sinal de clock para alternar os controladores durante um período T_i), enquanto que o outro fica sem o sinal de controle; depois a condição se inverte de maneira periódica.

Figura 23 – Controle distribuído com dois sistemas *Ball Balancer* sem o seletor de controle.

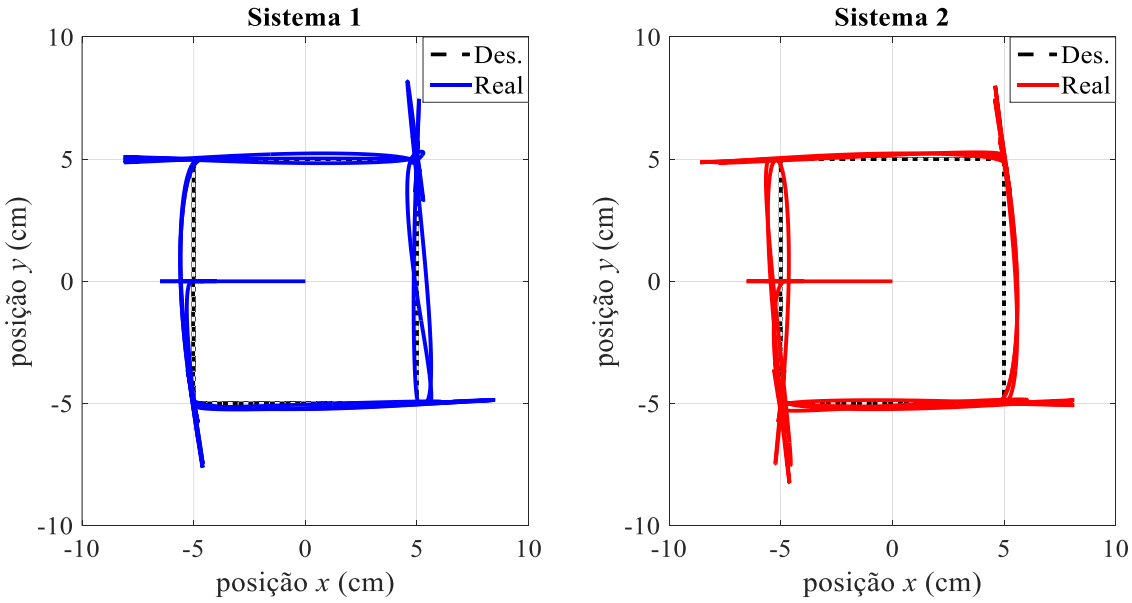


Fonte: próprio autor.

Na Figura 24 é apresentado o resultado do controle considerado para o rastreamento com a referência quadrada. Observa-se que nas Figuras 25 a 26, os resultados de simulação do

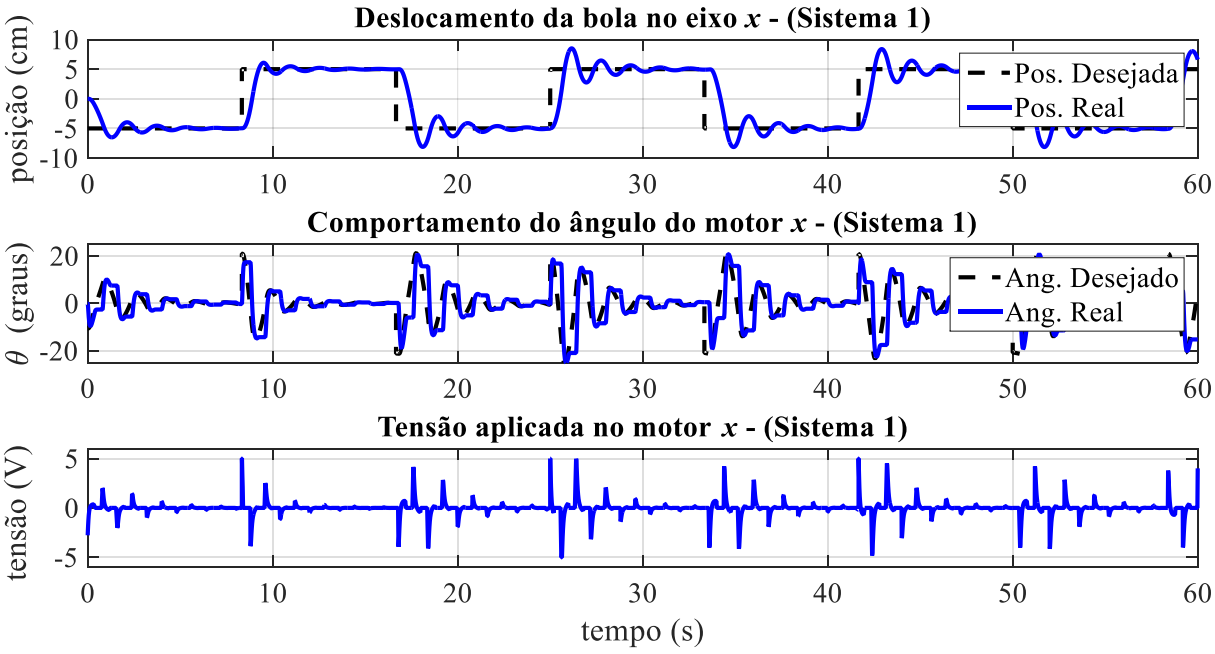
controle distribuído sem seletor, para o controle de posição das coordenadas *x* e *y*, apresentam uma resposta oscilatória, para o período de $400\ ms$, o qual foi escolhido de modo que apresentasse uma resposta aceitável.

Figura 24 – Resultado de simulação do controlador PID sem o seletor, para rastreamento da posição da bola nos eixos *x* e *y* de dois sistemas *Ball Balancer*, para referência quadrada.



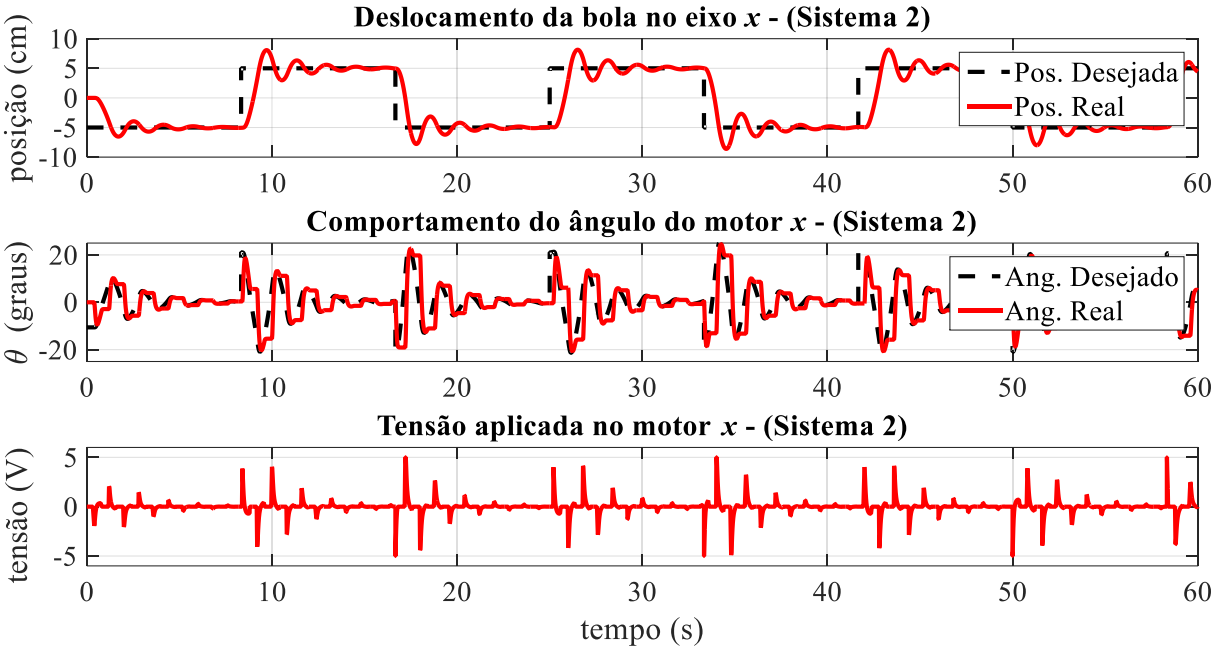
Fonte: próprio autor.

Figura 25 – Resultado de simulação do controlador PID do primeiro sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



Fonte: próprio autor.

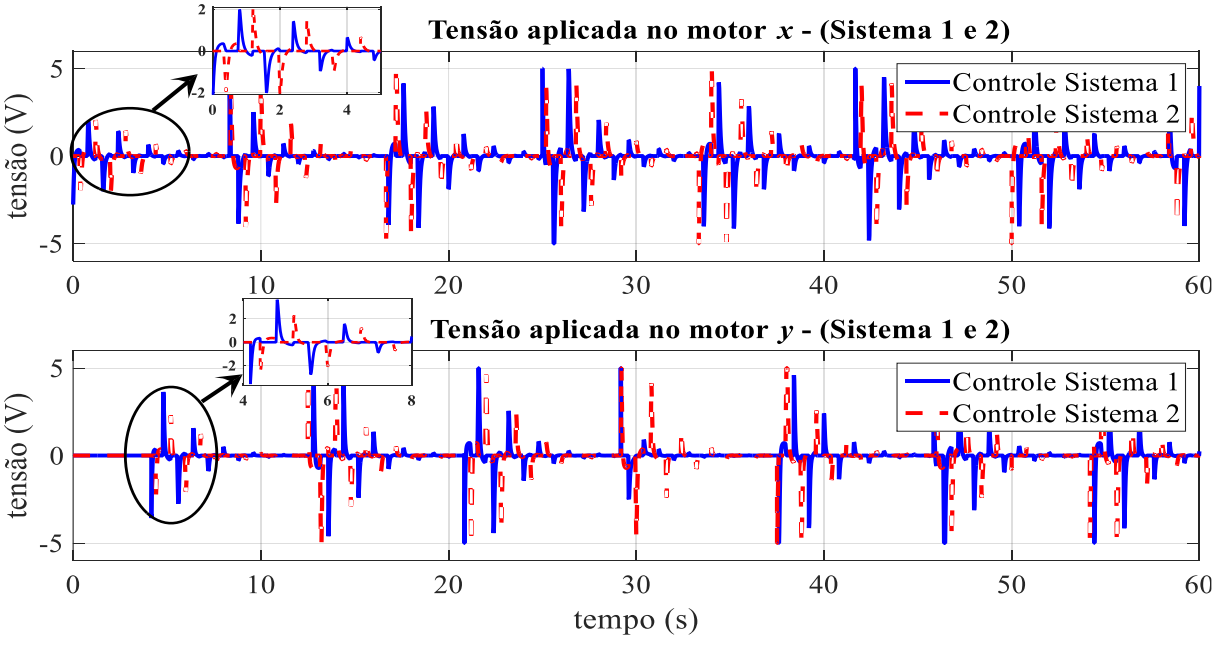
Figura 26 – Resultado de simulação do controlador PID do segundo sistema sem o seletor de controle, para posição da bola no eixo *x*, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



Fonte: próprio autor.

Na Figura 27 é mostrado o resultado do sinal de controle sem seletor. Observa-se na parte destacada, que o sistema não apresenta a natureza cooperativa do controle de forma semelhante quando se considera a estratégia de controle com seletor.

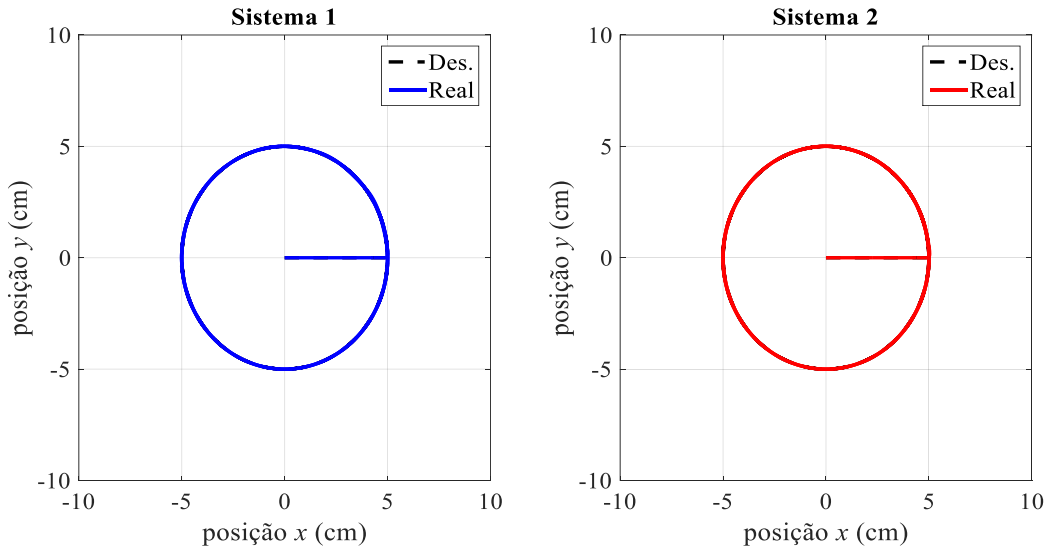
Figura 27 – Resultado de simulação do controlador PID de dois sistemas *Ball Balancer*, sem o seletor de controle para referência quadrada.



Fonte: próprio autor.

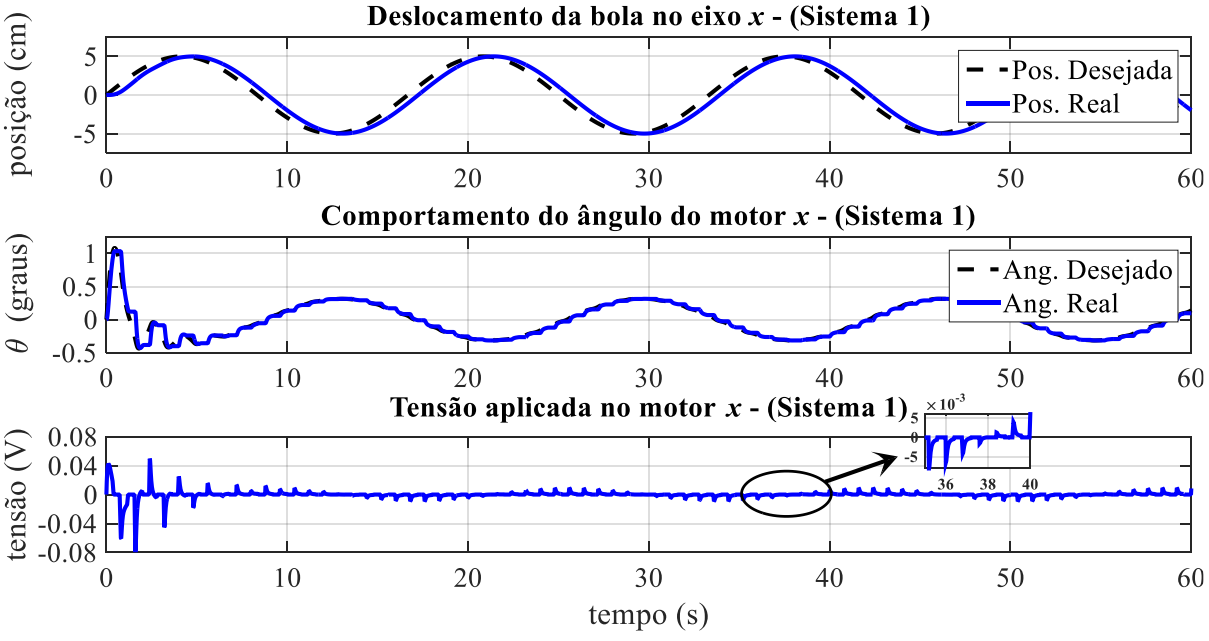
Portanto, nota-se na Figura 28 que o resultado de simulação para o rastreamento com a referência senoidal, não há prejuízos para o mesmo período considerado ($T_i = 400\ ms$) na referência quadrada. Observa-se nas Figuras 29 a 31, que os resultados para o controle de posição, apresentam boas respostas devido ao comportamento contínuo da referência

Figura 28 – Resultado de simulação do controlador PID sem o seletor de controle, para rastreamento da posição da bola nos eixos *x* e *y* de dois sistemas *Ball Balancer*, para referência senoidal.



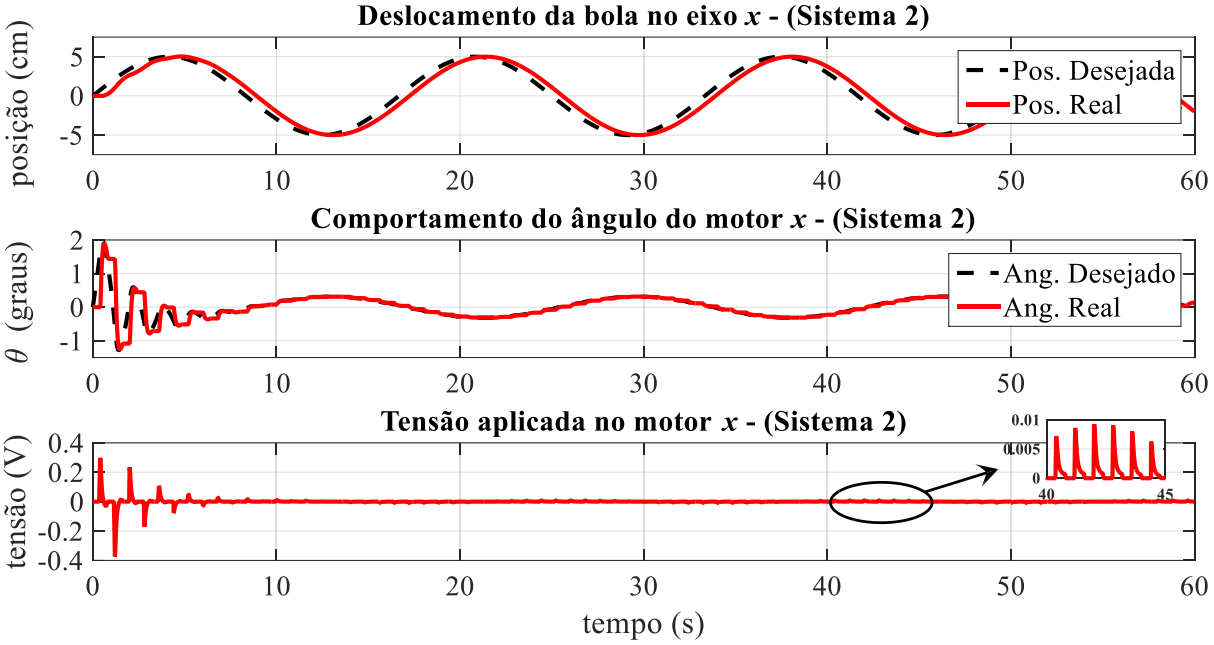
Fonte: próprio autor.

Figura 29 – Resultado de simulação do controlador PID do primeiro sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



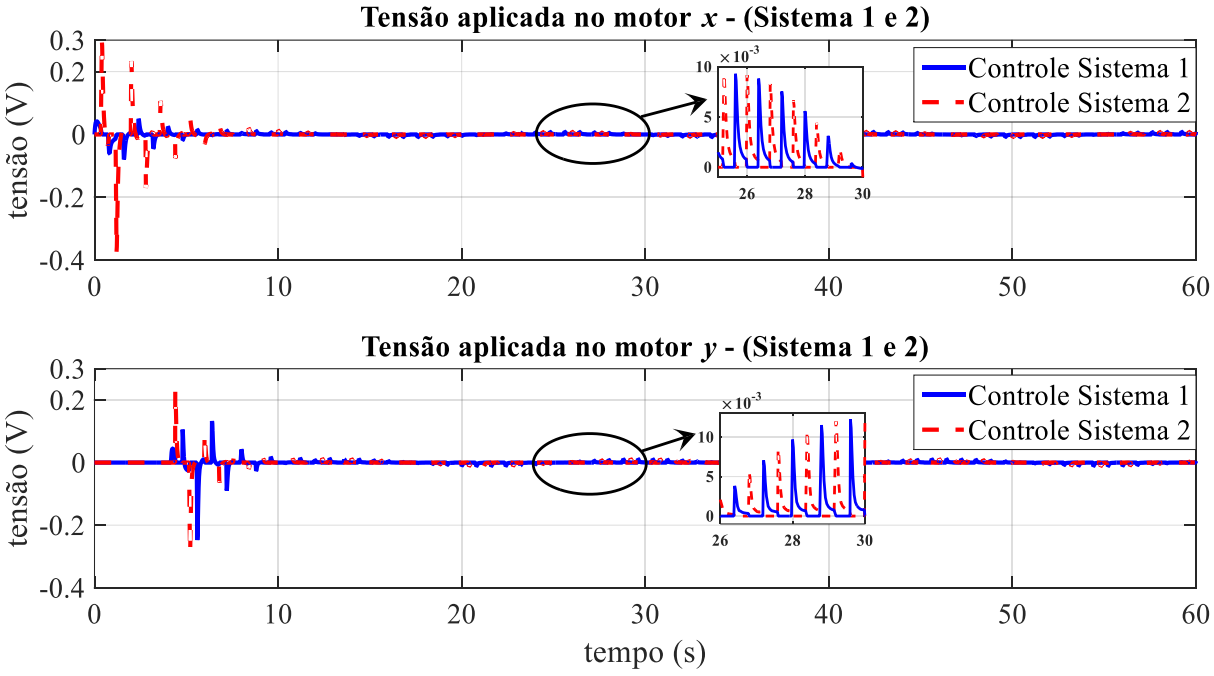
Fonte: próprio autor.

Figura 30 – Resultado de simulação do controlador PID do segundo sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



Fonte: próprio autor.

Figura 31 – Resultado de simulação do controlador PID de dois sistemas *Ball Balancer*, sem o seletor de controle para referência senoidal.



Fonte: próprio autor.

Na Figura 31 é mostrado o resultado do sinal de controle sem seletor para a referência senoidal. Nota-se que apesar do sistema apresentar uma natureza cooperativa do sinal de controle, observa-se na parte destacada, que o sistema apresenta um maior esforço de controle, e o resultado de comutação dos sinais, não está similar quando se considera a estratégia de controle com seletor.

Foi observado que para um período maior de que $T_i = 450\ ms$ o sistema fica instável. Também foi considerado para o período de $T_i = 250\ ms$, e os resultados para ambas as referências (quadrada e senoidal) apresentaram boas respostas.

Com os resultados apresentados, nota-se a importância da estratégia de controle para dois sistemas *Ball Balancer*. Contudo, o trabalho pode ser estendido para mais plantas, conforme será apresentado no Capítulo 6.

6 CONTROLE DISTRIBUÍDO COM TRÊS PLANTAS *BALL BALANCER*

Neste capítulo, apresenta-se os resultados de simulação, obtidos para as técnicas de controle PID para três plantas.

6.1 CONTROLE DISTRIBUÍDO DE TRÊS PLANTAS COM CONTROLE PID

Para simular o controlador PID, para três sistemas *Ball Balancer* idênticos, foi executado um programa no Matlab®, juntamente com o diagrama principal de blocos no Simulink®. Na Figura 32 apresenta-se o diagrama principal com os subsistemas: referências, controle em cascata, modelo não linear da planta, gráficos das três plantas, e chaveamento para o controle.

O resultado para o controle PID, foi obtido considerando as especificações de projeto do fabricante QUANSER, conforme consta nas Tabelas 2 e 3. Na Tabela 7 são mostrados os ganhos obtidos para o controle distribuído.

Tabela 7 – Ganhos para o controle PID para três sistemas *Ball Balancer*.

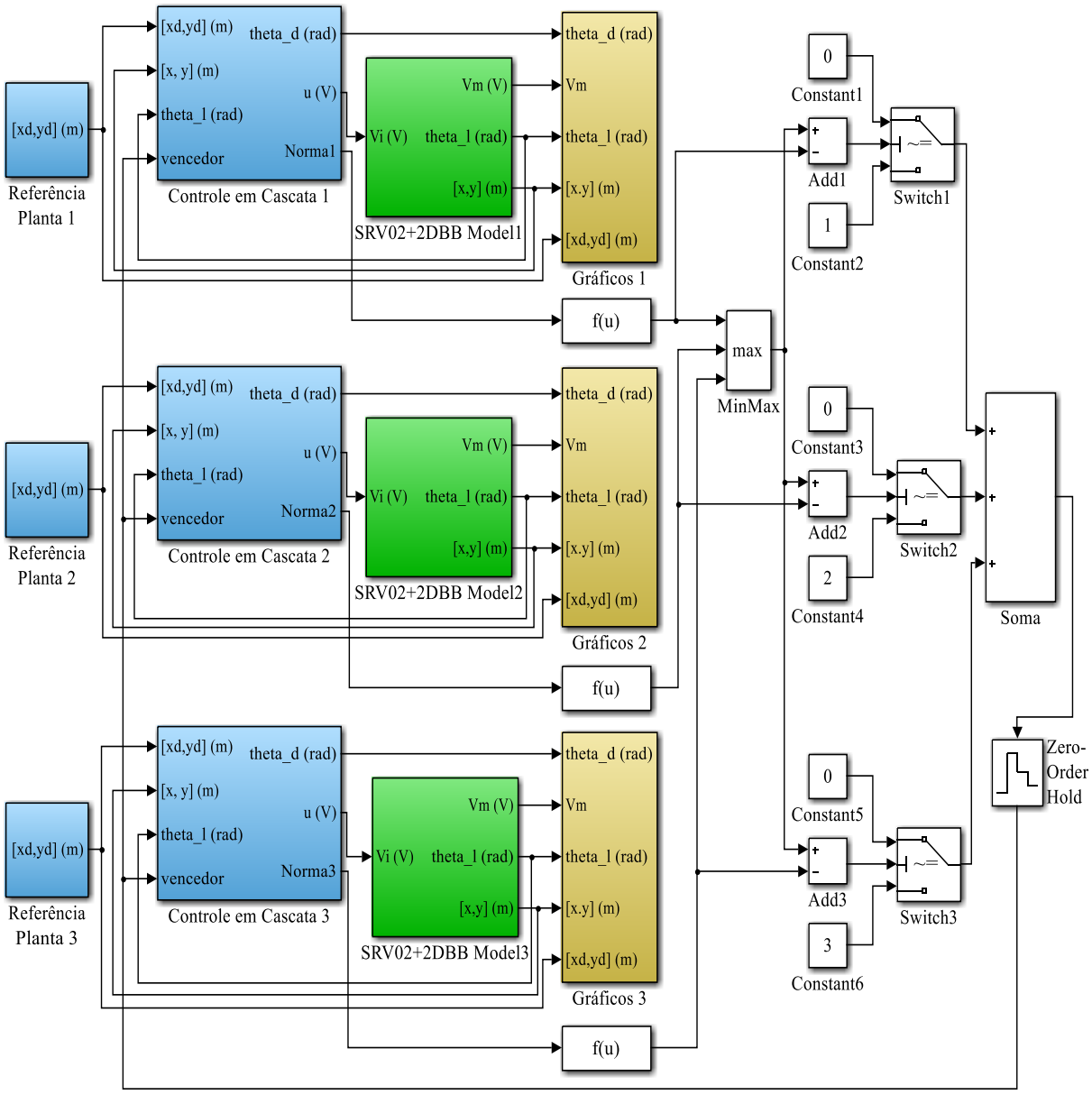
Ganhos	**Descrição**	**Eixo (*x*, *y*)**
K_{bb}	Constante da planta $(m/s^2/rad)$	1,29
K_p	Ganho K_p do motor (V/rad)	13,6
K_v	Ganho K_v do motor $(V.s/rad)$	$-0{,}00612$
$K_{p,bb}$	Ganho K_p do *Ball Balancer* (rad/m)	3,7
$K_{i,bb}$	Ganho K_i do *Ball Balancer* $(rad/m/s)$	0
$K_{d,bb}$	Ganho K_d do *Ball Balancer* $(rad.s/m)$	2,15

Fonte: próprio autor.

Na Figura 32, utiliza-se os blocos de máximo (*MinMax*), subtrator (*Add*) e seletor (*Switch*) para a seleção do vencedor, caso o sinal na saída do bloco soma (*Add*) for sinal um, o vencedor na entrada de cada bloco de controle (controle em cascata) recebe sinal um, então somente o sistema 1 será controlado nesse instante de tempo, caso o sinal na saída do bloco soma (*Add*) for dois ou três, os blocos de controle em cascata recebem sinal dois ou três na entrada vencedor, e somente o sistema correspondente ao sinal vencedor que será controlado. O bloco retenção de ordem zero (*Zero-Order Hold*), mantém sua entrada constante para o período de amostragem especificado (utilizou-se o padrão de $100\,ms$), e de acordo com a

tomada de decisão de qual planta apresenta a maior norma em um determinado instante de amostragem, o sinal é encaminhado para as plantas, e apenas a planta de maior norma compreenderá a decisão de ser controlada.

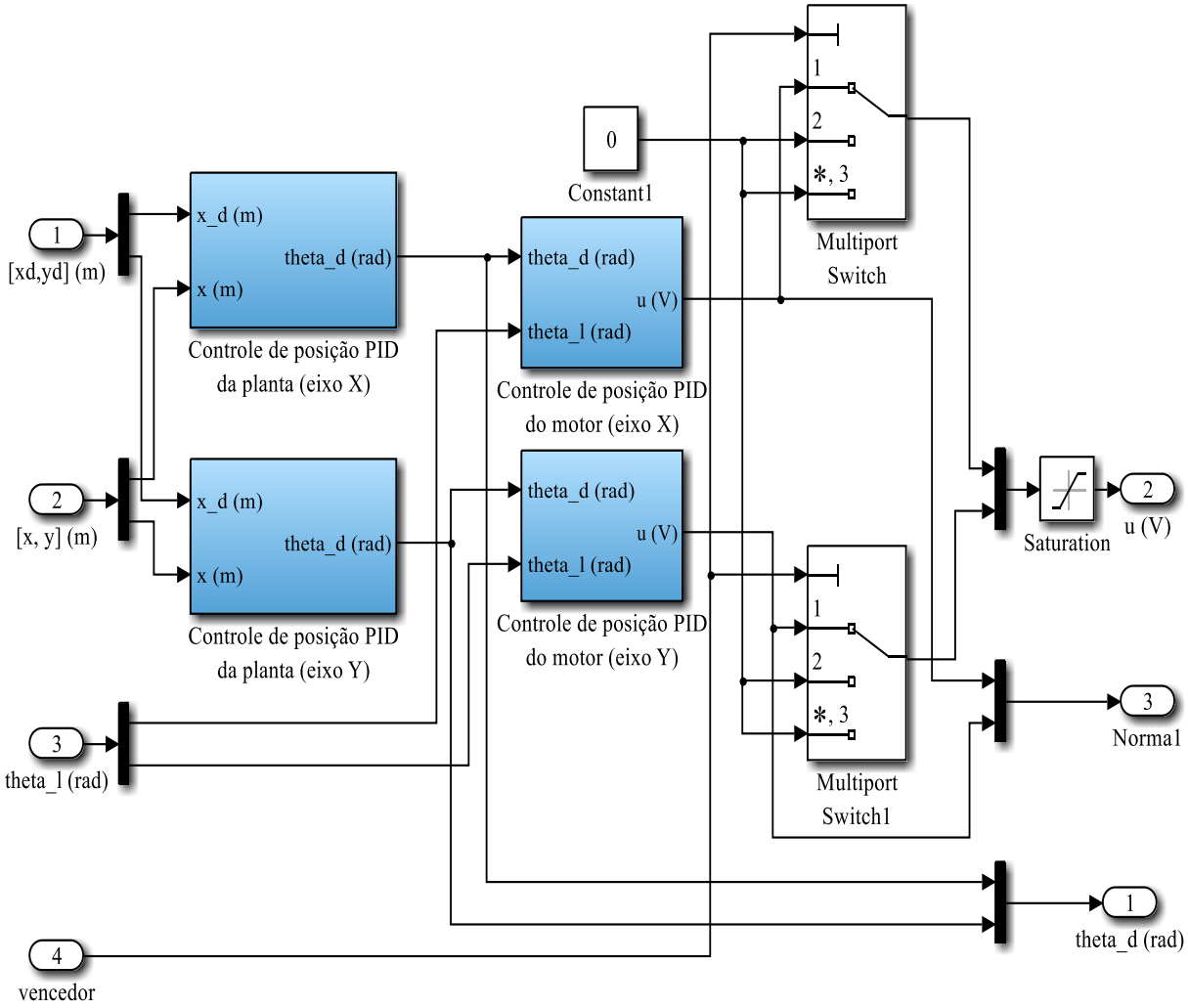
Figura 32 – Diagrama de blocos principal para simulação do controlador PID para três sistemas *Ball Balancer*.



Fonte: próprio autor.

Na Figura 33 é apresentado o subsistema do controle em cascata referente ao primeiro sistema *Ball Balancer*, com a entrada do sinal vencedor, caso o sinal na entrada vencedor receber índice 1, então o sistema 1 será controlado, contudo, se receber 2 ou 3, então seu sinal de controle será igual a zero.

Figura 33 – Subsistema do controle em cascata do primeiro sistema, com a entrada do sinal vencedor para simulação do controlador PID.

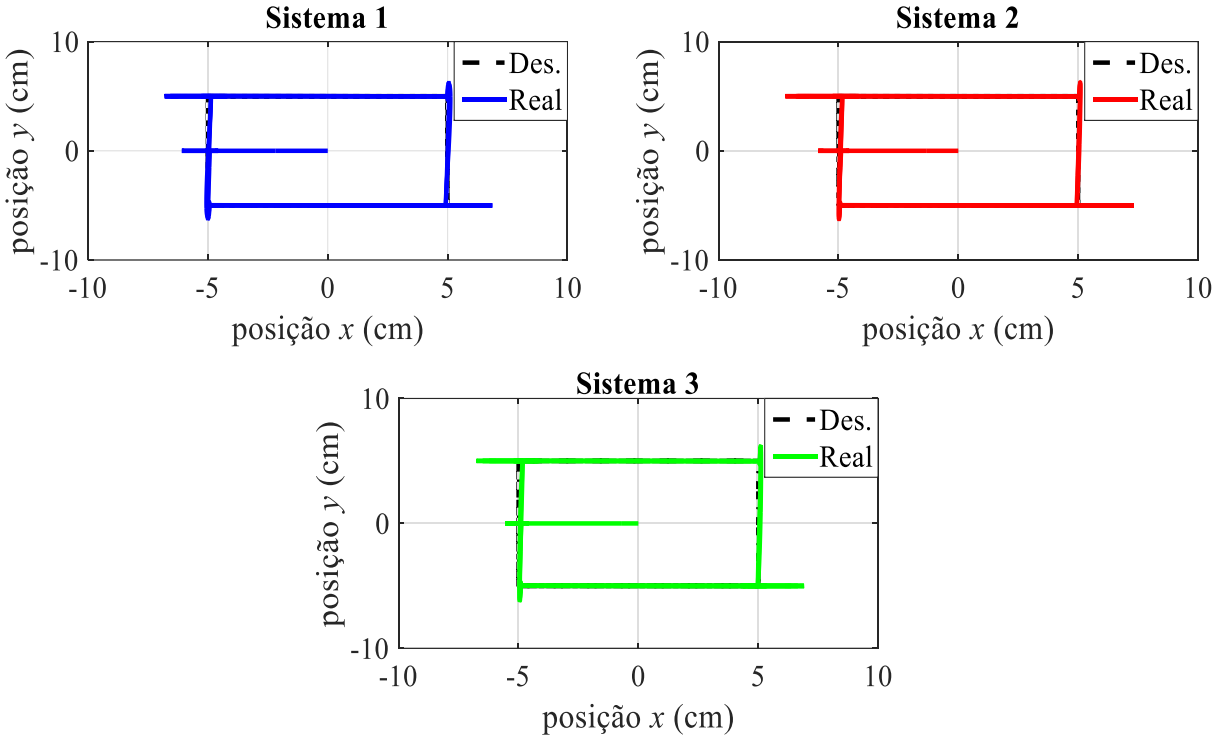


Fonte: próprio autor.

Na Figura 34 é apresentado o resultado de simulação do controle distribuído de três sistemas *Ball Balancer*, para o rastreamento em onda quadrada (referência em preto), é mostrado o controle de posição das coordenadas x e y dos sistemas.

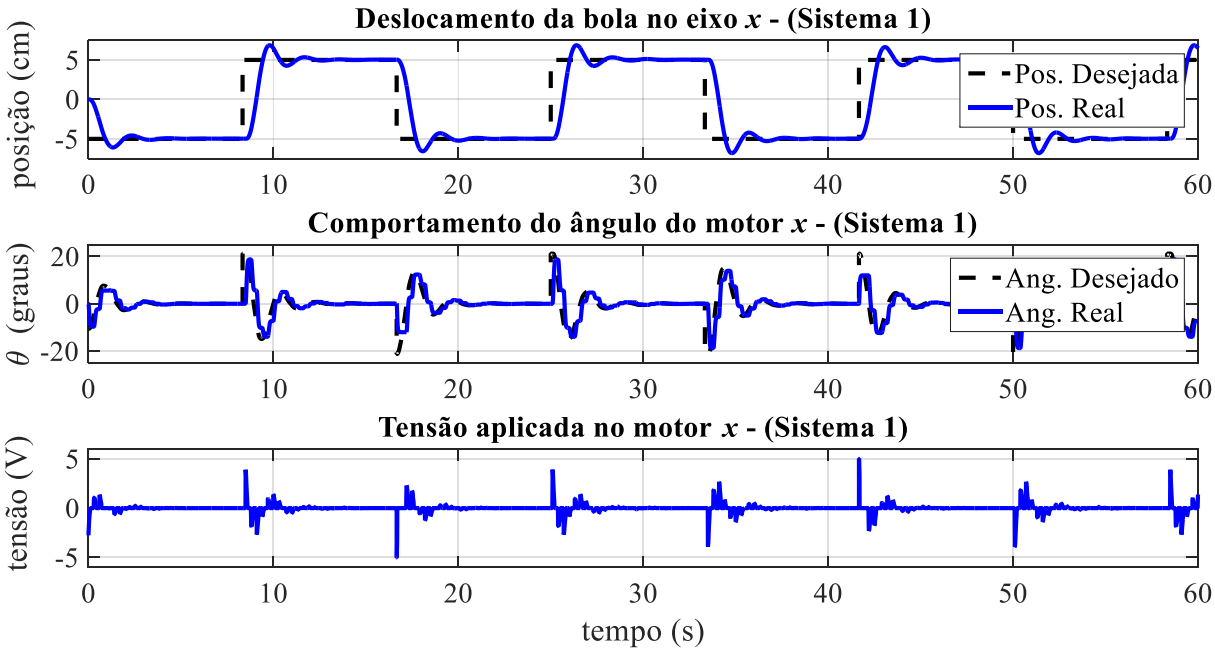
Observa-se que nas Figuras 35 a 37, os resultados de simulação do controle distribuído para três sistemas *Ball Balancer*, para o rastreamento da referência quadrada, o controle de posição das coordenadas *x* e *y* de ambos os sistemas, está adequado aos requisitos exigidos em projeto. Nota-se também que a resposta de malha interna apresenta uma rápida resposta.

Figura 34 – Resultado de simulação do controlador PID para rastreamento da posição da bola nos eixos *x* e *y* de três sistemas *Ball Balancer*, para referência quadrada.



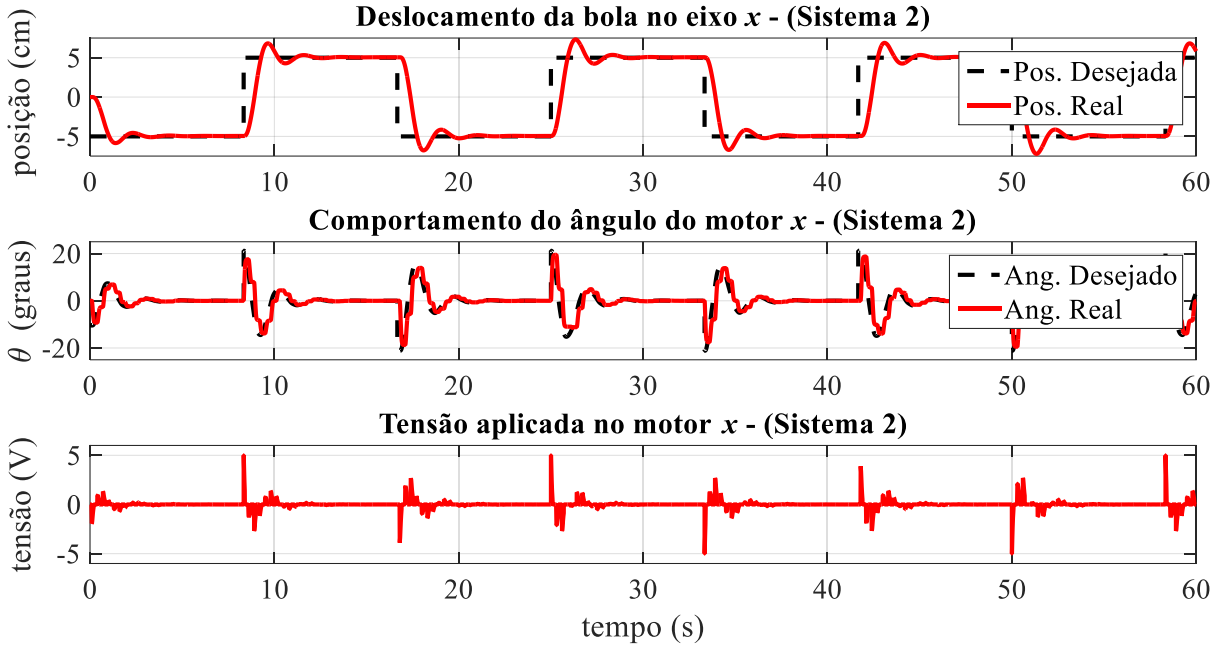
Fonte: próprio autor.

Figura 35 – Resultado de simulação do controlador PID do primeiro sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



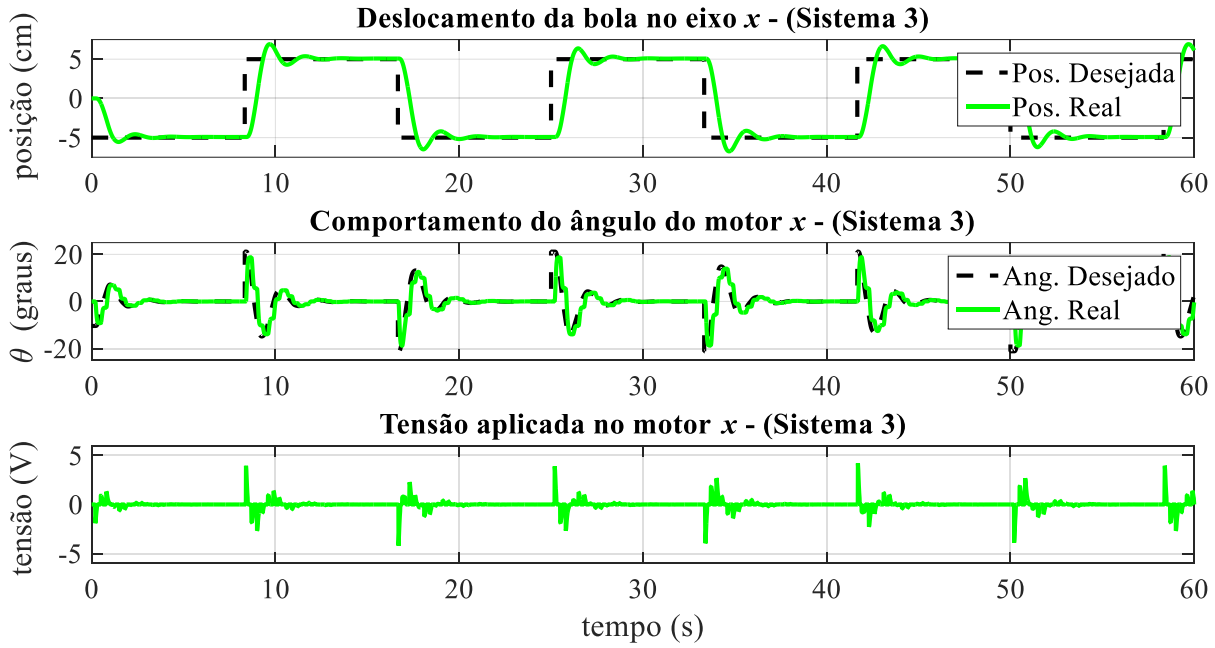
Fonte: próprio autor.

Figura 36 – Resultado de simulação do controlador PID do segundo sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



Fonte: próprio autor.

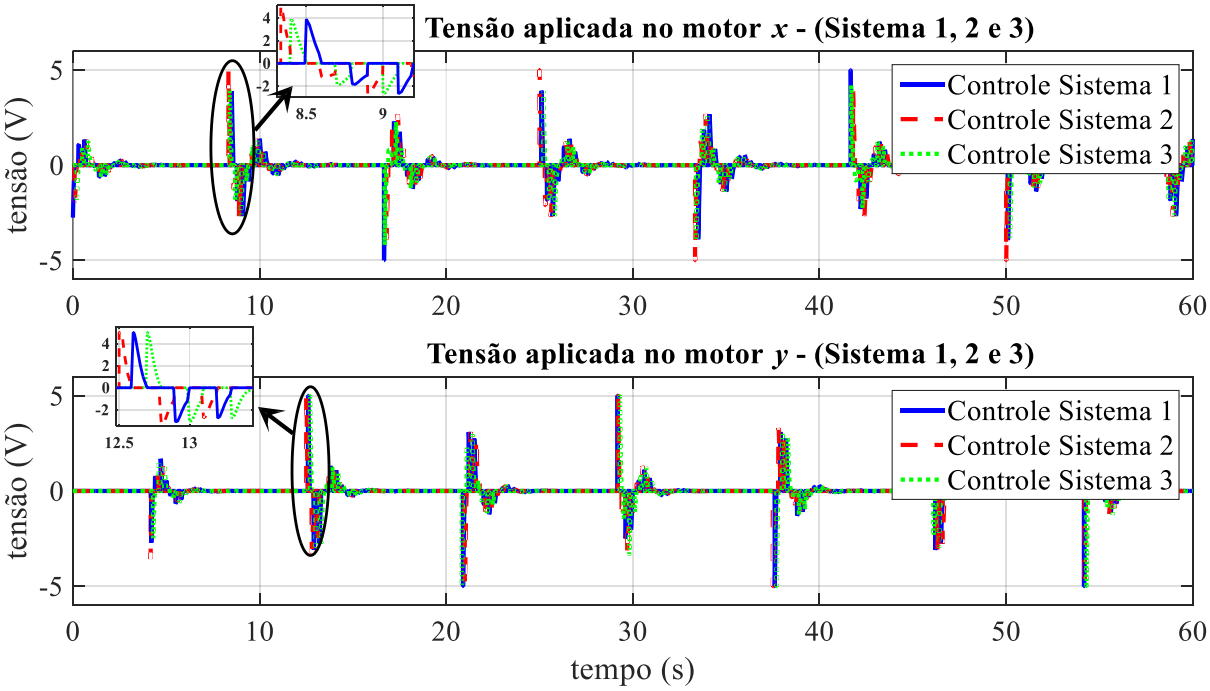
Figura 37 – Resultado de simulação do controlador PID do terceiro sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



Fonte: próprio autor.

Pode-se observar também que o esforço de controle dos três sistemas não sofrem alterações simultâneas, conforme Figura 38, observa-se a natureza cooperativa do controle na parte destacada, quando o sinal de controle referente ao primeiro sistema é atualizado, os sinais de controle do segundo e terceiro sistema, permanecem constante em zero e vice-versa.

Figura 38 – Resultado de simulação do controlador PID para o controle distribuído de três sistemas *Ball Balancer* para referência quadrada.

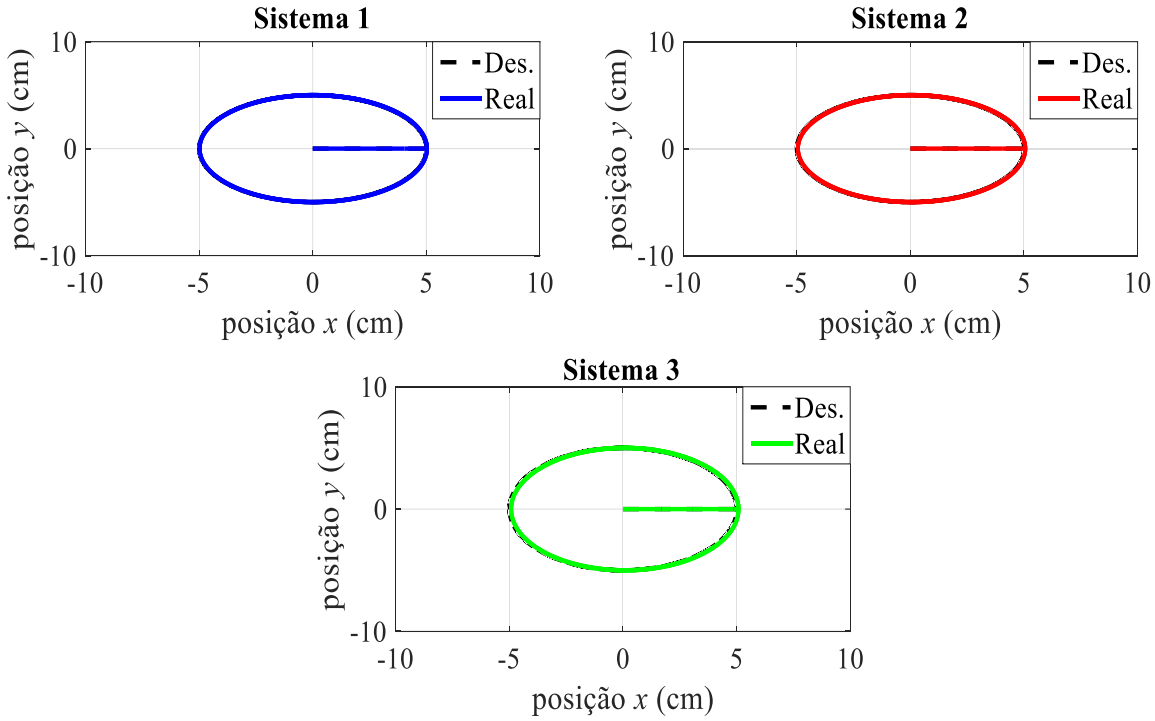


Fonte: próprio autor.

Na Figura 39 é apresentado o resultado de simulação do controle distribuído de três sistemas *Ball Balancer*, considerando o rastreamento em onda senoidal (referência em preto), é apresentado o controle de posição das coordenadas x e y dos três sistemas.

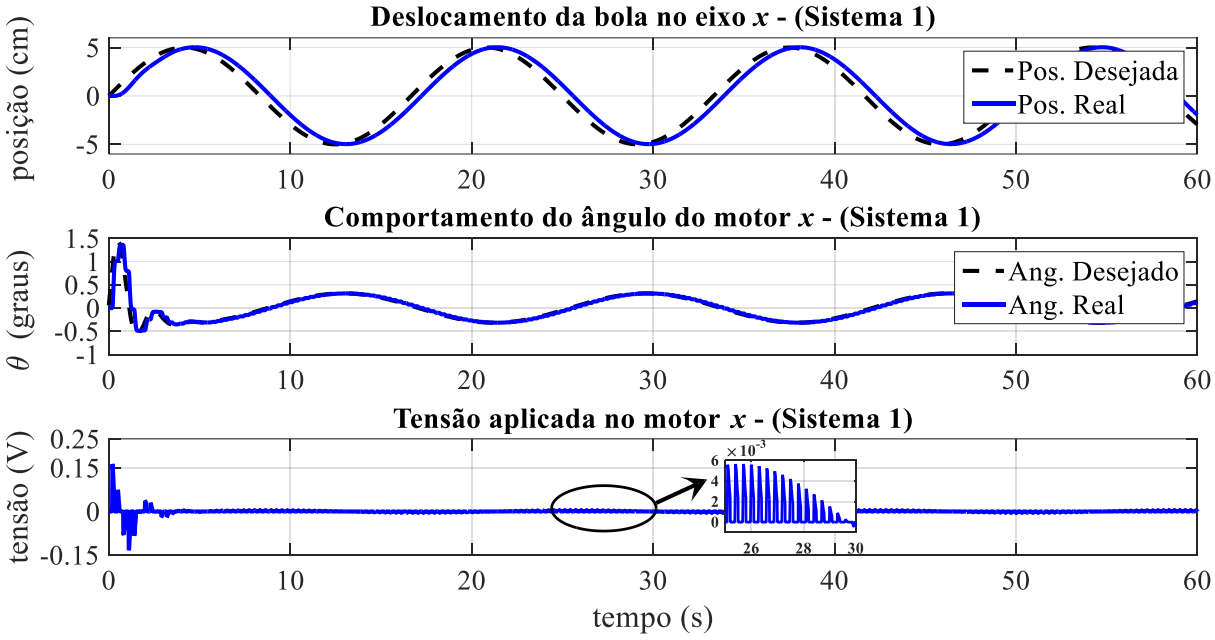
Observa-se que nas Figuras 40 a 42, os resultados de simulação do controle distribuído com os três sistemas, para o rastreamento da referência senoidal, o controle de posição das coordenadas x e y dos sistemas, está adequado aos requisitos exigidos em projeto. Nota-se também que na resposta de malha interna, o valor real do ângulo está praticamente sobreposto ao valor desejado, ou seja, apresenta uma rápida resposta, confirmando que a especificação do projeto foi atendida.

Figura 39 – Resultado de simulação do controlador PID para rastreamento da posição da bola nos eixos *x* e *y* de três sistemas *Ball Balancer*, para referência senoidal.



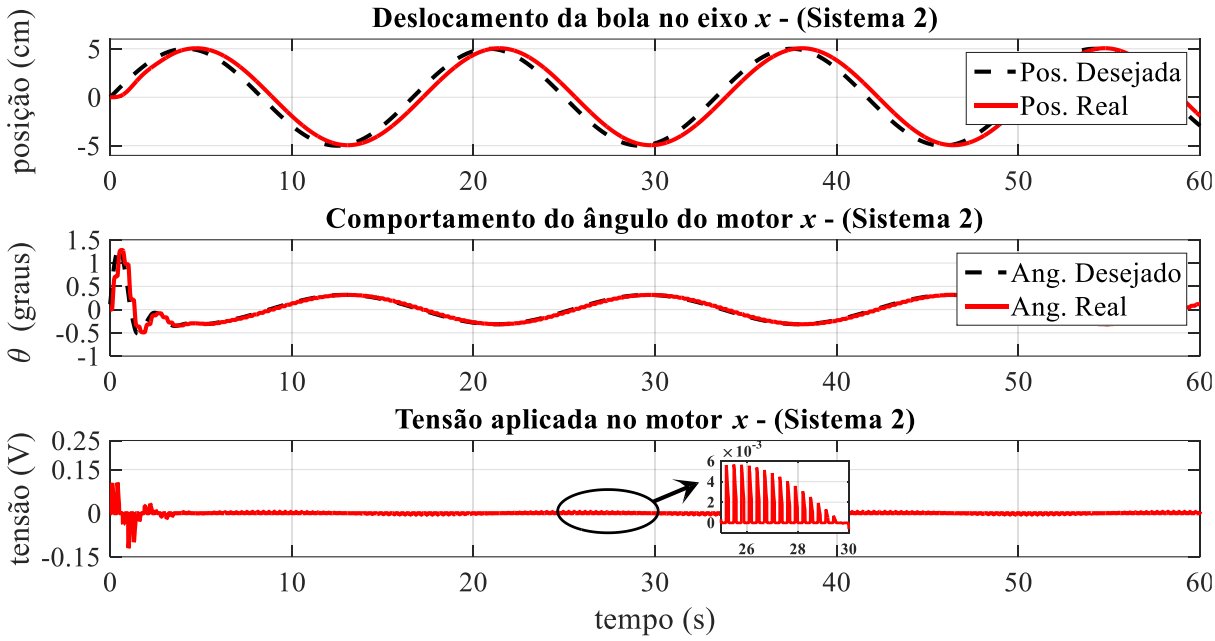
Fonte: próprio autor.

Figura 40 – Resultado de simulação do controlador PID do primeiro sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



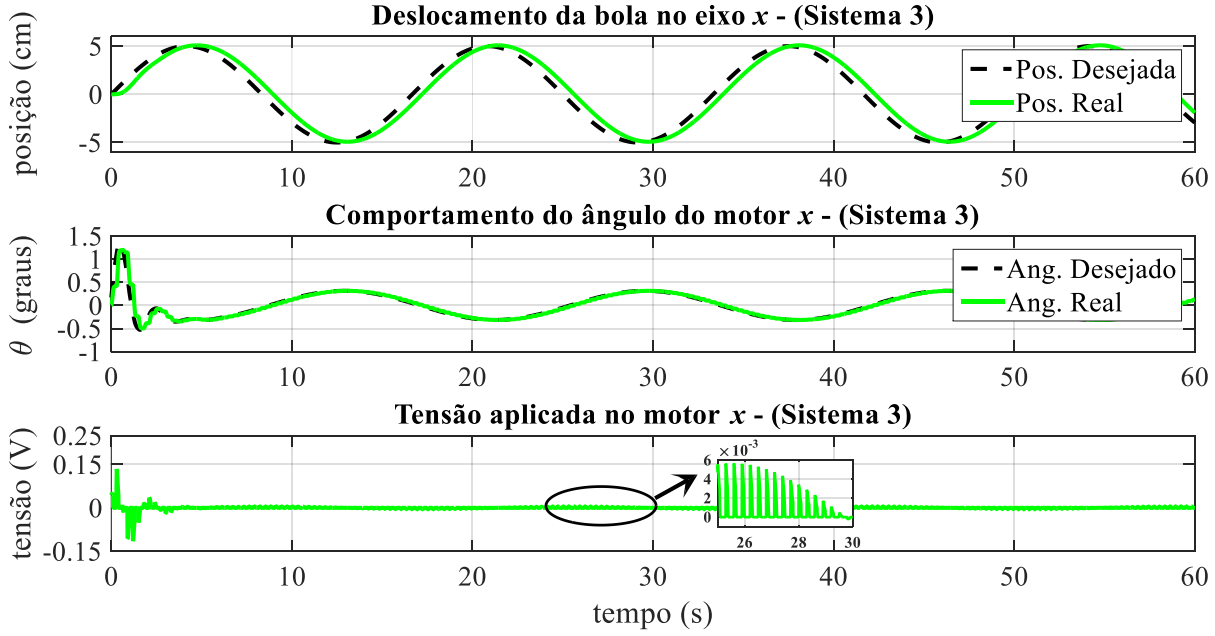
Fonte: próprio autor.

Figura 41 – Resultado de simulação do controlador PID do segundo sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



Fonte: próprio autor.

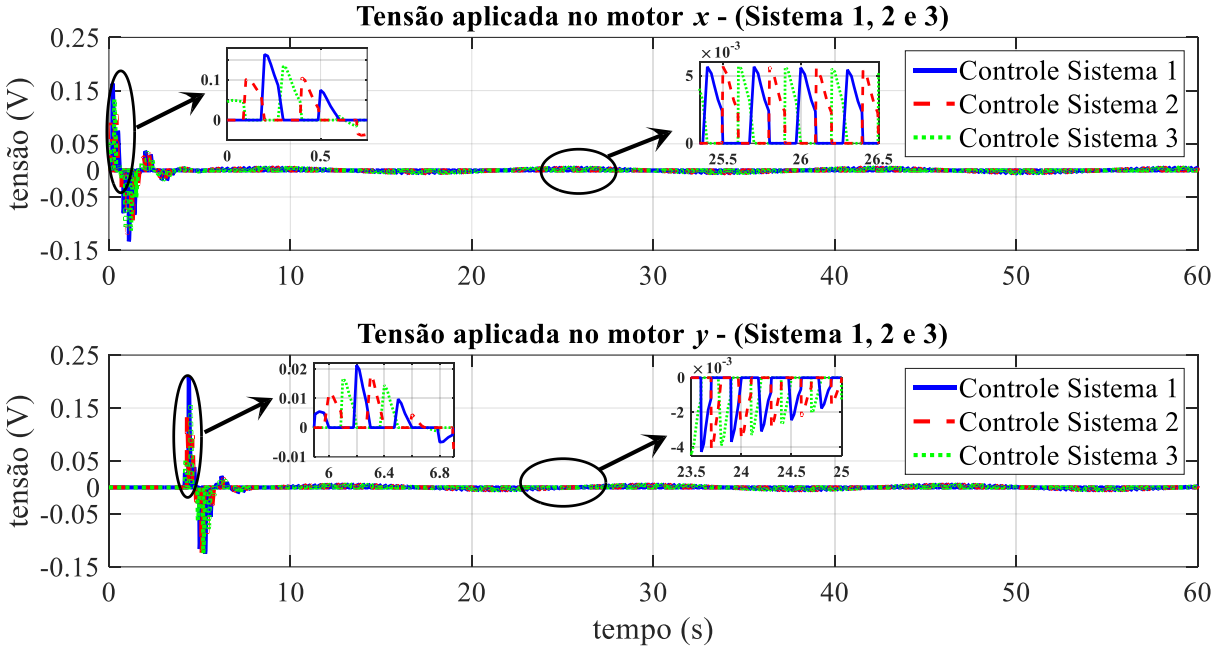
Figura 42 – Resultado de simulação do controlador PID do terceiro sistema, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



Fonte: próprio autor.

Nota-se que para a referência senoidal para o controle distribuído dos três sistemas, o esforço de controle também apresenta uma natureza cooperativa, conforme é mostrado na parte destacada na Figura 43. Observa-se que quando o sinal de controle referente ao primeiro sistema é atualizado, os sinais de controle do segundo e terceiro sistema, permanecem constante em zero e vice-versa.

Figura 43 – Resultado de simulação do controlador PID para o controle distribuído de três sistemas *Ball Balancer* para referência senoidal.



Fonte: próprio autor.

6.2 CONTROLE PID DE TRÊS PLANTAS SEM O SELETOR DE CONTROLE

Para verificar, através de comparação de resultados, o bom desempenho da estratégia proposta neste trabalho e simulada na Seção 6.1, foram realizadas simulações em que o sinal de controle é entregue para um sistema por vez, mas sem a tomada de decisão baseada na proposta deste trabalho, ou seja, cada sistema receberá o sinal de controle dentro de um intervalo de tempo.

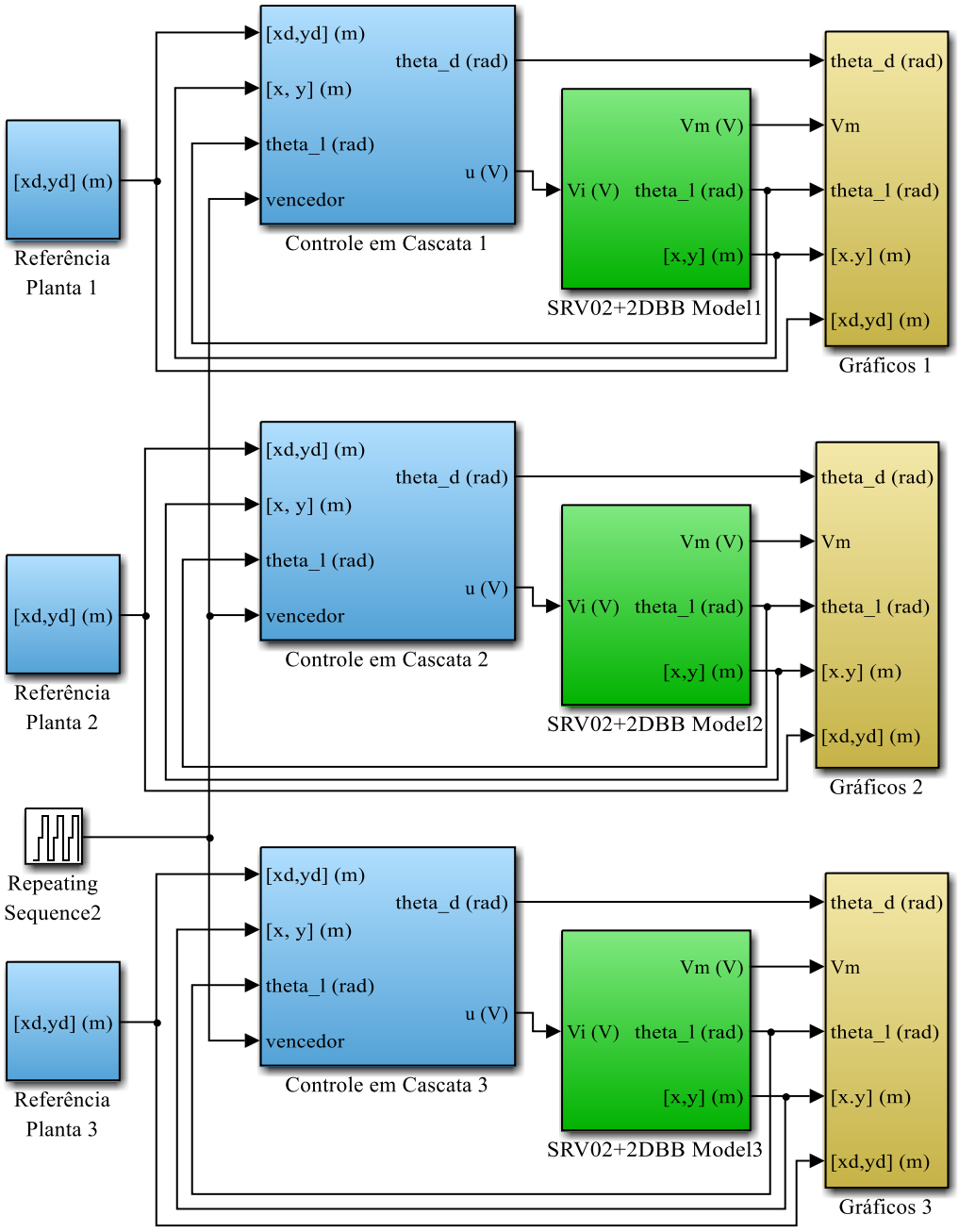
Para simular o controlador PID, para três plantas sem o seletor de controle, foi executado um programa no Matlab®, juntamente com o diagrama principal de blocos no Simulink®.

Na Figura 44 apresenta-se o diagrama principal com os subsistemas: referências, controle em cascata, modelo não linear da planta e gráficos das três plantas. O resultado para o

controle PID, foi obtido considerando as especificações de projeto do fabricante QUANSER®, conforme consta nas Tabelas 2 e 3.

Para a simulação com as três plantas sem o seletor de controle, foi considerado primeiramente que um *Ball Balancer* será controlado por $250\ ms$ (utilizou-se o bloco *Repeating Sequence* para gerar o sinal de clock para alternar os controladores ao longo de um período T_i), enquanto que as outras plantas ficam sem o sinal de controle (durante $250\ ms$); depois a condição se inverte de maneira periódica.

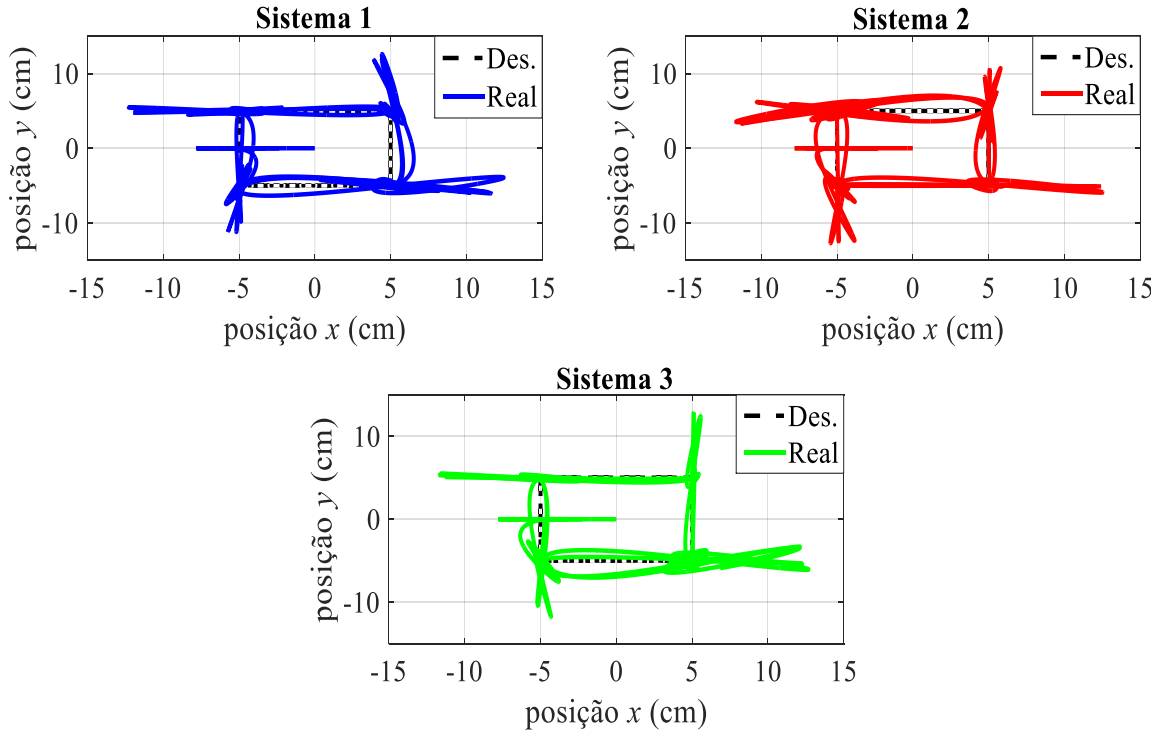
Figura 44 – Controle distribuído com três sistemas *Ball Balancer* sem o seletor de controle.



Fonte: próprio autor.

Na Figura 45 é apresentado o resultado do controle considerado, para o rastreamento com a referência quadrada.

Figura 45 – Resultado de simulação do controlador PID sem o seletor de controle, para rastreamento da posição da bola nos eixos *x* e *y* de três sistemas *Ball Balancer*, para referência quadrada.

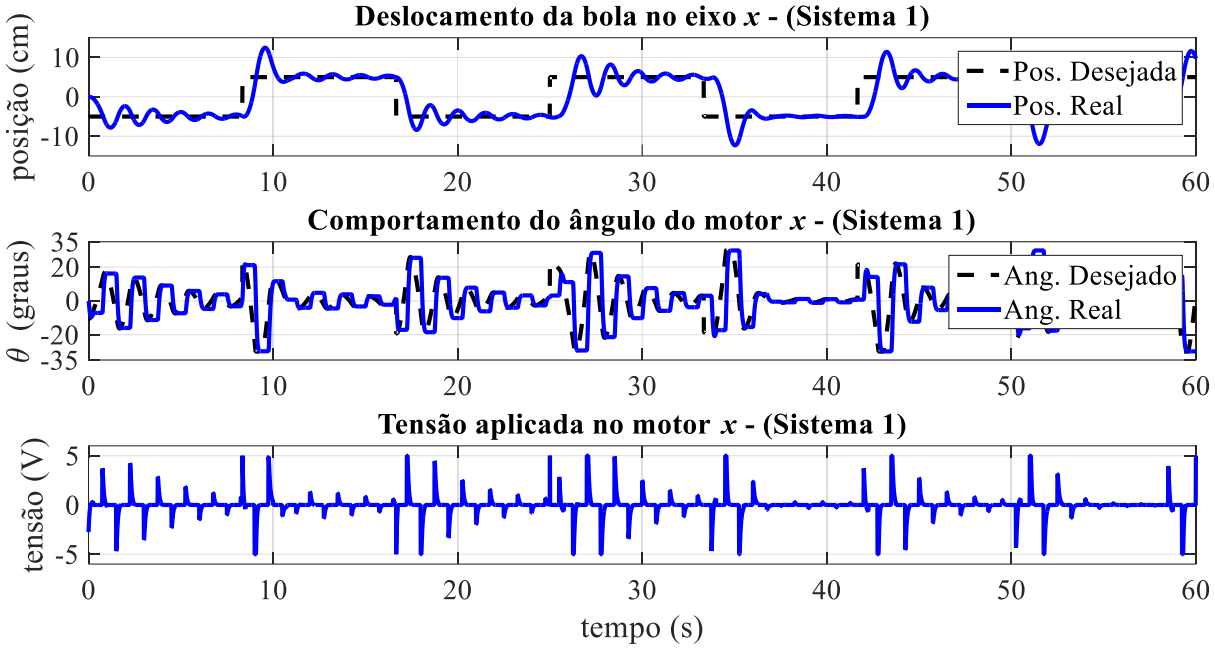


Fonte: próprio autor.

Nota-se que nas Figuras 46 a 48 os resultados de simulação do controle distribuído sem o seletor de controle para a referência quadrada, o controle já apresenta um prejuízo no sinal, diferente do que aconteceu no Capítulo 5 para o mesmo período de $T_i = 250\ ms$, uma vez que foi acrescentado um novo sistema *Ball Balancer*.

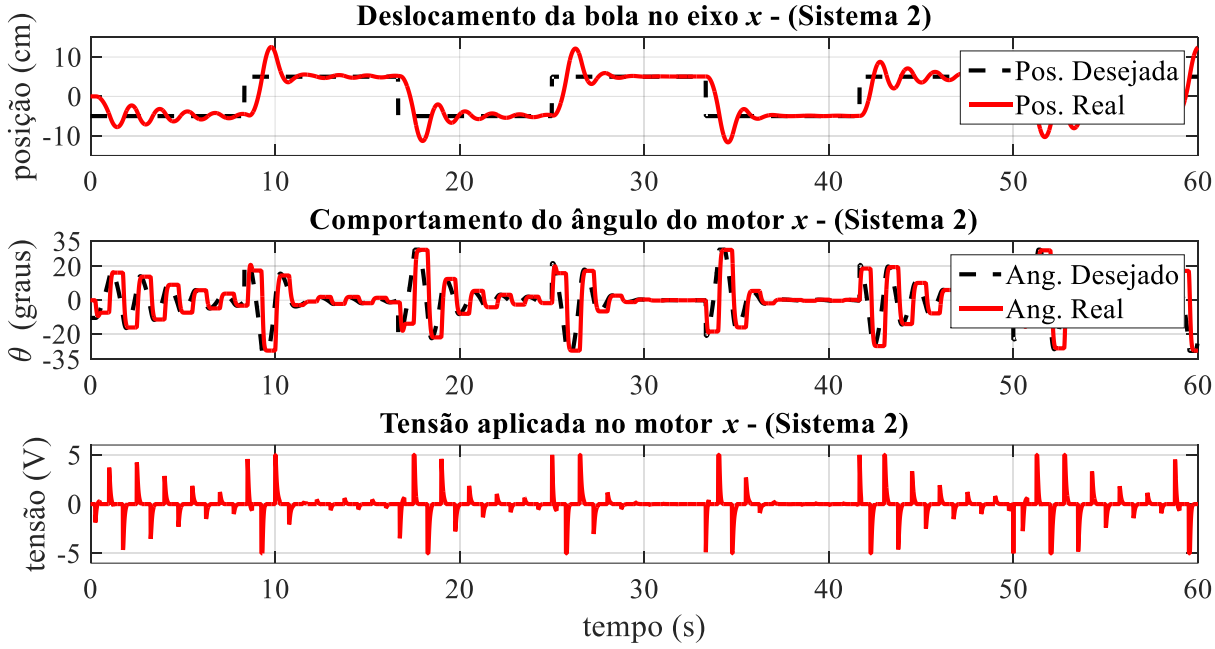
Na Figura 49 é mostrado o resultado do esforço de controle sem o seletor de controle para a referência quadrada. Observa-se que embora o sistema apresenta a natureza cooperativa de controle, nota-se na parte destacada, que o resultado de comutação dos sinais, não está similar quando se considera a estratégia de controle com seletor.

Figura 46 – Resultado de simulação do controlador PID do primeiro sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



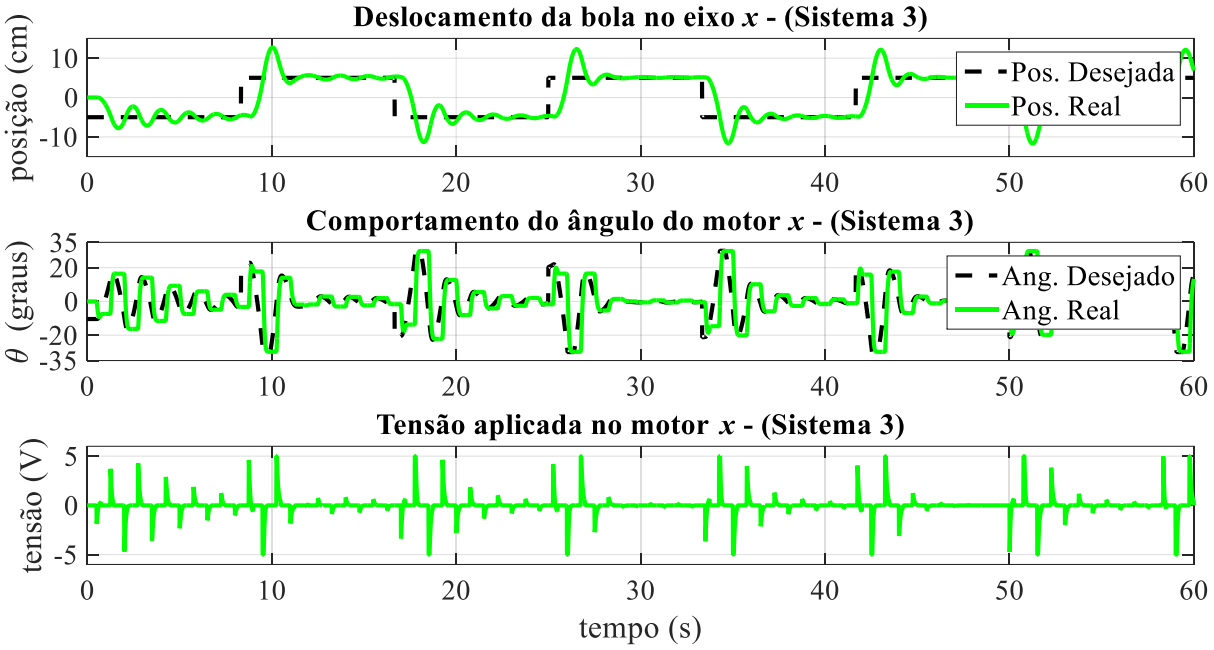
Fonte: próprio autor.

Figura 47 – Resultado de simulação do controlador PID do segundo sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



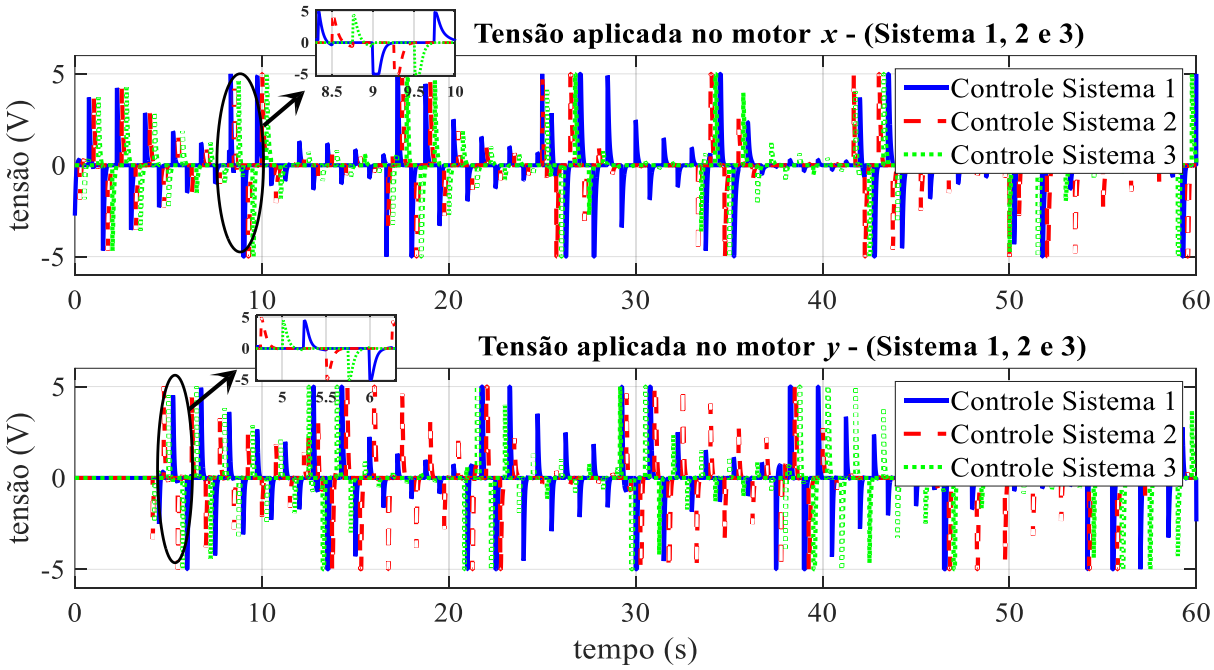
Fonte: próprio autor.

Figura 48 – Resultado de simulação do controlador PID do terceiro sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência quadrada em preto.



Fonte: próprio autor.

Figura 49 – Resultado de simulação do controlador PID de três sistemas *Ball Balancer*, sem o seletor de controle para referência quadrada.



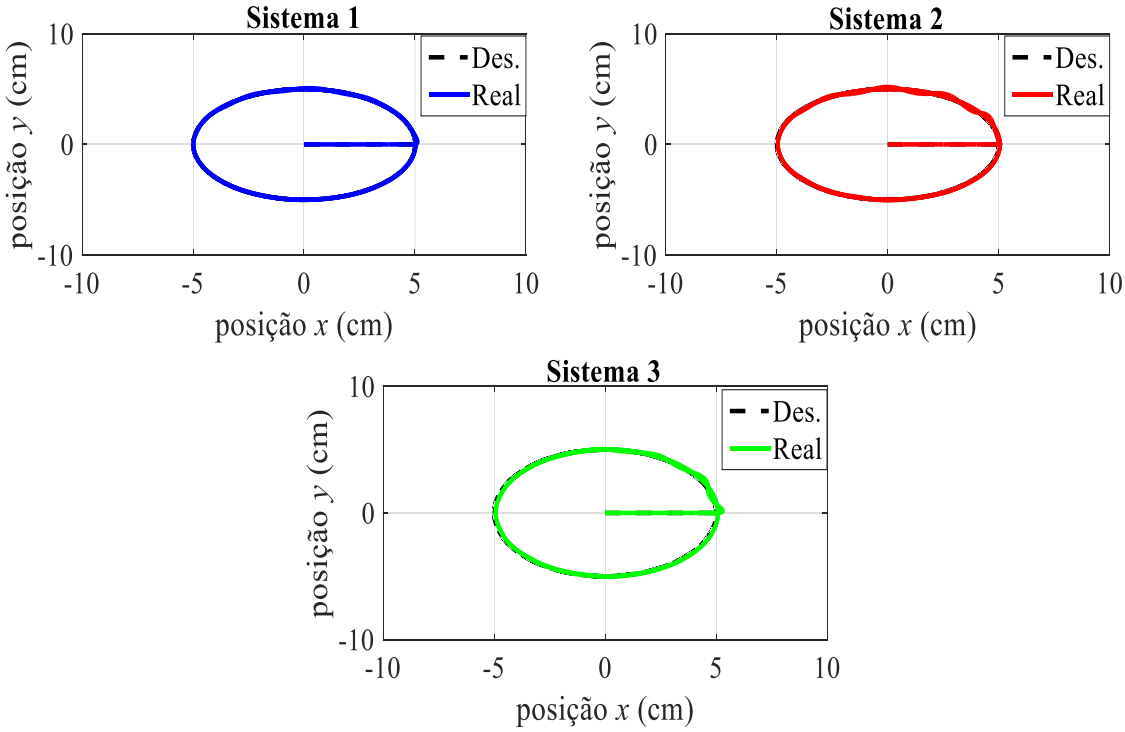
Fonte: próprio autor.

Na Figura 50 é apresentado o resultado do controle das posições das coordenadas *x* e *y*, para o rastreamento com a referência senoidal.

Nas Figuras 51 a 53, é apresentada as respostas dos três sistemas com $T_i = 250\,ms$, para a posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal. Portanto, verifica-se que o controle com referência senoidal, ficou adequado para o período considerado.

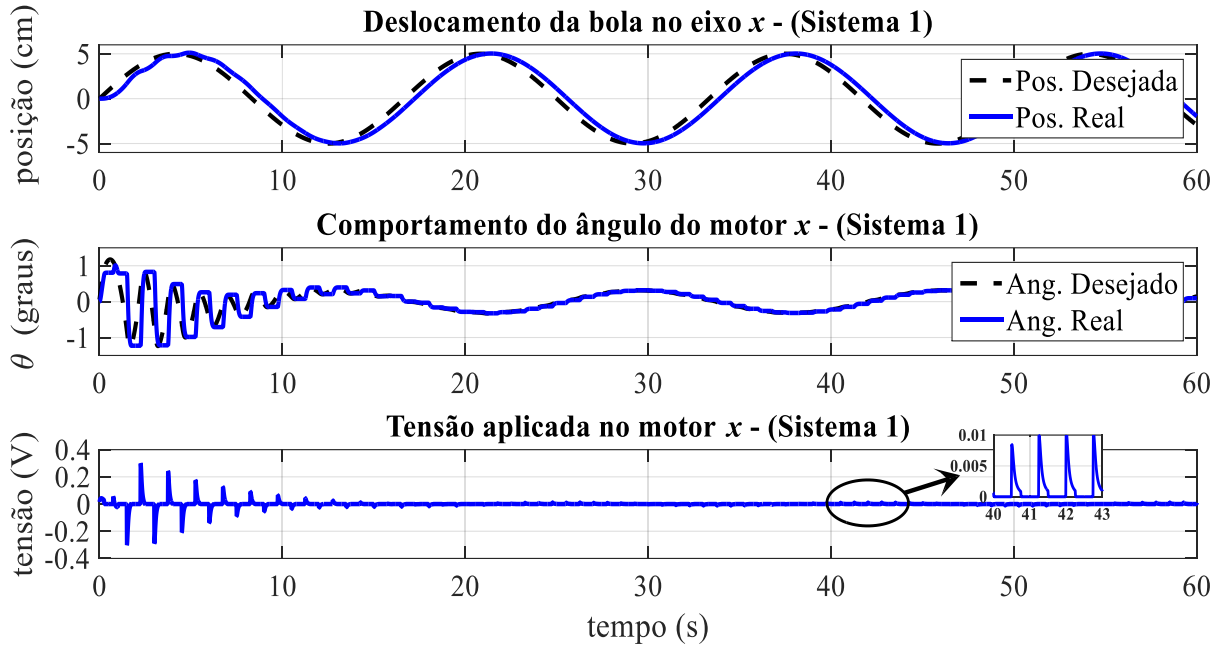
Na Figura 54 é apresentado o resultado do esforço de controle sem o seletor de controle para a referência senoidal. Observa-se que o sistema não apresenta a natureza cooperativa do controle de forma semelhante quando se considera a estratégia de controle distribuído com o seletor de controle.

Figura 50 – Resultado de simulação do controlador PID sem o seletor, para rastreamento da posição da bola nos eixos *x* e *y* de três sistemas *Ball Balancer*, para referência senoidal.



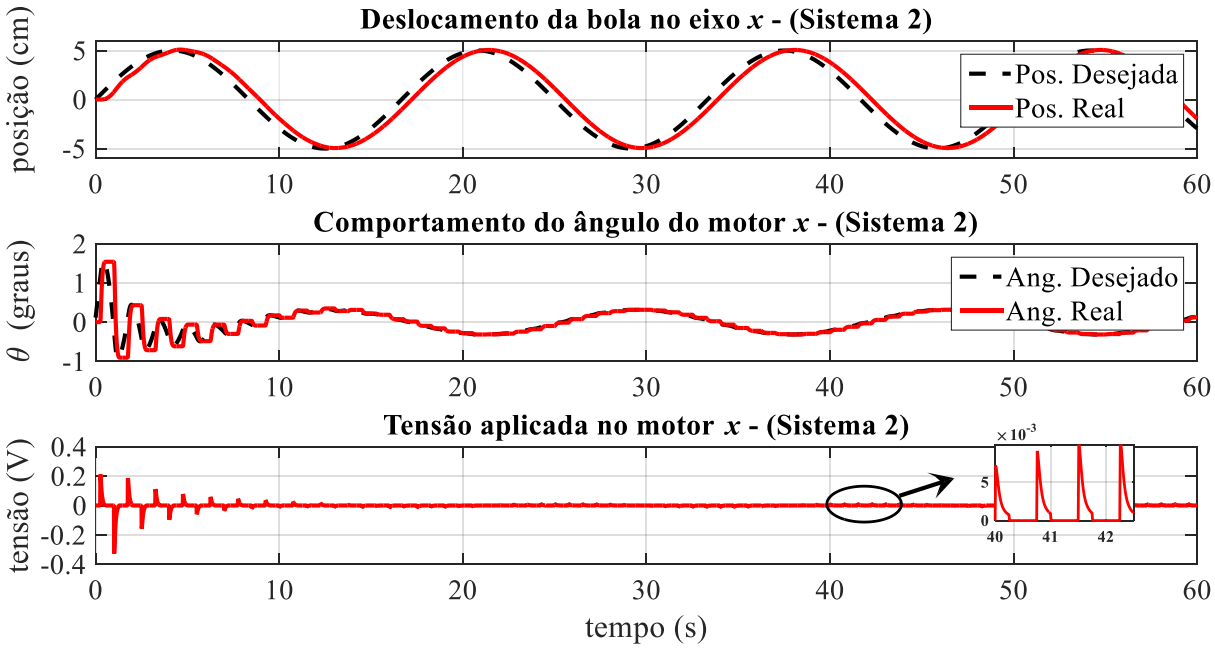
Fonte: próprio autor.

Figura 51 – Resultado de simulação do controlador PID do primeiro sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



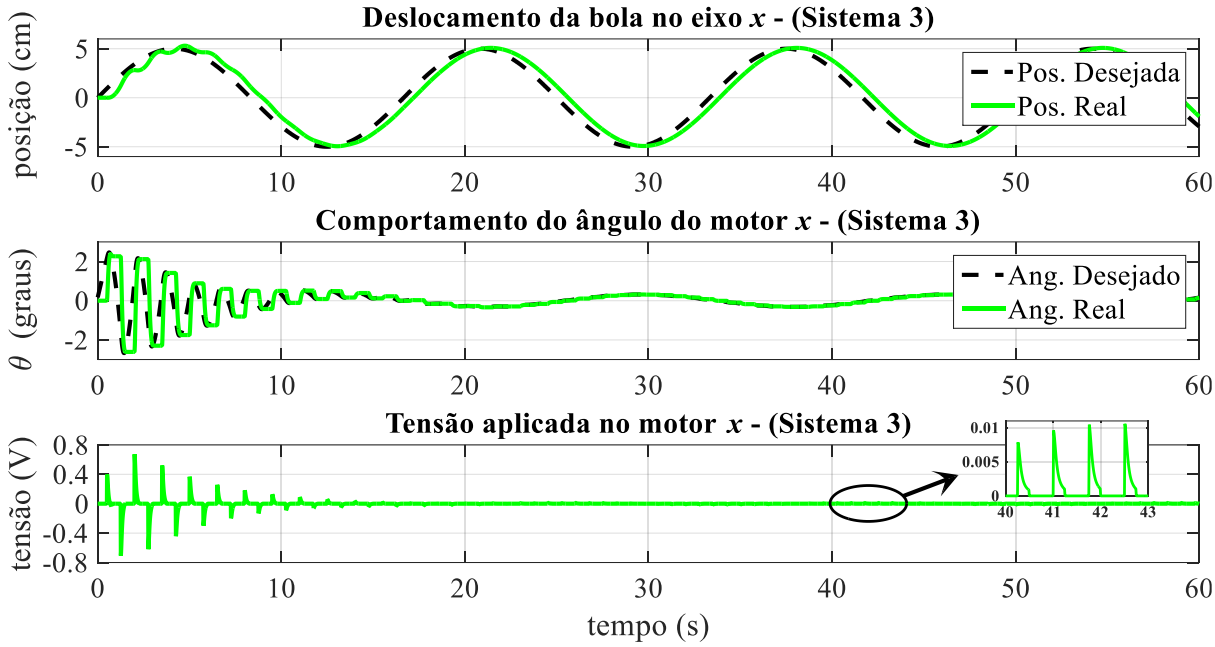
Fonte: próprio autor.

Figura 52 – Resultado de simulação do controlador PID do segundo sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



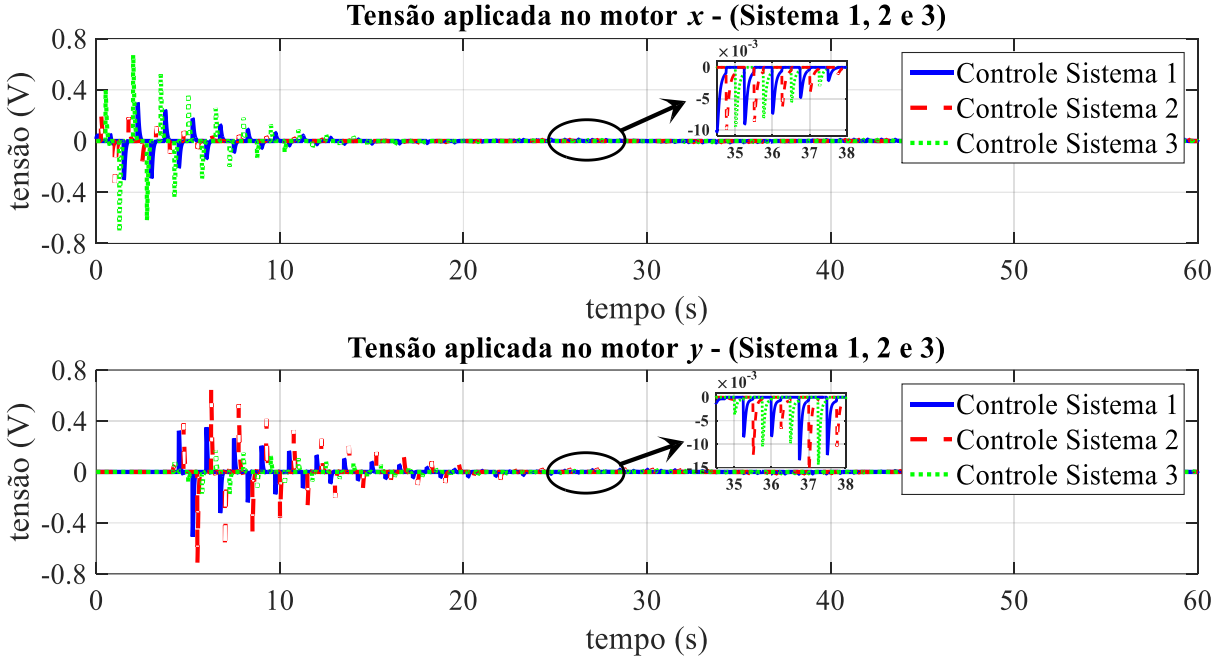
Fonte: próprio autor.

Figura 53 – Resultado de simulação do controlador PID do terceiro sistema sem o seletor de controle, para posição da bola, ângulo de carga, e tensão aplicada no motor referente ao eixo *x*, para referência senoidal em preto.



Fonte: próprio autor.

Figura 54 – Resultado de simulação do controlador PID de três sistemas *Ball Balancer*, sem o seletor de controle para referência senoidal.



Fonte: próprio autor.

Nesse capítulo observou-se que o controle sem seletor apresentou mau funcionamento com o tempo de atuação em cada *Ball Balancer* em $250\ ms$; com esse mesmo tempo não houve problemas quando a quantidade de sistemas *Ball Balancer* eram dois. Destaque-se que para a referência senoidal esse problema não acontece, visto que o sinal de referência da senoide é contínuo e para referência quadrada existe uma descontinuidade (degrau).

Assim mesmo, foram realizadas simulações que demonstraram que para tempos de atuação iguais a $900\ ms$ em cada *Ball Balancer*, o sistema ficou instável, mesmo para referência senoidal.

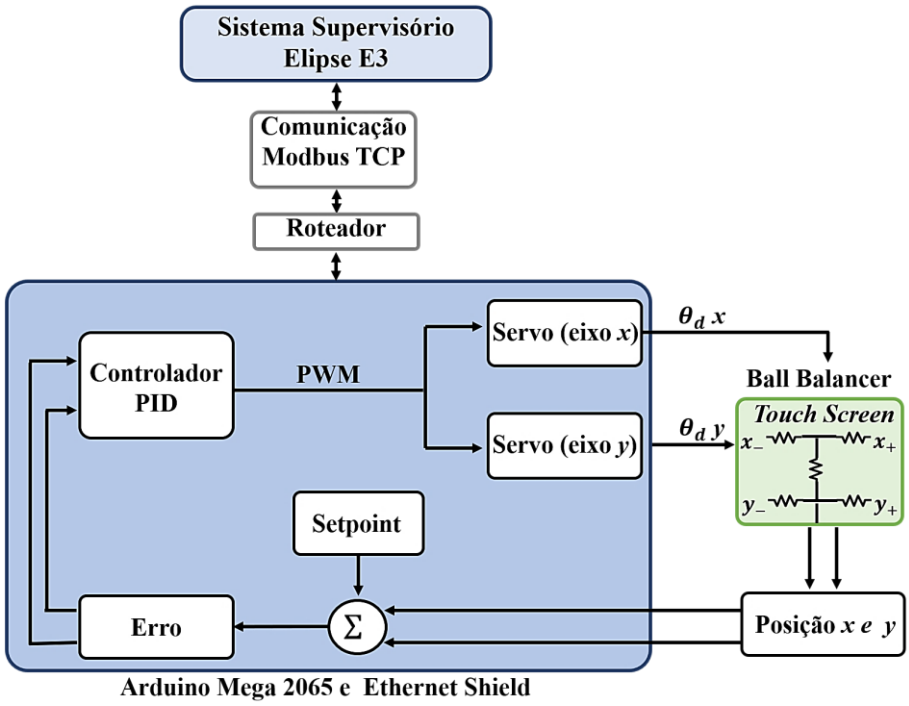
7 MATERIAIS E MÉTODOS

Este capítulo trata-se da descrição dos componentes eletrônicos e *softwares* utilizados para controlar e supervisionar o sistema.

7.1 DISPOSITIVOS DO SISTEMA

O sistema funciona pela interação de diversos dispositivos e sua estrutura geral pode ser visualizada através do diagrama de blocos apresentado na Figura 55, e os componentes do diagrama serão descritos em tópicos posteriores.

Figura 55 – Diagrama de blocos do projeto para um sistema *Ball Balancer*.



Fonte: próprio autor.

O sistema *2D Ball Balancer* utilizado possui uma tela *touch screen* retangular (com dimensões de 20 cm e 30 cm) sobre uma plataforma, que está acoplada a dois servomotores (um no eixo *x* e outro no eixo *y*), onde uma esfera pode ser colocada e movimenta-se com dois graus de liberdade (2 DOF). A posição da esfera é medida através da tela *touch screen* Playtix® resistiva 4 vias, e uma placa Arduino Mega 2560® realiza a interface de comunicação entre o computador (com sistema supervisório) e a tela *touch screen* para determinação da posição cartesiana da esfera sobre a placa, bem como a interface de comunicação entre computador e

os dispositivos de atuação do sistema (estes atuadores são servomotores HexTronik HXT12K® com controle de posição interno em malha-fechada).

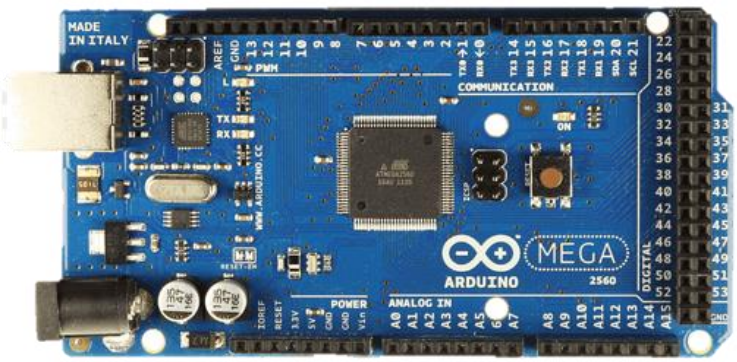
7.1.1 Central de Controle com Microcontrolador

É o *hardware* do sistema embarcado responsável por todo o gerenciamento do sistema. Foi implementado por meio da placa Arduino Mega 2560® que é baseada em um microcontrolador ATmega 2560® que trabalha com *clock* $16\,MHz$ e conversor analógico-digital (AD – Analógico Digital) de $10\,bits$ de resolução com 16 canais.

Este Arduino possui 54 pinos de entradas/saídas digitais (operam a $5\,Volts$, cada pino pode fornecer ou receber um máximo de $40\,mA$) onde 15 destes podem ser utilizados como saídas PWM (*Pulse Width Modulation* – Modulação por Largura de Pulso), 16 entradas analógicas, 4 portas seriais ou UARTs (*Universal Asynchronous Receiver/Transmitter* - Receptor/transmissor assíncrono universal), uma conexão USB (*Universal Serial Bus*), uma entrada de alimentação (tensão de entrada limite $6\text{-}20\,Volts$, tensão de operação da placa $5\,Volts$), uma conexão ICSP (*In-Circuit Serial Programming*) e um botão de reset. O Atmega2560® tem $256KB$ de memória *flash* para armazenamento de código (dos quais $8\,KB$ são usados pelo bootloader), $8KB$ de SRAM e $4KB$ de EEPROM (ARDUINO IDE, 2020).

O algoritmo para leitura das coordenadas da tela *touch screen* e do controlador PID está embarcado na central de controle (Arduino Mega), bem como para interface de comunicação entre o computador com sistema supervisório e tela *touch screen*. Na Figura 56 é apresentado o Arduino com pinos de Entrada/Saída Digital, as entradas analógicas, as portas de alimentação, entradas de comunicação serial, entre outros.

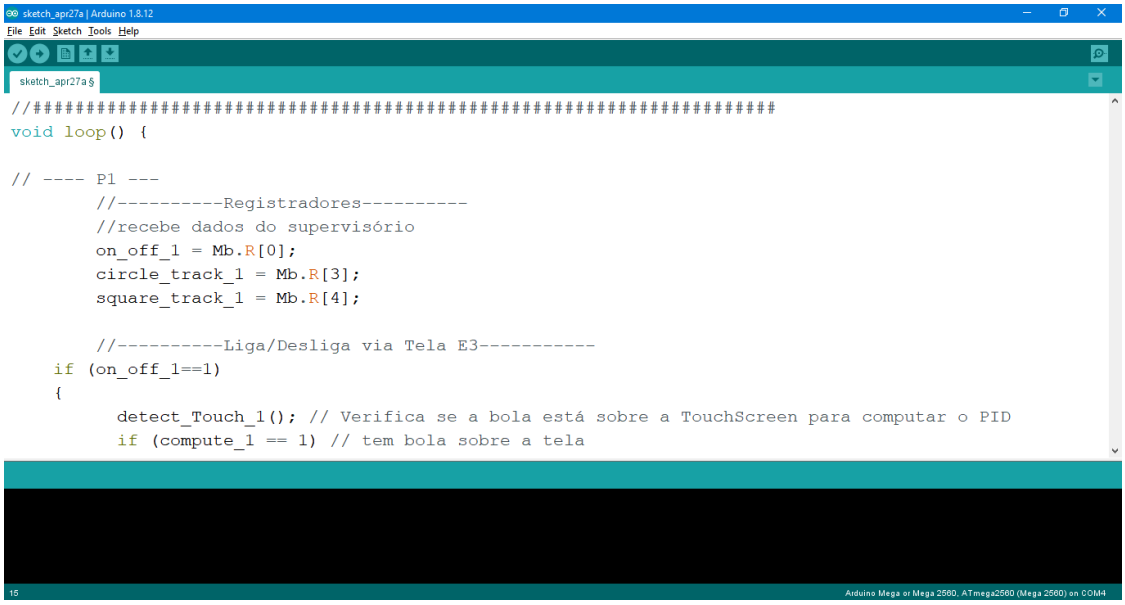
Figura 56 – Placa Arduino Mega 2560.



Fonte: adaptado pelo autor.

O *software* de programação do Arduino Mega está disponível para *download* no site do fabricante (ARDUINO IDE, 2020). Na Figura 57 é mostrado a interface do *software* Arduino (IDE - *integrated development environment*) 1.6.2 usado para programação do Arduino Mega. Este é um *software* livre (plataforma aberta) capaz de rodar em Windows, Mac OS X e Linux, e pode ser usado para programar qualquer placa Arduino. O ambiente é escrito em Java e baseada em processamento e outros *softwares* de código aberto (ARDUINO IDE, 2020).

Figura 57 – *Software* de ambiente de desenvolvimento integrado (IDE).

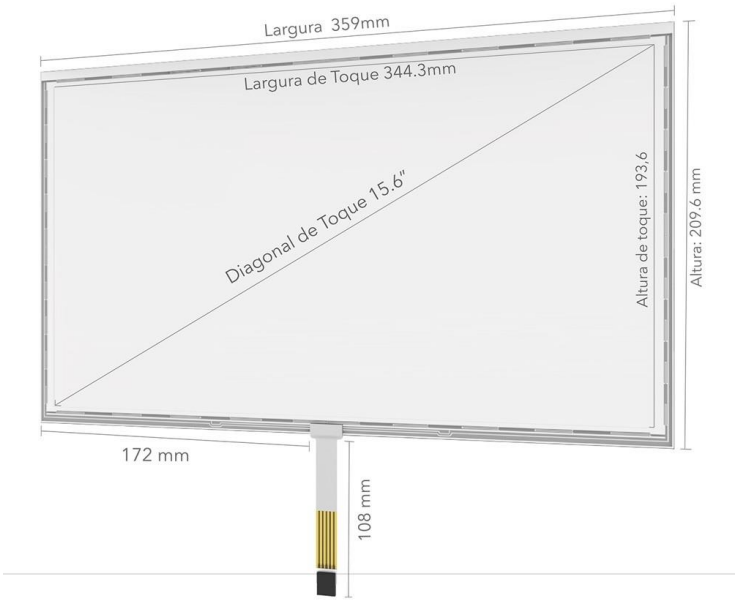


Fonte: próprio autor.

7.1.2 Tela Resistiva

Um dos principais aspectos para se ter um bom resultado na implementação de um controle para o sistema *Ball Balancer*, é a técnica de sensoriamento da esfera. Para realizar esta tarefa, a literatura está dividida principalmente entre dois métodos distinto (BRAESCU *et al*., 2012). Um dos modos consiste na utilização de uma tela sensível ao toque acoplado a placa, e outro método é o monitoramento por um dispositivo de aquisição de imagens como uma *webcam*. A tela *touch screen* apresenta vantagem de permitir um período de amostragem menor que o tempo de aquisição de quadros das *webcams* comerciais (BRAESCU *et al*., 2012; KOPICHEV; PUTOV; PASHENKO, 2019; SUMEGA *et al*., 2018; ZIA, 2011). Neste trabalho, o sistema *Ball Balancer* considerado utiliza uma tela resistiva sensível ao toque (*touch screen*) de dimensão 15,6 polegadas de 4 vias, na Figura 58 é mostrada a tela, e na Tabela 7 é mostrada as especificações da tela *touch screen*.

Figura 58 – Tela resistiva de 4 vias de dimensão 15,6 polegadas.



Fonte: (PLAYTIX, 2020).

Tabela 8 – Especificações da tela *touch screen*.

Tamanho da tela	15,6 polegadas
Tipo	Filme de vidro
Vida útil baseado em 250 gramas de força	1,000,000 de toques
Espessura do componente *touch*	3,80 mm de espessura
Temperatura de operação	Entre $-10°c$ e $+60°c$
Tensão de operação	Menor ou igual a 10 $Volts$
Tempo de resposta	Menor ou igual a 10 ms
Transmitância	Menor ou igual a 85%
Linearidade	Entre 1,5%

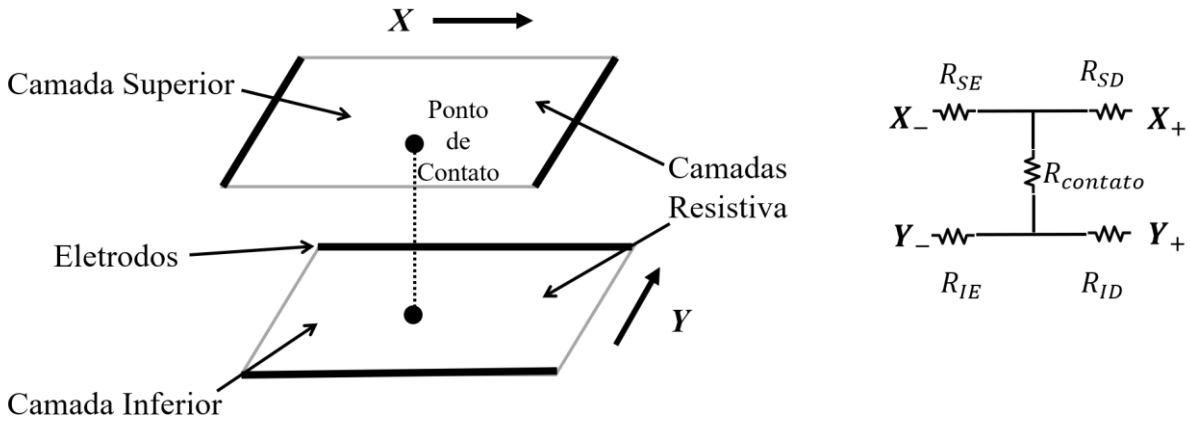
Fonte: (PLAYTIX, 2020).

As telas sensíveis ao toque, consiste em dois eletrodos (barramentos) em forma de tiras retangulares perpendiculares entre si, sendo cada par conectado à uma camada de ITO (*Indium Tin Oxide*) que reveste cada uma das placas. Em meio ao barramento superior e inferior do eletrodo, pontos espaçadores são fixados com um filme condutor e uma camada adesiva para manter os eletrodos separados. Enquanto o eletrodo superior é exposto à pressão com o eletrodo inferior, uma conexão é feita, respondendo à posição no painel *touch screen* em que a pressão

é aplicada. Portanto, o barramento do eletrodo superior é utilizado para determinar a tensão no ponto de pressão, dando uma leitura analógica da posição da tela. Além disso, dado a tela bidimensional, uma leitura analógica fornecerá posteriormente uma posição para cada eixo. O sinal elétrico analógico, será convertido em um sinal digital no microcontrolador usando um conversor AD (CALPE-MARAVILLA *et al*., 2014; TIMPEA; COSMA; SOSDEAN, 2019).

As telas de quatro vias utilizam um par simples de eletrodos em forma de tiras retangulares perpendiculares entre si. Uma placa possui distribuição equipotencial em *x*, e a outra possui distribuição potencial em *y*, conforme Figura 59. Os eletrodos são então conectados a cabos flexíveis que permitem comunicação com um dispositivo externo (BAI; CHEN, 2007). No caso da medição da coordenada *x*, a placa superior transmite um gradiente de tensão gerado pela aplicação de tensão entre os eletrodos superiores, e a leitura desta coordenada é realizada considerando *y* como entrada. A coordenada *x* do ponto de contato, é estimada considerando a resistência equivalente entre o ponto de contato para cada eletrodo linear na camada superior como pode ser observado na Figura 59. Aplicando o processo inverso, pode ser encontrada a coordenada *y* do ponto de contato (AGUILAR; MEIJER, 2002; BEGAM, 2017).

Figura 59 – Tela resistiva de quatro vias e circuito equivalente.



Fonte: adaptado de (AGUILAR; MEIJER, 2002).

7.1.3 Servomotor

Dois servomotores HexTronik HXT12K® são utilizados para garantir o movimento angular da plataforma nos eixos *x* e *y* para um sistema *Ball Balancer*. Estes motores, são acionados por um sinal PWM, e podem ser posicionados entre 0° e 180° na saída de acordo com a largura do pulso (*duty cycle*). Na Figura 60 é mostrado o servomotor utilizado, e na Tabela 8 é apresentado as especificações deste motor.

Figura 60 – Servomotor HexTronik HXT12K®.

Fonte: próprio autor.

Tabela 9 – Especificações do servomotor HexTronik HXT12K®

Modulação	Digital
Torque	4,8V : 12,96 kg − cm 6,0V: 14,98 kg − cm
Velocidade	4,8V: 0,17 sec/60° 6,0V: 0,13 sec/60°
Peso	47,9 g
Dimensão	Comprimento: 39,6 mm Largura: 20,1 mm Altura: 38,1 mm
Engrenagem	Metal

Fonte: (SERVODATABASE, 2020).

Os componentes básicos do servomotor padrão, são um potenciômetro, um motor de corrente contínua, e um circuito de controle. Conforme o motor gira, o potenciômetro administra uma mudança na resistência. O circuito de controle (malha fechada implementada internamente no próprio circuito) pode, assim, regular a rotação deste motor e determinar a direção. Neste projeto, o servomotor possui três conexões de alimentação disponíveis ao usuário, sendo utilizado como padrão o fio preto como referência, o fio vermelho como sinal de alimentação (geralmente 5 Volts) e o amarelo como sinal de controle PWM (SAWICZ, 2002). O controle destes motores ocorre através de um sinal de pulso com período de 20 ms, em que esses pulsos são gerados através da saída de PWM do microcontrolador (SAPUTRA, 2019). A largura de pulso determina a posição do eixo do motor de forma absoluta, sendo que

a largura válida para a movimentação do servomotor varia de 1 ms a 2 ms, ou seja, o período de aproximadamente 1,5 ms corresponde à metade do curso do servomotor (AKHMAD; TOLA; DJAMALUDDIN, 2018; MIT, 2009).

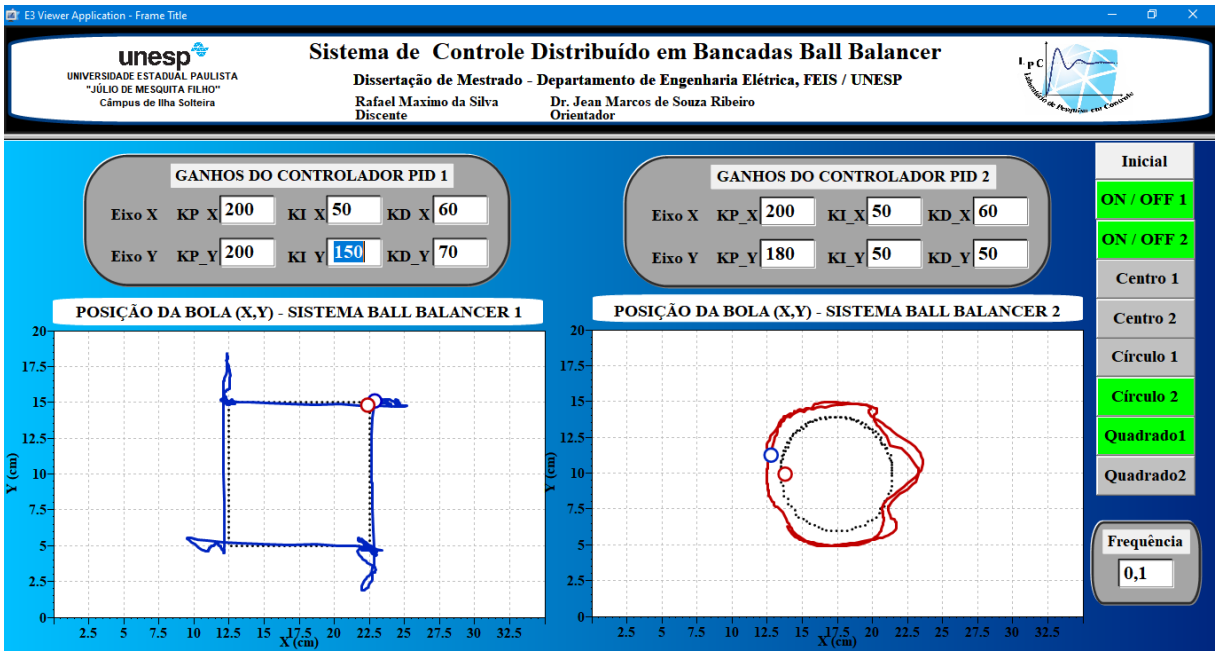
7.1.4 Sistema Supervisório

Os sistemas supervisórios, também chamados de SCADA (*Supervisory Control And Data Acquisition*), permitem monitorar e rastrear informações de vários tipos de processos. Isto é feito através de equipamentos de aquisição de dados relativos ao sistema, e em seguida essas informações são manipuladas, analisadas, armazenadas e, posteriormente, apresentadas ao usuário (ELIPSE, 2019). Portanto, o sistema de supervisão é uma representação virtual do processo ou planta física a controlar, que possui uma representação em tempo real das principais variáveis monitoradas ou controladas, tendo como objetivo principal de proporcionar uma interface de alto nível do operador com o processo, informando-o em tempo real de todos os eventos da planta, bem como possibilitar uma intervenção direta no processo (BOYER, 2004).

A principal funcionalidade de qualquer sistema SCADA está ligada à troca de informações, que pode ser uma comunicação com os equipamentos de campo (controlador etc.), realizada através de um protocolo em comum, cuja metodologia pode ser tanto de domínio público ou de acesso restrito (ELIPSE, 2019). Com a aplicação de protocolos de comunicação digital padronizados, tais como, Foundation Fielddbus, Profibus, Modbus e entre outras, em sistemas de automação, estas redes possibilitam a integração de equipamentos de vários fabricantes distintos (CASTRUCCI; MORAES, 2008).

Para elaboração do *software* supervisório desta dissertação, utilizou-se o Elipse E3® na sua versão gratuita para demonstração, e o protocolo de rede de comunicação utilizado foi o Modbus TCP por meio de uma porta Ethernet conectada entre o microcontrolador (Arduino e Ethernet Shield), roteador e computador. Na Figura 61 é apresentada a tela desenvolvida para supervisão de duas plantas *Ball Balancer* no Elipse E3®, onde pode ser definido as trajetórias desejadas para rastreamento, bem como determinar os ganhos do controlador PID.

Figura 61 – Sistema de supervisão para dois sistemas *Ball Balancer*.

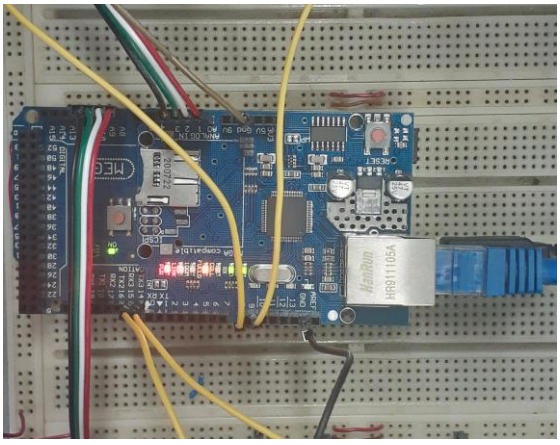


Fonte: próprio autor.

7.1.5 Arduino Ethernet Shield

Foi utilizado o Arduino Ethernet Shield para realizar a comunicação entre o Arduino Mega e o supervisório. O Ethernet Shield, conforme apresentado na Figura 62, baseia-se no chip WIZnet ethernet W5100 que fornece acesso à rede (IP – *Internet Protocol*) nos protocolos TCP (*Transmission Control Protocol*) ou UDP (*User Datagram Protocol*). Para estabelecer a comunicação, primeiramente adicionou-se uma biblioteca Modbus TCP no Arduino e logo após foi criada uma rede LAN (*Local Area Network*) por meio de um roteador.

Figura 62 – Arduino Ethernet Shield.



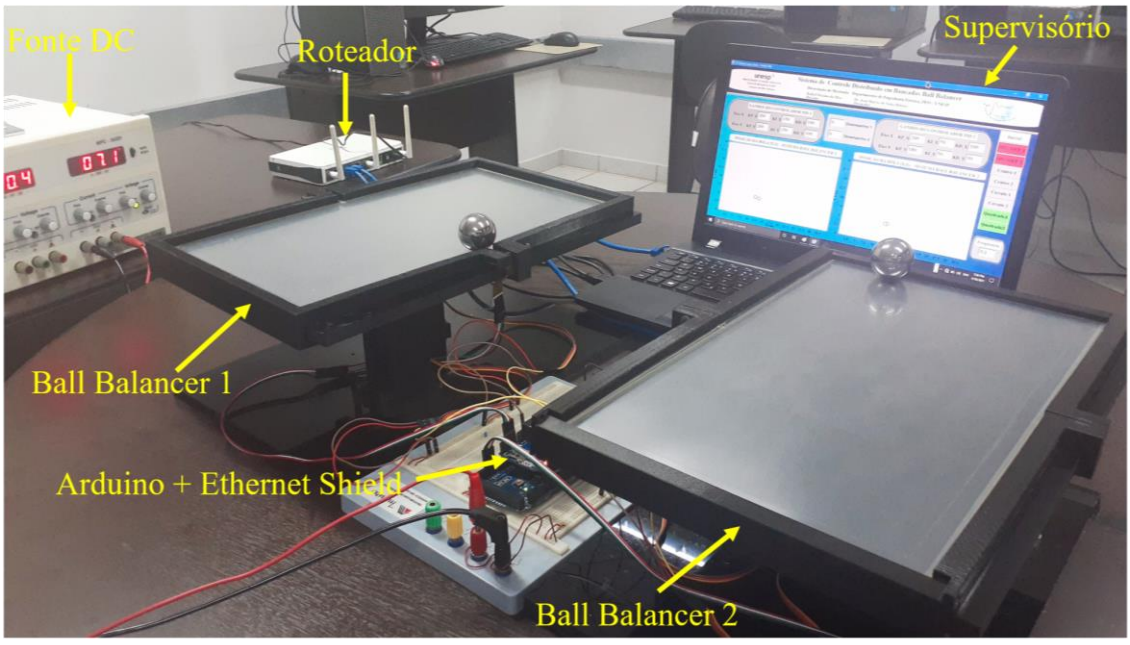
Fonte: próprio autor.

7.2 CONTROLE DISTRIBUÍDO APLICADO EM PLANTAS *BALL BALANCER*

O sistema implementado conforme Figura 63, possui duas plantas *Ball Balancer*, em que cada planta possui uma tela resistiva de quatro vias e dois servomotores, um para cada eixo (*x* e *y*). Para detectar as posições *x* e *y* da esfera sobre a tela, foi feita a calibração da tela através de um algoritmo para leitura analógica implementado no Arduino, considerando o mapeamento de tensões entre $0\,Volts$ e $5\,Volts$ para valores inteiros entre 0 e 1023 (considerando o conversor analógico-digital de 10 bits), em seguida fez-se a conversão de cada eixo para centímetro.

Para alimentação dos servomotores utilizou-se uma fonte CC com tensão de $7\,Volts$, pois o Arduino não fornece corrente suficiente para os servomotores. A comunicação entre Arduino e supervisório, foi realizada através de um Arduino Ethernet Shield conectado a uma LAN com o roteador. Incluiu-se as configurações do endereço Modbus TCP no Arduino, bem como as variáveis (registros Modbus) para comunicação com o supervisório. O Supervisório foi utilizado para ajuste de ganhos do controlador PID, escolha da posição desejada de rastreamento (centro, círculo ou quadrado), e visualização de gráficos de cada planta; para esse fim, adicionou-se um driver Modbus no supervisório e fez-se as configurações de endereço e variáveis, chamadas de *tags*, para permitir a comunicação com o Arduino.

Figura 63 – Sistema distribuído com duas plantas em laboratório.



Fonte: próprio autor.

8 RESULTADOS OBTIDOS

Este capítulo apresenta os resultados obtidos do controle PID do primeiro e segundo *Ball Balancer*, sistema de controle distribuído com duas plantas *Ball Balancer* sem o índice de desempenho da estratégia de controle proposta, e posteriormente apresenta-se o resultado obtido com a estratégia de controle distribuído proposta neste trabalho, que mostrará qual é o sistema que se encontra em pior situação, para assim o sinal de controle ser enviado para apenas uma planta por vez.

8.1 RESULTADOS DO CONTROLE PID DE DUAS PLANTAS *BALL BALANCER*

Os controles individuais das plantas e o controle simultâneo das duas plantas, sem utilizar o índice de desempenho, são apresentados nesta seção.

8.1.1 Controlador PID do primeiro *Ball Balancer*

Para que o sistema tenha uma resposta dentro do desempenho desejado, é necessário aplicar métodos de sintonia para que o controlador otimize o processo. Neste trabalho foi adotado o ajuste manual (ajuste fino) dos ganhos, que consiste em analisar o comportamento do processo e estabelecer os valores de ganho apropriados para o processo com base nas respostas obtidas. Desse modo, na Tabela 10 é mostrado os ganhos PID para o primeiro *Ball Balancer*, em que a planta apresentou uma melhor resposta considerando estes ganhos, para o controle de posição da esfera no centro, bem como para as referências quadrada e senoidal.

Tabela 10 – Ganhos do controlador PID para o primeiro *Ball Balancer*.

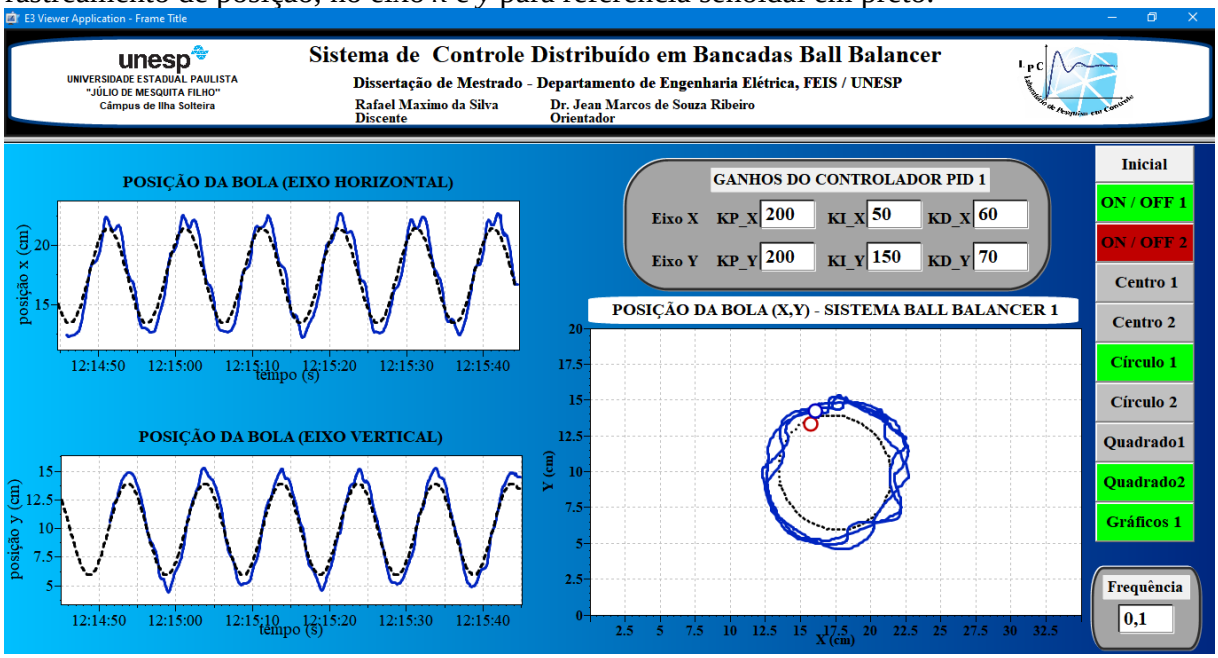
Ganhos	**Descrição**	**Eixo *x***	**Eixo *y***
K_p	Ganho K_p do *Ball Balancer* (rad/m)	0,20	0,20
K_i	Ganho K_i do *Ball Balancer* $(rad/m/s)$	0,05	0,15
K_d	Ganho K_d do *Ball Balancer* $(rad.s/m)$	0,06	0,07

Fonte: próprio autor.

Na Figura 64 é apresentada a tela do supervisório com o resultado de implementação do controle de um sistema *Ball Balancer*, para o rastreamento em onda senoidal (referência em preto), apresenta-se o controle de posição das coordenadas x e y ($x = 17{,}5 + 40.\cos(2\pi.f.t)$ cm e $y = 10 + 40.sen(2\pi.f.t)\ cm$) do primeiro sistema, considerando a frequência igual a $0{,}1\ Hz$. Observa-se que o resultado apresenta uma repetibilidade na resposta,

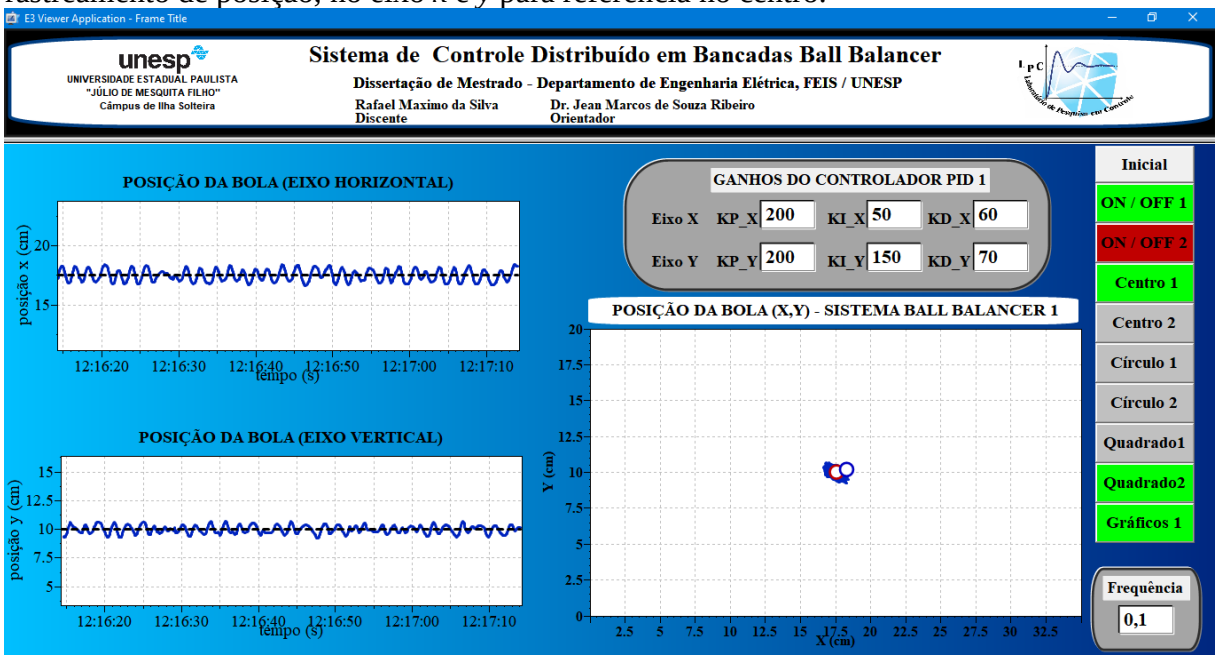
e baixo sobressinal para ambos os eixos. Na Figura 65 é mostrado o resultado de implementação do controle do primeiro sistema *Ball Balancer*, considerando a referência no centro da tela ($x = 17{,}5$ cm e $y = 10\,cm$), apresenta-se o controle de posição das coordenadas x e y. Também foi realizada a implementação para o rastreamento em onda quadrada para o primeiro sistema, conforme Anexo A.

Figura 64 – Resultado de implementação do controlador PID no primeiro sistema, do rastreamento de posição, no eixo *x* e *y* para referência senoidal em preto.



Fonte: próprio autor.

Figura 65 – Resultado de implementação do controlador PID no primeiro sistema, do rastreamento de posição, no eixo *x* e *y* para referência no centro.



Fonte: próprio autor.

8.1.2 Controlador PID do segundo *Ball Balancer*

Para a segunda planta, após o ajuste manual dos ganhos, realizado pela análise do comportamento das respostas obtidas experimentalmente, foram estabelecidos os valores de ganhos apropriados para o segundo sistema. Na Tabela 11 é mostrado os ganhos do controlador PID para o segundo *Ball Balancer*, em que a planta apresentou uma melhor resposta com os ganhos ajustados, para o controle de posição da esfera no centro, e para as referências quadrada e senoidal.

Tabela 11 – Ganhos do controlador PID para o segundo *Ball Balancer*.

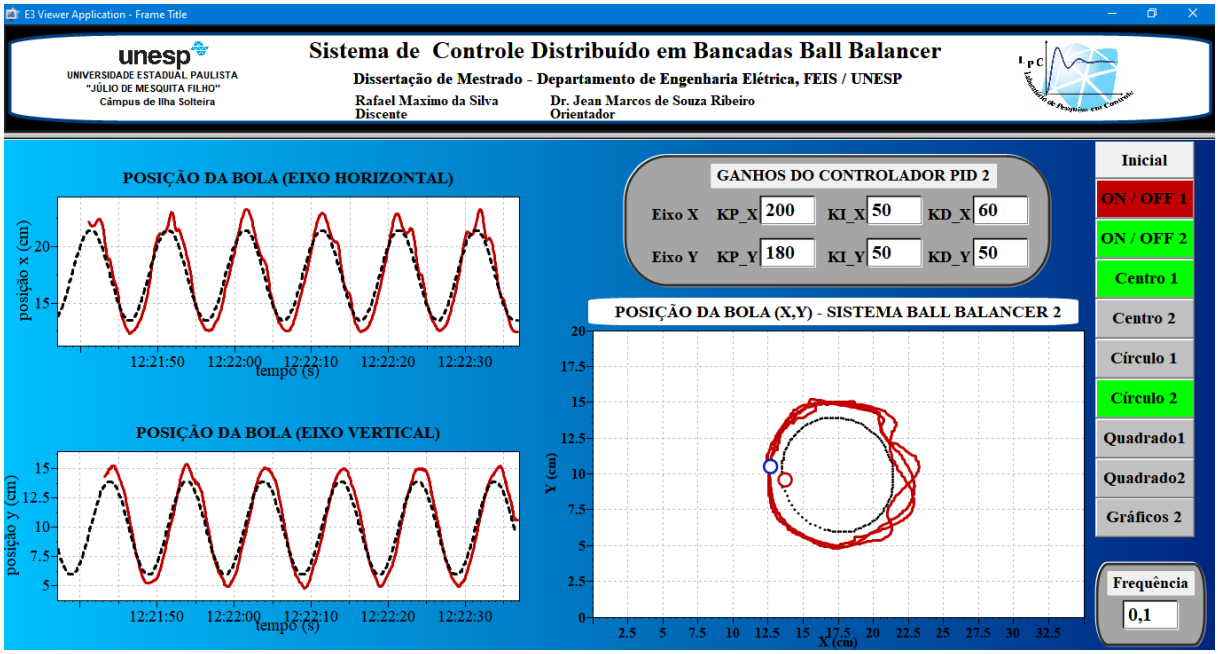
Ganhos	**Descrição**	**Eixo *x***	**Eixo *y***
K_p	Ganho K_p do *Ball Balancer* (rad/m)	0,20	0,18
K_i	Ganho K_i do *Ball Balancer* $(rad/m/s)$	0,05	0,05
K_d	Ganho K_d do *Ball Balancer* $(rad.s/m)$	0,06	0,05

Fonte: próprio autor.

Na Figura 66 é apresentada a tela do supervisório com o resultado de implementação do controle, considerando o segundo sistema *Ball Balancer*, para o rastreamento em onda senoidal (referência em preto), apresenta-se o controle de posição das coordenadas x e y ($x = 17{,}5 + 40.\cos(2\pi.f.t)$ cm e $y = 10 + 40.sen(2\pi.f.t)\ cm$), considerando a frequência igual a $0{,}1\ Hz$.

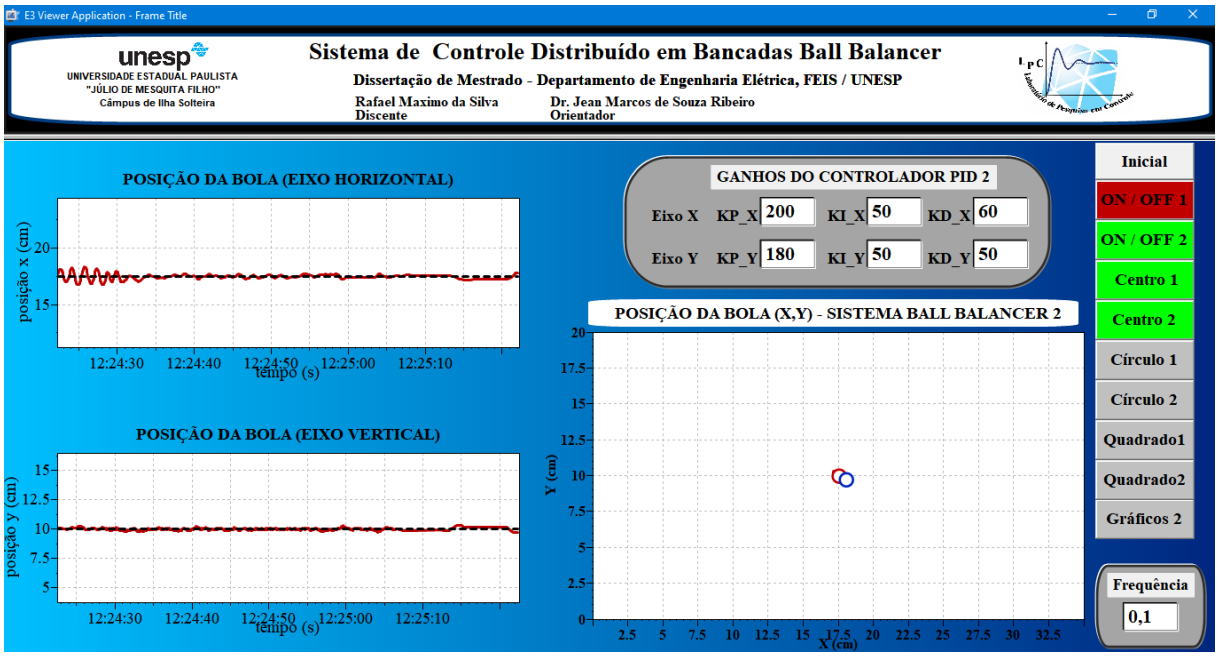
Na Figura 67 é mostrado o resultado de implementação do controle do segundo sistema *Ball Balancer*, considerando a referência no centro da tela ($x = 17{,}5$ cm e $y = 10\ cm$), apresenta-se o controle de posição das coordenadas x e y. Também foi realizada a implementação para o rastreamento em onda quadrada para o segundo sistema, conforme apresentado em Anexo A.

Figura 66 – Resultado de implementação do controlador PID no segundo sistema, do rastreamento de posição, no eixo *x* e *y* para referência senoidal em preto.



Fonte: próprio autor.

Figura 67 – Resultado de implementação do controlador PID no segundo sistema, do rastreamento de posição, no eixo *x* e *y* para referência no centro.



Fonte: próprio autor.

8.1.3 Resultado de implementação do controle PID com dois *Ball Balancer*

A análise com as duas plantas foi implementada, com objetivo de testar posteriormente a estratégia de controle distribuído levando em conta o índice de desempenho proposto neste trabalho. Deste modo, foi elaborada uma tela para visualização do controle de posição das duas plantas ligadas simultaneamente com o algoritmo de controle no Arduino Mega.

Os ganhos do controlador PID para cada planta encontra-se na Tabela 10 e 11. Para se obter o controle das duas plantas com um Arduino, foram realizados testes variando-se o tempo de amostragem (escolheu-se $T_s = 50\,ms$) para a avaliação do PID, já que o Arduino possui somente um conversor AD (de $10\,bits$), dificultando-se assim o processamento e conversão de leituras analógicas para digitais quando considerado as duas plantas *Ball Balancer*. Vale ressaltar que a ideia inicial era de considerar dois Arduinos para o controle, um para cada planta, contudo, não foi possível a realização da mesma por limitação do sistema supervisório de considerar apenas um Driver Modbus TCP na versão gratuita (permite a comunicação com apenas um dispositivo ou endereço IP).

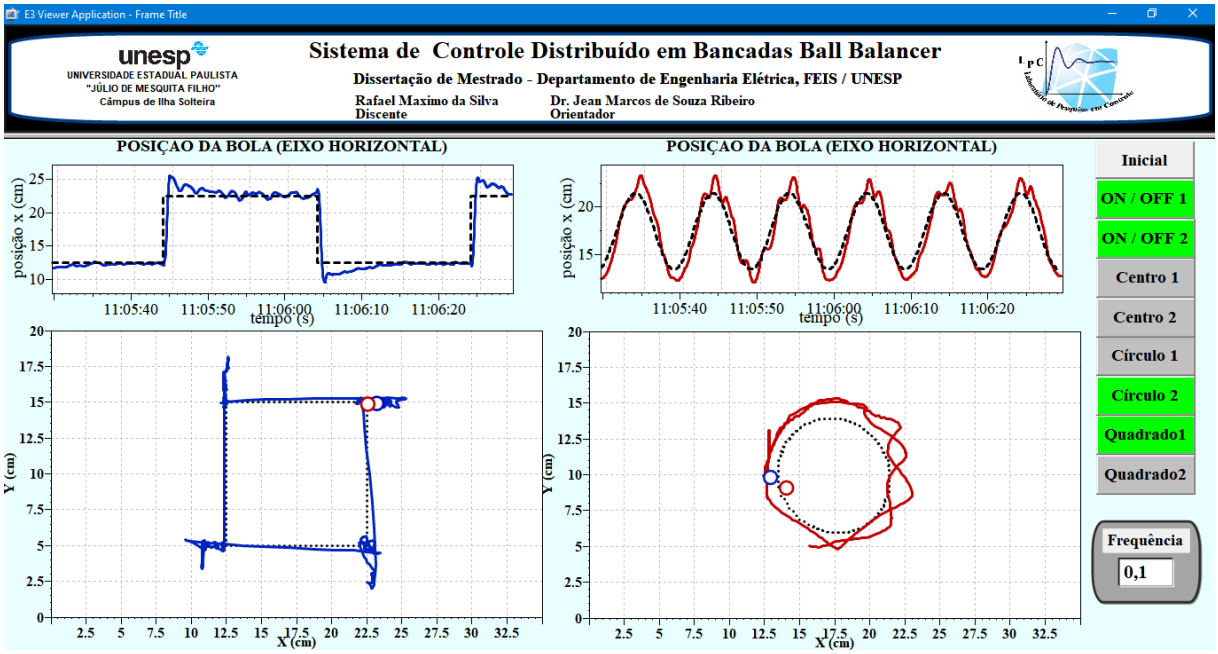
Na Figura 68 é apresentado o resultado de implementação do controle distribuído sem considerar a estratégia de controle proposta, considerando o rastreamento em onda quadrada (referência em preto) para o primeiro *Ball Balancer* (em azul), apresentando-se o controle de posição das coordenadas x e y, observa-se que a primeira planta apresenta tempo de resposta de aproximadamente $2\,s$. É apresentado também o resultado de implementação do controle, considerando o rastreamento em onda senoidal (referência em preto) para o segundo *Ball Balancer* (em vermelho).

Na Figura 69 é mostrado o resultado de implementação do controle para o primeiro e segundo sistema *Ball Balancer*, considerando a referência senoidal, apresenta-se o controle de posição das coordenadas x e y. Além disso, foram realizadas implementações para diferentes combinações de rastreamento (centro, quadrado, circulo) para essa mesma configuração de tela elaborada, conforme apresentado no Anexo B.

Na Figura 70 é apresentada a tela do supervisório com o resultado de implementação do controle de duas plantas *Ball Balancer*, para o rastreamento em onda senoidal (referência em preto) considerando a frequência igual a $0{,}1\,Hz$, apresenta-se o controle de posição das coordenadas x e y, e o sinal de controle aplicado no servomotor do eixo *x*. Nota-se que o resultado do sinal de controle da segunda planta, apresenta um valor maior comparado com a

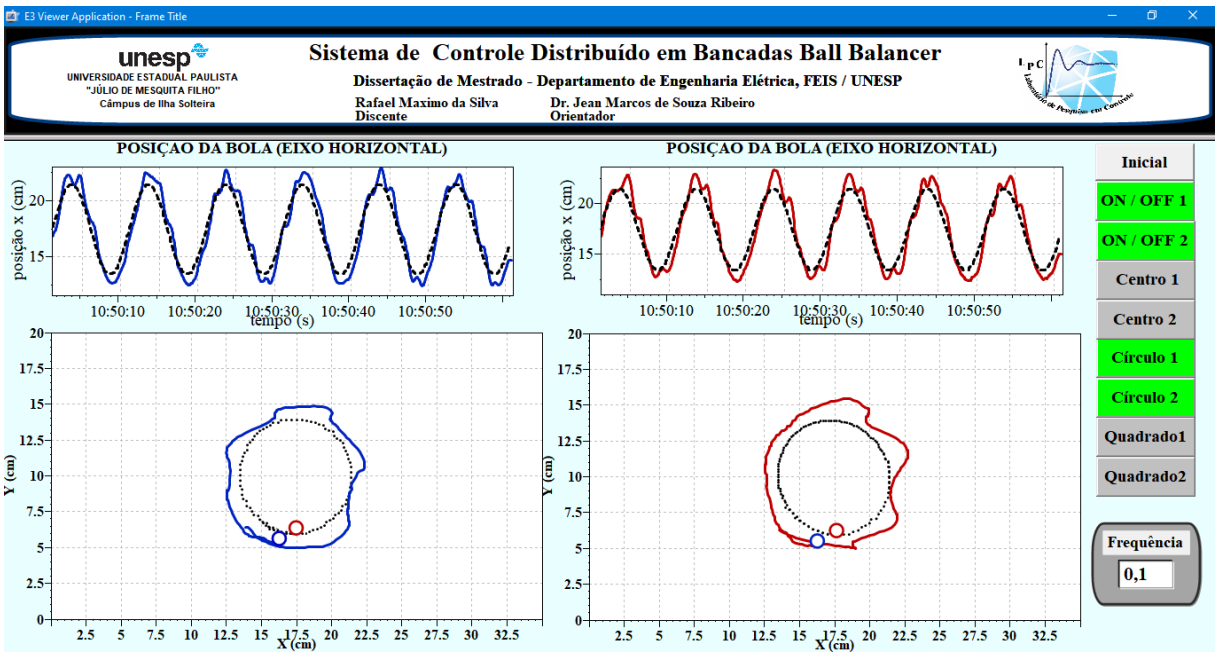
primeira planta devido ao diferente valor inicial de cada planta. Na Figura 71 é mostrado para o rastreamento em onda quadrada.

Figura 68 – Resultado de implementação do controlador PID no primeiro (em azul) e segundo (em vermelho) sistema, do rastreamento de posição no eixo *x* e y, para referência quadrada e senoidal em preto.



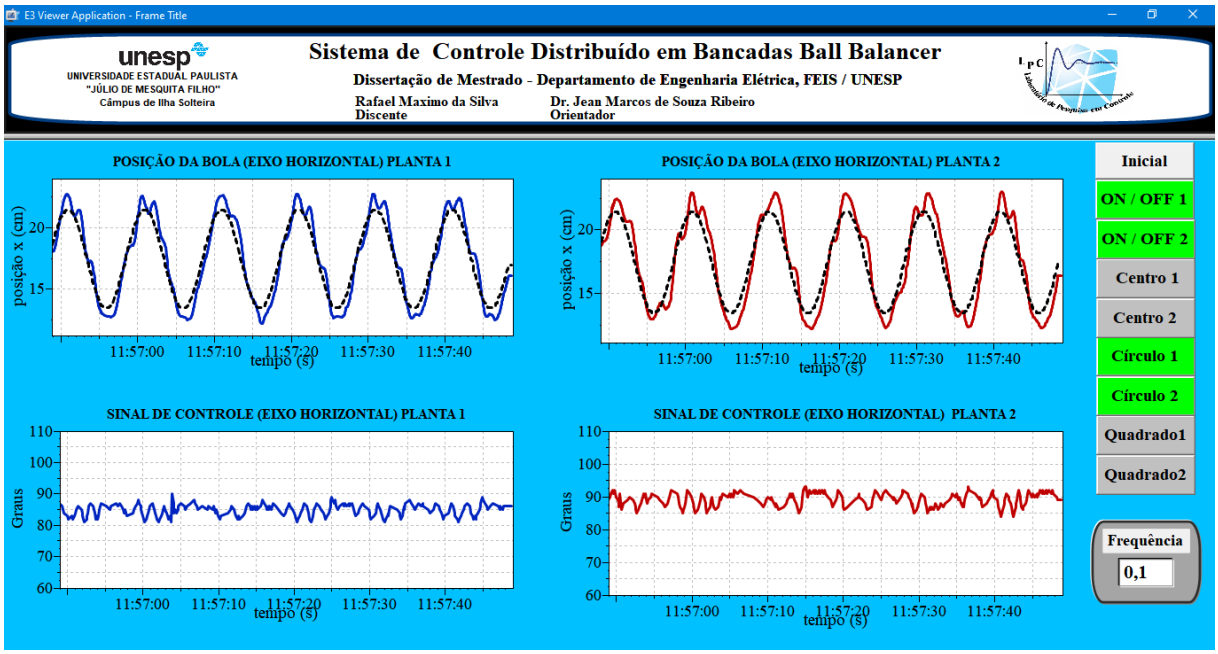
Fonte: próprio autor.

Figura 69 – Resultado de implementação do controlador PID no primeiro (em azul) e segundo (em vermelho) sistema, do rastreamento de posição no eixo *x* e y, para referência senoidal em preto.



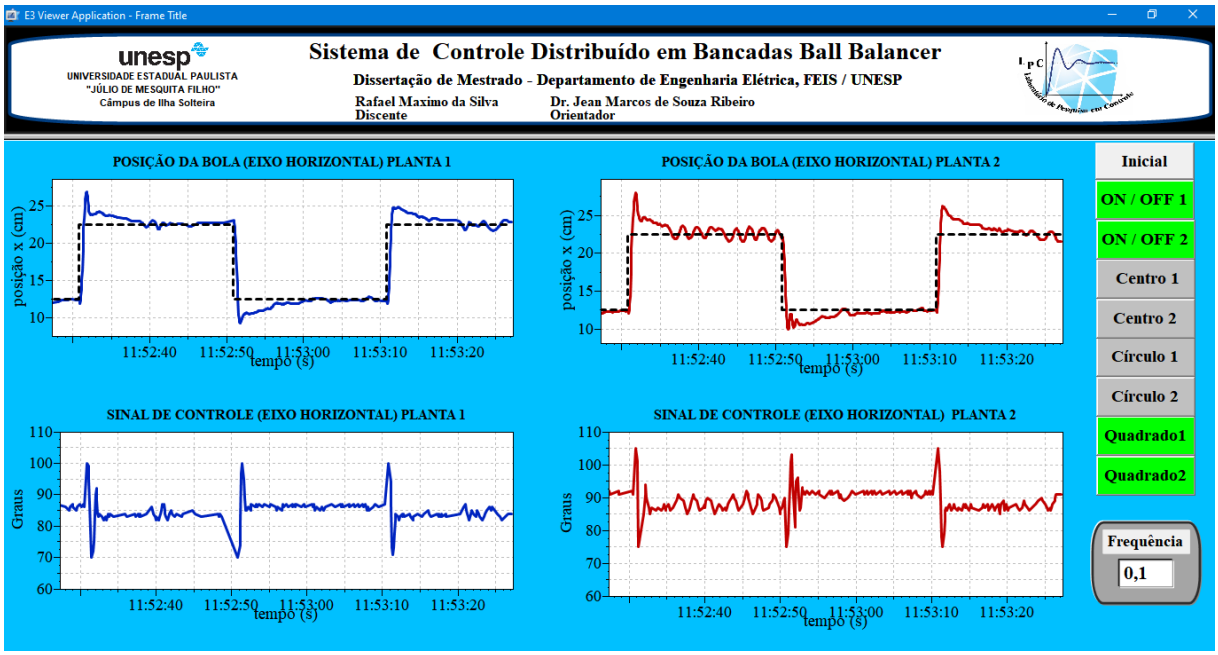
Fonte: próprio autor.

Figura 70 – Resultado de implementação do controlador PID no primeiro (em azul) e segundo (em vermelho) sistema, do rastreamento de posição no eixo *x* e y, para referência senoidal em preto, e o sinal de controle do eixo horizontal.



Fonte: próprio autor.

Figura 71 – Resultado de implementação do controlador PID no primeiro (em azul) e segundo (em vermelho) sistema, do rastreamento de posição no eixo *x* e y, para referência quadrada em preto, e o sinal de controle do eixo horizontal.



Fonte: próprio autor.

No Anexo B é apresentado as demais implementações com diferentes combinações de rastreamento (centro, quadrado, circulo).

8.2 RESULTADOS DO CONTROLE PID COM A ESTRATÉGIA DE CONTROLE DISTRIBUÍDO

Nesta seção, apresenta-se o resultado da implementação da estratégia de controle distribuído, que através de um índice de desempenho, apenas uma planta recebe o sinal de controle em cada período de amostragem. O algoritmo desta estratégia de controle distribuído, considera o tempo de amostragem de $T_s = 20\ ms$ para a avaliação do controlador PID.

Foram realizados vários testes no algoritmo levando em conta a condição de índice de desempenho, com a ideia de obter o controle das duas plantas. Desse modo, nas Tabelas 12 e 13 é apresentado os ganhos PID para o primeiro e segundo *Ball Balancer*, em que cada planta apresentou uma melhor resposta considerando estes ganhos ajustados manualmente, para o controle de posição da esfera no centro.

Tabela 12 – Ganhos do controlador PID distribuído do primeiro *Ball Balancer*.

Ganhos	**Descrição**	**Eixo *x***	**Eixo *y***
K_p	Ganho K_p do *Ball Balancer* (rad/m)	0,30	0,30
K_i	Ganho K_i do *Ball Balancer* $(rad/m/s)$	0,15	0,15
K_d	Ganho K_d do *Ball Balancer* $(rad.s/m)$	0,1	0,1

Fonte: próprio autor.

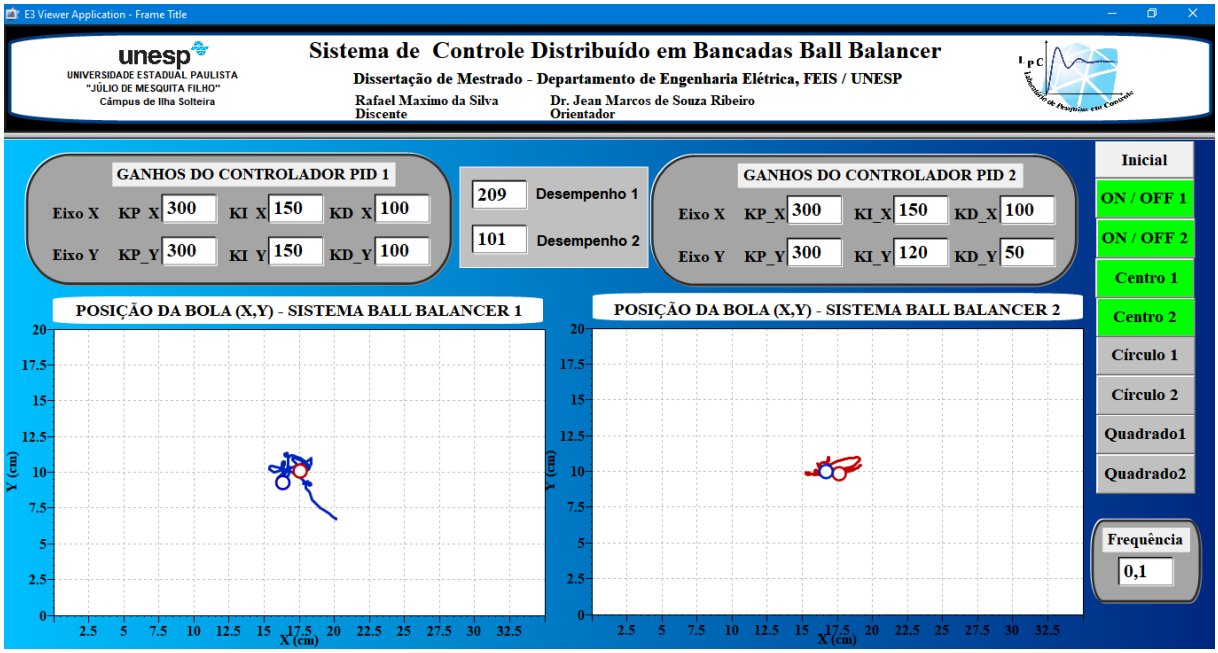
Tabela 13 – Ganhos do controlador PID distribuído do segundo *Ball Balancer*.

Ganhos	**Descrição**	**Eixo *x***	**Eixo *y***
K_p	Ganho K_p do *Ball Balancer* (rad/m)	0,30	0,30
K_i	Ganho K_i do *Ball Balancer* $(rad/m/s)$	0,15	0,12
K_d	Ganho K_d do *Ball Balancer* $(rad.s/m)$	0,10	0,05

Fonte: próprio autor.

Na Figura 72 é apresentada a tela do supervisório com o resultado de implementação do controle de duas plantas *Ball Balancer* considerando o índice de desempenho proposto, para o controle de posição das coordenadas x e y no centro ($x = 17{,}5$ cm e $y = 10\ cm$). Nota-se que o desempenho da planta 1 é maior comparado com o desempenho da planta 2 para o instante de amostragem, portanto as plantas disputam a cada instante de amostragem pelo sinal de controle.

Figura 72 – Resultado de implementação do controlador PID com a estratégia de controle distribuído aplicada em plantas *Ball Balancer*.



Fonte: próprio autor.

Foi testado também para o controle de posição, considerando a primeira planta com referência senoidal e a segunda com a referência no centro, portanto, as duas plantas ficaram instáveis, conforme Anexo C. Acredita-se que devido ao Arduino possuir apenas um conversor analógico (usado para leitura da posição da esfera de cada planta) bem como velocidade de processamento de apenas 16 MHz, limitou-se a validação da estratégia de controle proposta para as referências quadrada e senoidal (ficando-se instáveis), contudo, foi possível verificar que a estratégia funciona com duas plantas considerando o *setpoint* no centro para ambas as plantas.

9 CONCLUSÃO

Neste trabalho uma estratégia de controle distribuído foi aplicada em nível de simulação e implementação em bancada para controlar sistemas *Ball Balancer*. Os resultados de simulação mostraram a eficiência da estratégia de controle distribuído adotado, quando comparado com uma estratégia que não leva em consideração o índice de desempenho.

Foi feito um estudo para a seleção da planta que irá atuar em cada instante de amostragem, utilizando um índice de desempenho, baseado na norma dos sinais de controle dos sistemas controlado. A estratégia de decisão para a escolha da planta que receberá o sinal de controle pareceu adequada e de fácil processamento, demonstrando que é factível ser implementada em um sistema micro controlado. A validação de todas essas estratégias foi realizada através de implementação em bancada com 2 sistemas *Ball Balancer*.

Para implentação com dois *Ball Balancer,* foi considerado primeiramente o controle distribuído sem considerar a estratégia de controle proposta, com objetivo de analisar os dois sistemas sendo controlado com um Arduino Mega comunicando com o supervisório Elipse E3 ® através do protocolo Modbus TCP. Os resultados implementados com controlador PID nos dois *Ball Balancer* sem considerar a estratégia de controle proposta, apresentaram boas respostas para as resferências central, senoidal e quadrada. O algoritmo desta aplicação de controle PID de duas plantas *Ball Balancer* comunicando com o supervisório, está descrito em Anexo D.

Na implementação com duas plantas *Ball Balancer*, levando-se em conta o índice de desempenho proposto, foi testado a estratégia de decisão para escolha da planta que receberá o sinal de controle em cada período de amostragem, verificou-se que para o controle de posição da esfera com a referência no centro da tela, essa estratégia apresentou bom resultado para dois sistemas *Ball Balancer*. O algoritmo desta estratégia de controle distribuído está descrito em Anexo E. Porém é importante ressaltar que não foi possível controlar as plantas, através da análise de índice de desempenho, quando a referência muda dinamicamente através do sinal degrau (referência quadrada) e através do sinal senoidal (referência circular). Foi observado que a estratégia funciona de maneira adequada, porém, o *hardware* utilizado não tem capacidade de processamento e conversores AD suficientes para conseguir controlar esses sistemas, que possuem uma dinâmica muito rápida. Se o sistema possuísse uma dinâmica mais lenta, como controle de temperatura ou controle de nível, a estratégia funcionaria. Outra solução a este problema é utilizar microcontroladores com maior capacidade de processamento e com

mais conversores AD, tendo em vista que o Arduino Mega possui apenas um conversor AD que foi compartilhado para a realização das leituras dos quatro canais da tela *touch.*

Com o desenvolvimento deste trabalho foram disponibilizadas duas plantas *Ball Balancer* e, também, perspectivas de trabalhos no tema controle cooperativo e controle distribuído.

Como perspectivas para trabalhos futuros, pretende-se investigar a representação matemática dessa estratégia, demonstrando os limitantes da proposta de seleção da planta e, também, pretende-se utilizar de maneira complementar o chaveamento de controladores, que é um tema de pesquisa bastante estudado pelos pesquisadores do Laboratório de Pesquisa em Controle do DEE-FEIS (Departamento de Engenharia Elétrica - Faculdade de Engenharia de Ilha Solteira). Também, já foram adquiridos novos *hardwares* (Módulo ESP32), com capacidades melhoradas, para desenvolvimento de novos testes com a estratégia proposta.

REFERÊNCIAS

AGUILAR, R. N.; MEIJER, G. C. M. Fast interface electronics for a resistive touch-screen. **Proceedings of IEEE Sensors**, Piscataway, v. 1, n. 2, p. 1360–1363, 2002.

AKHMAD, S. S.; TOLA, M.; DJAMALUDDIN, R. Application of servo-motor control system at smart sprinkler. *In:* ELECTRICAL POWER, ELECTRONICS, COMMUNICATIONS, CONTROLS AND INFORMATICS SEMINAR, 2018, Batu. **Proceedings of the** [...]. Batu: IEEE, 2018. p. 159–163, 2018.

ARDUINO IDE. Disponível em: https://store.arduino.cc/usa/mega-2560-r3. Acesso em: 12 out. 2020.

BAI, Y.; CHEN, C. Using serial resistors to reduce the power consumption of resistive touch panels. *In*: IEEE INTERNATIONAL SYMPOSIUM ON CONSUMER ELECTRONICS, 2007, Irving. **Proceedings of the** [...]. Irving: IEEE, 2007. p. 1–6.

BEGAM, A. A. Touch panel based modern restaurant using robot. **IJASER**, Kangayam, v. 2, p. 571–575, 2017.

BOYER, S. **Supervisory control and data acquisition - SCADA**. 3. ed. [New Yorks]: The Instrumentation, Systems, and Automation Society, 2004.

BRAESCU, F. C. *et al*. Ball on plate balancing system for multi-discipline educational purposes. *In:* INTERNATIONAL CONFERENCE ON SYSTEM THEORY, CONTROL AND COMPUTING, ICSTCC, 16, 2012, **Proceedings o the** [...]. Sinai: IEEE, 2012. p. 1–5, 2012.

CALPE-MARAVILLA, J. *et al*. Dual touch and gesture recognition in 4-wire resistive touchscreens. **Proceedings of IEEE Sensors**, Piscataway, v. 2014, p. 787–790, 2014.

CASTRUCCI, P. L.; MORAES, C. **Engenharia de automação industrial**. 2. ed. Rio de Janeiro: LTC, 2008.

CHEN, Y. *et al*. Asynchronous observer-based H∞ control for switched stochastic systems with mixed delays under quantization and packet dropouts. **Nonlinear Analysis: Hybrid Systems**, London, v. 27, n. 2015, p. 225–238, 2018.

CHENG, L. *et al*. Sampled-data based average consensus of second-order integral multi-agent systems: switching topologies and communication noises. **Automatica**, Oxford, v. 49, n. 5, p. 1458–1464, 2013.

DORF, R. C.; BISHOP, R. H. **Sistemas de controle modernos**. 8. ed. Rio de Janeiro: LTC, 2001.

ELIPSE. O que são sistemas supervisórios. Disponível em: https://kb.elipse.com.br/o-que-sao-sistemas-supervisorios/. Acesso em: 30 out. 2020.

FADAEI, A.; SALAHSHOOR, K. Improving the control performance of networked control

systems using a new fuzzy PID. *In*: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, 2008, Cambridge. **Proceedin of the** [...]. Cambridge: IEEE, 2008. p. 2066–2071.

GUPTA, R.; CHOW, M. Y. Networked control system overview. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 57, n. 7, p. 2527–2535, 2010.

HALLIDAY, D.; RESNICK, R.; WALKER, J. **Fundamentos de física**: mecânica. 4. ed. Rio de Janeiro: LTC, 1996. V. 1.

HESPANHA, J. P.; NAGHSHTABRIZI, P.; XU, Y. A survey of recent results in networked control systems. **Proceedings of the IEEE**, Piscataway, v. 95, n. 1, p. 138–172, 2007.

KOPICHEV, M. M.; PUTOV, A. V.; PASHENKO, A. N. Ball on the plate balancing control system. **IOP Conference Series: Materials Science and Engineering**, Bristol, v. 638, n. 1, p. 0–6, 2019.

LI, H.; SHI, Y. Network-based predictive control for constrained nonlinear systems with two-channel packet dropouts. **IEEE Transactions on Industrial Electronics**, Piscataway, v. 61, n. 3, p. 1574–1582, 2014.

LI, Q. *et al*. Stochastic synchronization of semi-Markovian jump chaotic Lur'e systems with packet dropouts subject to multiple sampling periods. **Journal of the Franklin Institute**, Oxford, v. 356, n. 13, p. 6899–6925, 2019.

MAESTRELLI, R. **Análise de estabilidade e síntese de controladores para sistemas de controle via rede**. 2016. 171 f. Tese (Doutorado) - Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2016. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/173284/343971.pdf?sequence=1&isAllowed=y. Acesso em: 23 jun. 2021.

MIT. Mechanical Engineering. **Lab. 4**: motor control. Disponível em: https://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-electromechanical-robotic-systems-fall-2009/labs/MIT2_017JF09_lab4.pdf. Acesso em: 20 out. 2020.

NETTO, J. L. L. **Controle cooperativo H2 e H∞ via rede de comunicação**. 2018. [s.l.] Dissertação (Mestrado) - Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas, 2018. Disponível em: http://repositorio.unicamp.br/jspui/handle/REPOSIP/331713. Acesso em: 23 jun. 2021.

NISE, N. S. **Engenharia de sistemas de controle**. 6. ed. Rio de Janeiro: LTC, 2013.

OGATA, K. **Engenharia de controle moderno**. 3. ed. Rio de Janeiro: LTC, 1998.

PIN, G.; PARISINI, T. Networked predictive control of uncertain constrained nonlinear systems: recursive feasibility and input-to-state stability analysis. **IEEE Transactions on Automatic Control**, Piscataway, v. 56, n. 1, p. 72–87, 2011.

PLAYTIX. Disponível em: https://www.playtix.com.br/loja/p/kit-painel-touch-screen-resistivo-de-15-6-formato-widescreen-de-4-vias-usb.html. Acesso em: 05 out. 2020.

QU, H.; YANG, F.; HAN, Q. Distributed H ∞ -consensus filtering for a networked time-delay system with switching network topology and packet dropouts. *In:* AUSTRALIAN & NEW ZEALAND CONTROL CONFERENCE, ANZCC, 2018, Melbourne. **Proceedings of the** [...]. Melbourne: [s.n.], 2018. p. 334–339.

QUANSER. 2D Ball balancer 2d ball balancer control using QUARC instructor manual. 2008.

SAPUTRA, H. M. Servo motor controller device for stewart platform based on simple Pulse Generator. *In:* INTERNATIONAL CONFERENCE ON RADAR, ANTENNA, MICROWAVE, ELECTRONICS, AND TELECOMMUNICATIONS (ICRAMET), 2019, Tangerang. **Proceedings of the** [...]. Tangerang: IEEE, 2019. p. 104–108.

SAWICZ. Disponível em: https://www.princeton.edu/~mae412/-TEXT/NTRAK-2002/292-302.pdf. Acesso em: 22 out. 2020.

SERVODATABASE. Disponível em: https://servodatabase.com/servo-/hextronik-/hxt12k. Acesso em: 20 out. 2020.

SOUSA, T. T.; GEROMEL, J. C.; DEAECTO, G. S. Switching control resource allocation in Networked Control Systems. *In*: PROCEEDINGS OF THE IEEE CONFERENCE ON DECISION AND CONTROL, 54, 2015, Osaka. **Proceedings of the** [...]. Osaka: IEEE, 2015. p. 6862–6867.

SOUZA, M. **Controle de sistemas dinâmicos através de redes de comunicação**. 2012. Dissertação (Mestrado) - Escola de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas, 2012.

SUMEGA, M. *et al*. Experimental study of ball on plate platform. INTERNATIONAL CONFERENCE ELEKTRO, 12, 2018**,** Mikulov. **Proceedings of the** [...]. Mikulov: IEEE, 2018. p. 1–5.

TIMPEA, S. A.; COSMA, C.; SOSDEAN, D. Touch screen tester device end-effector. **Materiale Plastice**, Bucharest, v. 56, p. 445–448, 2019.

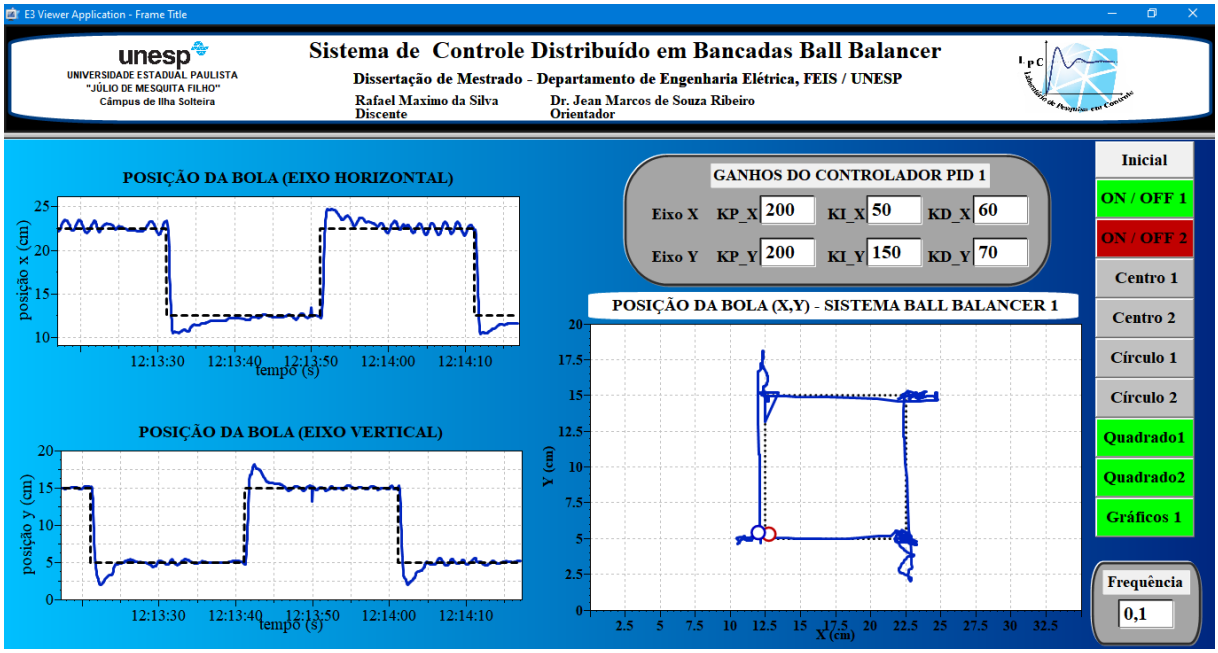
WANG, F.; LIU, D. **Networked control systems**: theory and applications. London: Springer-Verlag, 2008.

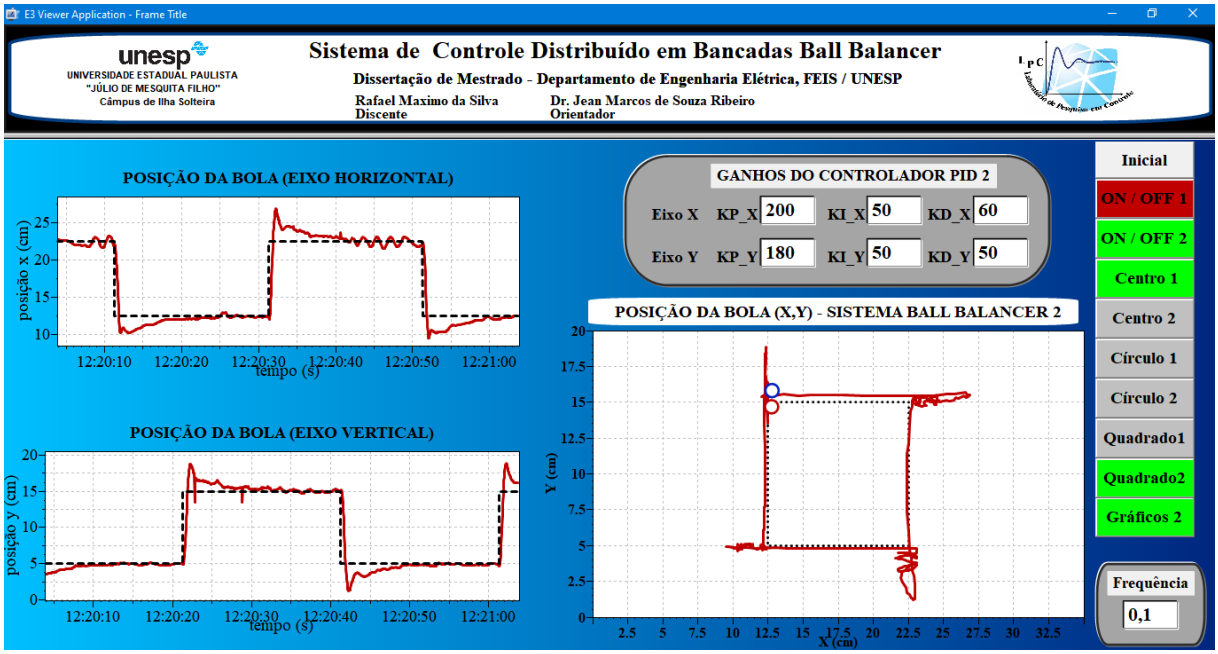
ZHANG, L.; GAO, H.; KAYNAK, O. Network-induced constraints in networked control systems: a survey. **IEEE Transactions on Industrial Informatics**, Piscataway, v. 9, n. 1, p. 403–416, 2013.

ZHANG, X. *et al*. Networked control systems : a survey of trends and techniques. **IEEE/CAA Journal of Automatica Sinica**, Piscataway, v. 7, n. 1, p. 1–17, 2019.

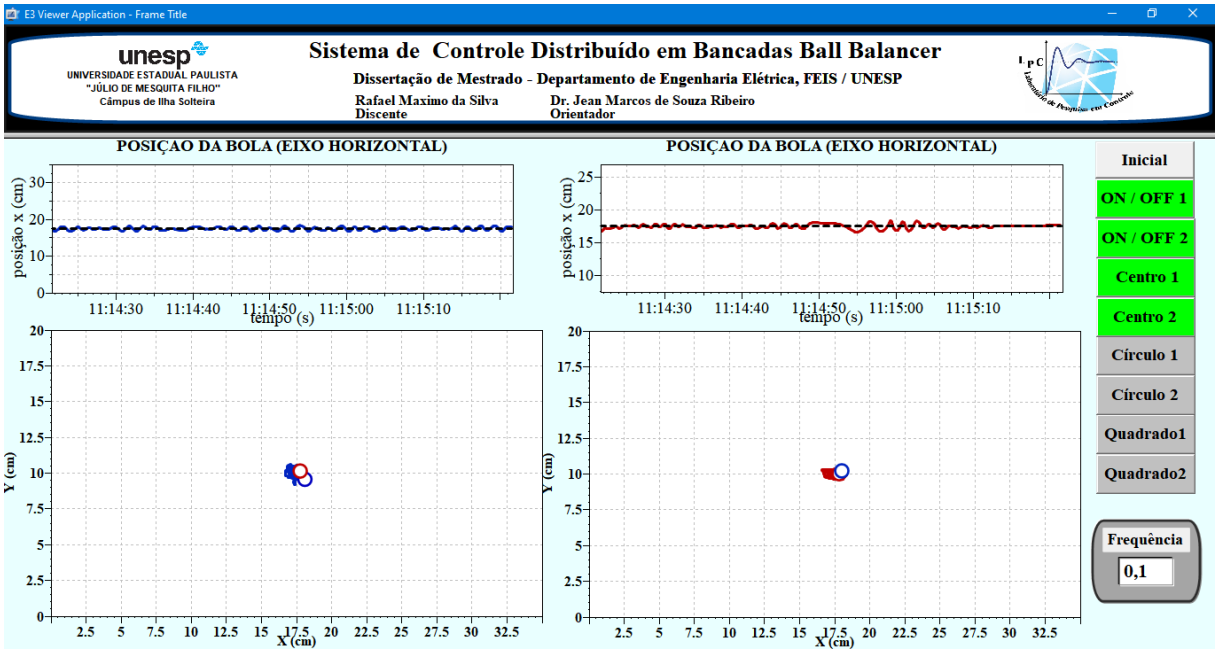
ZIA, A. Polar and polygon path traversal of a ball and plate system. *In:* INTERNATIONAL CONFERENCE ON ELECTRICAL AND CONTROL ENGINEERING, ICECE, 2011,Yichang. **Proceedings of the** [...]. Yichang: IEEE, 2011. p. 4005–4009.

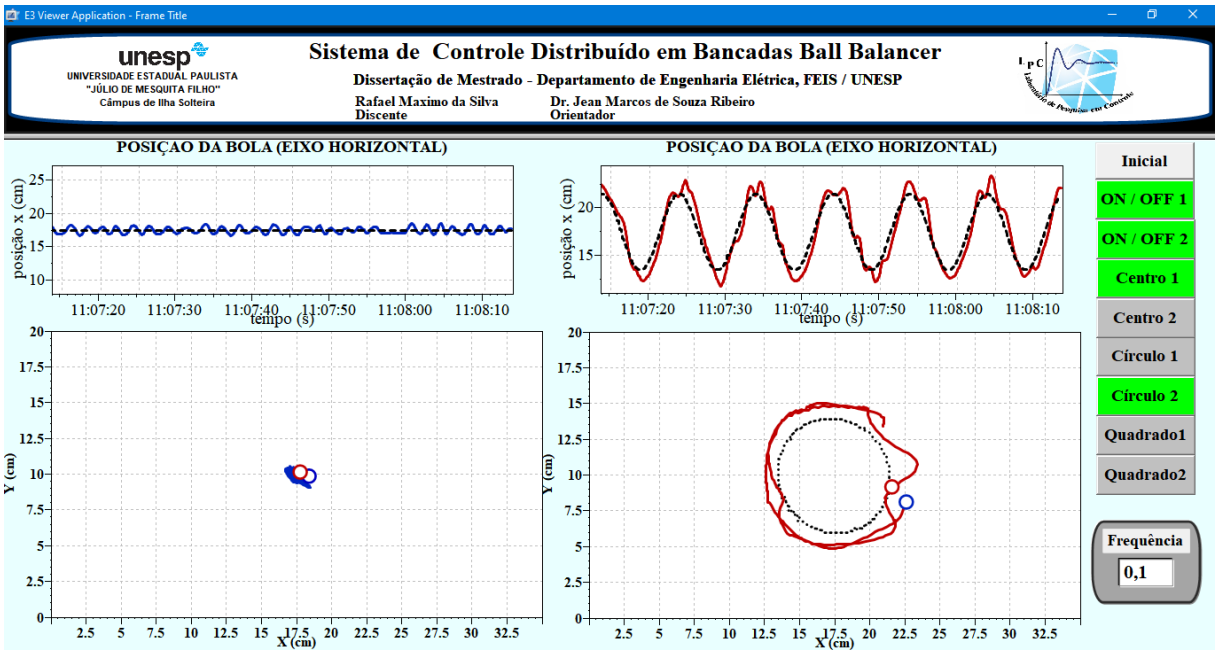
ANEXO A – *BALL BALANCER* 1 E 2

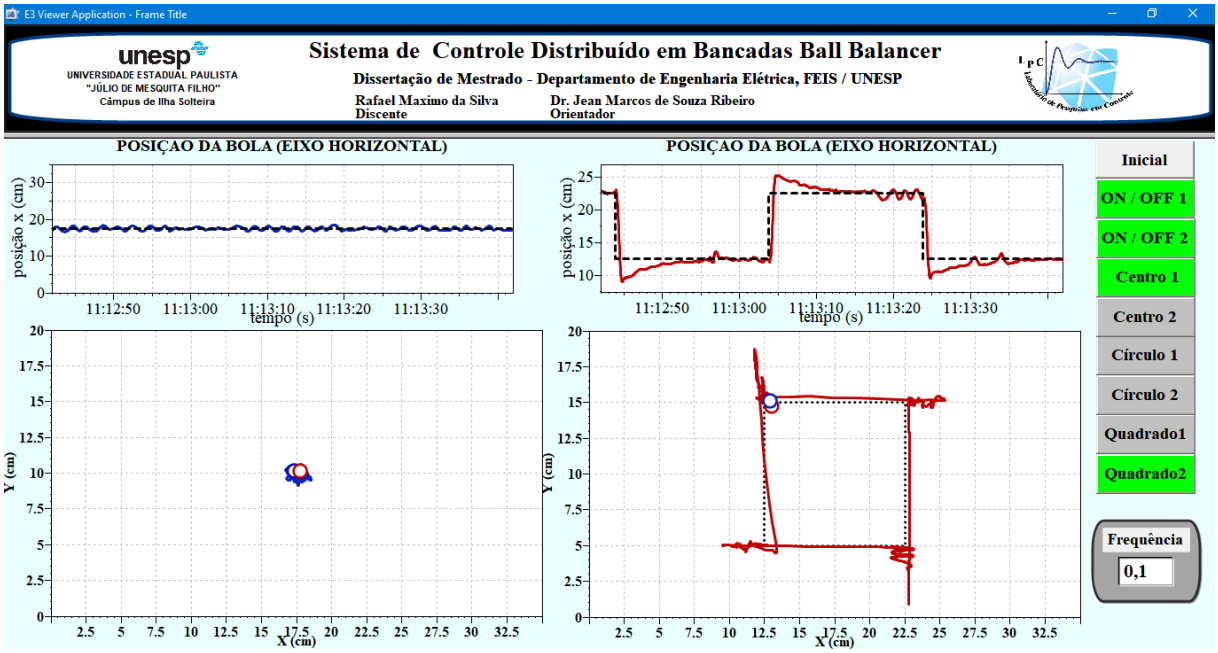


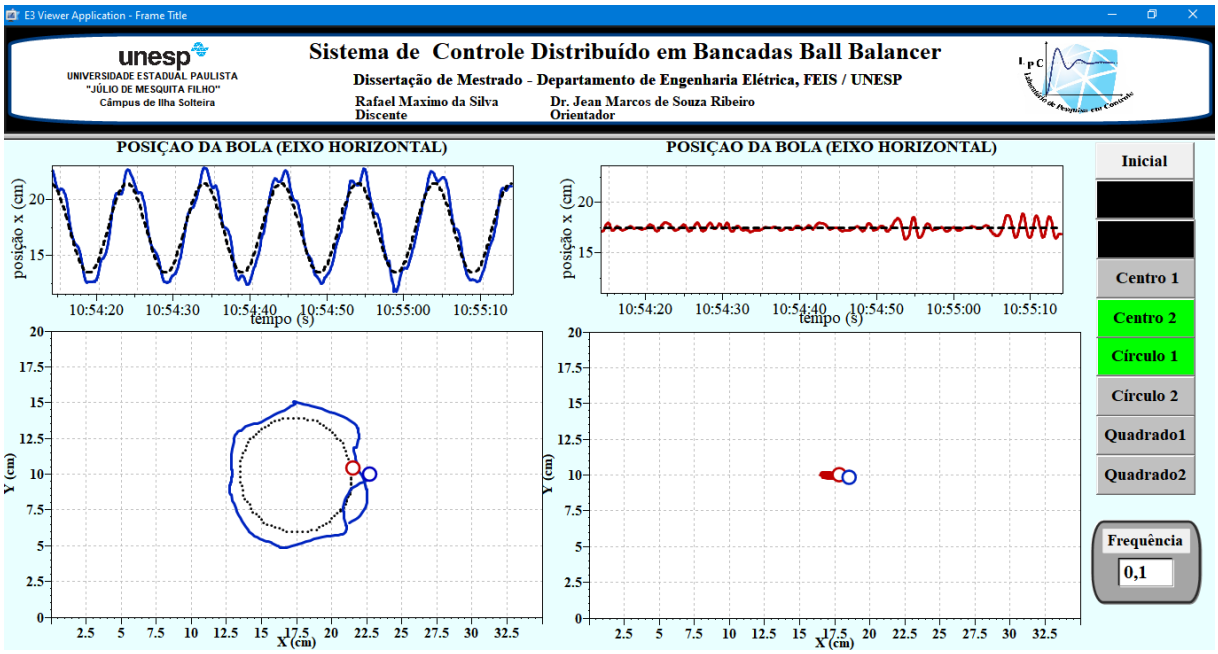


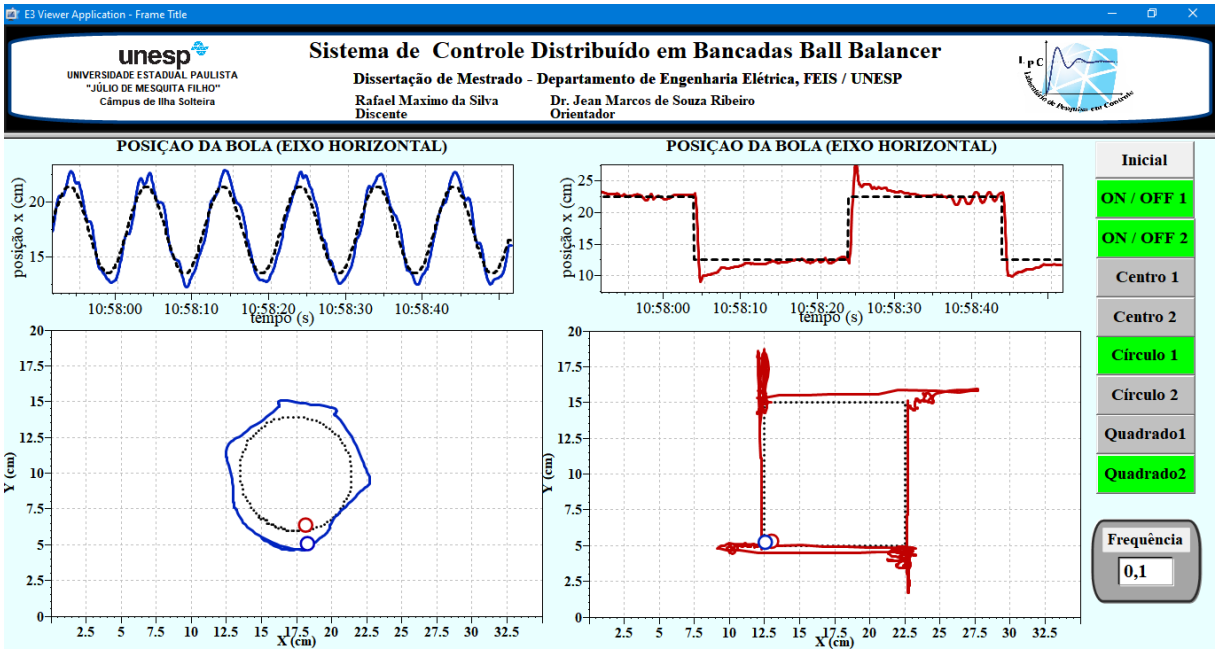
ANEXO B – IMPLEMENTAÇÃO COM DOIS *BALL BALANCER*

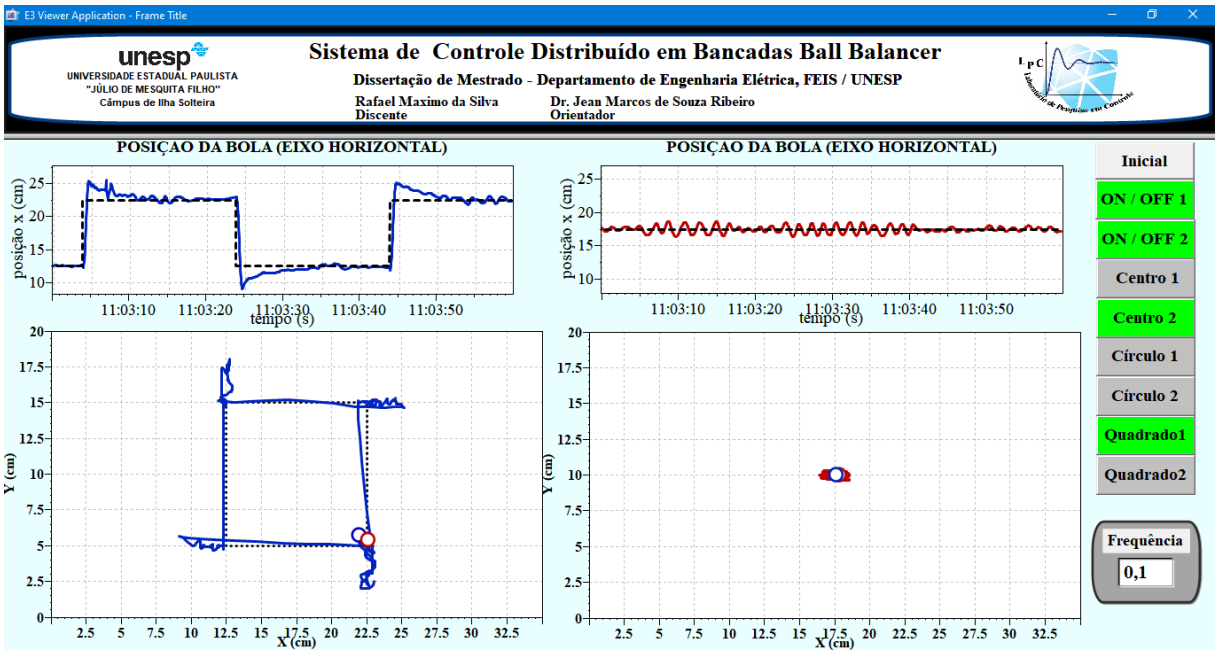


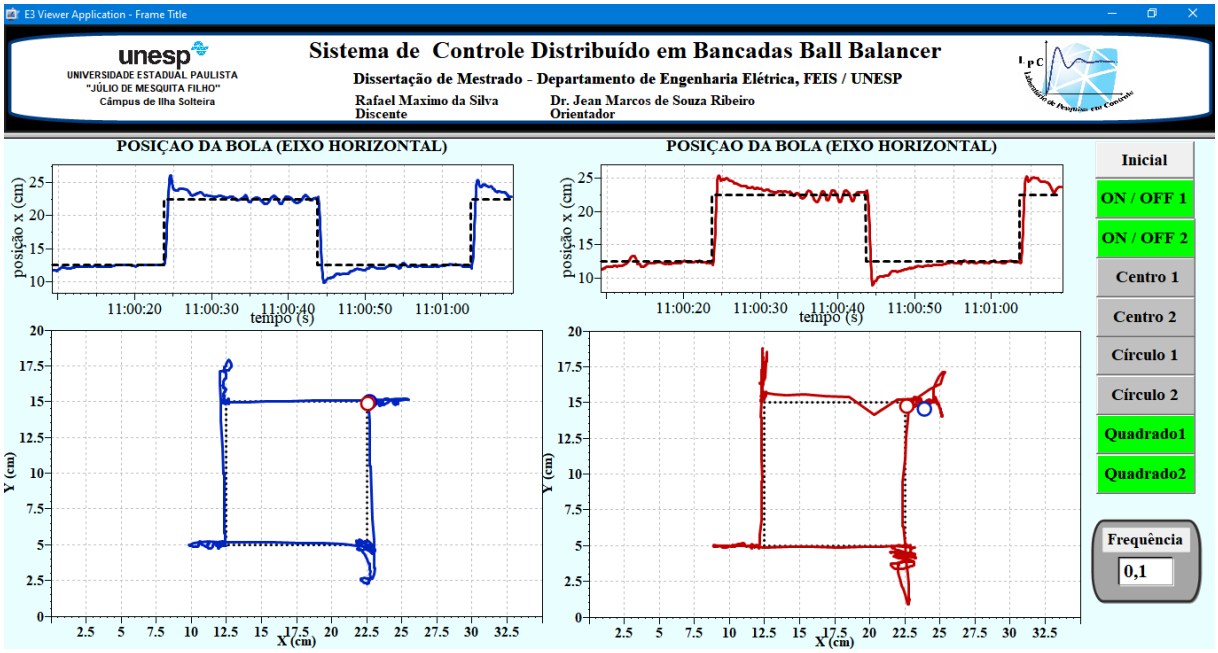


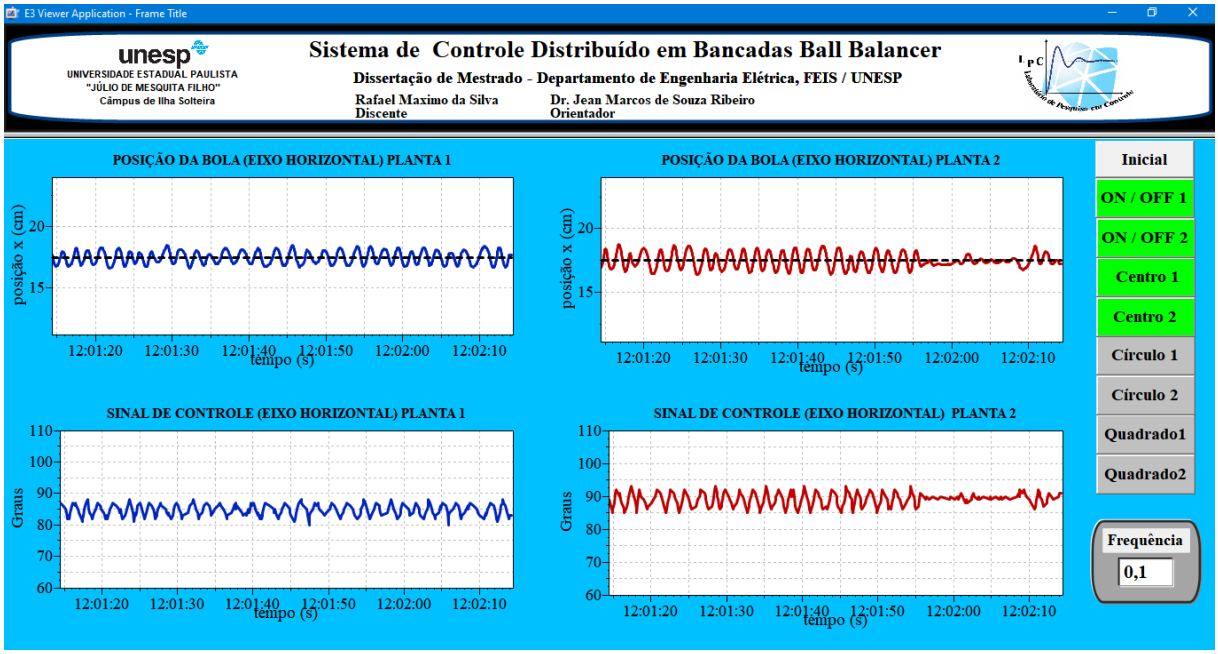

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
11:12:50
11:13:00
11:13:10
11:13:20
11:13:30
Y (cm)
X (cm)
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

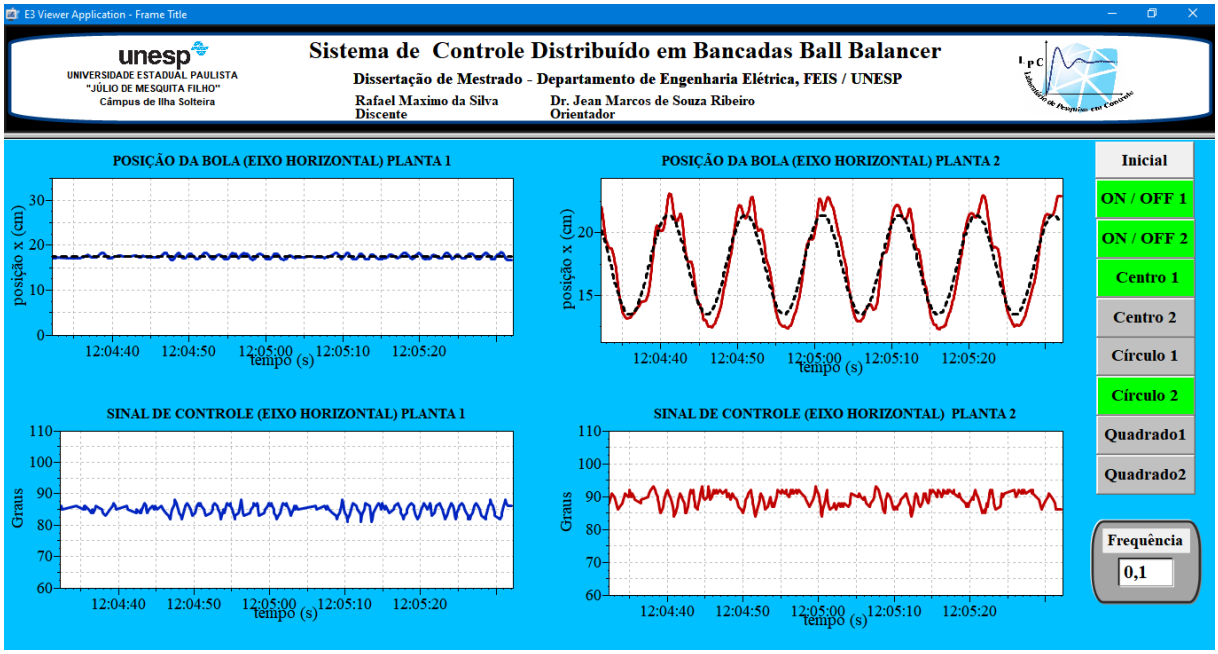

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
10:54:20
10:54:30
10:54:40
10:54:50
10:55:00
10:55:10
Y (cm)
X (cm)
Inicial
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

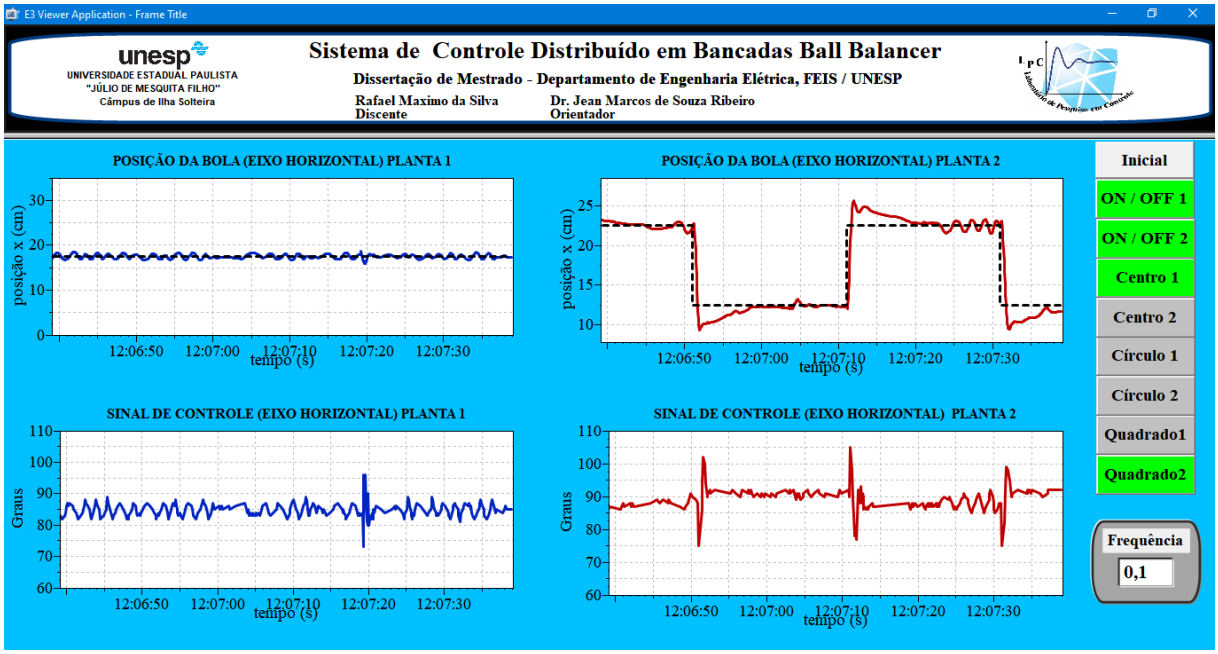

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
Y (cm)
X (cm)
Y (cm)
X (cm)
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

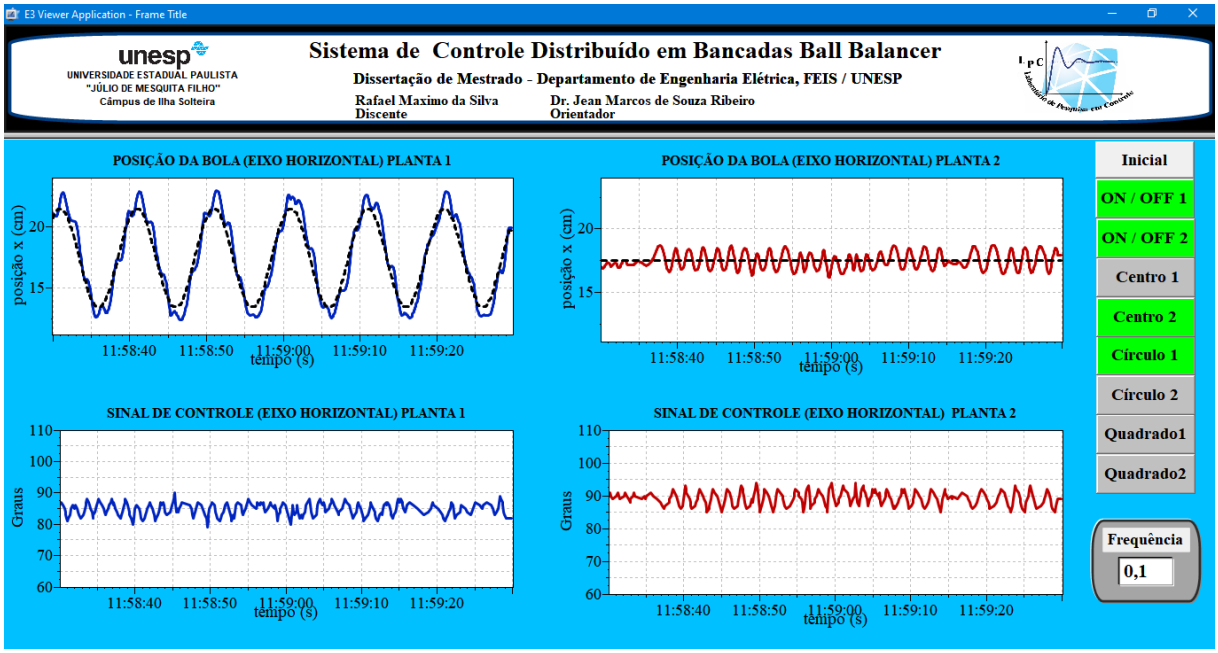

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
Y (cm)
X (cm)
Y (cm)
X (cm)
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

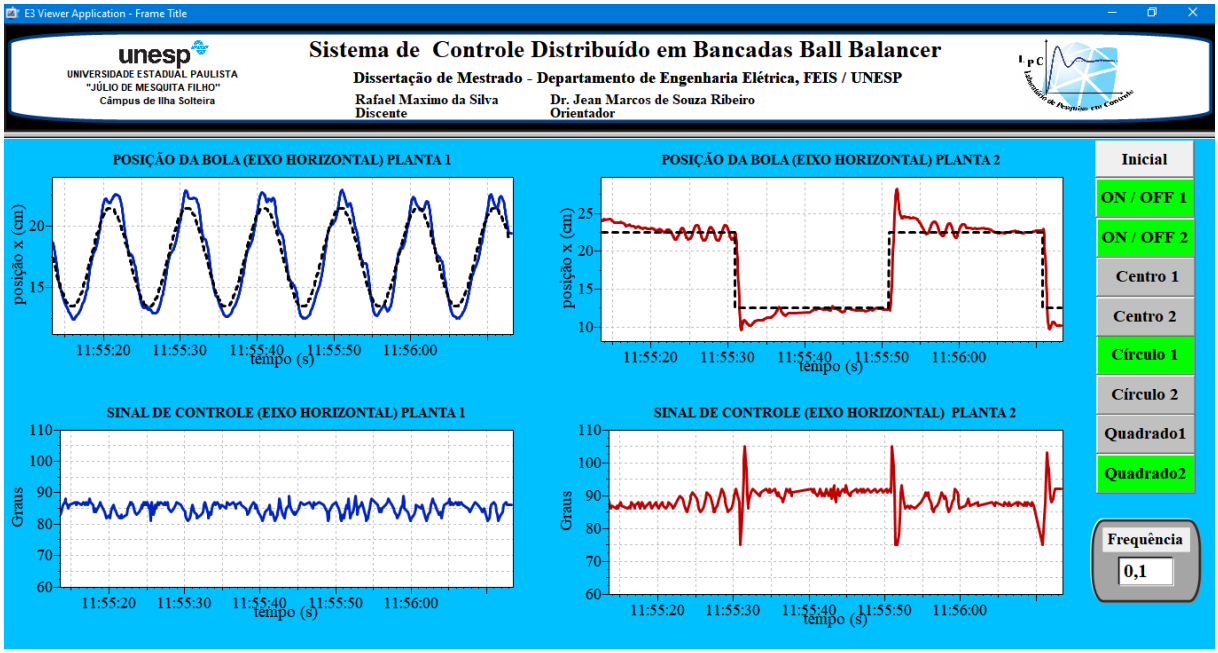

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
POSIÇAO DA BOLA (EIXO HORIZONTAL)
posição x (cm)
tempo (s)
Y (cm)
X (cm)
Y (cm)
X (cm)
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

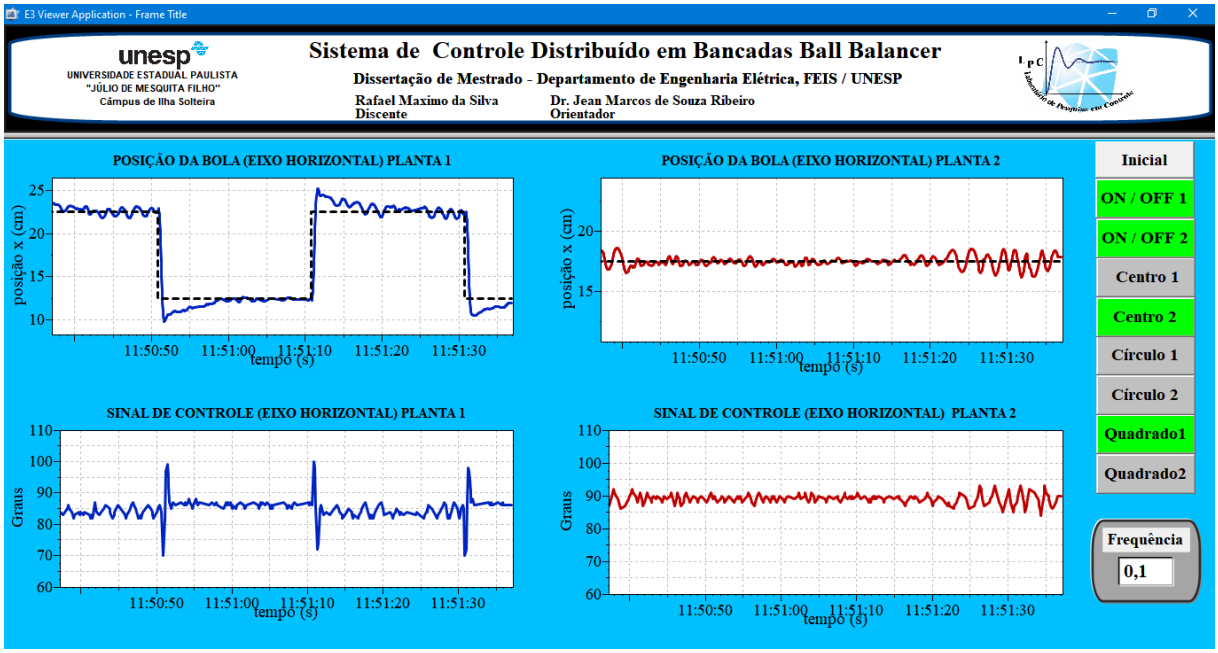

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 1
posição x (cm)
tempo (s)
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 2
posição x (cm)
tempo (s)
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 1
Graus
tempo (s)
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 2
Graus
tempo (s)
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

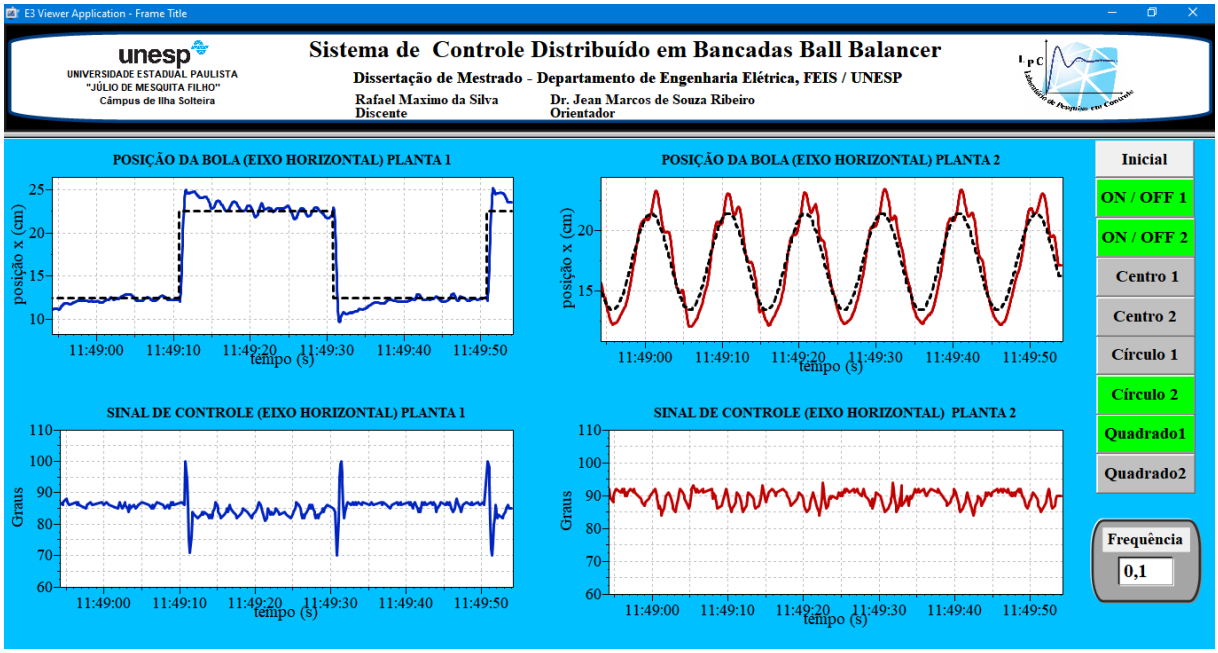

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 1
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 2
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 1
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 2
posição x (cm)
Graus
tempo (s)
12:04:40
12:04:50
12:05:00
12:05:10
12:05:20
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1


E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 1
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 2
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 1
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 2
posição x (cm)
Graus
tempo (s)
12:06:50
12:07:00
12:07:10
12:07:20
12:07:30
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

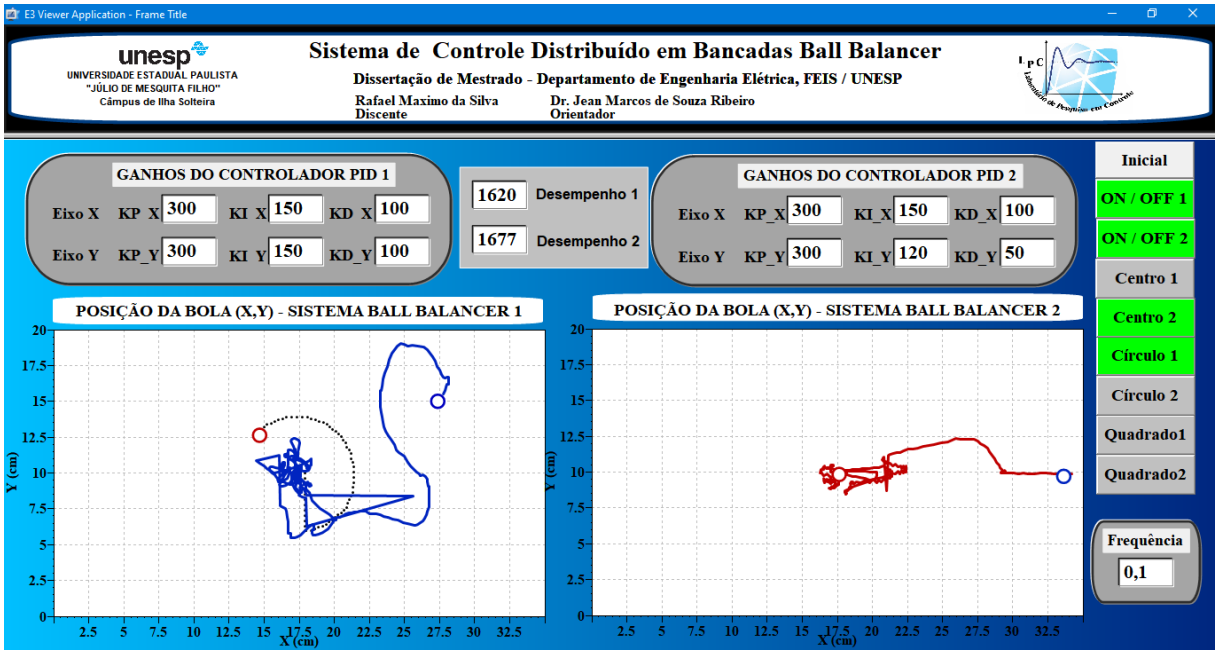

E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 1
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 2
posição x (cm)
tempo (s)
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 1
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 2
Graus
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1


E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 1
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 2
posição x (cm)
tempo (s)
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 1
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 2
Graus
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1


E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 1
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 2
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 1
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 2
posição x (cm)
Graus
tempo (s)
11:50:50
11:51:00
11:51:10
11:51:20
11:51:30
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1


E3 Viewer Application - Frame Title
unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira
Sistema de Controle Distribuído em Bancadas Ball Balancer
Dissertação de Mestrado - Departamento de Engenharia Elétrica, FEIS / UNESP
Rafael Maximo da Silva
Discente
Dr. Jean Marcos de Souza Ribeiro
Orientador
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 1
POSIÇÃO DA BOLA (EIXO HORIZONTAL) PLANTA 2
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 1
SINAL DE CONTROLE (EIXO HORIZONTAL) PLANTA 2
posição x (cm)
Graus
tempo (s)
11:49:00
11:49:10
11:49:20
11:49:30
11:49:40
11:49:50
Inicial
ON / OFF 1
ON / OFF 2
Centro 1
Centro 2
Círculo 1
Círculo 2
Quadrado1
Quadrado2
Frequência
0,1

ANEXO C – IMPLEMENTAÇÃO DA ESTRATÉGIA COM INSTABILIDADE



ANEXO D – PROGRAMA ARDUINO SEM ÍNDICE DE DESEMPENHO

```
/*#########################################################################
 *  Dissertação de Mestrado - Unesp Ilha Solteira
 *  Ball Balancer 2 DOF
 *  Discente Rafael M.Silva
 *  Orientador Dr. Jean M. S. Ribeiro
 *
 #########################################################################
*/

//---------------------- Bibliotecas ----------------------------------
#include <Servo.h>
#include <PID_v1.h>

#include <Mudbus.h>
#include <SPI.h>
#include <Ethernet.h>
//------------------ Final Bibliotecas --------------------------------

Mudbus Mb;

//---------------- Variáveis Supervisório -----------------------------

int referencia_1 = 0, referencia_2 = 3;

// --- Instanciando as variáveis P1 ---
int on_off_1=0,centre_1, circle_track_1, square_track_1;
int xs_1=0, ys_1=0;

int ang_Xs_1, ang_Ys_1;
int setpoint_xs_1, setpoint_ys_1;
float freq_s_1;

// --- P2---
int on_off_2=0,centre_2, circle_track_2, square_track_2;
int xs_2=0, ys_2=0;

int ang_xs_2, ang_ys_2;
int setpoint_xs_2, setpoint_ys_2;
float freq_s_2;
//---------------- Final Supervisório ---------------------------------

//--------------- Definição de variáveis ------------------------------
//--- P1 ---
// --- Pinos Touch Screen ---
#define x_gnd_1 A0
#define y_gnd_1 A1
#define x_5v_1  A2
```

```
#define y_5v_1  A3

// --- Pinos Servomotor da mesa 1 ---
#define servoXpin_1 8
#define servoYpin_1 9

// --- P2 ---
// --- Pinos Touch Screen ---
#define x_gnd_2 A8
#define y_gnd_2 A9
#define x_5v_2  A10
#define y_5v_2  A11

// --- Pinos Servomotor da mesa 2 ---
#define servoXpin_2 14
#define servoYpin_2 15

//------------ Final Definição de variáveis ----------------------------


//----------------- Tela Touch Screen ---------------------------------
unsigned long  t=0; //tempo para rastreamento
double pattern_counter_1; //contador padrão rastreamento

unsigned int noTouchCount_1 = 0; //variável para noTouch(sem toque)
int Z1_1;
boolean compute_1;

double pos_x_ball_1;
double pos_y_ball_1;

double pos_x_measured_1;
double pos_y_measured_1;

double pos_x_corrected_1;
double pos_y_corrected_1;


// --- Variáveis p/ filtrar spikes ---
double pos_x_last_1, max_x_variation_1 ;
double pos_y_last_1, max_y_variation_1 ;

int cont_1 = 0;

//--- P2 ---
double pattern_counter_2; //contador padrão rastreamento

unsigned int noTouchCount_2 = 0; //variável p/ noTouch(sem toque)
int Z1_2;
boolean compute_2;
```

```
double pos_x_ball_2;
double pos_y_ball_2;

double pos_x_measured_2;
double pos_y_measured_2;

double pos_x_corrected_2;
double pos_y_corrected_2;

// --- Variáveis p/ filtrar spikes ---
double pos_x_last_2, max_x_variation_2 ;
double pos_y_last_2, max_y_variation_2 ;

int cont_2 = 0;
//------------- Final Tela Touch  Screen ------------------------------


//----------------- controlador PID -----------------------------------
int Ts = 50;  //com que frequência, em milissegundos, o PID será avaliado.
            // padrão 100ms
//--- P1 ---
// --- Variáveis do controlador ---
double InputX_1 = 0, InputY_1 = 0;
double setpoint_x_1=175, setpoint_y_1=100;
double Output_SMX_1, Output_SMY_1; //ângulo motor

float KP_x_1 = 0.20, KP_y_1 = 0.20;
float KI_x_1 = 0.05, KI_y_1 = 0.15;
float KD_x_1 = 0.06, KD_y_1 = 0.07;

// variáveis usadas para transformar os ganhos em números inteiros
double KP_x_11 ;
double KP_y_12 ;
double KI_x_13 ;
double KI_y_14 ;
double KD_x_15 ;
double KD_y_16 ;

double KP_x_12;

//--- Objeto p/ controle PID do eixo X e Y ---
//--- PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)---
PID pid_x_1(&InputX_1, &Output_SMX_1, &setpoint_x_1, KP_x_1, KI_x_1,
KD_x_1,P_ON_E, DIRECT); //servo 1 / mesa 1
PID pid_y_1(&InputY_1, &Output_SMY_1, &setpoint_y_1, KP_y_1, KI_y_1,
KD_y_1,P_ON_E, DIRECT); //servo 2 / mesa 1

//--- P2 ---
// --- Variáveis do controlador ---
```

```
double InputX_2 = 0, InputY_2 = 0;
double setpoint_x_2=175, setpoint_y_2=100;
double Output_SMX_2, Output_SMY_2; //ângulo motor

float KP_x_2 = 0.20, KP_y_2 = 0.18;
float KI_x_2 = 0.05, KI_y_2 = 0.05;
float KD_x_2 = 0.06, KD_y_2 = 0.05;

// variáveis usadas para transformar os ganhos em números inteiros
double KP_x_21 ;
double KP_y_22 ;
double KI_x_23 ;
double KI_y_24 ;
double KD_x_25 ;
double KD_y_26 ;

//--- Objeto p/ controle PID do eixo X e Y ---
//--- PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)---
PID pid_x_2(&InputX_2, &Output_SMX_2, &setpoint_x_2, KP_x_2, KI_x_2,
KD_x_2,P_ON_E, DIRECT);
PID pid_y_2(&InputY_2, &Output_SMY_2, &setpoint_y_2, KP_y_2, KI_y_2,
KD_y_2,P_ON_E, DIRECT);
//----------------- Final controlador PID ------------------------------


// -------------------- Servomotor -----------------------------------
//--- Objeto p/ controlar o servomotor eixo X e Y ---
//--- P1 ---
Servo servoX_1;
Servo servoY_1;

//--- Ângulo inicial da plataforma (plana) ---
double thetaX_1=85, thetaY_1=58;

//--- Deslocamento angular máximo do servo X e Y ---
double displaceX_1 = 15; //Considera o limite mecânico
double displaceY_1 = 15;

//--- P2 ---
Servo servoX_2;
Servo servoY_2;

//--- Ângulo inicial da plataforma (plana) ---
double thetaX_2=90, thetaY_2=95;

//--- Deslocamento angular máximo do servo X e Y ---
double displaceX_2 = 15; //Considera o limite mecânico
double displaceY_2 = 15;
// ---------------- Final Servomotor ---------------------------------
```

```
//#########################################################################
//#########################################################################
void setup() {

 // Definição do endereço IP do microcontrolador
 byte mac[] = {
 0x90, 0xA2, 0xDA, 0x00, 0x51, 0x06 };
 IPAddress ip(192,168,0,201);
 Ethernet.begin(mac, ip);

 // --- Mapa Modbus ---
 // definição das condições iniciais dos endereços Modbus
 Mb.R[0] = on_off_1;
 Mb.R[1] = xs_1;
 Mb.R[2] = ys_1;
 Mb.R[3] = circle_track_1;
 Mb.R[4] = square_track_1;
 Mb.R[5] = ang_Xs_1;
 Mb.R[6] = ang_Ys_1;
 Mb.R[7] = setpoint_xs_1;
 Mb.R[8] = setpoint_ys_1;

 Mb.R[10] = on_off_2;
 Mb.R[11] = xs_2;
 Mb.R[12] = ys_2;
 Mb.R[13] = circle_track_2;
 Mb.R[14] = square_track_2;
 Mb.R[15] = ang_xs_2;
 Mb.R[16] = ang_ys_2;
 Mb.R[17] = setpoint_xs_2;
 Mb.R[18] = setpoint_ys_2;

 Mb.R[31] = KP_x_1*1000;
 Mb.R[32] = KP_y_1*1000 ;
 Mb.R[33] = KI_x_1*1000;
 Mb.R[34] = KI_y_1*1000;
 Mb.R[35] = KD_x_1*1000;
 Mb.R[36] = KD_y_1*1000;

 Mb.R[41] = KP_x_2*1000;
 Mb.R[42] = KP_y_2*1000;
 Mb.R[43] = KI_x_2*1000;
 Mb.R[44] = KI_y_2*1000;
 Mb.R[45] = KD_x_2*1000;
 Mb.R[46] = KD_y_2*1000;

 Mb.R[51] = referencia_1;
 Mb.R[52] = referencia_2;

 Mb.R[81] = Output_SMX_1;//Sinal de controle do eixo x mesa 1
```

```
  Mb.R[82] = Output_SMX_2;//Sinal de controle do eixo x mesa 2

  // --- Servo Motor / mesa 1 ---
  servoX_1.attach(servoXpin_1);  //ServoX ao pino digital 8
  Output_SMX_1 = thetaX_1;       //Posicionando o servoX(eixo Horizontal)
  servoX_1.write(Output_SMX_1);  //no ângulo inicial(plano)

  servoY_1.attach(servoYpin_1);  //ServoY ao pino digital 9
  Output_SMY_1 = thetaY_1;
  servoY_1.write(Output_SMY_1);

  // --- Configuração do PID ---
  pid_x_1.SetOutputLimits(thetaX_1-displaceX_1,thetaX_1+displaceX_1); // angulo x
//entre 70 e 100 graus
  pid_x_1.SetMode(AUTOMATIC); //Configuração para PID ligado (automático)
  pid_x_1.SetSampleTime(Ts);  //Determina com que frequência o algoritmo PID é
calculado (10ms)

  pid_y_1.SetOutputLimits(thetaY_1-displaceY_1,thetaY_1+displaceY_1); // angulo y
//entre 43 e 73 graus
  pid_y_1.SetMode(AUTOMATIC);
  pid_y_1.SetSampleTime(Ts);

// --- Servo Motor / mesa 2 ---
  servoX_2.attach(servoXpin_2);  //ServoX ao pino digital 14
  Output_SMX_2 = thetaX_2;       //Posicionando o servoX(eixo Horizontal)
  servoX_2.write(Output_SMX_2);  //no ângulo inicial(plano)

  servoY_2.attach(servoYpin_2);  //ServoY ao pino digital 15
  Output_SMY_2 = thetaY_2;
  servoY_2.write(Output_SMY_2);

  // --- Configuração do PID ---
  pid_x_2.SetOutputLimits(thetaX_2-displaceX_2,thetaX_2+displaceX_2);// angulo x
//entre 75 e 105 graus
  pid_x_2.SetMode(AUTOMATIC); //Configuração para PID ligado (automático)
  pid_x_2.SetSampleTime(Ts);  //Determina com que frequência o algoritmo PID é
//calculado

  pid_y_2.SetOutputLimits(thetaY_2-displaceY_2,thetaY_2+displaceY_2); // angulo x
//entre 80 e 110 graus
  pid_y_2.SetMode(AUTOMATIC);
  pid_y_2.SetSampleTime(Ts);

  Mb.Run();

}  ////////////// Final do Setup  ***************************************


//#########################################################################
```

```
//#########################################################################
void loop() {

// ---- P1 ---
      //----------Registradores----------
      //recebe dados do supervisório
      on_off_1 = Mb.R[0];
      circle_track_1 = Mb.R[3];
      square_track_1 = Mb.R[4];

      //----------Liga/Desliga via Tela E3-----------
   if (on_off_1==1)
   {
       detect_Touch_1(); // Verifica se a bola está sobre a TouchScreen para computar o
PID
       if (compute_1 == 1) // tem bola sobre a tela
       {
        servoX_1.attach(servoXpin_1);
        servoY_1.attach(servoYpin_1);
        noTouchCount_1 = 0;

        desired_position_1(); //Lê a posição desejada

        spikes_Filter_1();

        InputX_1 = pos_x_corrected_1; //leitura da posição vinda do void
spikes_Filter_1()
        InputY_1 = pos_y_corrected_1;

        pid_x_1.Compute(); //na frequência especificada no SetSampleTime
        pid_y_1.Compute();

        servoX_1.write(Output_SMX_1);
        servoY_1.write(Output_SMY_1);

        compute_1 = 0; // usado no void detect_Touch()

     }

     else
     {  //Caso não houver contato na tela
        noTouchCount_1++;//incrementa o contador (sem toque)

        if (noTouchCount_1 > 20) //Posiciona a plataforma na posição inicial(plana)
        {
         Output_SMX_1 = thetaX_1;
         servoX_1.write(Output_SMX_1);

         Output_SMY_1 = thetaY_1;
         servoY_1.write(Output_SMY_1);
```

```
          }

          //Caso não existir contato por muito tempo

          if(noTouchCount_1 == 200)
          {
            servoX_1.detach(); //desliga o servo
            servoY_1.detach();
          }
        }
      }

      xs_1 = InputX_1*10;  //valor de X para o regs do supervisório
      ys_1 = InputY_1*10;  //valor de Y para o regs do supervisório
      ang_Xs_1 = Output_SMX_1;
      ang_Ys_1 = Output_SMY_1;

      // Atualiza as variáveis modbus
      Mb.R[1] = xs_1;
      Mb.R[2] = ys_1;
      Mb.R[5] = ((ang_Xs_1-70)*(30-0))/(100-70)*100;
      Mb.R[6] = ((ang_Ys_1-43)*(30-0))/(73-43) *100;

      setpoint_xs_1 = setpoint_x_1;
      setpoint_ys_1 = setpoint_y_1;
      Mb.R[7] = setpoint_xs_1;
      Mb.R[8] = setpoint_ys_1;


// ---- P2 ---
       //----------Registradores----------
       on_off_2 = Mb.R[10]; //recebe dados do supervisório
       circle_track_2 = Mb.R[13];
       square_track_2 = Mb.R[14];

       //----------Liga/Desliga via Tela E3-----------
      if (on_off_2==1)
      {
        detect_Touch_2();
        if (compute_2 == 1) //Verifica se a bola está sobre a TouchScreen para computar o
PID
        {
          servoX_2.attach(servoXpin_2);
          servoY_2.attach(servoYpin_2);
          noTouchCount_2 = 0;

          desired_position_2();  //Lê a posição desejada

          spikes_Filter_2();
```

```
        InputX_2 = pos_x_corrected_2; //leitura da posição vinda do void
spikes_Filter_2()
        InputY_2 = pos_y_corrected_2;

        pid_x_2.Compute();
        pid_y_2.Compute();

        servoX_2.write(Output_SMX_2);
        servoY_2.write(Output_SMY_2);

        compute_2 = 0;

       }

       //Caso não houver contato na tela
       else
       {
        noTouchCount_2++;//incrementa o contador (sem toque)

        //Posiciona a plataforma na posição inicial(plana)
        if (noTouchCount_2 > 50)
        {
         Output_SMX_2 = thetaX_2;
         servoX_2.write(Output_SMX_2);

         Output_SMY_2 = thetaY_2;
         servoY_2.write(Output_SMY_2);
        }

        //Caso não existir contato por muito tempo
        if(noTouchCount_2 == 200)
        {
         servoX_2.detach(); //desliga o servo
         servoY_2.detach();
        }
       }
      }


    xs_2 = InputX_2*10;  //valor de X para o regs do supervisório
    ys_2 = InputY_2*10;  //valor de Y para o regs do supervisório
    ang_xs_2 = Output_SMX_2;
    ang_ys_2 = Output_SMY_2;

    // Atualiza as variáveis modbus
    Mb.R[11] = xs_2;
    Mb.R[12] = ys_2;
    Mb.R[15] = ((ang_xs_2-75)*(30-0))/(105-75)*100;
    Mb.R[16] = ((ang_ys_2-80)*(30-0))/(110-80) *100;
```

```
setpoint_xs_2 = setpoint_x_2;
setpoint_ys_2 = setpoint_y_2;
Mb.R[17] = setpoint_xs_2;
Mb.R[18] = setpoint_ys_2;

// Atualiza as variáveis do Arduino com valores 100 vezes maior
KP_x_11 = Mb.R[31];
KP_y_12 = Mb.R[32];
KI_x_13 = Mb.R[33];
KI_y_14 = Mb.R[34];
KD_x_15 = Mb.R[35];
KD_y_16 = Mb.R[36];

KP_x_21 = Mb.R[41];
KP_y_22 = Mb.R[42];
KI_x_23 = Mb.R[43];
KI_y_24 = Mb.R[44];
KD_x_25 = Mb.R[45];
KD_y_26 = Mb.R[46];

referencia_1 = Mb.R[51];
referencia_2 = Mb.R[52];

Mb.R[81] = Output_SMX_1;//Sinal de controle do eixo x mesa 1
Mb.R[82] = Output_SMX_2;//Sinal de controle do eixo x mesa 2


// Adequa as variáveis do Arduino com a divisão por 100
KP_x_1 = (KP_x_11/1000);
KP_y_1 = (KP_y_12/1000);
KI_x_1 = (KI_x_13/1000);
KI_y_1 = (KI_y_14/1000);
KD_x_1 = (KD_x_15/1000);
KD_y_1 = (KD_y_16/1000);

KP_x_2 = (KP_x_21/1000);
KP_y_2 = (KP_y_22/1000);
KI_x_2 = (KI_x_23/1000);
KI_y_2 = (KI_y_24/1000);
KD_x_2 = (KD_x_25/1000);
KD_y_2 = (KD_y_26/1000);


 Mb.Run();

 // Atualiza os ganhos do PID
 pid_x_1.SetTunings(KP_x_1, KI_x_1, KD_x_1);//atualiza os valores em tempo de
execução
 pid_y_1.SetTunings(KP_y_1, KI_y_1, KD_y_1);
```

```
      pid_x_2.SetTunings(KP_x_2, KI_x_2, KD_x_2);//atualiza os valores em tempo de
execução
      pid_y_2.SetTunings(KP_y_2, KI_y_2, KD_y_2);

}  // Fim do Void Loop( ) ************************************


//#########################################################################
//#########################################################################

void desired_position_1()
{
 t=millis();
 pattern_counter_1 = (t+40000)%40000; //função resto da divisão
                              //(zera a cada 40000)

 //controle de posição e de rastreamento da trajetória
 //---  Referência no Centro ---
 //if(centre_1==1)
 if(referencia_1==0)
 {
  setpoint_x_1 = 175;
  setpoint_y_1 = 100;
 }

 //--- Referência circular ---
 if(referencia_1==1)
 {
  setpoint_x_1 = 175+ 40*cos(2*3.1415*0.1*t/1000);
  setpoint_y_1 = 100+ 40*sin(2*3.1415*0.1*t/1000);
 }

 //--- Referência Quadrada ---
 if(referencia_1==2)
 {
   if(pattern_counter_1 <= 10000)
   {
    setpoint_x_1 = 225;
    setpoint_y_1 = 50;
   }
   if(pattern_counter_1 > 10000 && pattern_counter_1 <= 20000)
   {
    setpoint_x_1 = 125;
    setpoint_y_1 = 50;
   }
   if(pattern_counter_1 > 20000 && pattern_counter_1 <= 30000)
   {
    setpoint_x_1 = 125;
    setpoint_y_1 = 150;
   }
```

```
    if(pattern_counter_1 > 30000 && pattern_counter_1 <= 40000)
    {
     setpoint_x_1 = 225;
     setpoint_y_1 = 150;
    }
   }
}


//#########################################################################
//#########################################################################
void read_Touch_1() //
{
    //Leitura posição-x Planta 1
    pinMode(x_gnd_1,OUTPUT);     //Xb - gnd (prepara sinais da tela p/
    pinMode(x_5v_1,OUTPUT);      //Xa - 5V   leitura da coordenada X)
    pinMode(y_gnd_1,INPUT);      //Yb - gnd
    pinMode(y_5v_1,INPUT);       //Ya - 5V
    digitalWrite(x_gnd_1,LOW);   //xb
    digitalWrite(x_5v_1,HIGH);   //xa   //aplicando 5volts no eixo X
    delay(6);
    pos_x_ball_1=analogRead(y_5v_1); //lê coordenada X

    delay(3);
    //Leitura posição-y Planta 1
    pinMode(x_gnd_1,INPUT);    //Xb - gnd
    pinMode(x_5v_1 ,INPUT);    //Xa - 5V
    pinMode(y_gnd_1,OUTPUT);   //Yb - gnd
    pinMode(y_5v_1 ,OUTPUT);   //Ya - 5V
    digitalWrite(y_gnd_1,LOW); //yb
    digitalWrite(y_5v_1,HIGH); //aplicando 5volts no eixo Y
    delay(6);
    pos_y_ball_1=analogRead(x_5v_1);  //lê coordenada Y
}


//#########################################################################
//#########################################################################
void correct_Position_1()  //media_touch
{
 detect_Touch_1(); //a bola esta sobre a tela?
 if(Z1_1>0)
 {
  read_Touch_1();

  pos_x_measured_1 = pos_x_ball_1;
  pos_y_measured_1 = pos_y_ball_1;

  //Leitura em  cm
  pos_x_corrected_1 = ((pos_x_measured_1-51)*(350-0))/(973-51);
```

```
    pos_y_corrected_1 = ((pos_y_measured_1-107)*(200-0))/(880-107);

  }
}

//#########################################################################
//#########################################################################
void detect_Touch_1()
{
     //--------- Lê Z1 -----------
     //  Lê e verifica se há contato
     pinMode(x_gnd_1,OUTPUT);    // x_gnd_1
     pinMode(x_5v_1,INPUT);      // x_5v_1
     pinMode(y_gnd_1,INPUT);     // y_gnd_1
     pinMode(y_5v_1,OUTPUT);     // y_5v_1
     digitalWrite(x_gnd_1,LOW);  // x_gnd_1
     digitalWrite(y_5v_1,HIGH);  // y_5v_1
     delay(6);//10
     Z1_1 = analogRead(x_5v_1);  // x_5v_1

     if(Z1_1>0)
     {
       compute_1 = 1;
     }

    //--------- Lê Z1 ----------------------
}


//#########################################################################
//#########################################################################
void spikes_Filter_1() //Filtra spikes com variação acima de ___
{
   correct_Position_1();

   if(cont_1==0)
   {
     pos_x_last_1 = pos_x_corrected_1; //armazena o X anterior
     pos_y_last_1 = pos_y_corrected_1; //armazena o Y anterior
     cont_1++;
   }

   max_x_variation_1 = 20; //cálculo da variação máxima permitida
   max_y_variation_1 = 20; //cálculo da variação máxima permitida 2cm

   //Condição para filtrar spikes do eixo X
   if(abs(pos_x_corrected_1 - pos_x_last_1) > max_x_variation_1)
   {
      correct_Position_1(); //Chama a função p/ nova leitura de X
```

```
      if(abs(pos_x_corrected_1-pos_x_last_1) > max_x_variation_1)
      {
       pos_x_corrected_1 = pos_x_corrected_1; //Caso X novo for maior que variação
                                  //considera que a bola está em movimento
      }
      else
      {
       pos_x_corrected_1 = pos_x_last_1;  //Senão, desconsidera a leitura espúria
      }

   }

   //Condição para filtrar spikes do eixo Y
   if(abs(pos_y_corrected_1 - pos_y_last_1) > max_y_variation_1)
   {
      correct_Position_1(); //Chama a função p/ leitura de Y

      if(abs(pos_y_corrected_1-pos_y_last_1) > max_y_variation_1)
      {
       pos_y_corrected_1 = pos_y_corrected_1;
      }

      else
      {
       pos_y_corrected_1 = pos_y_last_1;
      }
   }

   pos_x_corrected_1 = pos_x_corrected_1;
   pos_y_corrected_1 = pos_y_corrected_1;

   pos_x_last_1 = pos_x_corrected_1; //armazena o X anterior
   pos_y_last_1 = pos_y_corrected_1; //armazena o Y anterior

}


//#########################################################################
//#########################################################################

//#######################  Planta 2  ###################################

void desired_position_2()
{
  t=millis();
  pattern_counter_2 = (t+40000)%40000; //função resto da divisão
                            //(zera a cada 40000)

 //Controle de posição e de rastreamento da trajetória
```

```
  //--- Referência no Centro ---
  if(referencia_2==3)
  {
   setpoint_x_2 = 175;
   setpoint_y_2 = 100;
  }

  //--- Referência Circular ---
  if(referencia_2==4)
  {
   setpoint_x_2 = 175+ 40*cos(2*3.1415*0.1*t/1000);
   setpoint_y_2 = 100+ 40*sin(2*3.1415*0.1*t/1000);
  }
  //--- Referência Quadrada ---
  if(referencia_2==5)
  {
   if(pattern_counter_2 <= 10000)
   {
    setpoint_x_2 = 225;
    setpoint_y_2 = 50;
   }
   else if(pattern_counter_2 > 10000 && pattern_counter_2 <= 20000)
   {
    setpoint_x_2 = 125;
    setpoint_y_2 = 50;
   }

   else if(pattern_counter_2 > 20000 && pattern_counter_2 <= 30000)
   {
    setpoint_x_2 = 125;
    setpoint_y_2 = 150;
   }
   else if(pattern_counter_2 > 30000 && pattern_counter_2 <= 40000)
   {
    setpoint_x_2 = 225;
    setpoint_y_2 = 150;
   }
  }
}


//#########################################################################
//#########################################################################
void read_Touch_2() //
{
    //Leitura posição-x Planta 2
    pinMode(x_gnd_2,OUTPUT);    //Xb - gnd (prepara sinais da tela p/
    pinMode(x_5v_2,OUTPUT);     //Xa - 5V   leitura da coordenada X)
    pinMode(y_gnd_2,INPUT);     //Yb - gnd
    pinMode(y_5v_2,INPUT);      //Ya - 5V
```

```
    digitalWrite(x_gnd_2,LOW);   //xb
    digitalWrite(x_5v_2,HIGH);   //xa   //aplicando 5volts no eixo X
    delay(6);
    pos_x_ball_2=analogRead(y_5v_2); //lê coordenada X

    delay(3);
    //Leitura posição-y
    pinMode(x_gnd_2,INPUT);    //Xb - gnd
    pinMode(x_5v_2 ,INPUT);    //Xa - 5V
    pinMode(y_gnd_2,OUTPUT);   //Yb - gnd
    pinMode(y_5v_2 ,OUTPUT);   //Ya - 5V
    digitalWrite(y_gnd_2,LOW); //yb
    digitalWrite(y_5v_2,HIGH); //aplicando 5volts no eixo Y
    delay(6);
    pos_y_ball_2=analogRead(x_5v_2);  //lê coordenada Y
}


//#########################################################################
//#########################################################################
void correct_Position_2()  //media_touch
{
 detect_Touch_2(); //a bola esta sobre a tela?

 if(Z1_2>0)
 {
  read_Touch_2();

  pos_x_measured_2 = pos_x_ball_2;
  pos_y_measured_2 = pos_y_ball_2;

  //Leitura em cm
  pos_x_corrected_2=((pos_x_measured_2-54)*(350-0))/(964-54);

  pos_y_corrected_2 =((pos_y_measured_2-97)*(200-0))/(888-97);

 }
}


//#########################################################################
//#########################################################################
void detect_Touch_2()
{
    //--------- Lê Z1 -----------
    //  Lê e verifica se há contato
    pinMode(x_gnd_2,OUTPUT);    // x_gnd_2
    pinMode(x_5v_2,INPUT);      // x_5v_2
    pinMode(y_gnd_2,INPUT);     // y_gnd_2
    pinMode(y_5v_2,OUTPUT);     // y_5v_2
```

```
    digitalWrite(x_gnd_2,LOW);  // x_gnd_2
    digitalWrite(y_5v_2,HIGH);  // y_5v_2
    delay(6);//10
    Z1_2 = analogRead(x_5v_2);  // x_5v_2

    if(Z1_2>0)
    {
     compute_2 = 1;
    }

  //--------- Lê Z1 ----------------------
}


//#############################################################################
//#############################################################################
void spikes_Filter_2() //Filtra spikes com variação acima de ___
{
  correct_Position_2();

  if(cont_2==0)
  {
   pos_x_last_2 = pos_x_corrected_2; //armazena o X anterior
   pos_y_last_2 = pos_y_corrected_2; //armazena o Y anterior
   cont_2++;
  }

  max_x_variation_2 = 20; //cálculo da variação máxima permitida
  max_y_variation_2 = 20; //cálculo da variação máxima permitida 2cm


  //Condição para filtrar spikes do eixo X
  if(abs(pos_x_corrected_2 - pos_x_last_2) > max_x_variation_2)
  {

    correct_Position_2(); //Chama a função p/  nova leitura de X

    if(abs(pos_x_corrected_2-pos_x_last_2) > max_x_variation_2)
    {
     pos_x_corrected_2 = pos_x_corrected_2; //Caso X novo for maior que variação
                                  //considera que a bola está em movimento
    }
    else
    {
     pos_x_corrected_2 = pos_x_last_2;  //Senão, desconsidera a leitura espúria
    }
  }

  //Condição para filtrar spikes do eixo Y
  if(abs(pos_y_corrected_2 - pos_y_last_2) > max_y_variation_2)
```

```
    {
      correct_Position_2(); //Chama a função p/ leitura de Y

      if(abs(pos_y_corrected_2-pos_y_last_2) > max_y_variation_2)
      {
       pos_y_corrected_2 = pos_y_corrected_2;
      }

      else
      {
       pos_y_corrected_2 = pos_y_last_2;
      }
    }

    pos_x_corrected_2 = pos_x_corrected_2;
    pos_y_corrected_2 = pos_y_corrected_2;

    pos_x_last_2 = pos_x_corrected_2; //armazena o X anterior
    pos_y_last_2 = pos_y_corrected_2; //armazena o Y anterior
}

//#########################################################################
//#########################################################################
```

ANEXO E – PROGRAMA ARDUINO COM ÍNDICE DE DESEMPENHO

```
/*#########################################################################
 *  Dissertação de Mestrado - Unesp Ilha Solteira
 *  Ball Balancer 2 DOF
 *  Discente Rafael M.Silva
 *  Orientador Dr. Jean M. S. Ribeiro
 *
 #########################################################################
*/

//---------------------- Bibliotecas ----------------------------------
#include <Servo.h>
#include <PID_v1.h>

#include <Mudbus.h>
#include <SPI.h>
#include <Ethernet.h>
//------------------ Final Bibliotecas --------------------------------

Mudbus Mb;

//---------------- Variáveis Supervisório -----------------------------

int referencia_1 = 0, referencia_2 = 3;


// --- Instanciando as variáveis P1 ---
int on_off_1=0,centre_1, circle_track_1, square_track_1;
int xs_1=0, ys_1=0;

int ang_Xs_1, ang_Ys_1;
int setpoint_xs_1, setpoint_ys_1;
float freq_s_1;

// --- P2---
int on_off_2=0,centre_2, circle_track_2, square_track_2;
int xs_2=0, ys_2=0;

int ang_xs_2, ang_ys_2;
int setpoint_xs_2, setpoint_ys_2;
float freq_s_2;

//---------------- Final Supervisório ---------------------------------


//--------------- Definição de variáveis ------------------------------
```

```
//--- P1 ---
// --- Pinos Touch Screen ---
#define x_gnd_1 A0
#define y_gnd_1 A1
#define x_5v_1  A2
#define y_5v_1  A3

// --- Pinos Servomotor da mesa 1 ---
#define servoXpin_1 8
#define servoYpin_1 9

//--- P2 ---
// --- Pinos Touch Screen ---
#define x_gnd_2 A8
#define y_gnd_2 A9
#define x_5v_2  A10
#define y_5v_2  A11

// --- Pinos Servomotor da mesa 2 ---
#define servoXpin_2 14
#define servoYpin_2 15

//------------ Final Definição de variáveis ----------------------------


//----------------- Tela Touch Screen ---------------------------------
// Declaração da função RunningMedian (Objeto)
//--- P1 ---
unsigned long  t=0; //tempo para rastreamento
double pattern_counter_1; //contador padrão rastreamento

unsigned int noTouchCount_1 = 0; //variável para noTouch(sem toque)
int Z1_1;
boolean compute_1;

double pos_x_ball_1;
double pos_y_ball_1;

double pos_x_measured_1;
double pos_y_measured_1;

double pos_x_corrected_1;
double pos_y_corrected_1;


// --- Variáveis p/ filtrar spikes ---
double pos_x_last_1, max_x_variation_1 ;
double pos_y_last_1, max_y_variation_1 ;

int cont_1 = 0;
```

```
//--- P2 ---
//unsigned long  t=0; //tempo para rastreamento
double pattern_counter_2; //contador padrão rastreamento

unsigned int noTouchCount_2 = 0; //variável p/ noTouch(sem toque)
int Z1_2;
boolean compute_2;

double pos_x_ball_2;
double pos_y_ball_2;

double pos_x_measured_2;
double pos_y_measured_2;

double pos_x_corrected_2;
double pos_y_corrected_2;

// --- Variáveis p/ filtrar spikes ---
double pos_x_last_2, max_x_variation_2 ;
double pos_y_last_2, max_y_variation_2 ;

int cont_2 = 0;


//------------- Final Tela Touch  Screen ------------------------------


//----------------- controlador PID -----------------------------------
int Ts = 20;  //com que frequência, em milissegundos, o PID será avaliado.
          // padrão 100ms
//--- P1 ---
// --- Variáveis do controlador ---
double InputX_1 = 0, InputY_1 = 0;
double setpoint_x_1=175, setpoint_y_1=100;
double Output_SMX_1, Output_SMY_1; //ângulo motor

float KP_x_1 = 0.30, KP_y_1 = 0.30;
float KI_x_1 = 0.15, KI_y_1 = 0.15;
float KD_x_1 = 0.1,  KD_y_1 = 0.1;


// variáveis usadas para transformar os ganhos em números inteiros
double KP_x_11 ;
double KP_y_12 ;
double KI_x_13 ;
double KI_y_14 ;
double KD_x_15 ;
double KD_y_16 ;
```

```
int Desempenho_1 = 0, Desempenho_2 = 0;

//--- Objeto p/ controle PID do eixo X e Y ---
//--- PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)---
PID pid_x_1(&InputX_1, &Output_SMX_1, &setpoint_x_1, KP_x_1, KI_x_1,
KD_x_1,P_ON_E, DIRECT); //servo 1 / mesa 1
PID pid_y_1(&InputY_1, &Output_SMY_1, &setpoint_y_1, KP_y_1, KI_y_1,
KD_y_1,P_ON_E, DIRECT); //servo 2 / mesa 2

// --- P2 ---
// --- Variáveis do controlador ---
double InputX_2 = 0, InputY_2 = 0;
double setpoint_x_2=175, setpoint_y_2=100;
double Output_SMX_2, Output_SMY_2; //ângulo motor

float KP_x_2 = 0.30, KP_y_2 = 0.30;
float KI_x_2 = 0.15, KI_y_2 = 0.12;
float KD_x_2 = 0.10, KD_y_2 = 0.05;

// variáveis usadas para transformar os ganhos em números inteiros
double KP_x_21 ;
double KP_y_22 ;
double KI_x_23 ;
double KI_y_24 ;
double KD_x_25 ;
double KD_y_26 ;

//--- Objeto p/ controle PID do eixo X e Y ---
//--- PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)---
PID pid_x_2(&InputX_2, &Output_SMX_2, &setpoint_x_2, KP_x_2, KI_x_2,
KD_x_2,P_ON_E, DIRECT);
PID pid_y_2(&InputY_2, &Output_SMY_2, &setpoint_y_2, KP_y_2, KI_y_2,
KD_y_2,P_ON_E, DIRECT);
//----------------- Final controlador PID ------------------------------

// -------------------- Servomotor -------------------------------------
//--- Objeto p/ controlar o servomotor eixo X e Y ---
//--- P1 ---
Servo servoX_1;
Servo servoY_1;

//--- Ângulo inicial da plataforma (plana) ---
double thetaX_1=85, thetaY_1=58;

//--- Deslocamento angular máximo do servo X e Y ---
double displaceX_1 = 4; //Considera o limite mecânico
double displaceY_1 = 4;

//--- P2 ---
Servo servoX_2;
```

```
Servo servoY_2;

//--- Ângulo inicial da plataforma (plana) ---
double thetaX_2=90, thetaY_2=95;

//--- Deslocamento angular máximo do servo X e Y ---
double displaceX_2 = 4; //Considera o limite mecânico
double displaceY_2 = 4;

// ---------------- Final Servomotor ---------------------------------

//#########################################################################
//#########################################################################
void setup() {

  // Definição do endereço IP do microcontrolador
  byte mac[] = {
  0x90, 0xA2, 0xDA, 0x00, 0x51, 0x06 };
  IPAddress ip(192,168,0,201);
  Ethernet.begin(mac, ip);


  // --- Mapa Modbus ---
  // definição das condições iniciais dos endereços Modbus
  Mb.R[0] = on_off_1;
  Mb.R[1] = xs_1;
  Mb.R[2] = ys_1;
  Mb.R[3] = circle_track_1;
  Mb.R[4] = square_track_1;
  Mb.R[5] = ang_Xs_1;
  Mb.R[6] = ang_Ys_1;
  Mb.R[7] = setpoint_xs_1;
  Mb.R[8] = setpoint_ys_1;

  Mb.R[10] = on_off_2;
  Mb.R[11] = xs_2;
  Mb.R[12] = ys_2;
  Mb.R[13] = circle_track_2;
  Mb.R[14] = square_track_2;
  Mb.R[15] = ang_xs_2;
  Mb.R[16] = ang_ys_2;
  Mb.R[17] = setpoint_xs_2;
  Mb.R[18] = setpoint_ys_2;

  Mb.R[31] = KP_x_1*1000;
  Mb.R[32] = KP_y_1*1000 ;
  Mb.R[33] = KI_x_1*1000;
  Mb.R[34] = KI_y_1*1000;
  Mb.R[35] = KD_x_1*1000;
  Mb.R[36] = KD_y_1*1000;
```

```
  Mb.R[41] = KP_x_2*1000;
  Mb.R[42] = KP_y_2*1000;
  Mb.R[43] = KI_x_2*1000;
  Mb.R[44] = KI_y_2*1000;
  Mb.R[45] = KD_x_2*1000;
  Mb.R[46] = KD_y_2*1000;

  Mb.R[51] = referencia_1;
  Mb.R[52] = referencia_2;

  // --- Servo Motor / mesa 1 ---
  servoX_1.attach(servoXpin_1);  //ServoX ao pino digital 8
  Output_SMX_1 = thetaX_1;       //Posicionando o servoX(eixo Horizontal)
  servoX_1.write(Output_SMX_1);  //no ângulo inicial(plano)

  servoY_1.attach(servoYpin_1);  //ServoY ao pino digital 9
  Output_SMY_1 = thetaY_1;
  servoY_1.write(Output_SMY_1);

  // --- Configuração do PID ---
  pid_x_1.SetOutputLimits(thetaX_1-displaceX_1,thetaX_1+displaceX_1); // angulo x
//entre 70 e 100 graus
  pid_x_1.SetMode(AUTOMATIC); //Configuração para PID ligado (automático)
  pid_x_1.SetSampleTime(Ts);  //Determina com que frequência o algoritmo PID é
calculado (10ms)

  pid_y_1.SetOutputLimits(thetaY_1-displaceY_1,thetaY_1+displaceY_1); // angulo y
//entre 43 e 73 graus
  pid_y_1.SetMode(AUTOMATIC);
  pid_y_1.SetSampleTime(Ts);


// --- Servo Motor / mesa 2 ---
  servoX_2.attach(servoXpin_2);  //ServoX ao pino digital 14
  Output_SMX_2 = 90;//thetaX_2;       //Posicionando o servoX(eixo Horizontal)
  servoX_2.write(Output_SMX_2);  //no ângulo inicial(plano)

  servoY_2.attach(servoYpin_2);  //ServoY ao pino digital 15
  Output_SMY_2 = 95; //thetaY_2;
  servoY_2.write(Output_SMY_2);

  // --- Configuração do PID ---
  pid_x_2.SetOutputLimits(thetaX_2-displaceX_2,thetaX_2+displaceX_2);// angulo x
entre 75 e 105 graus
  pid_x_2.SetMode(AUTOMATIC); //Configuração para PID ligado (automático)
  pid_x_2.SetSampleTime(Ts);  //Determina com que frequência o algoritmo PID é
calculado
```

```
  pid_y_2.SetOutputLimits(thetaY_2-displaceY_2,thetaY_2+displaceY_2); // angulo x
entre 80 e 110 graus
  pid_y_2.SetMode(AUTOMATIC);
  pid_y_2.SetSampleTime(Ts);

Mb.Run();

}  ////////////// Final do Setup  ****************************************


//#########################################################################
//#########################################################################
void loop() {

// ---- P1 ---
      //----------Registradores----------
      //recebe dados do supervisório
      on_off_1 = Mb.R[0];
      circle_track_1 = Mb.R[3];
      square_track_1 = Mb.R[4];

      //----------Liga/Desliga via Tela E3-----------
   if (on_off_1==1)
   {
       compute_1 = 0;
       detect_Touch_1(); // Verifica se a bola está sobre a TouchScreen para computar o
PID
       if (compute_1 == 1) // tem bola sobre a tela
       {
         servoX_1.attach(servoXpin_1);
         servoY_1.attach(servoYpin_1);
         noTouchCount_1 = 0;

         desired_position_1(); //Lê a posição desejada

         //Posição sem a média
         //correct_Position_1();
         spikes_Filter_1();

         //running_Media_1();
         InputX_1 = pos_x_corrected_1; //leitura da posição vinda do void
spikes_Filter_1()
         InputY_1 = pos_y_corrected_1;

         //Desempenho_1 = sqrt((Output_SMX_1 * Output_SMX_1)/(85 * 85) +
(Output_SMY_1 * Output_SMY_1) / (58 * 58 ));
         Desempenho_1 = abs(10*setpoint_xs_1-xs_1)+ abs(10*setpoint_ys_1-ys_1);

      }
```

```
 else
 {  //Caso não houver contato na tela
     Desempenho_1 = 0;
     noTouchCount_1++;//incrementa o contador (sem toque)

     if (noTouchCount_1 > 20) //Posiciona a plataforma na posição inicial(plana)
     {
      Output_SMX_1 = thetaX_1;
      servoX_1.write(Output_SMX_1);

      Output_SMY_1 = thetaY_1;
      servoY_1.write(Output_SMY_1);
     }

     //Caso não existir contato por muito tempo

     if(noTouchCount_1 == 200)
     {
      servoX_1.detach(); //desliga o servo
      servoY_1.detach();
     }
   }
}

 xs_1 = InputX_1*10;  //valor de X para o regs do supervisório
 ys_1 = InputY_1*10;  //valor de Y para o regs do supervisório
 ang_Xs_1 = Output_SMX_1;
 ang_Ys_1 = Output_SMY_1;

 // Atualiza as variáveis modbus
 Mb.R[1] = xs_1;
 Mb.R[2] = ys_1;
 Mb.R[5] = ((ang_Xs_1-70)*(30-0))/(100-70)*100;
 Mb.R[6] = ((ang_Ys_1-43)*(30-0))/(73-43) *100;

 setpoint_xs_1 = setpoint_x_1;
 setpoint_ys_1 = setpoint_y_1;
 Mb.R[7] = setpoint_xs_1;
 Mb.R[8] = setpoint_ys_1;


 // ---- P2 ---
  //----------Registradores----------
  on_off_2 = Mb.R[10]; //recebe dados do supervisório
  circle_track_2 = Mb.R[13];
  square_track_2 = Mb.R[14];

  //----------Liga/Desliga via Tela E3-----------
 if (on_off_2==1)
 {
```

```
          compute_2 = 0;
          detect_Touch_2();
          if (compute_2 == 1) //Verifica se a bola está sobre a TouchScreen p/ computar ctrl
          {
           servoX_2.attach(servoXpin_2);
           servoY_2.attach(servoYpin_2);
           noTouchCount_2 = 0;

           desired_position_2();  //Lê a posição desejada

           //Posição sem a média
           //correct_Position_2();
           spikes_Filter_2();

           //running_Media_2();

           InputX_2 = pos_x_corrected_2; //leitura da posição vinda do spikes_Filter_2()
           InputY_2 = pos_y_corrected_2;

          //Desempenho_2 = sqrt((Output_SMX_2 * Output_SMX_2) / (90 * 90)+
(Output_SMY_2 * Output_SMY_2) / (95 * 95));
          Desempenho_2 = abs(10*setpoint_xs_2-xs_2)+ abs(10*setpoint_ys_2-ys_2);

          }

          //Caso não houver contato na tela
          else
          {
           Desempenho_2 = 0;
           noTouchCount_2++;//incrementa o contador (sem toque)

           //Posiciona a plataforma na posição inicial(plana)
           if (noTouchCount_2 > 50)
           {
            Output_SMX_2 = thetaX_2;
            servoX_2.write(Output_SMX_2);

            Output_SMY_2 = thetaY_2;
            servoY_2.write(Output_SMY_2);
           }

           //Caso não existir contato por muito tempo
           if(noTouchCount_2 == 200)
           {
            servoX_2.detach(); //desliga o servo
            servoY_2.detach();
           }
          }
         }
```

```
 if (on_off_1==0)
 {
  Desempenho_1 = 0;
 }
if (on_off_2==0)
 {
  Desempenho_2 = 0;
 }

xs_2 = InputX_2*10;  //valor de X para o regs do supervisório
ys_2 = InputY_2*10;  //valor de Y para o regs do supervisório
ang_xs_2 = Output_SMX_2;
ang_ys_2 = Output_SMY_2;

// Atualiza as variáveis modbus
Mb.R[11] = xs_2;
Mb.R[12] = ys_2;
Mb.R[15] = ((ang_xs_2-75)*(30-0))/(105-75)*100;
Mb.R[16] = ((ang_ys_2-80)*(30-0))/(110-80) *100;

setpoint_xs_2 = setpoint_x_2;
setpoint_ys_2 = setpoint_y_2;
Mb.R[17] = setpoint_xs_2;
Mb.R[18] = setpoint_ys_2;

// Atualiza as variáveis do Arduino com valores 100 vezes maior
KP_x_11 = Mb.R[31];
KP_y_12 = Mb.R[32];
KI_x_13 = Mb.R[33];
KI_y_14 = Mb.R[34];
KD_x_15 = Mb.R[35];
KD_y_16 = Mb.R[36];

KP_x_21 = Mb.R[41];
KP_y_22 = Mb.R[42];
KI_x_23 = Mb.R[43];
KI_y_24 = Mb.R[44];
KD_x_25 = Mb.R[45];
KD_y_26 = Mb.R[46];

referencia_1 = Mb.R[51];
referencia_2 = Mb.R[52];


// Adequa as variáveis do Arduino com a divisão por 100
KP_x_1 = (KP_x_11/1000);
KP_y_1 = (KP_y_12/1000);
KI_x_1 = (KI_x_13/1000);
KI_y_1 = (KI_y_14/1000);
KD_x_1 = (KD_x_15/1000);
```

```
      KD_y_1 = (KD_y_16/1000);

      KP_x_2 = (KP_x_21/1000);
      KP_y_2 = (KP_y_22/1000);
      KI_x_2 = (KI_x_23/1000);
      KI_y_2 = (KI_y_24/1000);
      KD_x_2 = (KD_x_25/1000);
      KD_y_2 = (KD_y_26/1000);


       Mb.Run();

       //Atualiza os ganhos do PID
       pid_x_1.SetTunings(KP_x_1, KI_x_1, KD_x_1);//atualiza os valores em tempo de
execução
       pid_y_1.SetTunings(KP_y_1, KI_y_1, KD_y_1);

       pid_x_2.SetTunings(KP_x_2, KI_x_2, KD_x_2);//atualiza os valores em tempo de
execução
       pid_y_2.SetTunings(KP_y_2, KI_y_2, KD_y_2);

    if(Desempenho_1 > Desempenho_2)
    {
      pid_x_1.Compute();
      pid_y_1.Compute();
      servoX_1.write(Output_SMX_1);
      servoY_1.write(Output_SMY_1);
      servoX_2.write(90);
      servoY_2.write(95);
    }
    if(Desempenho_2 > Desempenho_1)
    {
      pid_x_2.Compute();
      pid_y_2.Compute();
      servoX_2.write(Output_SMX_2);
      servoY_2.write(Output_SMY_2);
      servoX_1.write(85);
      servoY_1.write(58);
    }

    Mb.R[71] = Desempenho_1;
    Mb.R[72] = Desempenho_2;

}  // Fim do Void Loop( ) ************************************


//#########################################################################
//#########################################################################

void desired_position_1()
```

```
{
 t=millis();
 pattern_counter_1 = (t+40000)%40000; //função resto da divisão
                                //(zera a cada 40000)

 //controle de posição e de rastreamento da trajetória
 //---  Referência no Centro ---
 //if(centre_1==1)
 if(referencia_1==0)
 {
  setpoint_x_1 = 175;
  setpoint_y_1 = 100;
 }

 //--- Referência circular ---
 if(referencia_1==1)
 {
  setpoint_x_1 = 175+ 40*cos(2*3.1415*0.1*t/1000);
  setpoint_y_1 = 100+ 40*sin(2*3.1415*0.1*t/1000);
 }

 //--- Referência Quadrada ---
 if(referencia_1==2)
 {
   if(pattern_counter_1 <= 10000)
   {
    setpoint_x_1 = 225;
    setpoint_y_1 = 50;
   }
   if(pattern_counter_1 > 10000 && pattern_counter_1 <= 20000)
   {
    setpoint_x_1 = 125;
    setpoint_y_1 = 50;
   }
   if(pattern_counter_1 > 20000 && pattern_counter_1 <= 30000)
   {
    setpoint_x_1 = 125;
    setpoint_y_1 = 150;
   }
   if(pattern_counter_1 > 30000 && pattern_counter_1 <= 40000)
   {
    setpoint_x_1 = 225;
    setpoint_y_1 = 150;
   }
  }
}


//#########################################################################
//#########################################################################
```

```
void read_Touch_1() //
{
    //Leitura posição-x Planta 1
    pinMode(x_gnd_1,OUTPUT);     //Xb - gnd (prepara sinais da tela p/
    pinMode(x_5v_1,OUTPUT);      //Xa - 5V   leitura da coordenada X)
    pinMode(y_gnd_1,INPUT);      //Yb - gnd
    pinMode(y_5v_1,INPUT);       //Ya - 5V
    digitalWrite(x_gnd_1,LOW);   //xb
    digitalWrite(x_5v_1,HIGH);   //xa   //aplicando 5volts no eixo X
    delay(5);
    pos_x_ball_1=analogRead(y_5v_1); //lê coordenada X

    delay(2);
    //Leitura posição-y Planta 1
    pinMode(x_gnd_1,INPUT);    //Xb - gnd
    pinMode(x_5v_1 ,INPUT);    //Xa - 5V
    pinMode(y_gnd_1,OUTPUT);   //Yb - gnd
    pinMode(y_5v_1 ,OUTPUT);   //Ya - 5V
    digitalWrite(y_gnd_1,LOW); //yb
    digitalWrite(y_5v_1,HIGH); //aplicando 5volts no eixo Y
    delay(5);
    pos_y_ball_1=analogRead(x_5v_1);  //lê coordenada Y
}


//#########################################################################
//#########################################################################
void correct_Position_1()  //media_touch
{
 detect_Touch_1(); //a bola esta sobre a tela?
 if(Z1_1>0)
 {
  read_Touch_1();

  pos_x_measured_1 = pos_x_ball_1;
  pos_y_measured_1 = pos_y_ball_1;

  //Leitura em  milímetro
  pos_x_corrected_1=((pos_x_measured_1-51)*(350-0))/(973-51);

  pos_y_corrected_1 =((pos_y_measured_1-107)*(200-0))/(880-107);

 }
}

//#########################################################################
//#########################################################################
void detect_Touch_1()
{
    //--------- Lê Z1 -----------
```

```
    //  Lê e verifica se há contato
    pinMode(x_gnd_1,OUTPUT);    // x_gnd_1
    pinMode(x_5v_1,INPUT);      // x_5v_1
    pinMode(y_gnd_1,INPUT);     // y_gnd_1
    pinMode(y_5v_1,OUTPUT);     // y_5v_1
    digitalWrite(x_gnd_1,LOW);  // x_gnd_1
    digitalWrite(y_5v_1,HIGH);  // y_5v_1
    delay(5);//10
    Z1_1 = analogRead(x_5v_1);  // x_5v_1

    if(Z1_1>0)
    {
     compute_1 = 1;
    }

  //--------- Lê Z1 ----------------------
}


//#########################################################################
//#########################################################################
void spikes_Filter_1() //Filtra spikes com variação acima de ___
{

  correct_Position_1();

  if(cont_1==0)
  {
   pos_x_last_1 = pos_x_corrected_1; //armazena o X anterior
   pos_y_last_1 = pos_y_corrected_1; //armazena o Y anterior
   cont_1++;
  }

  max_x_variation_1 = 20; //cálculo da variação máxima permitida
  max_y_variation_1 = 20; //cálculo da variação máxima permitida 2cm

  //Condição para filtrar spikes do eixo X
  if(abs(pos_x_corrected_1 - pos_x_last_1) > max_x_variation_1)
  {
     correct_Position_1(); //Chama a função p/ nova leitura de X

     if(abs(pos_x_corrected_1-pos_x_last_1) > max_x_variation_1)
     {
      pos_x_corrected_1 = pos_x_corrected_1; //Caso X novo for maior que variação
                                  //considera que a bola está em movimento
     }
     else
     {
      pos_x_corrected_1 = pos_x_last_1;  //Senão, desconsidera a leitura espúria
     }
```

```
    }

    //Condição para filtrar spikes do eixo Y
    if(abs(pos_y_corrected_1 - pos_y_last_1) > max_y_variation_1)
    {
       correct_Position_1(); //Chama a função p/ leitura de Y

       if(abs(pos_y_corrected_1-pos_y_last_1) > max_y_variation_1)
       {
        pos_y_corrected_1 = pos_y_corrected_1;
       }

       else
       {
        pos_y_corrected_1 = pos_y_last_1;
       }
    }

    pos_x_corrected_1 = pos_x_corrected_1;
    pos_y_corrected_1 = pos_y_corrected_1;

    pos_x_last_1 = pos_x_corrected_1; //armazena o X anterior
    pos_y_last_1 = pos_y_corrected_1; //armazena o Y anterior

}


//#########################################################################
//#########################################################################


//######################  Planta 2  ###################################

void desired_position_2()
{
   t=millis();
   pattern_counter_2 = (t+40000)%40000; //função resto da divisão
                                //(zera a cada 40000)

  //Controle de posição e de rastreamento da trajetória
  //--- Referência no Centro ---
  if(referencia_2==3)
  {
    setpoint_x_2 = 175;
    setpoint_y_2 = 100;
  }

  //--- Referência Circular ---
  if(referencia_2==4)
```

```
  {
   setpoint_x_2 = 175+ 40*cos(2*3.1415*0.1*t/1000);
   setpoint_y_2 = 100+ 40*sin(2*3.1415*0.1*t/1000);
  }
  //--- Referência Quadrada ---
  if(referencia_2==5)
  {
   if(pattern_counter_2 <= 10000)
   {
    setpoint_x_2 = 225;
    setpoint_y_2 = 50;
   }
   else if(pattern_counter_2 > 10000 && pattern_counter_2 <= 20000)
   {
    setpoint_x_2 = 125;
    setpoint_y_2 = 50;
   }

   else if(pattern_counter_2 > 20000 && pattern_counter_2 <= 30000)
   {
    setpoint_x_2 = 125;
    setpoint_y_2 = 150;
   }
   else if(pattern_counter_2 > 30000 && pattern_counter_2 <= 40000)
   {
    setpoint_x_2 = 225;
    setpoint_y_2 = 150;
   }
  }
}


//#########################################################################
//#########################################################################
void read_Touch_2() //
{
    //Leitura posição-x Planta 2
    pinMode(x_gnd_2,OUTPUT);     //Xb - gnd (prepara sinais da tela p/
    pinMode(x_5v_2,OUTPUT);      //Xa - 5V   leitura da coordenada X)
    pinMode(y_gnd_2,INPUT);      //Yb - gnd
    pinMode(y_5v_2,INPUT);       //Ya - 5V
    digitalWrite(x_gnd_2,LOW);   //xb
    digitalWrite(x_5v_2,HIGH);   //xa   //aplicando 5volts no eixo X
    delay(5);
    pos_x_ball_2=analogRead(y_5v_2); //lê coordenada X

    delay(2);
    //Leitura posição-y
    pinMode(x_gnd_2,INPUT);    //Xb - gnd
    pinMode(x_5v_2 ,INPUT);    //Xa - 5V
```

```
    pinMode(y_gnd_2,OUTPUT);   //Yb - gnd
    pinMode(y_5v_2 ,OUTPUT);   //Ya - 5V
    digitalWrite(y_gnd_2,LOW); //yb
    digitalWrite(y_5v_2,HIGH); //aplicando 5volts no eixo Y
    delay(5);
    pos_y_ball_2=analogRead(x_5v_2);  //lê coordenada Y
}


//#########################################################################
//#########################################################################
void correct_Position_2()  //media_touch
{
 detect_Touch_2(); //a bola esta sobre a tela?

 if(Z1_2>0)
 {
  read_Touch_2();

  pos_x_measured_2 = pos_x_ball_2;
  pos_y_measured_2 = pos_y_ball_2;

  //Leitura em  milímetro
  pos_x_corrected_2=((pos_x_measured_2-54)*(350-0))/(964-54);

  pos_y_corrected_2 =((pos_y_measured_2-97)*(200-0))/(888-97);

 }
}


//#########################################################################
//#########################################################################
void detect_Touch_2()
{
    //--------- Lê Z1 -----------
    //  Lê e verifica se há contato
    pinMode(x_gnd_2,OUTPUT);    // x_gnd_2
    pinMode(x_5v_2,INPUT);      // x_5v_2
    pinMode(y_gnd_2,INPUT);     // y_gnd_2
    pinMode(y_5v_2,OUTPUT);     // y_5v_2
    digitalWrite(x_gnd_2,LOW);  // x_gnd_2
    digitalWrite(y_5v_2,HIGH);  // y_5v_2
    delay(5);//10
    Z1_2 = analogRead(x_5v_2);  // x_5v_2

    if(Z1_2>0)
    {
     compute_2 = 1;
    }
```

```
  //--------- Lê Z1 ---------------------
}


//#########################################################################
//#########################################################################
void spikes_Filter_2() //Filtra spikes com variação acima de ___
{
  correct_Position_2();

  if(cont_2==0)
  {
   pos_x_last_2 = pos_x_corrected_2; //armazena o X anterior
   pos_y_last_2 = pos_y_corrected_2; //armazena o Y anterior
   cont_2++;
  }

  max_x_variation_2 = 20; //cálculo da variação máxima permitida
  max_y_variation_2 = 20; //cálculo da variação máxima permitida 2cm


  //Condição para filtrar spikes do eixo X
  if(abs(pos_x_corrected_2 - pos_x_last_2) > max_x_variation_2)
  {

    correct_Position_2(); //Chama a função p/  nova leitura de X

    if(abs(pos_x_corrected_2-pos_x_last_2) > max_x_variation_2)
    {
     pos_x_corrected_2 = pos_x_corrected_2; //Caso X novo for maior que variação
                                  //considera que a bola está em movimento
    }
    else
    {
     pos_x_corrected_2 = pos_x_last_2;  //Senão, desconsidera a leitura espúria
    }
  }

  //Condição para filtrar spikes do eixo Y
  if(abs(pos_y_corrected_2 - pos_y_last_2) > max_y_variation_2)
  {
    correct_Position_2(); //Chama a função p/ leitura de Y

    if(abs(pos_y_corrected_2-pos_y_last_2) > max_y_variation_2)
    {
     pos_y_corrected_2 = pos_y_corrected_2;
    }

    else
```

```
        {
          pos_y_corrected_2 = pos_y_last_2;
        }
    }

    pos_x_corrected_2 = pos_x_corrected_2;
    pos_y_corrected_2 = pos_y_corrected_2;

    pos_x_last_2 = pos_x_corrected_2; //armazena o X anterior
    pos_y_last_2 = pos_y_corrected_2; //armazena o Y anterior
}


//#########################################################################
//#########################################################################
```