UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA ELÉTRICA

ROGÉRIO FONSECA CÔRTE REAL

BALL BEAM: PROJETO, CONSTRUÇÃO E CONTROLE DE UM PROTÓTIPO DE BANCADA DIDÁTICA

TUCURUÍ
2021

ROGÉRIO FONSECA CÔRTE REAL

BALL BEAM: PROJETO, CONSTRUÇÃO E CONTROLE DE UM PROTÓTIPO DE BANCADA DIDÁTICA

Trabalho de Conclusão de Curso, apresentado como requisito parcial para a obtenção do grau de Engenheiro Eletricista pela Universidade Federal do Pará.

Orientador: Prof. Dr. Eng. Rafael Suzuki Bayma.

TUCURUÍ
2021

ROGÉRIO FONSECA CÔRTE REAL

BALL BEAM: PROJETO, CONSTRUÇÃO E CONTROLE DE UM PROTÓTIPO DE BANCADA DIDÁTICA

Trabalho de Conclusão de Curso, apresentado como requisito parcial para a obtenção do grau de Engenheiro Eletricista pela Universidade Federal do Pará.

DATA DE APROVAÇÃO: _____/_____/_____

CONCEITO: _____

BANCA EXAMINADORA

_____
Prof. Dr. Eng. Rafael Suzuki Bayma
Orientador - UFPA

_____
Prof. Dr. Eng. Cleison Daniel Silva
Membro - UFPA

_____
Prof. Me. Eng. André Felipe Souza da Cruz
Membro - UFPA

RESUMO

O presente trabalho visa o desenvolvimento de um protótipo de bancada didática para um sistema *ball beam*, sua modelagem matemática e implementação de um controlador digital. O desafio havia sido lançado ainda no primeiro ano do curso de Engenharia Elétrica, pelo professor Raphael Teixeira, que desenvolvia atividades no Laboratório de Controle com alunos de todos os períodos, desde calouros à concluintes. O projeto do protótipo foi desenvolvido após pesquisas onde avaliou-se diversos modelos de *ball beam* existentes, além de fornecedores para os materiais necessários à sua confecção. Os componentes utilizados, atuador e transdutores, foram objeto de extensos estudos e ensaios, determinando-se diversas características, a princípio ocultas, que impactaram na implementação do controlador. O projeto do controlador iniciou-se pela modelagem matemática, seguida pela definição das especificações de desempenho. Adotou-se uma arquitetura de controle PD sem a derivada do erro, mas devido as limitações impostas pelo atuador e transdutores, fez-se uso de recursos como compensadores, filtragem com processamento estatístico, estimador recursivo e preditor. Os resultados construtivos e de funcionamento ficaram próximos aos resultados teóricos, de forma que os recursos utilizados para mitigar as limitações dos componentes utilizados se mostraram satisfatórios.

Palavras-chave: Ball Beam. Controlador PD. Controlador Digital. Processamento de Sinais.

ABSTRACT

The present work aims at the development of a didactic bench prototype for a ball beam system, its mathematical modeling and the implementation of a digital controller. The challenge had been launched in the first year of the Electrical Engineering course, by Professor Raphael Teixeira, who developed activities in the Control Laboratory with students from all periods, from freshmen to seniors. The prototype project was developed after research in which several existing ball beam models were evaluated, in addition to suppliers for the materials needed for their manufacture. The components used, actuator and transducers, were the object of extensive studies and tests, determining several characteristics, at first hidden, which impacted the implementation of the controller. The controller project started with mathematical modeling, followed by the definition of performance specifications. A PD control architecture was adopted without the error derivative, but due to the limitations imposed by the actuator and transducers, resources such as compensators, filtering with statistical processing, recursive estimator and predictor were used. The constructive and operational results were close to the theoretical results, so that the resources used to mitigate the limitations of the components used proved to be satisfactory.

Keywords: Ball Beam. PD Controller. Digital Controller. Signal Processing.

LISTA DE SÍMBOLOS

β	Deslocamento angular da barra nos ensaios de bancada
h	Deslocamento linear da barra nos ensaios de bancada
g	Aceleração da gravidade
m	Massa da esfera
W	Distância entre os pontos de contato da esfera com a barra
J	Momento de inércia da esfera
R	Raio da esfera
r	Raio de revolução do movimento de rolamento
x	Posição linear da esfera sobre a barra
θ	Posição angular da barra
K	Constante em função de g, R e W
f_e	Força de atrito estático
t_{pwm}	Período do pulso PWM em nível alto
t_0	*Offset* do pulso PWM para manter a barra na horizontal
θ_s	Posição angular equivalente ao sinal aplicado ao servomotor
θ_f	Desvio de posição angular devido a folga nas engrenagens
θ_t	Desvio de posição angular devido ao torque externo
M_p	Especificação de desempenho: sobressinal ou *overshot*
t_r	Especificação de desempenho: tempo de subida
t_p	Especificação de desempenho: tempo de pico
t_s	Especificação de desempenho: tempo de acomodação
m_s	Especificação de desempenho: critério do tempo de acomodação

K_P	Ganho proporcional
K_D	Ganho derivativo
G_p	Função de transferência do sistema *ball beam*
G_{cp}	Função de transferência do controlador proporcional
G_{cd}	Função de transferência do controlador derivativo
G_{mf}	Função de transferência de malha fechada
ω_n	Frequência natural não amortecida
ξ	Coeficiente de amortecimento
f_b^{mf}	Largura de banda do processo em malha fechada
T	Período de amostragem
xt_k	Leitura de posição da esfera na iteração k
y_k	Saída do filtro de média móvel na iteração k
k_e	Ganho do filtro estimador
$\hat{x}_k$	Posição estimada da esfera na iteração k
x_k	Saída do filtro estimador na iteração k
$x_{k\|k+d}$	Predição da posição da esfera na iteração k para o estado futuro $k+d$
u_k	Saída do controlador PD após o bloco saturador na iteração k

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES GERAIS

Amplamente utilizado no âmbito acadêmico, o sistema *ball beam* é um sistema clássico dos cursos de controle (YU e ORTIZ, 2005), utilizado como configuração de controle de referência para avaliação de estratégias de controle.

O sistema *ball beam* consiste basicamente em um sistema composto por uma esfera que desliza sobre uma barra, onde o objetivo é manter a esfera sob controle em uma posição de referência, estática ou dinâmica, através do rolamento da esfera sobre a barra, devido à ação de controle que atua sobre a inclinação da barra.

É possível encontrar comercialmente no mercado uma grande variedade de bancadas didáticas *ball beam*, contudo possuem elevados custos de aquisição. Outrossim, o desenvolvimento de protótipos didáticos, comum em cursos de engenharia, contribui para uma formação mais consistente dos alunos e na associação da teoria à prática. Os protótipos de *ball beam* podem ser construídos de maneiras diversas, com diferentes formas de atuadores, pontos de articulação, entre outros, cada qual apresentando particularidades próprias de projeto e controle.

1.2 OBJETIVO

O objetivo do presente trabalho é o projeto e construção de um protótipo de bancada didática *ball beam*, bem como o desenvolvimento de um sistema de controle e sua implementação através de uma plataforma eletrônica com microcontrolador.

1.3 METODOLOGIA

Para o projeto do protótipo, realizou-se uma pesquisa onde foram avaliados uma gama de modelos de *ball beam*, bem como fornecedores dos materiais necessários à sua confecção.

Após a etapa de pesquisa, definiu-se um modelo de bancada e seguiu-se para o projeto e aquisição dos materiais e equipamentos que foram utilizados em sua construção, os quais foram estudados e ensaiados para determinar seus princípios de funcionamento e suas limitações.

Para o desenvolvimento do sistema de controle, inicialmente realizou-se a modelagem matemática do sistema, sendo seguida pelo projeto do controlador a partir da definição das especificações de desempenho.

A arquitetura de controle utilizada foi um controlador PD sem a derivada do erro, onde os ganhos foram determinados a partir das especificações de desempenho.

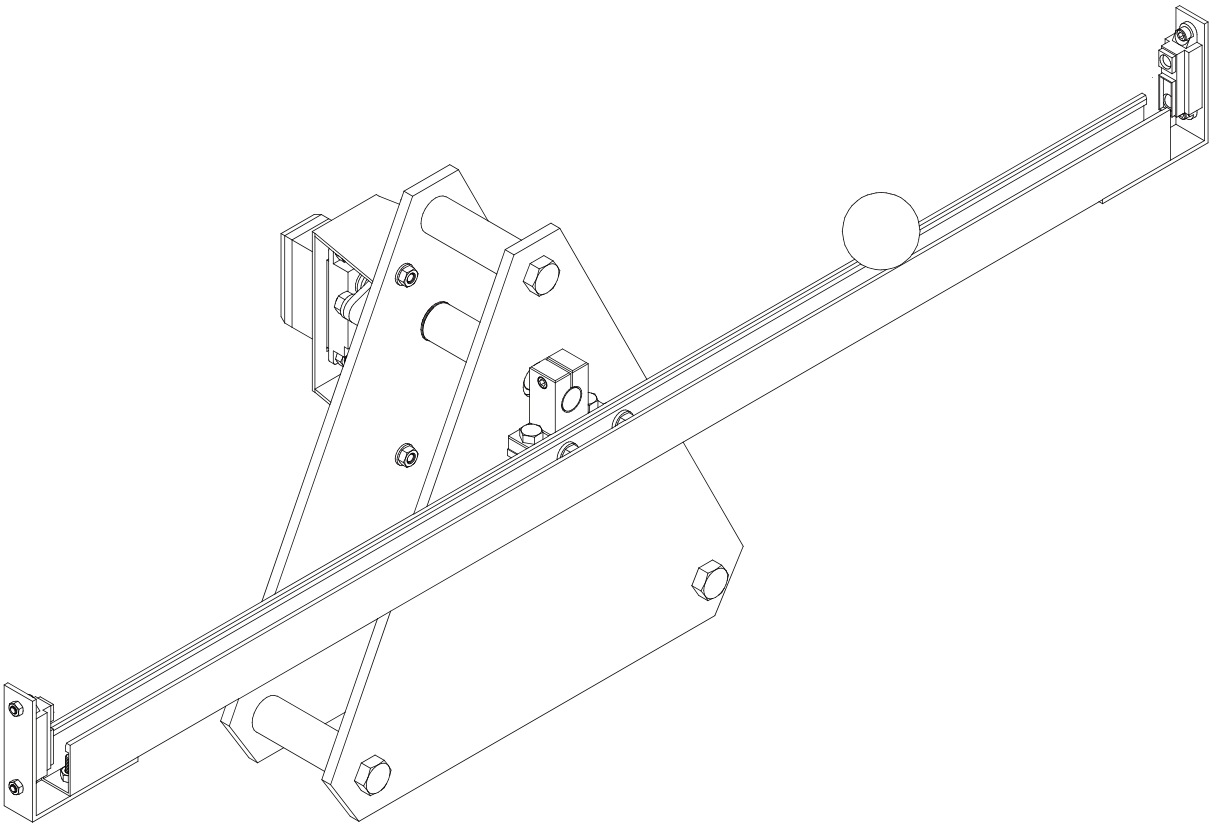
O atuador e transdutores utilizados na construção da bancada apresentaram algumas limitações de projeto, de forma que, a fim de se obter resultados satisfatórios no controle do sistema, mostrou-se necessário adotar soluções tanto para o sinal do atuador, através de compensadores, quanto para o sinal proveniente dos transdutores, através de processamento de sinal utilizando-se de técnicas de filtragem de moda e de média móvel, estimador recursivo e preditor.

2 BANCADA BALL BEAM

Para confecção da bancada, inicialmente realizou-se uma pesquisa onde foram avaliados uma gama de modelos de bancada *ball beam*, como por exemplo COSTA, 2018, NASCIMENTO FILHO, 2008 e CONTROLE DE UM BALL AND BEAM UTILIZANDO SISTEMA SEGUIDOR COM ATRIBUIÇÃO DE AUTO-ESTRUTURA COMPLETA, 2014. Buscou-se também fornecedores dos materiais necessários: servomotor, eixo de transmissão, rolamentos, suportes e etc.

Após a pesquisa adotou-se por conveniência um modelo que possui um chassi composto por duas chapas de aço paralelas, fixadas entre si por parafusos, porcas e espaçadores. Ao chassi é fixado o suporte do servomotor. A transmissão do torque do motor a barra de rolagem da esfera é realizada por um eixo, que atravessa o chassi, apoiado em dois mancais os quais são fixados ao chassi através de um espaçador. São utilizados suportes para fixar o servomotor e a barra ao eixo. Nas extremidades da barra foram colocados dois sensores de distância, um de cada lado, os quais tem a função de fazer a leitura de posição da esfera. Na Figura 1 observa-se o projeto da bancada em perspectiva.

Figura 1 – Vista em perspectiva do projeto da bancada *ball beam*



Fonte: Elaborado pelo autor do trabalho.

Definiu-se o comprimento da barra em 600mm e o chassi foi projetado de forma a proporcionar um movimento de giro de 20° para a barra em ambas as direções.

2.1 MATERIAIS UTILIZADOS

Parte dos materiais utilizados na bancada foram adquiridos através da internet em sites estrangeiros devido ao baixo custo. Outros foram adquiridos de fornecedores locais. A Tabela 1 apresenta a lista dos materiais utilizados com seus valores de custo. Os itens foram agrupados de acordo com a parte construtiva a que pertencem, a saber: Chassi; Atuador; Transmissão; Barra e Periféricos.

Tabela 1 – Lista de materiais utilizados na construção da bancada com valores de custo

Item	Quantidade	Descrição	Custo (R$)

		Chassi	
1	2	Chapa de aço triangular 250 x 250mm x 3/16"	38,00
2	3	Tubo Ø5/8" x 1,5 x 55mm	–
3	3	Parafuso hexagonal M8 x 70mm	4,50
4	3	Porca M8	0,90
5	3	Arruela lisa M8	0,30
		Subtotal	43,70
		Atuador	
6	1	Servomotor GS-5515MG 15kgf·cm	42,36
7	1	Chapa de aço 1,2 x 50 x 174mm	–
8	4	Parafuso allen M4 x 10mm	0,40
9	4	Porca M4	0,20
10	4	Parafuso hexagonal M5 x 15mm	0,60
11	4	Porca M5	0,20
12	4	Arruela lisa M5	0,20
		Subtotal	43,96

Tabela 1 – Lista de materiais utilizados na construção da bancada com valores de custo

Item	Quantidade	Descrição	Custo (R$)
		Transmissão	
13	1	Eixo de aço Ø10 x 100mm	18,99
14	2	Mancal rolamento para eixo Ø10mm	10,00
15	1	Tubo Ø5/8" x 1,5 x 53,5mm	–
16	1	Flange para servomotor	14,49
17	1	Suporte horizontal para eixo Ø10mm	15,99
18	2	Parafuso hexagonal M4 x 15mm	0,30
19	1	Suporte *stand-up* para eixo Ø10mm	15,99
20	1	Cantoneira 3 x 20 x 25 x 50mm	–
21	2	Parafuso hexagonal M5 x 15mm	0,30
22	2	Parafuso phillips M5 x 25mm	0,30
23	8	Porca M5	0,40
24	4	Arruela lisa M5	0,20
		Subtotal	76,96
		Barra	
25	1	Perfil U alumínio 1 x 15,5 x 24 x 600mm	5,00
26	1	Esfera de aço Ø30mm	22,00
27	2	Sensor de distância GP2Y0A21YK0F	100,00
28	2	Cantoneira 2 x 15,5 x 60 x 60mm	–
29	4	Parafuso phillips M3 x 6mm	0,10
30	4	Porca M3	0,10
		Subtotal	127,20
		Periféricos	
31	1	Arduino Uno	46,31
32	1	Fonte de alimentação 6V/5A	60,00
		Subtotal	106,31
		Total geral	398,13

Fonte: Elaborado pelo autor do trabalho.

2.2 CONSTRUÇÃO DA BANCADA

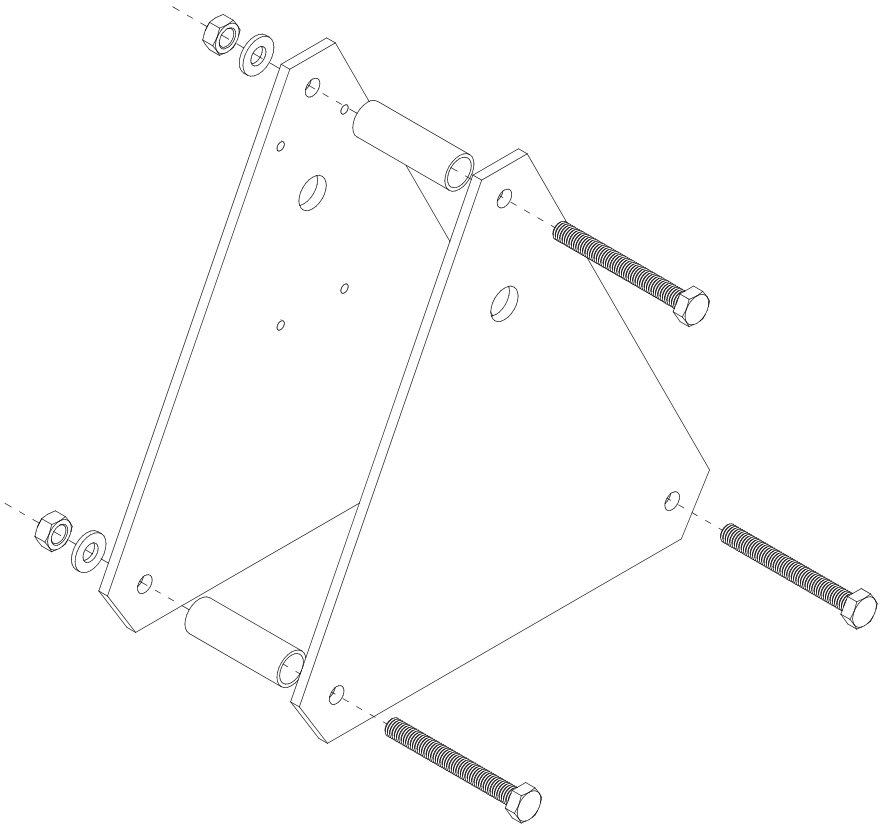
A bancada foi construída a partir dos materiais listados na Tabela 1. Foram necessários serviços de corte, furação e dobra nas chapas de aço, os quais foram executados em oficina

própria. Também foi necessária uma pequena adaptação para fixar o flange do servomotor ao suporte horizontal do eixo.

2.2.1 CHASSI

O chassi é constituído por duas placas de aço em formato de um triângulo isósceles com base e altura de 250mm. Os cantos dos triângulos foram chanfrados a cerca de 25mm dos vértices para eliminar pontas agudas. Ambas as chapas possuem furação para fixação entre si, apoio dos mancais e passagem do eixo. A placa traseira possui ainda furação para fixação do suporte do servomotor. Na Figura 2 é apresentada uma vista explodida do chassi. Observa-se que os espaçadores determinam a largura final do chassi.

Figura 2 – Vista explodida do chassi



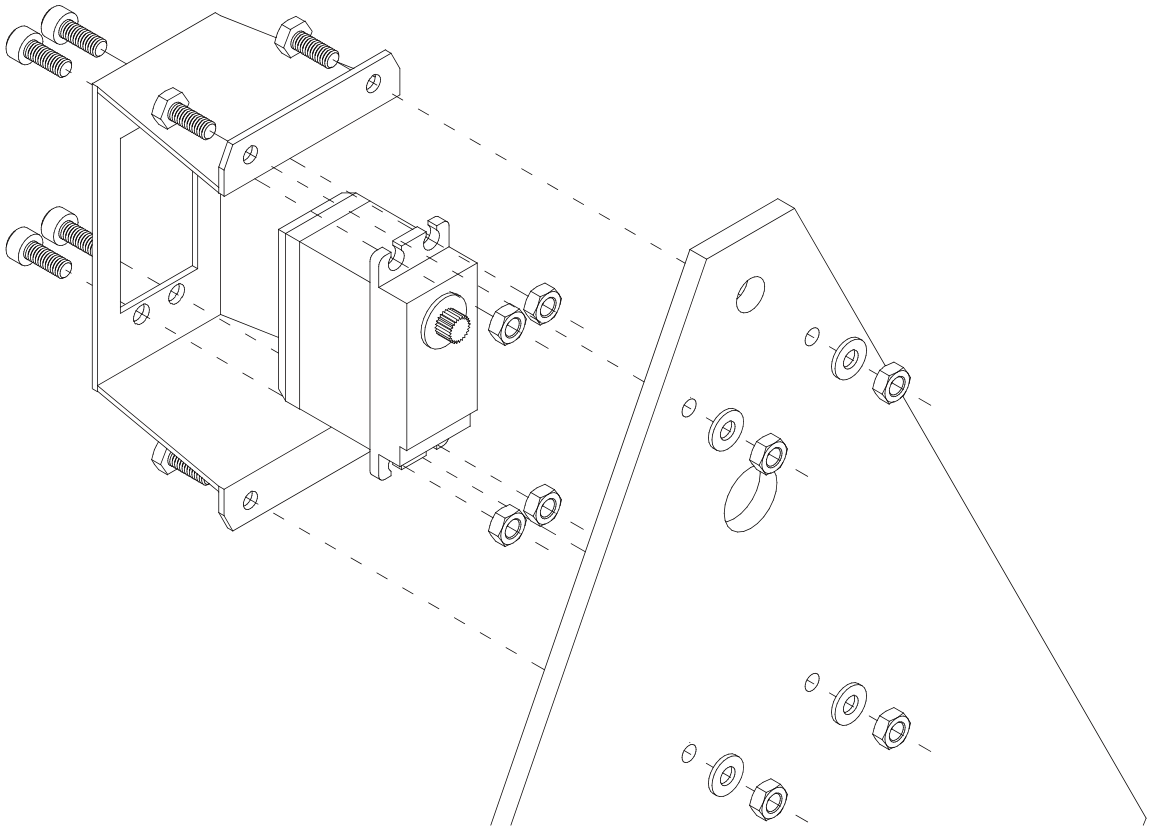
Fonte: Elaborado pelo autor do trabalho.

2.2.2 ATUADOR

O atuador é composto por um servomotor marca GOTECK modelo GS-5515MG com torque de 15kgf·cm (GOTECK RC INC, 2010). O servomotor é fixado com 4 parafusos allen

M4 x 10mm e porcas a um suporte fabricado a partir de uma chapa de aço furada e dobrada. O suporte por sua vez é fixado ao chassi através de 4 parafusos hexagonais M5 x 15mm, arruela e porcas. A Figura 3 apresenta uma vista explodida dos componentes do atuador.

Figura 3 – Vista explodida do atuador

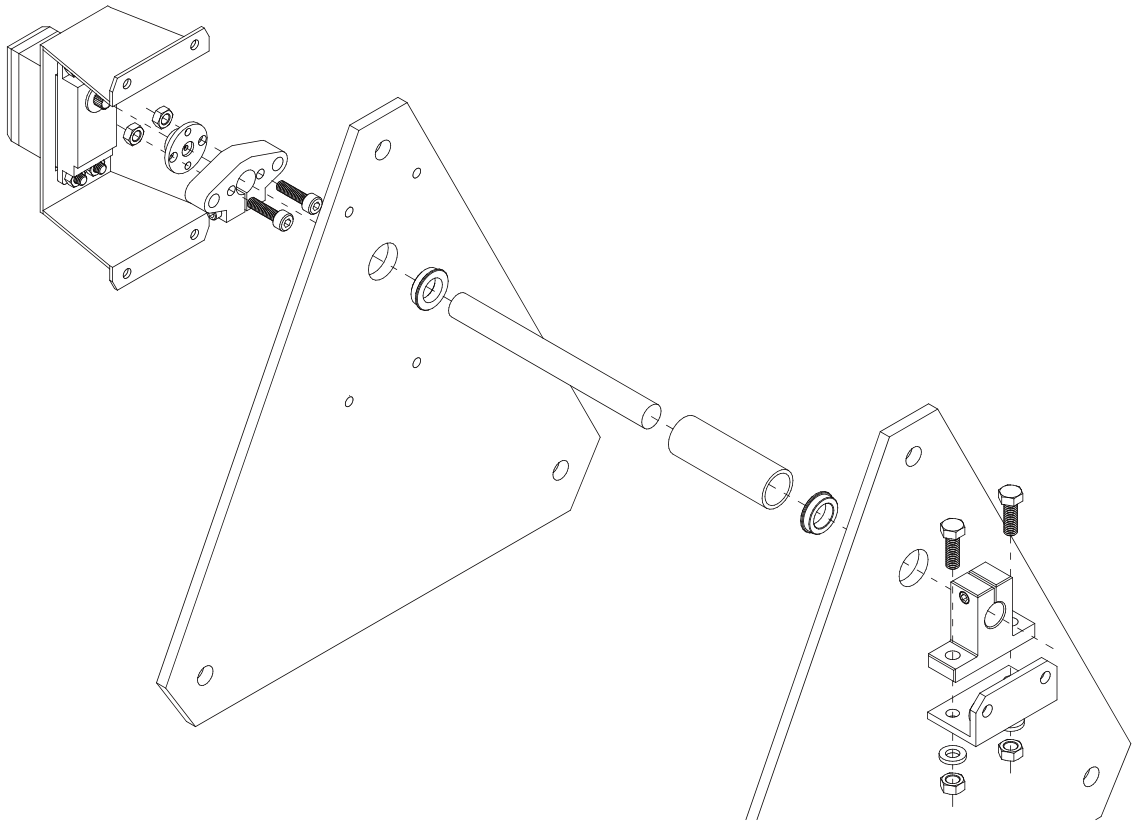


Fonte: Elaborado pelo autor do trabalho.

2.2.3 TRANSMISSÃO

A transmissão é responsável por transmitir a barra o torque fornecido pelo servomotor. Seu componente principal é um eixo de aço de Ø10 x 100mm o qual é apoiado por dois rolamentos tipo mancal com abas. Os rolamentos são introduzidos nos orifícios das chapas do chassi com as abas viradas para parte interna do chassi. Um espaçador mantém os rolamentos fixados aos orifícios. A fixação do eixo ao servomotor é feita por um conjunto de flange com um suporte tipo horizontal, os quais são fixados entre si através de uma furação adaptada e um conjunto de parafusos e porcas. No lado da barra é fixado ao eixo um conjunto formado por suporte tipo *stand-up*, cantoneira, parafusos, arruelas e porcas, sendo a cantoneira o suporte para fixação da barra. Na Figura 4 é apresentada uma vista explodida da transmissão.

Figura 4 – Vista explodida da transmissão

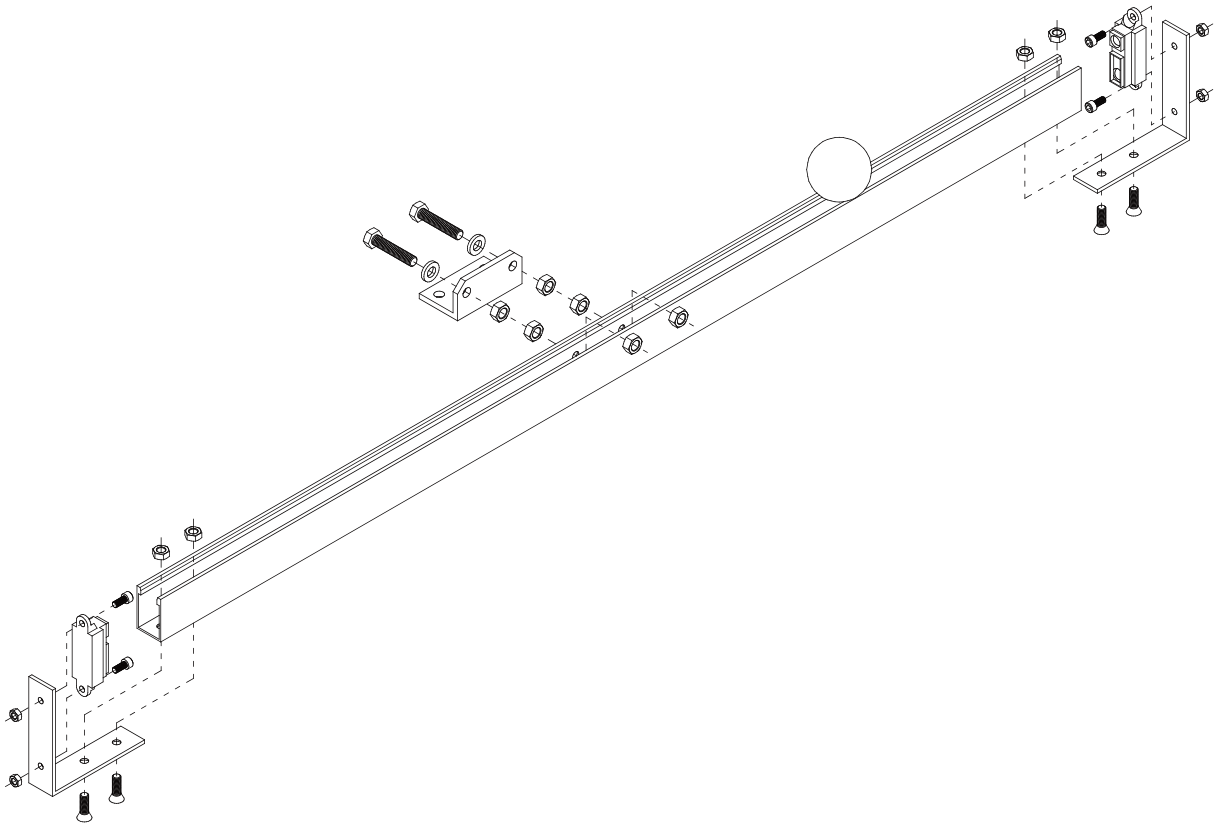


Fonte: Elaborado pelo autor do trabalho.

2.2.4 BARRA E ESFERA

A barra e a esfera podem ser considerados os principais componentes do sistema sendo responsáveis pela sua denominação. A barra é o elemento que serve de pista de rolamento para esfera, sendo fixada a cantoneira da transmissão através de um conjunto de parafusos e porcas utilizadas como espaçador. Em cada extremidade da barra é fixado um suporte com sensor de distância, os quais são utilizados na determinação da posição da esfera sobre a barra. A barra em si é um perfil de alumínio em formato de “U” com 15,5mm de largura, 24mm de altura e 600mm de comprimento. A esfera é de aço maciço com diâmetro de 30mm. Os suportes dos sensores são feitos de cantoneira de aço, sendo fixadas a barra através de um conjunto de parafusos e porcas. Os sensores de distância são do tipo infravermelho marca SHARP modelo GP2Y0A21YK0F com faixa de medição de 10 a 80cm (SHARP CORPORATION, 2006). Os sensores são fixados por parafusos e porcas. A Figura 5 apresenta uma vista explodida da barra.

Figura 5 – Vista explodida da barra



Fonte: Elaborado pelo autor do trabalho.

2.2.5 PERIFÉRICOS

Os componentes periféricos são os componentes externos a bancada e que são utilizados para sua colocação em funcionamento. Tem-se no presente trabalho como componentes periféricos uma fonte de alimentação de 6V/5A, uma placa Arduino Uno, uma protoboard e cabos.

2.3 FUNCIONAMENTO DA BANCADA

Para se atingir os resultados desejados, além das técnicas de controle, é necessário conhecer o correto funcionamento dos componentes de um sistema, além de suas limitações e/ou recursos. Os componentes utilizados possuem princípios de funcionamento próprios, além de limitações que precisam ser entendidas e mensuradas a fim de se extrair todo o potencial da bancada. Nem todas as informações necessárias estão disponíveis na documentação dos componentes, o que torna essencial uma investigação experimental de suas características.

2.3.1 SERVOMOTOR

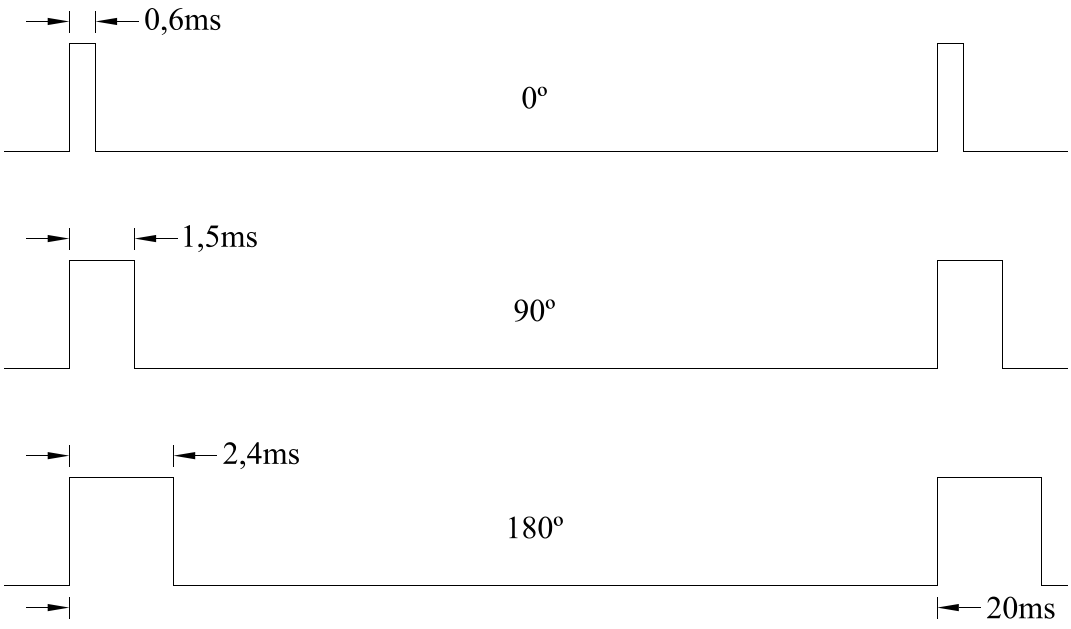
O servomotor é um componente eletromecânico composto por um motor DC, uma caixa de redução, um circuito de controle e um potenciômetro que é responsável por identificar a posição angular do eixo de saída, a qual é determinada por um sinal de controle.

2.3.1.1 Princípio de Funcionamento

A posição angular de um servomotor é determinada por um sinal de controle do tipo modulação por largura de pulso, mais conhecido por *Pulse Width Modulation* (PWM), que é um sinal digital e periódico onde durante uma parte do ciclo o sinal encontra-se em nível lógico alto e no restante do ciclo em nível lógico baixo. Para o controle do servomotor o sinal PWM deve ter um período de 20ms sendo o tempo em que o sinal fica em nível lógico alto o fator que determina sua posição angular. O servomotor utilizado possui curso de 180° e para uma posição de curso médio, ou seja, de 90°, deve-se aplicar um pulso de 1,5ms, conforme especificado em seu *datasheet*. A largura de pulso para as posições de 0° e 180° foram determinadas a partir de testes na bancada e são respectivamente de 0,6ms e 2,4ms.

Na Figura 6 é apresentado o sinal PWM para as posições de 0°, 90° e 180°.

Figura 6 – Sinal PWM para as posições de 0°, 90° e 180°



Fonte: Elaborado pelo autor do trabalho.

Quaisquer posições intermediárias podem ser obtidas aplicando-se um sinal linearmente proporcional, onde cada incremento de 10μs corresponde a 1°, conforme demonstrado na Equação (1).

$$\frac{2400\mu s - 600\mu s}{180° - 0°} = 10\,\mu s/° \tag{1}$$

2.3.1.2 Limitações

Foram identificadas as seguintes limitações no servomotor:

a) largura de banda morta;
b) velocidade angular;
c) folga nas engrenagens;
d) capacidade de resposta à aplicação de um torque externo.

2.3.1.2.1 Largura de banda morta

O servomotor utilizado possui uma largura de banda morta de 8μs de acordo com o *datasheet*. Desta forma o servomotor tem sua posição angular praticamente discretizada em intervalos de 8μs da largura de pulso do sinal de controle. Isto evita também que o servomotor fique respondendo à ruídos, porém acarreta uma não linearidade que afeta o desempenho do sistema, pois o servomotor deixa de atuar quando são necessárias pequenas correções na inclinação da barra a fim de estabilizar a esfera, fazendo com que possa ocorrer uma oscilação da esfera em torno da posição de referência.

2.3.1.2.2 Velocidade angular

A velocidade angular do servomotor, especificada no *datasheet*, é de $0,28s/60°$ quando operando sob uma alimentação de 6Vdc, ou seja, menos de $5ms/°$.

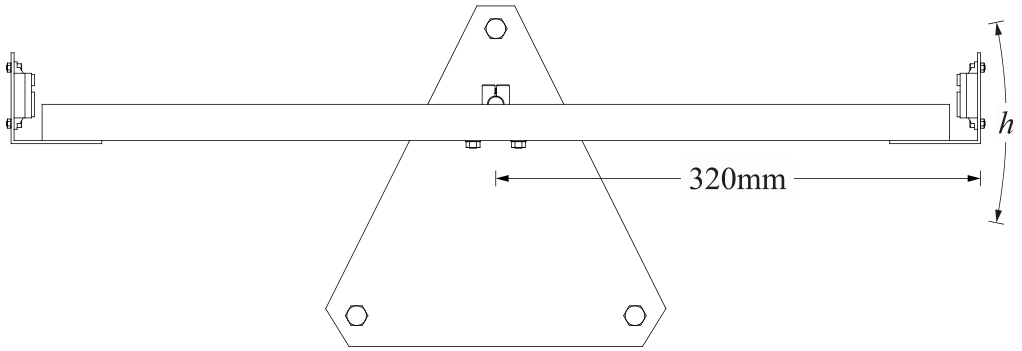
2.3.1.2.3 Folga nas engrenagens

Foi observado durante os testes com a bancada uma folga na barra, proveniente das engrenagens da caixa de redução do servomotor. Com o auxílio de uma régua determinou-se a folga na extremidade da barra, a uma distância de 320mm do eixo de rotação. A variação

vertical proveniente da folga, medida na extremidade da barra, foi de 19mm, conforme representado na Figura 7.

A variação total é de 19mm, de forma que a variação em torno do ponto médio é de ±9,5mm. O desvio devido a folga, em termos angulares, é de ±1,7°, determinada com o auxílio da Equação (2), onde β é a variação angular e h a variação vertical em mm.

Figura 7 – Folga proveniente das engrenagens do servomotor



Fonte: Elaborado pelo autor do trabalho.

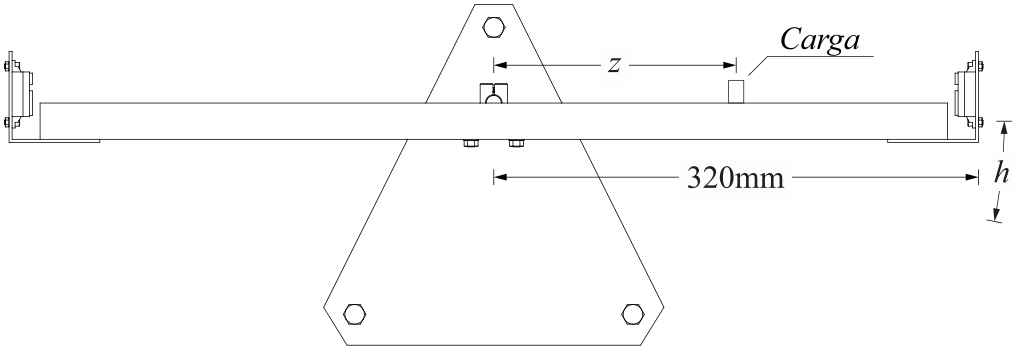
$$\beta = \arctan\left(\frac{h}{320}\right) \tag{2}$$

2.3.1.2.4 Capacidade de resposta à aplicação de um torque externo

Quando se aplica um torque externo ao eixo do servo motor, o circuito de controle atua na tentativa de manter a posição angular de acordo com o sinal de controle recebido. Contudo nesta condição é observado um desvio na posição angular proporcional ao torque aplicado.

A fim de se determinar o comportamento deste efeito foram executados testes com uma carga de 110g, mesma massa da esfera, em diferentes pontos da barra, haja vista que o torque exercido pela carga é proporcional à distância do centro em que esta é colocada. Foi aplicado um sinal de controle que levasse a barra para horizontal, compensando também o desvio produzido pela folga descrita na seção anterior de acordo com o lado da barra em que a carga era posicionada. Com o auxílio de uma régua determinou-se o deslocamento vertical produzido na extremidade da barra, a uma distância de 320mm do eixo de rotação, para cada posição em que se colocou a carga, conforme ilustrado na Figura 8. Com o auxílio da Equação (2) determinou-se o desvio como sendo de $4^{\circ}/\mathrm{m}$.

Figura 8 – Teste para determinar o desvio da posição angular devido à aplicação de um torque externo



Fonte: Elaborado pelo autor do trabalho.

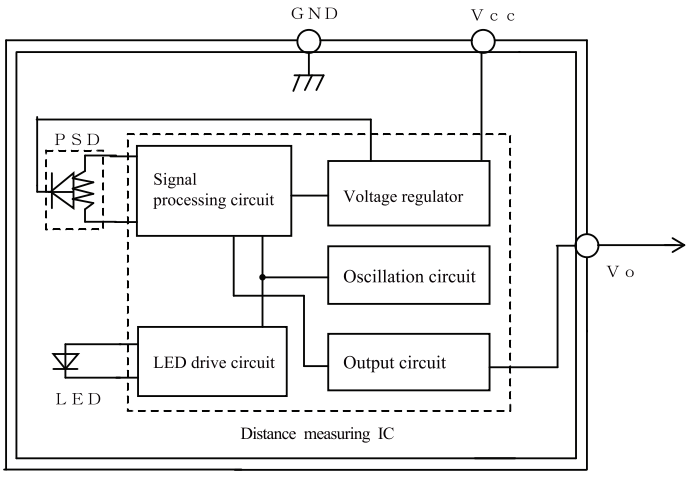
2.3.2 SENSOR DE DISTÂNCIA

Sensores de distância medem a distância entre um ponto de referência e um objeto alvo, neste caso o próprio sensor e a esfera, respectivamente.

2.3.2.1 Princípios de Funcionamento

Os sensores de distância utilizados na bancada possuem um conjunto de transmissor e receptor de luz infravermelha. A distância é determinada pelo circuito de processamento e uma tensão correspondente pode ser observada no circuito de saída. A Figura 9 apresenta um diagrama de blocos dos sensores utilizados.

Figura 9 – Diagrama de blocos do sensor de distância

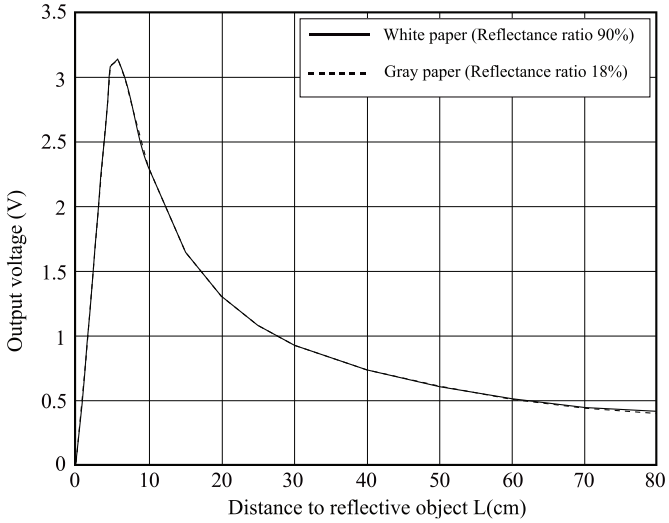


Fonte: SHARP CORPORATION, 2006, p. 2.

O sinal de saída do sensor, *Output voltage* (V), em função da distância do objeto refletivo, *Distance to reflective object* L(cm), é apresentado na Figura 10. É um sinal inversamente proporcional à distância medida, o que é melhor demonstrado na Figura 11.

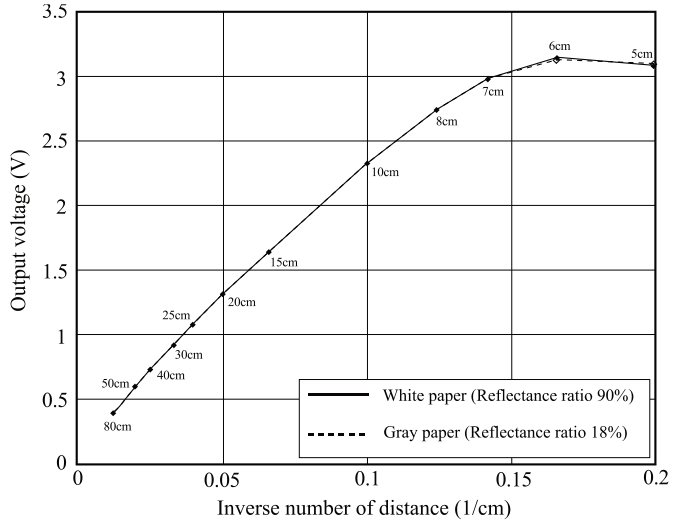
Observando a Figura 11, verifica-se que o sinal de saída em função do inverso da distância medida, *Inverse number of distance* (1/cm), é praticamente linear dentro da faixa de operação do sensor, podendo ainda ser mapeado por zonas.

Figura 10 – Sinal de saída do sensor em função da distância



Fonte: SHARP CORPORATION, 2006, p. 5.

Figura 11 – Sinal de saída do sensor em função do inverso da distância



Fonte: SHARP CORPORATION, 2006, p.5.

2.3.2.2 Limitações

Foram identificadas as seguintes limitações nos sensores de distância:

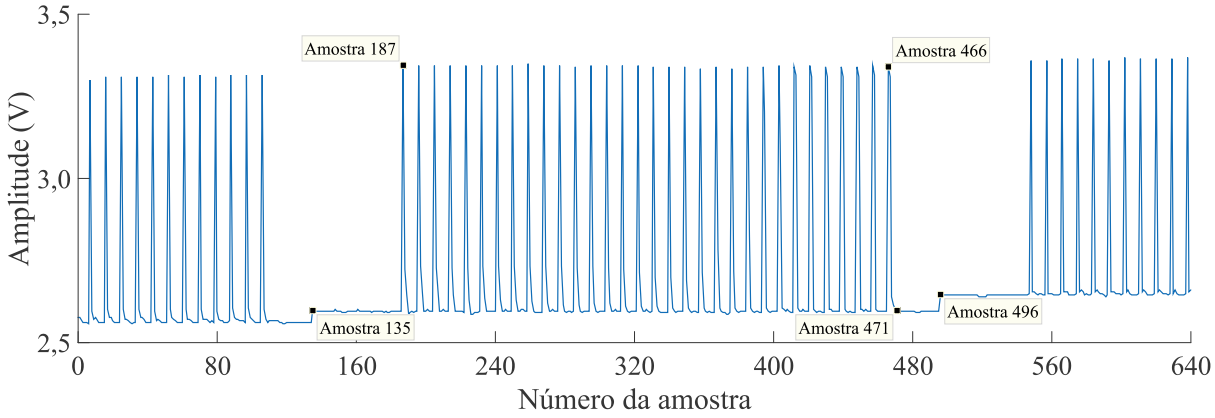
a) ruído do sinal de saída;
b) interferência óptica entre os sensores;
c) taxa de atualização das leituras;
d) discretização do sinal de saída.

2.3.2.2.1 Ruído do sinal de saída

O sinal lido a partir da saída dos sensores apresentou um ruído conforme apresentado na Figura 12. Depois de várias amostragens concluiu-se que não se trata de um ruído aleatório, mas sim um ruído periódico, o qual é produzido durante o processamento de uma nova leitura de distância. Na Figura 12 são apresentadas 640 amostras tomadas com espaçamento uniforme de 104μs. Para evidenciar os instantes em que o sensor atualiza o sinal de saída em função de uma nova leitura, conforme descrito em 2.3.2.2.1, a amostragem foi realizada com a esfera em leve movimento, de forma que duas leituras consecutivas tenham níveis de tensão diferente.

Na Figura 12 existem dois instantes em que houve atualização da leitura de distância, nas amostras de número 135 e 496, evidenciadas por um pequeno degrau em cada atualização. A partir dos dados obtidos observa-se que após a atualização do sinal de saída na amostra 135, o sinal permanece praticamente sem ruído até a amostra 186, então a partir da amostra 187 são observados 32 pulsos no sinal de saída, sendo o último observado na amostra 466. A partir da amostra 471 o sinal retorna ao nível em que praticamente não há ruído, permanecendo neste até a atualização da nova leitura na amostra 496. Este padrão se repetiu, com ambos os sensores, nas diversas amostragens realizadas em testes com a bancada.

Figura 12 – Amostragem do sensor direito



Fonte: Elaborado pelo autor do trabalho.

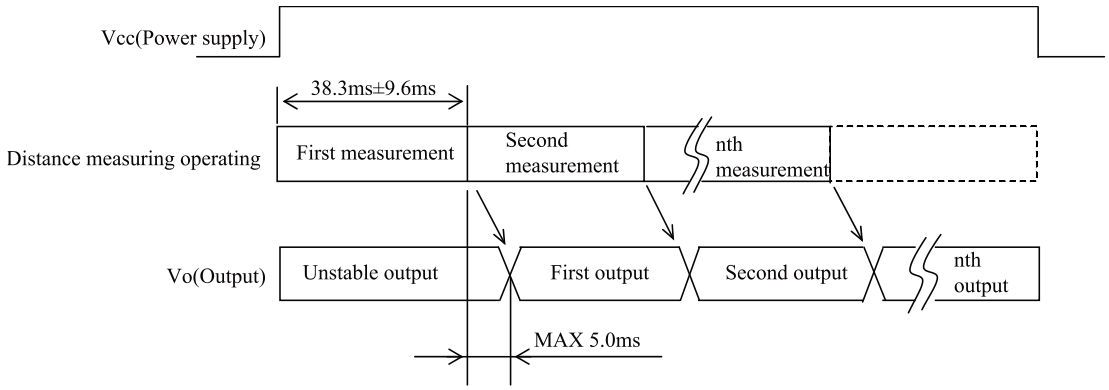
2.3.2.2.2 Interferência óptica entre os sensores

Durante os testes com a bancada, observou-se um comportamento na resposta dos sensores quando ambos foram ligados simultaneamente. Quando foi ligado apenas um dos sensores de cada vez, suas respostas foram bastante limpas e precisas, apresentando respostas satisfatórias para distâncias de até mais de 400mm. Contudo quando ambos os sensores foram ligados simultaneamente, observou-se respostas variantes mesmo com a esfera parada, reduzindo a precisão a curta distância e degradando-a para distância superiores a 350mm. Atribui-se a este comportamento o fato dos sensores estarem um de frente para o outro, de forma que a luz emitida por um sensor, interfere na recepção do outro, isto é, uma interferência óptica entre eles.

2.3.2.2.3 Taxa de atualização das leituras

O sinal de saída do sensor não é alterado continuamente. O processamento de uma nova leitura de distância ocorre a cada 38,3ms ±9,6ms, conforme apresentado na Figura 13. Isto não quer dizer que as leituras são atualizadas a intervalos de tempo diferentes, mas que cada sensor possui uma taxa de atualização própria dentro da faixa especificada. Após testes determinou-se a taxa de atualização dos sensores utilizados na bancada, as quais são de 39,4ms para o sensor esquerdo e de 37,5ms para o sensor direito.

Figura 13 – Diagrama de tempo do sensor de distância



Fonte: SHARP CORPORATION, 2006, p. 4.

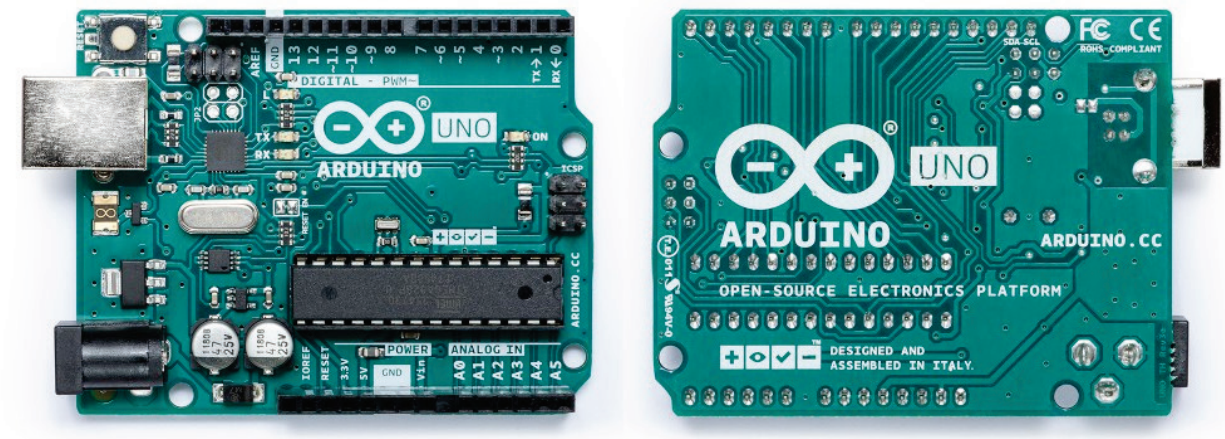
2.3.2.2.4 Discretização do sinal de saída

O *datasheet* do sensor especifica que seu sinal de saída é analógico, contudo após testes na bancada concluiu-se que o sinal é discretizado em níveis de aproximadamente 19,5mV, ou seja, aparenta ser gerado por um conversor digital analógico (DAC) de 8 bits sobre a tensão de alimentação (5V) o que implicaria em um *least significant bit* (LSB) de 19,53mV.

2.3.3 ARDUINO UNO

O Arduino é uma plataforma eletrônica de código aberto baseada em hardware e software fáceis de usar. É uma plataforma que nasceu como uma ferramenta fácil para prototipagem rápida, destinada a estudantes sem formação em eletrônica e programação.

Figura 14 – Arduino Uno



Fonte: ARDUINO UNO REV3.

Existem diversos tipos de placas Arduino no mercado, sendo que no presente trabalho utilizou-se uma placa Arduino Uno similar a apresentada na Figura 14, a qual é baseada em um microcontrolador de 8 bits modelo ATmega328 fabricado pela Atmel.

2.3.3.1 Recursos Úteis ao Projeto

O Arduino, através de seu microcontrolador, possui um gama de recursos. No escopo deste trabalho os principais recursos do Arduino a serem utilizados, além de memória, processamento matemático, entre outros, serão o conversor analógico digital (ADC), para leitura dos sensores, e o conversor digital analógico, implementado através de modulação por largura de pulso (PWM), utilizado na geração do sinal enviado ao servomotor.

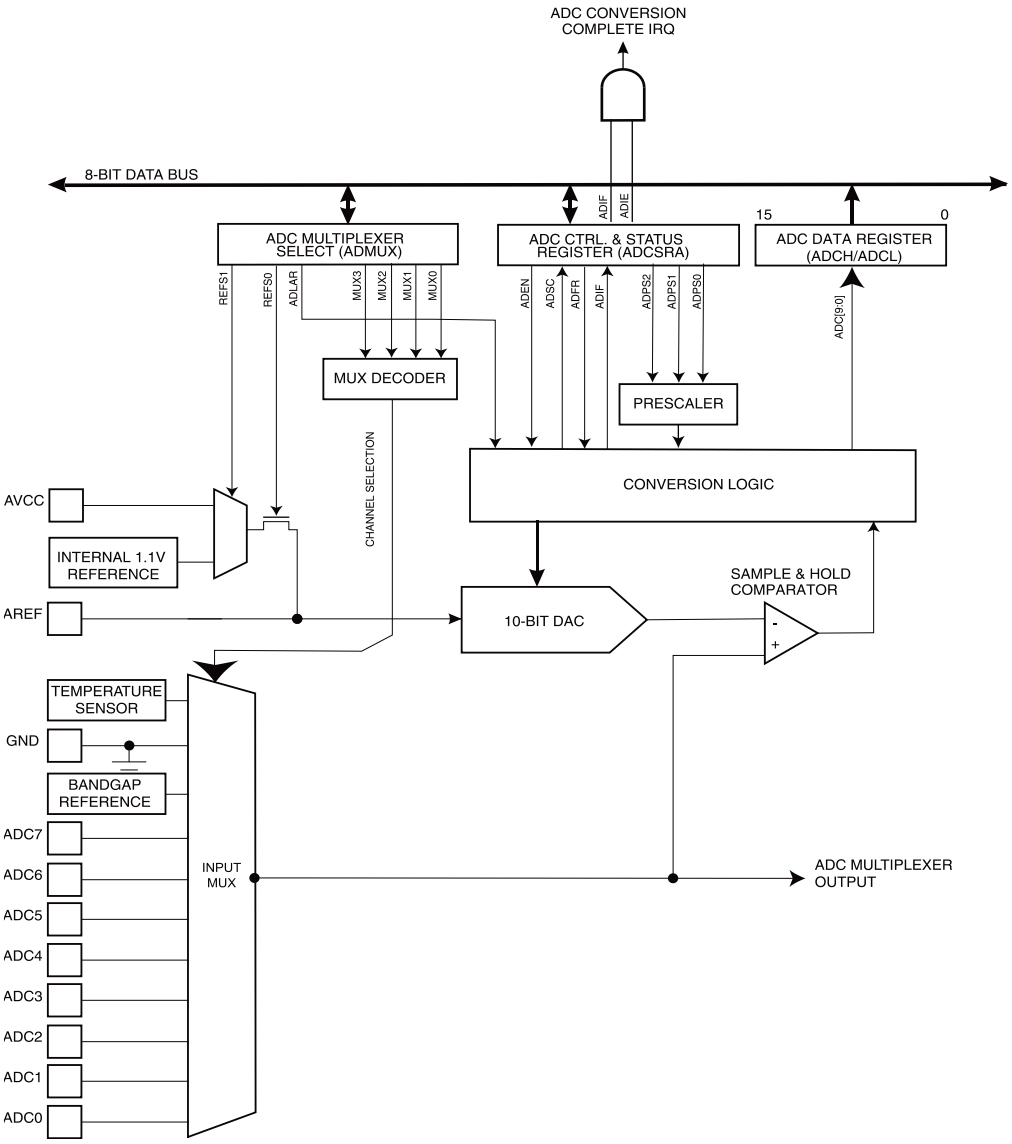
2.3.3.1.1 Conversor analógico digital

O ADC do ATmega328 presente no Arduino UNO utiliza um processo de aproximações sucessivas de 10 bits e é conectado a um multiplexador de 6 canais. Possui um circuito de amostragem e retenção, o que assegura que a tensão de entrada do ADC seja mantida num nível constante durante a conversão. Na Figura 15 é apresentado o diagrama de blocos do ADC.

O ADC converte uma tensão de entrada analógica em um valor digital de 10 bits, onde o valor mínimo representa terra (GND) e o valor máximo representa a tensão no pino de referência analógica (AREF) menos 1 LSB.

O canal de entrada analógica é selecionado através do multiplexador e o resultado da conversão é apresentado nos registradores de dados.

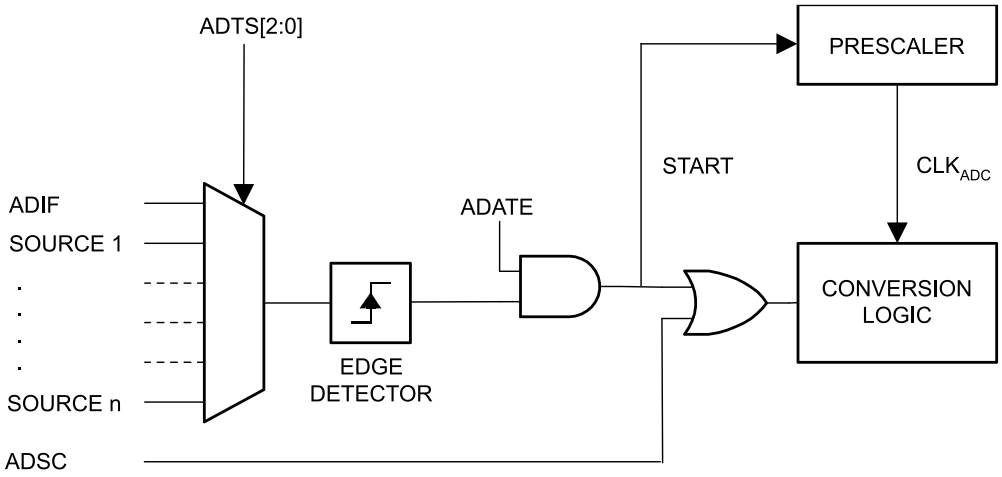
Figura 15 – Diagrama de blocos do ADC do ATmega328



Fonte: ATMEL, 2016, p. 238.

Podem ser realizadas várias amostragens consecutivas através do disparador automático, o qual tem seu diagrama lógico apresentado na Figura 16.

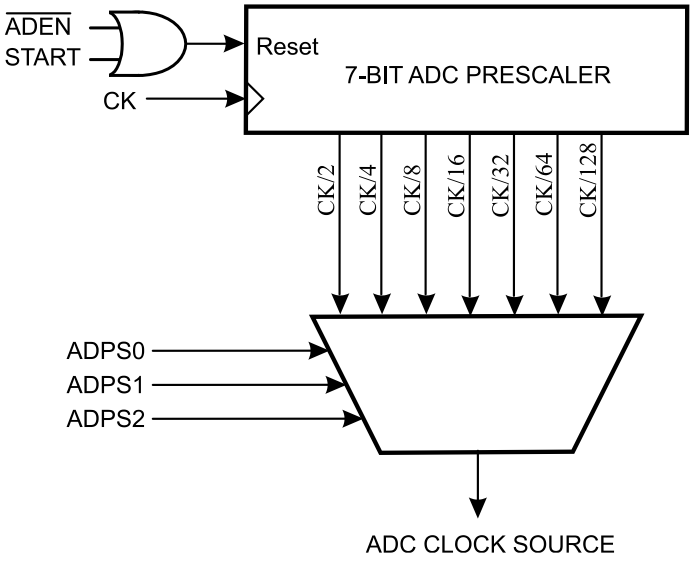
Figura 16 – Diagrama lógico do disparador automático do ADC



Fonte: ATMEL, 2016, p. 239.

Por padrão o circuito de aproximação sucessiva requer uma frequência de *clock* entre 50kHz e 200kHz para obter a resolução máxima. O módulo ADC contém um divisor de frequência conhecido como *prescaler*, que gera uma frequência de *clock* ADC aceitável a partir de qualquer frequência de CPU acima de 100kHz. Na Figura 17 é apresentado o diagrama lógico do *prescaler*.

Figura 17 – Diagrama lógico do *prescaler* do ADC

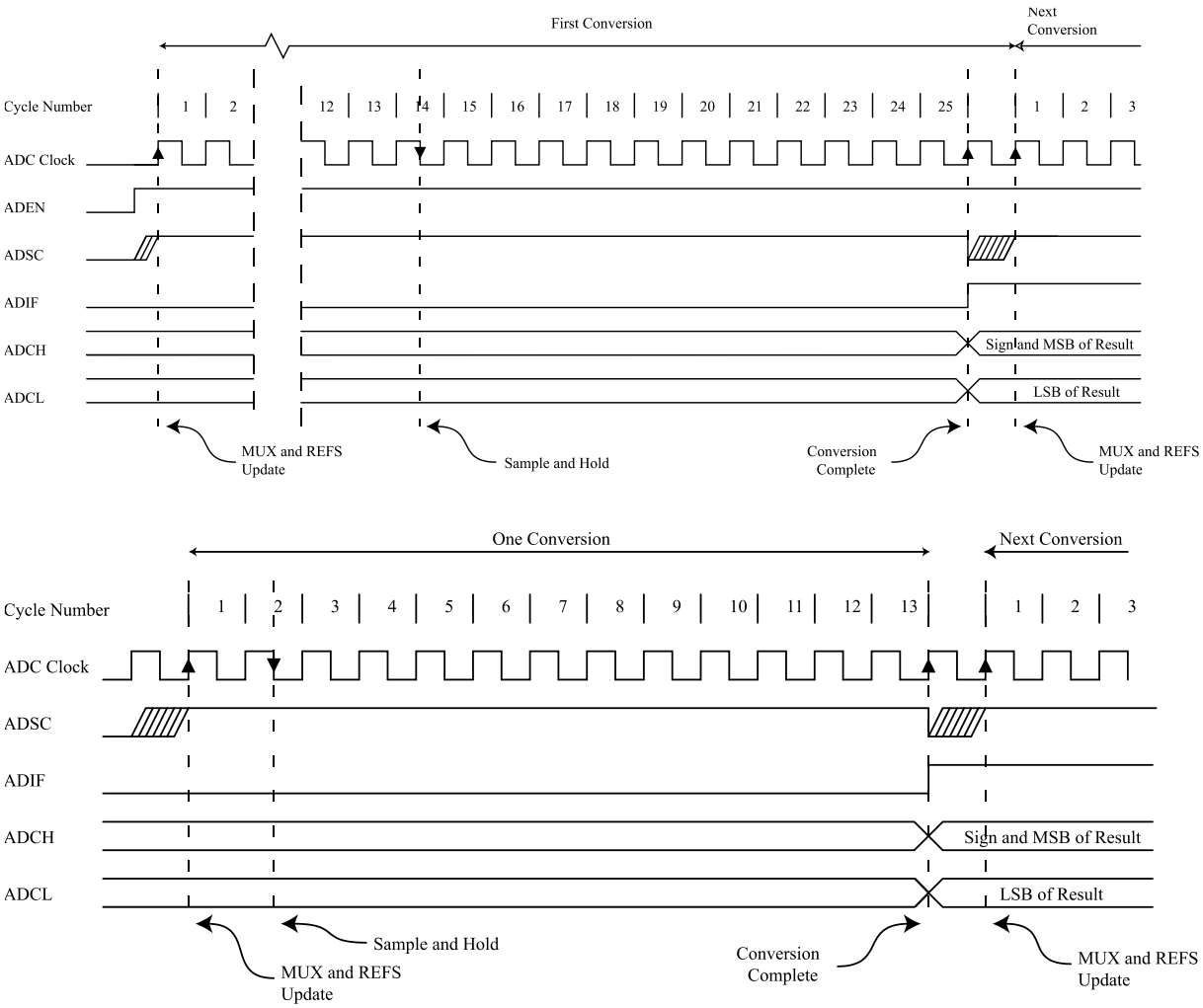


Fonte: ATMEL, 2016, p. 240.

Uma conversão normal leva 13 ciclos de *clock* do ADC. A primeira conversão depois que o ADC é ligado, isto é, quando no registrador ADCSRA o bit *ADC Enable* (ADEN) é colocado em nível lógico 1, leva 25 ciclos de *clock* do ADC devido a inicialização do circuito

analógico. Um diagrama de tempo é apresentado na Figura 18. Quando uma conversão é concluída, o resultado é gravado nos registros de dados. No modo *Free Running*, uma nova conversão será iniciada imediatamente após a conclusão da conversão conforme apresentado na Figura 19.

Figura 18 – Diagrama de tempo do ADC



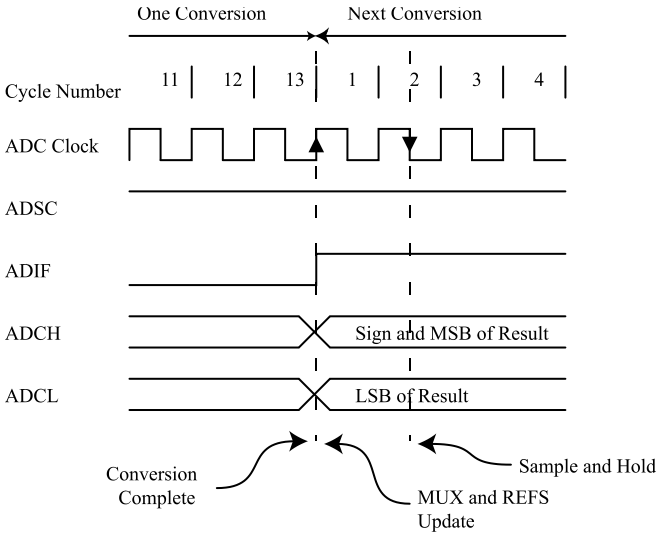
Fonte: ATMEL, 2016, p. 241.

2.3.3.1.2 Modulação por largura de pulso

Para geração de um sinal PWM através do ATmega328 é necessário a utilização de um de seus temporizadores contadores (TCs). Basicamente os TCs realizam contagem de pulsos de *clock* e desta forma podem gerar eventos periódicos, como um sinal PWM. Um estouro do TC ocorre quando este atinge o valor máximo de contagem (TOP). Os pulsos de *clock* a serem contados podem antes passar por um *prescaler* a fim de ajustar o período do sinal a ser gerado,

T, o qual pode ser obtido a partir da Equação (3), onde f_{osc} é a frequência do *clock*. A Figura 20 ilustra o funcionamento de um TC do ATmega328.

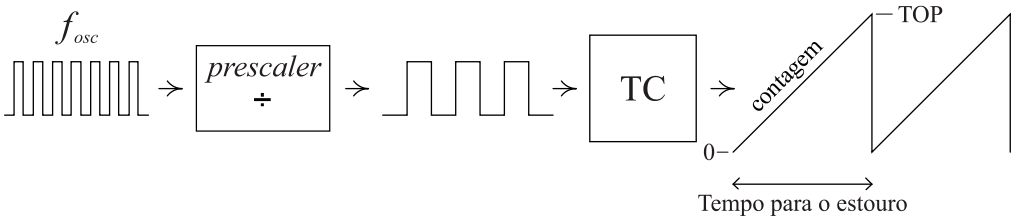
Figura 19 – Diagrama de tempo do ADC no modo *Free Running*



Fonte: ATMEL, 2016, p. 242.

$$T = \frac{(\text{TOP}+1)\times prescaler}{f_{osc}} \tag{3}$$

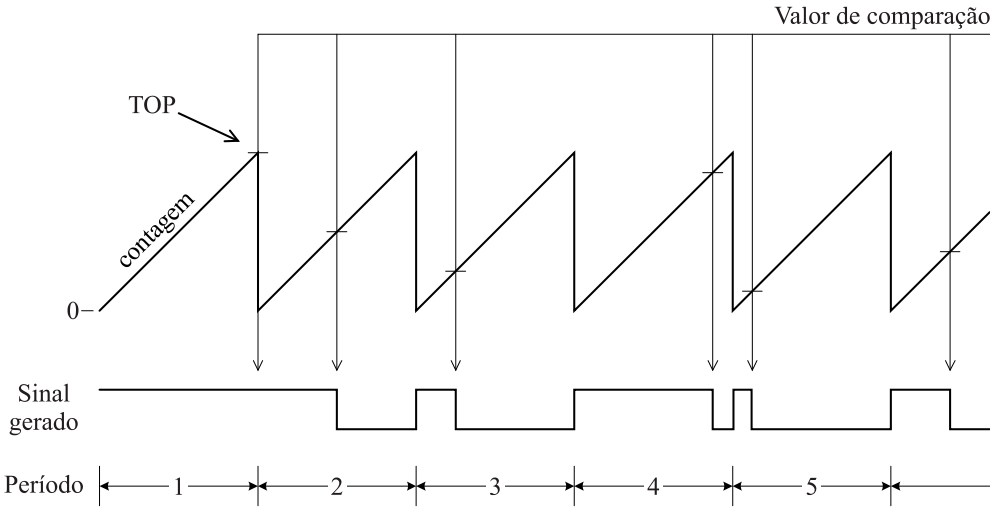
Figura 20 – Diagrama de funcionamento de um TC do ATmega328



Fonte: Adaptado de LIMA e VILLAÇA, 2012, p. 187.

Um sinal PWM pode ser obtido a partir do modo de operação *Fast PWM*, onde no modo não invertido o sinal gerado é colocado em nível lógico alto no estouro da contagem e colocado em nível lógico baixo quando a contagem alcança um determinado valor de comparação, conforme apresentado na Figura 21.

Figura 21 – Geração de um sinal PWM com um TC do ATmega328

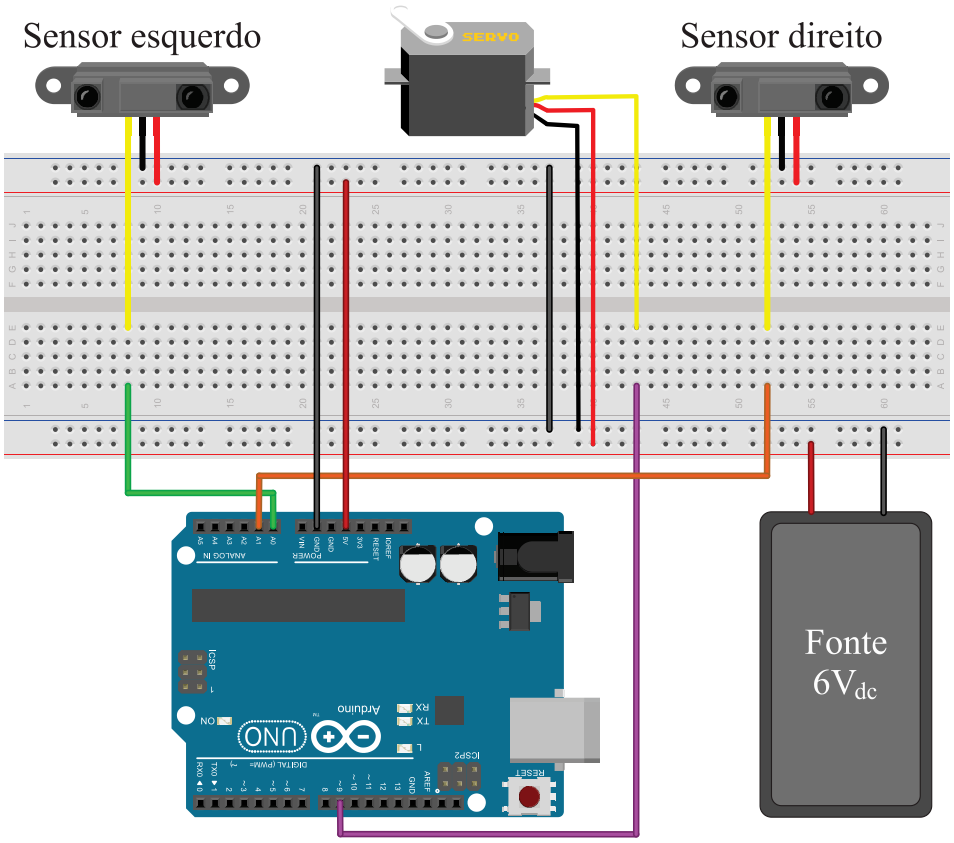


Fonte: Adaptado de LIMA e VILLAÇA, 2012, p. 214.

2.3.4 LIGAÇÕES ELÉTRICAS

As ligações elétricas entre os dispositivos da bancada são ilustradas na Figura 22.

Figura 22 – Ligações elétricas entre os dispositivos da bancada



Fonte: Elaborado pelo autor do trabalho

A interface com o usuário se dá através da porta USB, a qual também fornece a alimentação ao Arduino e indiretamente aos sensores através da linha de 5Vdc do Arduino. A fonte de alimentação de 6Vdc é responsável por fornecer a potência demandada pelo servomotor. É necessário interligar o negativo da fonte com o pino GND do Arduino de forma a fechar o circuito de controle do servomotor que é feito a partir do pino 9 do Arduino.

3 MODELAGEM MATEMÁTICA

Para controlar sistemas complexos, é importante obter modelos matemáticos que permitam analisar as relações entre suas variáveis. Uma vez que muitos sistemas são dinâmicos por natureza, as equações que os descrevem normalmente são equações diferenciais. Se estas equações puderem ser linearizadas, pode-se utilizar a transformada de Laplace de forma a simplificar o método de solução. Na prática, a complexidade dos sistemas e o desconhecimento de todos seus componentes requerem a introdução de hipóteses relativas à sua operação. É útil considerar o funcionamento físico, elaborar hipóteses simplificadoras e linearizar o sistema (DORF e BISHOP, 2001, p. 25). Seguindo este raciocínio percebe-se que, embora não seja necessário um entendimento profundo sobre mecânica clássica, para o projeto do sistema de controle, é importante, do ponto de vista da engenharia, um entendimento rudimentar das hipóteses simplificadoras.

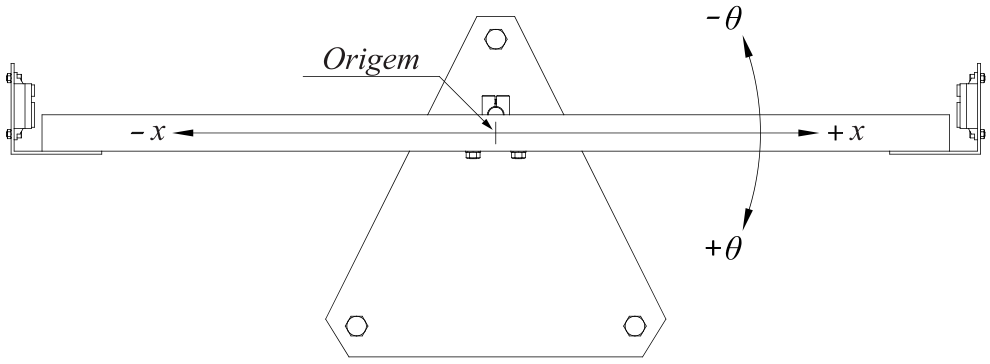
3.1 MODELAGEM DO SISTEMA BARRA-ESFERA

A modelagem do sistema barra-esfera consiste apenas na reação da esfera em função da movimentação da barra, isto é, a relação entre a posição da esfera e a posição angular da barra.

3.1.1 SISTEMA DE COORDENADAS

Por conveniência, adotou-se o sistema de coordenadas apresentado na Figura 23, onde x é a posição da esfera e θ a posição angular da barra. É importante observar que o eixo x gira em conjunto com a barra.

Figura 23 – Sistema de coordenadas



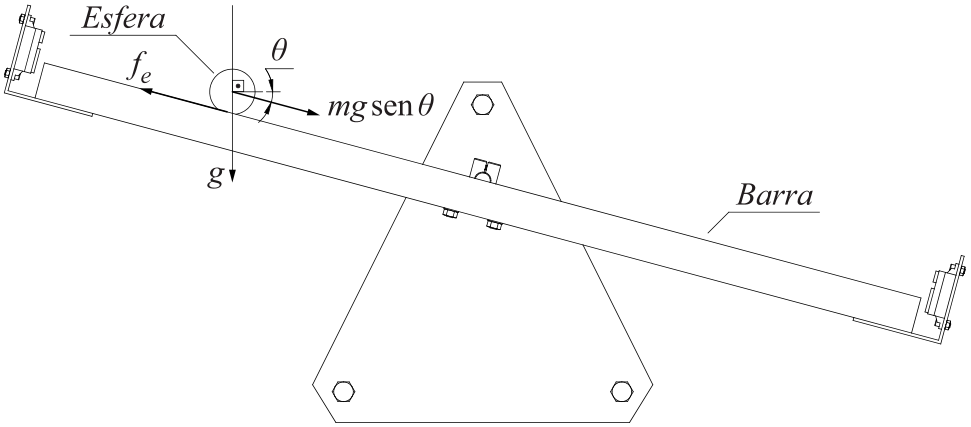
Fonte: Elaborado pelo autor do trabalho.

3.1.2 ACELERAÇÃO DA ESFERA

Como hipótese simplificadora será considerada apenas a força peso da esfera, a força normal e a força de atrito estático, desconsiderando-se assim a força centrífuga que atua quando a barra executa um movimento de rotação, principalmente por este movimento ter velocidade angular relativamente pequena quando a esfera está próxima ao ponto de referência, bem como quaisquer outras forças atuantes.

A força peso irá acelerar a esfera quando a posição angular da barra for diferente de zero, isto é, quando a barra estiver inclinada, conforme ilustrado na Figura 24, onde g é aceleração da gravidade, m é a massa da esfera e f_e é a força de atrito estático.

Figura 24 – Forças atuantes sobre a esfera



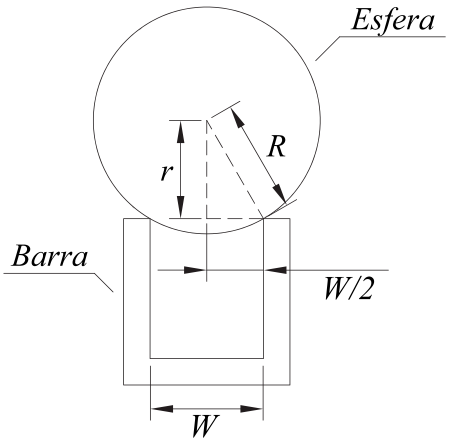
Fonte: Elaborado pelo autor do trabalho.

Pode-se considerar que o peso atua no centro de massa da esfera, que está localizado no centro deste corpo uniforme. A força normal e a de atrito estático atuam na porção do corpo que está em contato com a barra. A rotação da esfera é o resultado de um torque em sentido oposto ao movimento linear da esfera devido a força de atrito, a qual possui um braço de alavanca em relação ao eixo de rotação ligeiramente menor que o raio da esfera devido a distância entre os pontos de contato da esfera com a barra, conforme observa-se na Figura 25, onde R é o raio da esfera, r é o raio de revolução do movimento de rolamento e W é a distância entre os pontos de contato da esfera com a barra.

Aplicando a forma linear da segunda lei de Newton ao longo da barra, obtém-se a Equação (4).

$$\sum F = mg \operatorname{sen}\theta - f_e = m\ddot{x} \tag{4}$$

Figura 25 – Contato da esfera com a barra



Fonte: Elaborado pelo autor do trabalho.

Aplicando a forma angular da segunda lei de Newton em torno do eixo de rotação, o qual passa pelo centro de massa da esfera, obtém-se a Equação (5), onde *J* é o momento de inércia da esfera.

$$\sum \tau = f_e r = J\frac{\ddot{x}}{r} \tag{5}$$

A aceleração da esfera pode ser obtida ao isolar-se a força de atrito na Equação (5) e substituí-la Equação (4), conforme apresentado na Equação (6).

$$\ddot{x} = \frac{g \operatorname{sen}\theta}{1 + J/mr^2} \tag{6}$$

O momento de inércia de uma esfera sólida em relação a um eixo que passe por seu centro é dado pela relação $J = 2mR^2/5$ (HALLIDAY, RESNICK e WALKER, 1996, p. 249). O raio de revolução pode ser obtido a partir da relação trigonométrica observada na Figura 25, de forma que $r^2 = R^2 - W^2/4$. Aplicando-se estas relações a Equação (6), a aceleração da esfera passa a ser função de apenas uma variável, a posição angular da barra, conforme apresentado na Equação (7), onde $K = g\left(20R^2 - 5W^2\right)/\left(28R^2 - 5W^2\right)$ é constante, uma vez que *R*, *W* e *g* são constantes.

$$\ddot{x} = K \operatorname{sen}\theta \tag{7}$$

3.1.3 LINEARIZAÇÃO

A Equação (7) possui uma função seno, a qual é não linear. Para o controle de posição da esfera a barra irá operar em torno de um deslocamento angular de 0°, com variações relativamente pequenas para um lado ou para o outro. Desta forma será necessário linearizar a função seno em torno de 0°. A expansão da função seno em série de Taylor é apresentada na Equação (8).

$$\operatorname{sen}\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \cdots \tag{8}$$

Observa-se que para valores pequenos de θ, a função sen θ será aproximadamente igual a θ. De fato, para ângulos de até $\pm\pi/9$ (20°), limite de inclinação da barra quando operando sobre uma mesa, a diferença entre sen θ e θ será no máximo de 2%. Desta forma a Equação (7) para a aceleração da esfera pode ser reescrita de forma linearizada, conforme apresentado na Equação (9).

$$\ddot{x} = K\theta \tag{9}$$

3.1.4 FUNÇÃO DE TRANSFERÊNCIA DO SISTEMA BALL BEAM

A função de transferência de um sistema linear é definida como a relação entre a transformada de Laplace da variável de saída e a transformada de Laplace da variável de entrada, com todas as condições iniciais nulas (DORF e BISHOP, 2001, p. 37).

Por simplicidade, até aqui foi omitido das equações a variável de tempo. São funções do tempo a inclinação da barra e a aceleração da esfera. Desta forma, fazendo-se as devidas correções a Equação (9), aplicando-se a transformada de Laplace em ambos os membros e considerando condições iniciais nulas, tem-se:

$$\mathcal{L}\left(\ddot{x}(t)\right) = \mathcal{L}\left(K\theta(t)\right)$$

$$s^2 X(s) = K\Theta(s)$$

$$G_p(s) = \frac{X(s)}{\Theta(s)} = \frac{K}{s^2} \tag{10}$$

A Equação (10) apresenta a função de transferência do sistema *ball beam*, $G_p(s)$. As dimensões da bancada foram determinados com o auxílio de um paquímetro, sendo $W = 11\text{mm}$

e $R = 15\text{mm}$, e a aceleração da gravidade foi considerada como $9{,}8\,\text{m/s}^2$. Aplicando os valores das constante obtém-se a expressão final da função de transferência, a qual é apresentada na Equação (11).

$$G_p(s) = \frac{6{,}7}{s^2} \tag{11}$$

3.2 MODELAGEM DO SERVOMOTOR

A modelagem do servomotor descreve como a posição angular da barra responde tanto ao sinal de controle PWM, quanto a folga nas engrenagens e ao torque externo produzido pelo sistema barra-esfera.

3.2.1 SINAL PWM

O curso do servomotor é de 180°, ver 2.3.1.1, sendo que para uma posição angular de 0° é necessário aplicar um sinal PWM de período de 20ms com nível lógico alto de $600\mu\text{s}$. A cada incremento de $10\mu\text{s}$ ao nível lógico alto do sinal PWM, a posição angular do servomotor é incrementada em $1°$, conforme apresentado na Equação (1).

Para correlacionar a posição angular do sinal do servomotor ao sistema de coordenadas adotado em 3.1.1, é necessário aplicar um *offset*, isto é, um valor que represente a largura do pulso em nível alto quando a barra está na posição horizontal. Este valor varia em função da posição angular em que o eixo de saída do servomotor é fixado a barra e pode ser obtido experimentalmente. O eixo deve ser fixado a barra de tal forma que haja curso suficiente para variar a posição da barra em ambos os sentidos, horário e anti-horário, dentro de limites pré-estabelecidos, sem que se alcance os limites de inclinação de 0° e de 180° do servomotor.

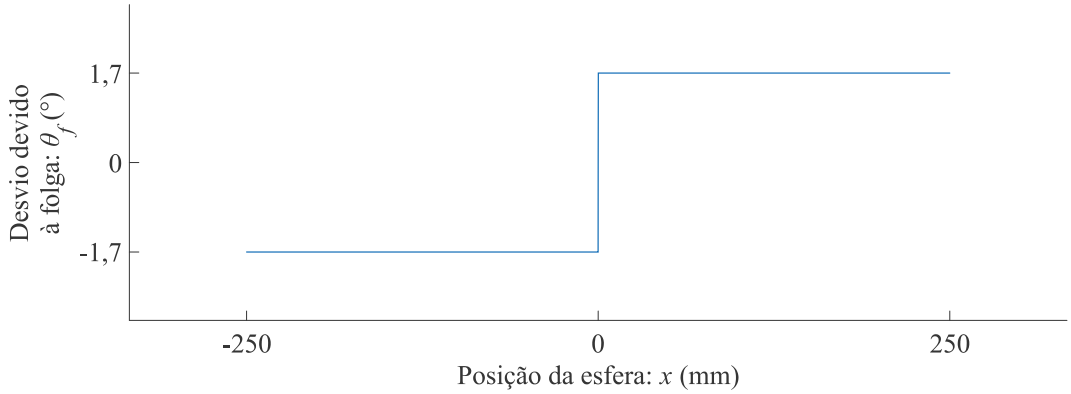
A Equação (12) descreve a posição angular equivalente ao sinal aplicado ao servomotor, θ_s, em função da largura ou período do pulso em nível alto, t_{pwm}, e do valor do *offset*, t_0.

$$\theta_s = \frac{t_{pwm} - t_0}{10\,\mu\text{s}/°} \tag{12}$$

3.2.2 FOLGA NAS ENGRENAGENS

A folga nas engrenagens da caixa de redução do servomotor, ver 2.3.1.2.3, ocasiona um desvio de posição angular da barra $\theta_f = \pm 1,7°$ ou $\theta_f = \pm 1,7°$, quando a esfera se encontra fora do centro, conforme representado na Figura 26, sem que o circuito de controle do servomotor o perceba.

Figura 26 – Desvio angular devido à folga das engrenagens do servomotor



Fonte: Elaborado pelo autor do trabalho.

A Equação (13) descreve seu comportamento.

$$\theta_f = \begin{cases} -1,7° \text{ para } x < 0 \\ 0° \quad \text{para } x = 0 \\ +1,7° \text{ para } x > 0 \end{cases} \tag{13}$$

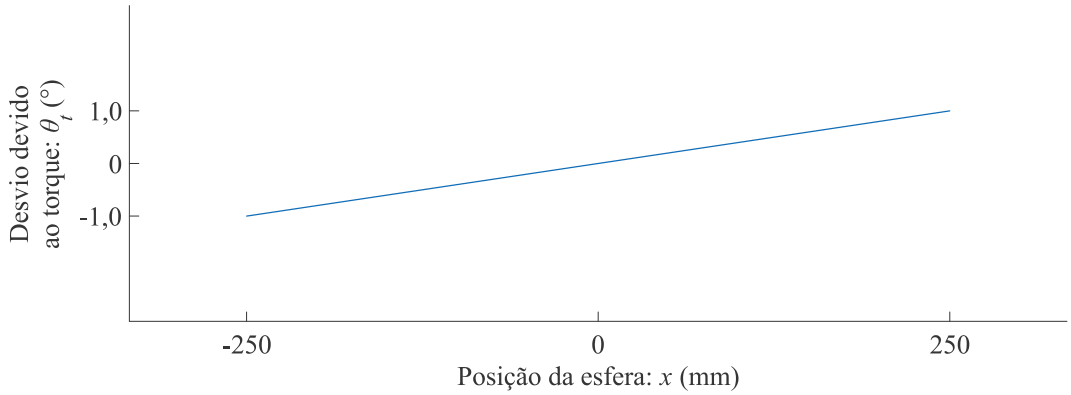
3.2.3 TORQUE EXTERNO APLICADO AO EIXO DE SAÍDA

Devido a limitação de resposta do servomotor à um torque externo aplicado ao eixo de saída, ver 2.3.1.2.4, observa-se um desvio na posição angular da barra quando a esfera se encontra fora da posição central. Diferentemente do desvio devido à folga, este é percebido pelo circuito de controle do servomotor, podendo ser considerado como um erro de regime permanente de seu sistema de controle.

O desvio observado, θ_t, é de $4°/\text{m}$, ou $1°/250\text{mm}$, conforme ilustrado na Figura 27. A Equação (14) descreve seu comportamento.

$$\theta_t = x \cdot 4°/\text{m} \tag{14}$$

Figura 27 – Desvio angular devido ao torque produzido pelo peso da esfera



Fonte: Elaborado pelo autor do trabalho.

3.2.4 Posição Angular da Barra

A posição angular da barra pode ser obtida a partir da Equação (15).

$$\theta = \theta_s + \theta_f + \theta_t \tag{15}$$

4 CONTROLE DIGITAL

A principal vantagem do controlador digital sobre o controlador contínuo é a flexibilidade de implementação. Como os controladores contínuos são implementados por meio de componentes eletrônicos, a complexidade desses controladores depende da complexidade e da precisão dos componentes, além do que uma alteração na lei de controle implica em mudanças de componentes e interligações. Já os controladores digitais são implementados na forma de programas, sendo portanto igualmente fácil implementar controladores simples ou complexos, além do que mudanças na lei de controle também são muito mais simples de serem realizadas (CASTRUCCI, BITTAR e SALES, 2011, p. 410).

Basicamente, o projeto de controladores de tempo discreto pode ser realizado de duas formas (CASTRUCCI, BITTAR e SALES, 2011, p. 410):

a) através de aproximações discretas do projeto realizado no domínio contínuo por meio da transformada de Laplace;
b) diretamente no domínio discreto por meio da transformada *Z*.

4.1 APROXIMAÇÕES DE TEMPO DISCRETO

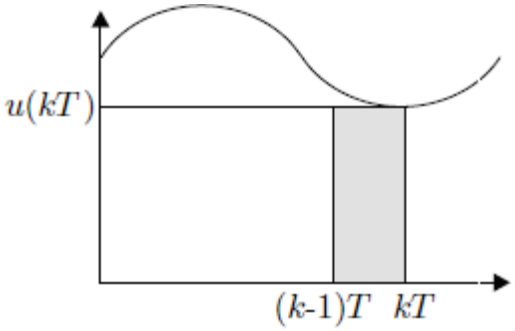
Pode-se obter funções de transferência em tempo discreto no domínio *z*, aproximadas a partir de funções de transferência em tempo contínuo no domínio *s*, através de diversos métodos diferentes, entre eles:

a) Retangular para trás;
b) Método de Tustin;
c) Mapeamento polo-zero.

4.1.1 RETANGULAR PARA TRÁS

Neste método o cálculo da integral é aproximado pela área de um retângulo, cujo comprimento de um dos lados é igual ao período *T* e o comprimento do outro é igual ao valor da função no instante *kT*, conforme mostrado na Figura 28.

Figura 28 – Método numérico de integração retangular para trás

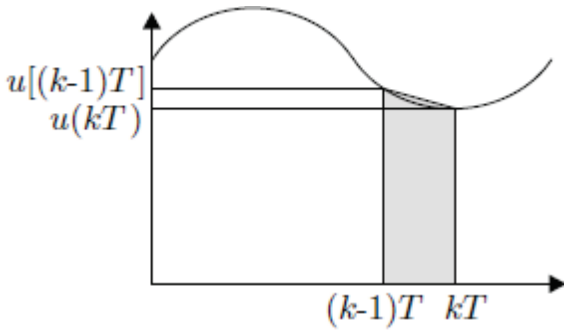


Fonte: CASTRUCCI, BITTAR e SALES, 2011, p. 411.

4.1.2 MÉTODO DE TUSTIN

No método de Tustin o cálculo da integral é aproximado pela área de um trapézio, cuja altura é igual ao período T e cujas bases correspondem aos valores da função nos instantes kT e $(k\text{-}1)T$, conforme mostrado na figura

Figura 29 - Método numérico de integração de trapézio



Fonte: CASTRUCCI, BITTAR e SALES, 2011, p. 411.

4.1.3 MAPEAMENTO POLO-ZERO

O método consiste em mapear os polos e zeros da função de transferência $G(s)$ no plano z, através da relação $z = e^{sT}$. Inicialmente deve-se fatorar $G(s)$ na forma de polos e zeros, sendo n o número de polos e m o número de zeros.

Um polo de $G(s)$ em $s = p_n$ é mapeado no plano z em $z = e^{p_n T}$, e um zero finito de $G(s)$ em $s = z_m$ é mapeado no plano z em $z = e^{z_m T}$.

Quando o número de zeros de $G(s)$ é menor que o número de polos $(m < n)$, $G(s)$ possui zeros no infinito que são mapeados em $z = -1$.

Por último, deve-se ajustar o ganho de $G(z)$, onde normalmente o ganho é ajustado para que em baixas frequências o ganho do sistema contínuo seja igual ao do sistema discreto, isto é, $G(s)\big|_{s=0} = G(z)\big|_{z=1}$.

4.2 PROJETO DO CONTROLADOR

Projetar é o processo de conceber ou inventar formas, partes e detalhes de um sistema para alcançar um fim específico (DORF e BISHOP, 2001, p. 14).

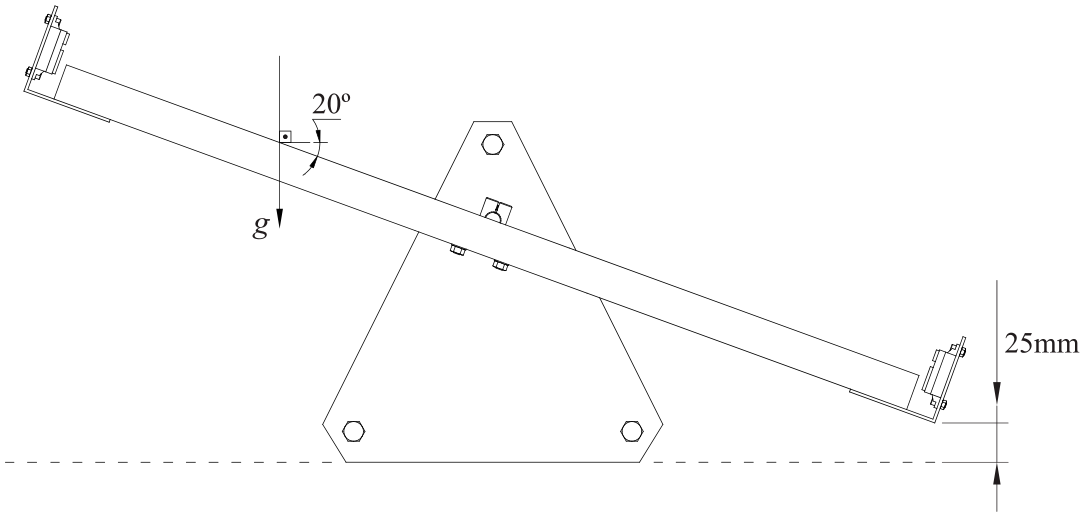
As especificações de desempenho descreverão como o sistema em malha fechada deverá se comportar e incluirão boa regulação contra perturbações, resposta desejada aos comandos, sinais reais do atuador, baixas sensibilidades e robustez (DORF e BISHOP, 2001, p. 15).

4.2.1 ESPECIFICAÇÕES DE DESEMPENHO

As especificações de desempenho de projetos de controle costumam ser realizadas em termos dos parâmetros da resposta temporal, como por exemplo sobressinal (M_p), tempo de subida (t_r), tempo de pico (t_p) ou tempo de acomodação (t_s) (CASTRUCCI, BITTAR e SALES, 2011, p. 419). Estas especificações devem levar em consideração as capacidades e limites físicos do sistema.

A bancada *ball beam* desenvolvida possui um limite de inclinação da barra de ±20°, conforme ilustrado na Figura 30, de forma que esta não toque a superfície de apoio. A barra possui ainda um comprimento de 600mm com o curso da esfera limitado aos 450mm centrais, pois existe um limite de distância mínimo que os sensores podem detectar, ver 2.3.2.1. Desta forma a especificação de desempenho de tempo de resposta deve estar dentro da capacidade de resposta do sistema, atendendo critérios que sejam fisicamente realizáveis.

Figura 30 – Limite de inclinação da barra



Fonte: Elaborado pelo autor do trabalho.

A partir destas considerações definiu-se o tempo de acomodação, para um critério de 2%, em $t_s^{2\%} < 3\text{s}$ e o sobressinal em $M_p < 1\%$.

4.2.2 ARQUITETURA DE CONTROLE

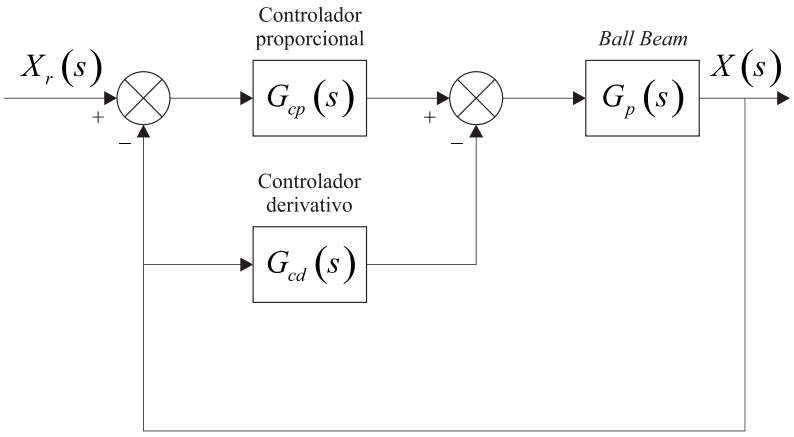
Ao longo das últimas décadas teorias matemáticas dedicadas ao controle ótimo e ao controle robusto têm produzido importantes resultados, indispensáveis nas aplicações aeronáuticas, aeroespaciais e similares. No entanto, algumas arquiteturas especiais utilizando algoritmos clássicos, chamados de PID ou P+I+D, têm demonstrado notável eficácia e praticidade no controle dos processos industriais. Esses controladores PID, ainda mais quando inseridos nos computadores industriais e nos controladores lógicos programáveis, mantêm-se como um dos principais equipamentos de controle. O nome PID deriva do fato de que sua função de transferência contém a soma das ações Proporcional, Integradora e Derivadora (CASTRUCCI, BITTAR e SALES, 2011, p. 229).

A arquitetura de controle adotada para o *ball beam* é um controlador proporcional derivativo (PD), haja vista que a função de transferência da planta possui um duplo integrador, implementado sem a derivada do erro, conforme ilustrado na Figura 31.

Nos problemas de regulação da saída da planta, em que o sinal de referência é mantido em níveis constantes, a derivada do erro pode produzir picos nos amplificadores e atuadores quando se muda o nível do sinal de referência. Nesta situação, ao se aplicar um sinal do tipo

degrau na referência ocorre também um degrau no sinal de erro, e o termo derivativo, por sua vez, produz picos na saída do controlador (CASTRUCCI, BITTAR e SALES, 2011, p. 241).

Figura 31 – Arquitetura de controle



Fonte: Elaborado pelo autor do trabalho.

As funções de transferência dos controladores proporcional e derivativo são apresentadas nas Equações (16) e (17), respectivamente, onde K_P é o ganho proporcional e K_D o ganho derivativo.

$$G_{cp}(s) = K_P \tag{16}$$

$$G_{cd}(s) = K_D s \tag{17}$$

4.2.3 Função de Transferência do Sistema em Malha Fechada

A função de transferência de malha fechada, G_{mf}, pode ser determinada a partir de análise do diagrama de blocos ilustrado na Figura 31, onde:

$$X(s) = X_r(s)G_{cp}(s)G_p(s) - X(s)\left(G_{cp}(s) + G_{cd}(s)\right)G_p(s)$$

$$G_{mf}(s) = \frac{X(s)}{X_r(s)} = \frac{G_{cp}(s)G_p(s)}{1 + \left(G_{cp}(s) + G_{cd}(s)\right)G_p(s)} = \frac{K_P\, 6,7/s^2}{1 + \left(K_P + K_D s\right)6,7/s^2}$$

$$G_{mf}(s) = \frac{6,7K_P}{s^2 + 6,7K_D s + 6,7K_P} \tag{18}$$

4.2.4 DETERMINAÇÃO DOS GANHOS DO CONTROLADOR

Observa-se que o sistema em malha fechada é de segunda ordem. A função de transferência de um sistema de segunda ordem normalizado com ganho unitário é apresentada na Equação (19), onde ω_n é a frequência natural não amortecida e ξ é o coeficiente de amortecimento.

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{19}$$

O comportamento dinâmico dos sistemas de segunda ordem é determinado inteiramente pelos parâmetros ω_n e ξ (CASTRUCCI, BITTAR e SALES, 2011, p. 70).

A partir das especificações de desempenho os parâmetros do sistema foram determinados em $\xi = 0,87$ e $\omega_n = 1,74\,\mathrm{rad/s}$, com o auxílio das Equações (20) e (21), onde m_s é o critério do tempo de acomodação.

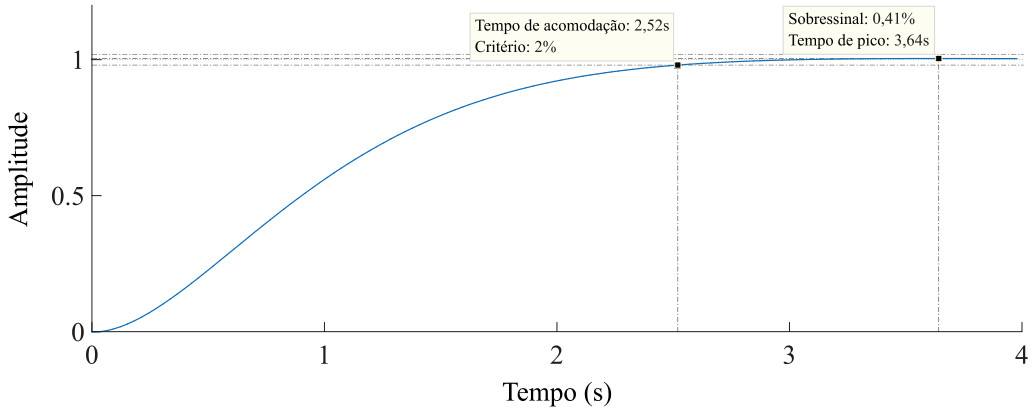
$$\xi > \sqrt{\frac{\ln\left(M_p\right)^2}{\pi^2 + \ln\left(M_p\right)^2}} \tag{20}$$

$$\omega_n > -\frac{\ln\left(m_s\right)}{\xi t_s^{m_s\,\%}} \tag{21}$$

Comparando as Equações (18) e (19) pode-se determinar os ganhos do controlador, os quais foram definidos em $K_P = 0,45$ e $K_D = 0,45$. Desta forma a função de transferência do sistema em malha fechada é apresentada na Equação (22) e a resposta ao degrau unitário na Figura 32.

$$G_{mf}(s) = \frac{3}{s^2 + 3s + 3} \tag{22}$$

Figura 32 – Resposta ao degrau unitário em malha fechada



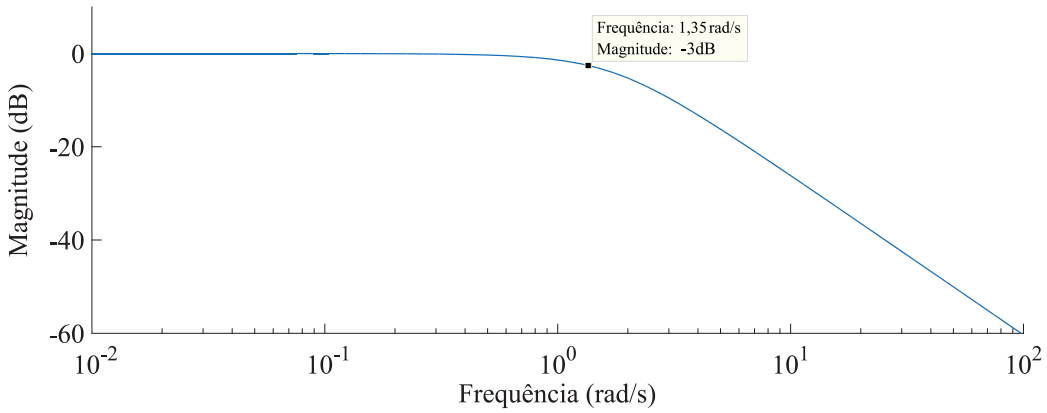
Fonte: Elaborado pelo autor do trabalho.

4.2.5 PERÍODO DE AMOSTRAGEM

Segundo o teorema de Nyquist, para se reconstruir um sinal contínuo da sequência amostrada, a frequência de amostragem deve obedecer a condição $f_s > 2f_{máx}$, onde f_s é a frequência de amostragem e $f_{máx}$ é a maior frequência presente no sinal (LANDAU e ZITO, 2006, p. 28). Este é um limite teórico, na prática deve-se escolher uma frequência de amostragem mais alta (LANDAU e ZITO, 2006, p. 29).

A frequência de amostragem para sistemas de controle digital é escolhida de acordo com a largura de banda desejada do sistema de malha fechada, independentemente de como as performances desejadas são especificadas (LANDAU e ZITO, 2006, p. 31).

Figura 33 – Resposta em frequência do sistema em malha fechada



Fonte: Elaborado pelo autor do trabalho.

A resposta em frequência do sistema em malha fechada é apresentada na Figura 33. A largura de banda do processo em malha fechada é $f_b^{mf} = 0,21\text{Hz}$ ou $1,35\,\text{rad/s}$.

A regra usada para escolha da frequência de amostragem é $f_s = (6 \text{ à } 25) f_b^{mf}$ (LANDAU e ZITO, 2006, p. 31). Em sentido restrito, os sensores de distância utilizados não são contínuos, mas sim discretos, com taxas de atualização de 39,4ms para o sensor esquerdo e de 37,5ms para o sensor direito, ver 2.3.2.2.1. Por sua vez o servomotor opera sob um sinal de controle com período de 20ms, ou 50Hz.

Considerando-se além da largura de banda também a taxa de atualização dos sensores e o período do sinal de controle, o período de amostragem foi definido em $T = 20\text{ms}$, de forma que todas as posições capturadas pelos sensores serão amostradas.

4.2.6 DISCRETIZAÇÃO DO CONTROLADOR

A discretização do controlador proporcional é direta, contudo diversas aproximações podem ser empregadas para discretizar o controlador derivativo. Uma boa aproximação para um controlador derivativo é obtida com o método retangular para trás (CASTRUCCI, BITTAR e SALES, 2011, p. 437), conforme apresentado na Equação (23).

$$G_{cd}^{*}(z) = \frac{K_D}{T}\left(1 - z^{-1}\right) \tag{23}$$

Por ser imprópria e pelo fato de possuir um ganho que cresce com o aumento da frequência, amplificando ruídos de alta frequência, a função de transferência da Equação (17) não é implementada na prática. Então esta função de transferência é substituída pela Equação (24), sendo que N é um parâmetro tipicamente adotado entre 0,05 e 0,17 (CASTRUCCI, BITTAR e SALES, 2011, p. 437). O método utilizado para discretizar esta função de transferência foi o mapeamento polo-zero, conforme apresentado na Equação (25).

$$G_{cd}(s) = \frac{K_D s}{N K_D s + 1} \tag{24}$$

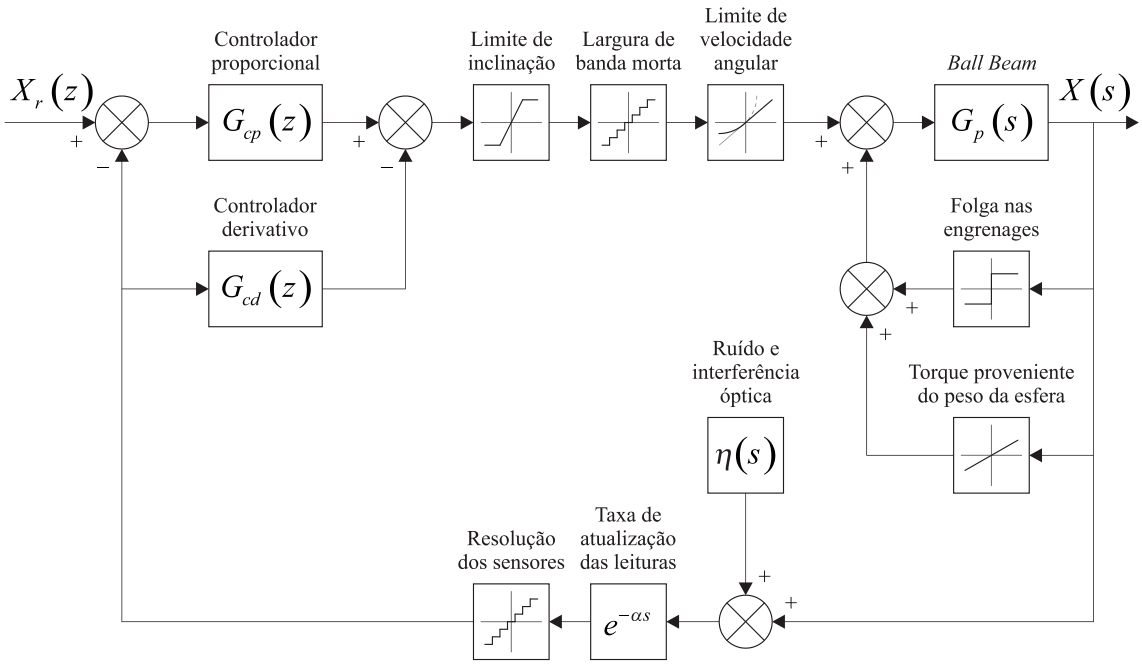
$$G_{cd}(z) = \frac{z - 1}{N\left(z - e^{-T/NK_D}\right)} \tag{25}$$

4.3 LIMITAÇÕES DE PROJETO

Até aqui na modelagem do sistema e no projeto do controlador não foram considerados diversos fatores limitadores que degradam o desempenho do sistema. Uma representação do sistema que leva em consideração as diversas limitações observadas tanto na forma construtiva da bancada, no servomotor como também nos sensores de distância utilizados, é apresentada na Figura 34, onde:

a) o bloco limite de inclinação representa o limite de ±20° permitido para inclinação da barra devido as características construtivas da bancada, ver 4.2.1;
b) o bloco largura de banda morta representa a discretização de posição angular do servomotor, ver 2.3.1.2.1;
c) o bloco limite de velocidade angular representa o limite da taxa de variação da posição angular do servomotor, ver 2.3.1.2.2;
d) o bloco folga nas engrenagens representa desvio de posição angular pela ação do peso da esfera sobre a folga observada na barra, proveniente da caixa de redução do servomotor, ver 2.3.1.2.3;
e) o bloco torque devido ao peso da esfera representa o desvio de posição angular do servomotor sob ação de um torque externo, ver 2.3.1.2.4;
f) o bloco ruído e interferência óptica representam respectivamente os efeitos do ruído do sinal de saída dos sensores, ver 2.3.2.2.1, e a interferência óptica entre os sensores, ver 2.3.2.2.2;
g) o bloco taxa de atualização das leituras representa o atraso do sinal proveniente dos circuitos de processamento de sinal dos sensores, ver 2.3.2.2.3;
h) o bloco resolução dos sensores representa a discretização do sinal de saída dos sensores, ver 2.3.2.2.4.

Figura 34 – Diagrama de blocos considerando as limitações do sistema



Fonte: Elaborado pelo autor do trabalho.

Percebe-se que, para além da simples discretização do controlador, muitos são os fatores que contribuem negativamente na controlabilidade do sistema. Para que se atinja uma performance aceitável é necessário mitigar estes efeitos lançando mão de técnicas mais sofisticadas.

O possível atraso no tempo de resposta do servomotor, ver 2.3.1.2, não foi incluído no diagrama. Contudo para efeitos práticos este poderá ser adicionado ao tempo de atraso do sinal dos sensores sem prejuízo as técnicas que poderão ser adotadas.

4.3.1 PROCESSAMENTO DE SINAL

Para que o controlador atinja as especificações de desempenho, é necessário que o sinal de realimentação represente o mais fielmente possível a saída do processo. Considerando as diversas limitações apresentadas, faz-se necessário um processamento no sinal a fim de reduzir as incertezas introduzidas pelos ruídos e tempo de atraso.

4.3.1.1 Filtro de Moda

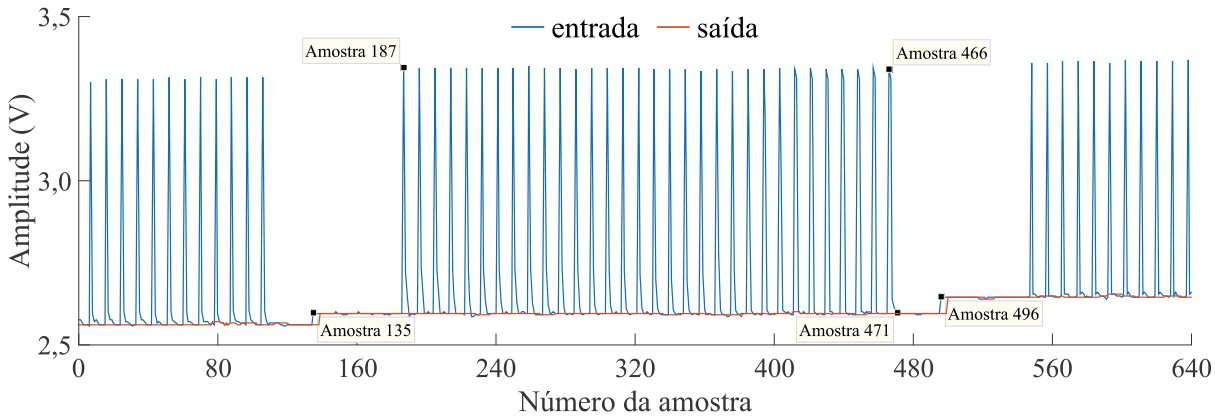
O ruído presente no sinal dos sensores tem uma característica própria que requer uma filtragem a fim de se eliminar os picos observados na Figura 12. Estes picos não fazem parte

da informação dos sensores, tornando-se um componente que interfere diretamente no correto funcionamento do controlador.

Os picos se repetem de maneira uniforme a cada 9 amostras, de forma que uma janela de amostragem com 36 amostras de cada sensor é mais que suficiente para se extrair a correta informação do sinal.

Para eliminar os picos, aplicou-se um filtro de moda a janela de amostragem, de forma que o valor a ser considerado será o valor de amostra mais frequente. O resultado da filtragem, sobre uma janela de apenas 9 amostras, é apresentado na Figura 35.

Figura 35 – Filtro de moda aplicado à uma janela de nove amostras



Fonte: Elaborado pelo autor do trabalho.

4.3.1.2 Filtro de Média Móvel

Para atenuar os efeitos causados pela diferença entre a taxa de atualização dos sensores e a frequência de amostragem, aplicou-se um filtro de média móvel simples de terceira ordem à leitura de posição proveniente dos transdutores, o qual consiste de se tomar como saída do filtro na iteração *k*, a média aritmética das últimas 3 entradas, conforme Equação (26), onde os subíndices indicam o número da iteração a qual as variáveis pertencem, *y* é a saída do filtro de média móvel e *xt* é a leitura de posição.

$$y_k = \frac{xt_k + xt_{k-1} + xt_{k-2}}{3} \tag{26}$$

4.3.1.3 Estimador Recursivo

Técnicas recursivas são muito úteis quando deseja-se estimar parâmetros de um determinado modelo à medida que os dados do processo são disponibilizados. Algoritmos desenvolvidos para esse fim são denominados de algoritmos recursivos (AGUIRRE, 2015, p. 331 e 332).

Nas situações em que se conhece a lei de movimento de um determinado objeto, é possível estimar a posição deste num instante t_2 a partir apenas de sua posição estimada anteriormente em t_1 e da lei que descreve esse movimento. A lei de movimento fornece a melhor predição de posição em t_2 sem levar em conta a medição em t_2 (AGUIRRE, 2015, p. 363 e 364).

Para contornar o ruído no sinal dos sensores proveniente da interferência óptica, optou-se por utilizar um estimador recursivo a partir da função de transferência discretizada do sistema *ball beam*, de forma a se obter uma estimativa da posição da esfera, a qual combinada a leitura proveniente dos sensores produz um sinal para posição da esfera menos ruidoso. A função de transferência contínua do sistema *ball beam* apresentada na Equação (10) é composta por um duplo integrador e, desta forma, a melhor aproximação de discretização é obtida pelo método de Tustin (CASTRUCCI, BITTAR e SALES, 2011, p. 437). Aplicando o método de Tustin a Equação (10) e considerando-se o tempo de atraso na resposta do sistema, a posição da esfera pode ser estimada a partir da Equação (27), onde os subíndices indicam o número da iteração a qual as variáveis pertencem, $\hat{x}$ é a posição estimada da esfera, u é o sinal de controle do controlador PD após o bloco limitador de inclinação, x é a posição da esfera a partir da combinação entre a posição estimada e a posição proveniente da leitura dos sensores e d representa o tempo de atraso discreto na resposta do sistema em número de amostras.

$$\hat{x}_k = \frac{K \cdot T^2}{4}\left(u_{k-d} + 2u_{k-d-1} + u_{k-d-2}\right) + 2x_{k-1} - x_{k-2} \tag{27}$$

A combinação entre a posição estimada e a leitura dos sensores é obtida a partir da Equação (28), onde k_e é o ganho do filtro estimador, de forma que $0 < k_e \leq 1$, e y é a posição da esfera proveniente da leitura dos sensores após aplicação do filtro de moda e de média móvel. Quanto menor for o valor de k_e, maior será o peso da estimativa em relação a leitura dos sensores. Para $k_e = 1$ a posição da esfera será determinada unicamente pela leitura dos sensores.

$$x_k = \hat{x}_k + k_e\left(y_k - \hat{x}_k\right) \tag{28}$$

4.3.1.4 Preditor Baseado no Estimador Recursivo

O estimador recursivo também pode ser usado como um preditor, assim como num filtro de Kalman, o que lhe confere uma estrutura equivalente ao preditor de Smith filtrado para sistemas lineares (LIMA, LIMA e NORMEY-RICO, 2018, p. 24). Pode-se então utilizar a posição estimada da esfera como realimentação dos controladores, de forma a se obter uma melhor rejeição a distúrbios e maior robustez.

A predição da posição da esfera no instante *k* para o estado futuro *k*+*d*, indicada por $x_{k|k+d}$, pode ser obtida recursivamente através da Equação (29), iniciando com $n=1$ e incrementando n até $n=d$. As variáveis $x_{k|k}$ e $x_{k|k-1}$ devem ser inicializadas com os valores de x_k e x_{k-1}, respectivamente.

$$x_{k|k+n} = \frac{K\cdot T^2}{4}\left(u_{k+n-d} + 2u_{k+n-d-1} + u_{k+n-d-2}\right) + 2x_{k|k+n-1} - x_{k|k+n-2} \tag{29}$$

4.3.2 COMPENSADORES

Para que a inclinação da barra seja efetivamente a inclinação determinada pelo controlador PD é necessário compensar, paralelamente ao controlador, os desvios na inclinação produzidos tanto pela folga nas engrenagens quanto pelo torque devido ao peso da esfera. Estes desvios podem ser facilmente mitigados no domínio do tempo através de compensadores que atuam em função da posição da esfera sobre a barra.

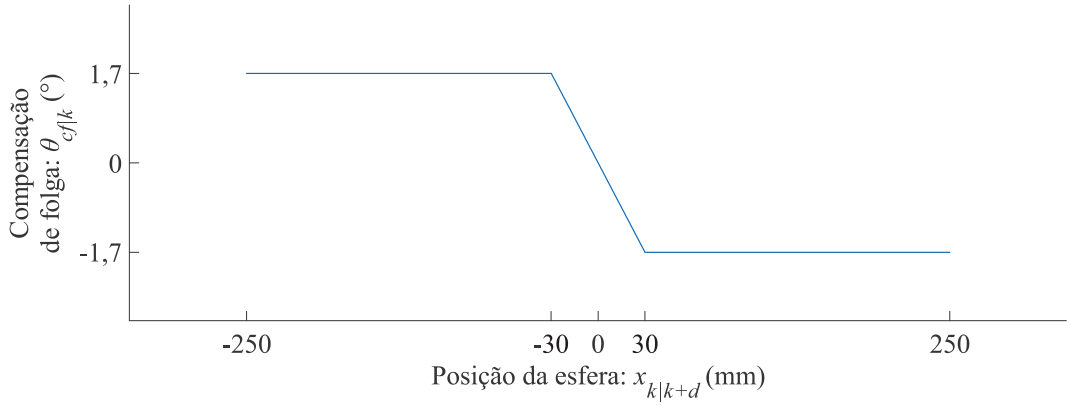
4.3.2.1 Compensador da Folga nas Engrenagens

A compensação deste desvio pode ser feita acrescendo-se $\pm 1,7°$ à inclinação de referência passada ao servomotor

Contudo quando a esfera se encontra próxima ao centro da barra pode haver uma instabilidade devido tanto a incerteza da posição da esfera, que no centro da barra é maior devido à queda de precisão dos sensores em função da distância além de um maior efeito da interferência óptica, quanto a mudança abrupta de 3,4° na transição de um estado para o outro.

Desta forma a implementação do compensador é feita acrescentando-se uma suavização na transição de estado. A resposta da compensação implementada é apresentada na Figura 36.

Figura 36 – Resposta do compensador de folga em função da posição da esfera

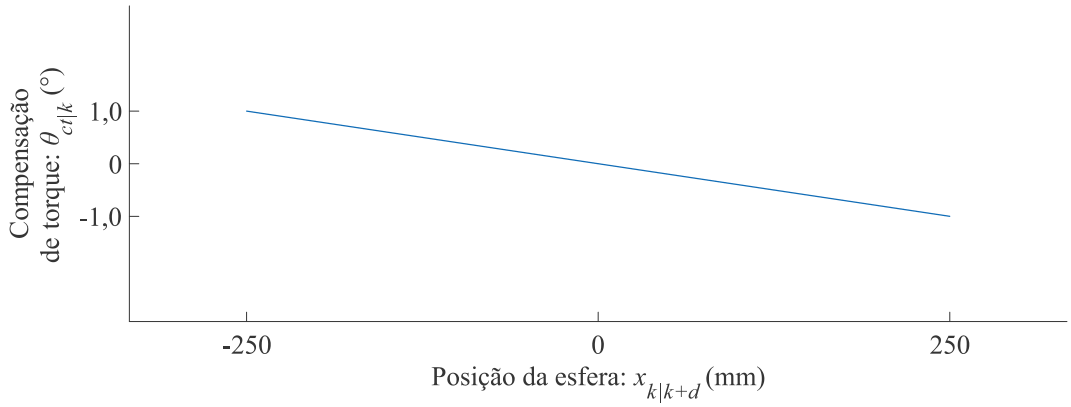


Fonte: Elaborado pelo autor do trabalho.

4.3.2.2 Compensador do Torque Originado pelo Peso da Esfera

A compensação deste desvio pode ser feita acrescendo-se um inclinação de $-\theta_t$ à inclinação de referência passada ao servomotor, conforme representado na Figura 37.

Figura 37 – Resposta do compensador de torque em função da posição da esfera

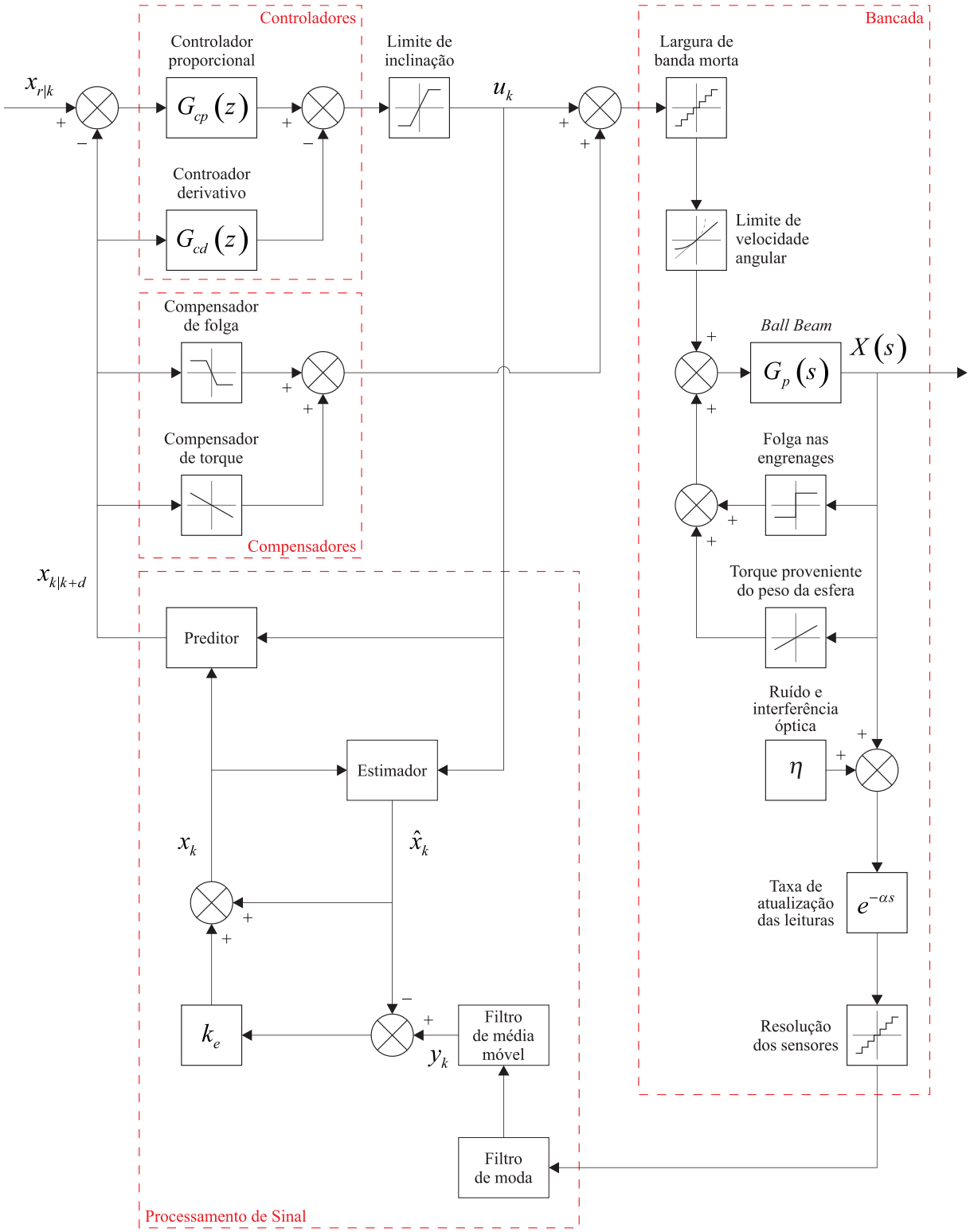


Fonte: Elaborado pelo autor do trabalho.

4.3.3 Representação Completa do Sistema

Uma representação completa do sistema é apresentada na Figura 38.

Figura 38 – Diagrama de blocos do sistema completo



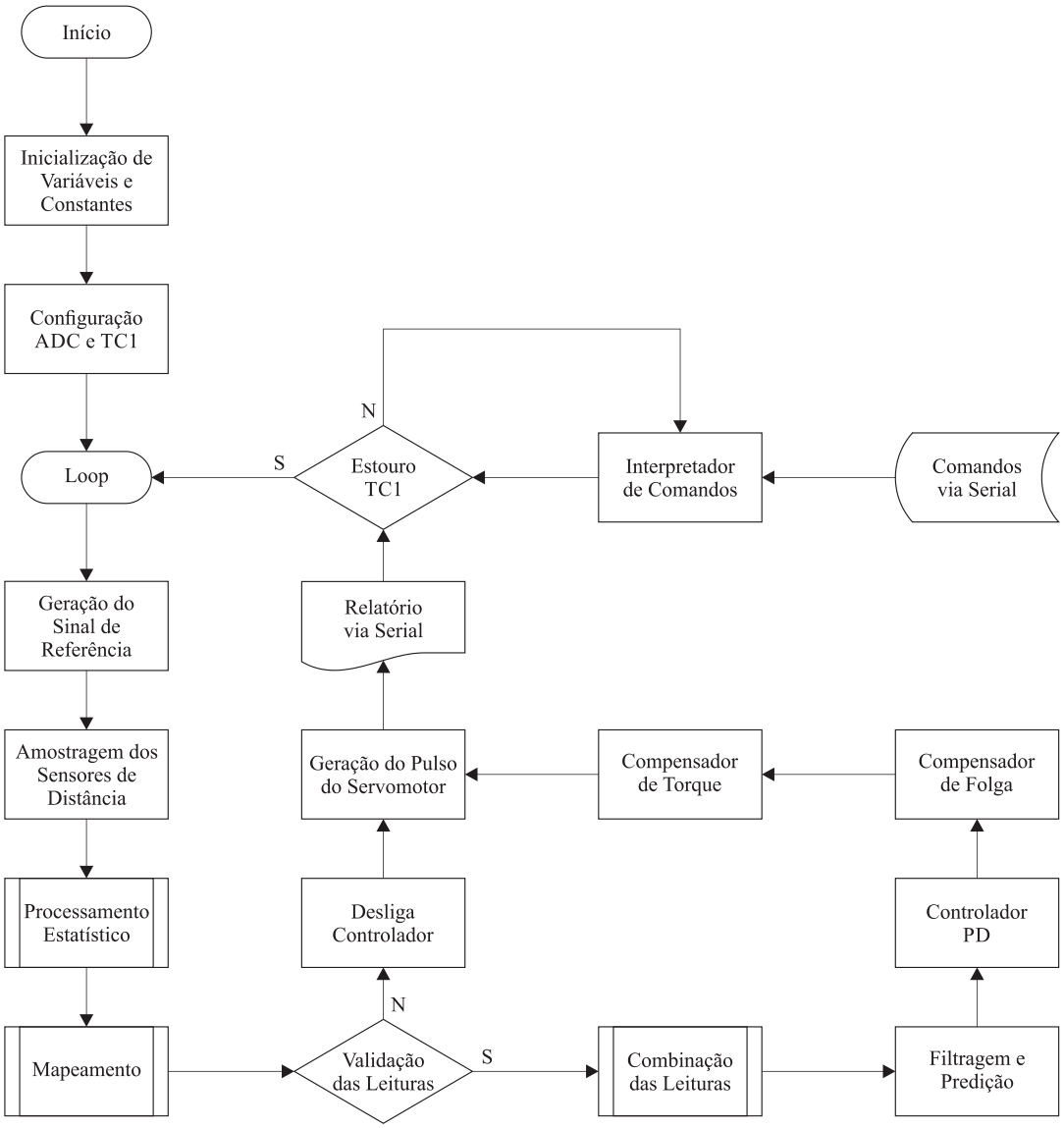
Fonte: Elaborado pelo autor do trabalho.

4.4 PROGRAMAÇÃO DO MICROCONTROLADOR

Uma vez definida a arquitetura de controle e seus parâmetros, além de técnicas mitigadoras para as limitações do sistema, é necessário elaborar o programa que irá implementar todo o trabalho desenvolvido até aqui, explorando os recursos disponíveis no microcontrolador.

O fluxo de trabalho do microcontrolador é apresentado na Figura 39.

Figura 39 – Fluxo de trabalho do microcontrolador



Fonte: Elaborado pelo autor do trabalho.

O código fonte é disponibilizado no APÊNDICE A.

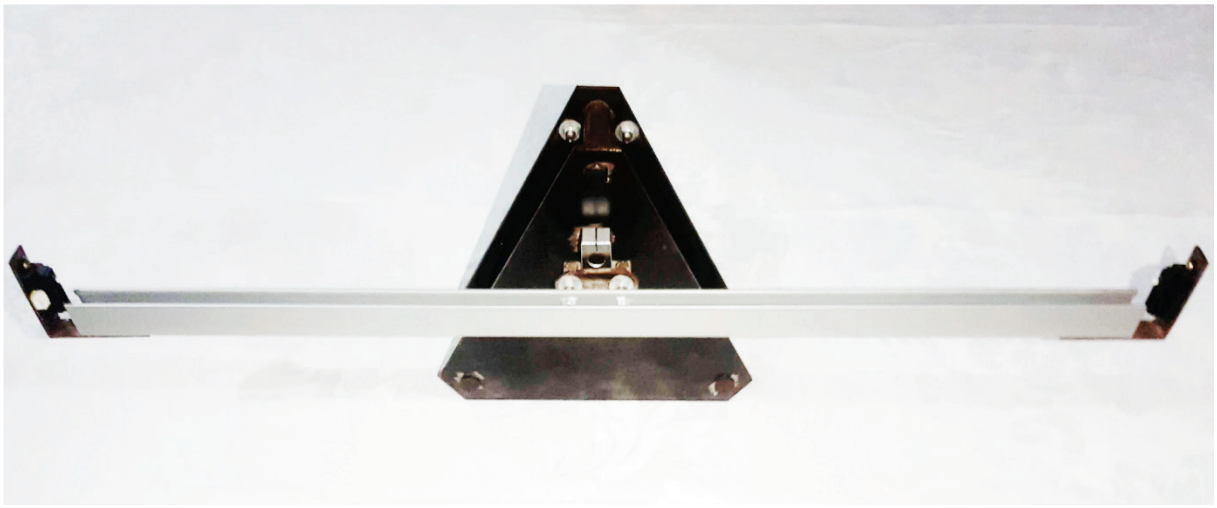
5 RESULTADOS

Neste capítulo serão apresentados os resultados construtivos da bancada, como também os resultados experimentais obtidos a partir do pleno funcionamento da bancada.

5.1 RESULTADOS CONSTRUTIVOS

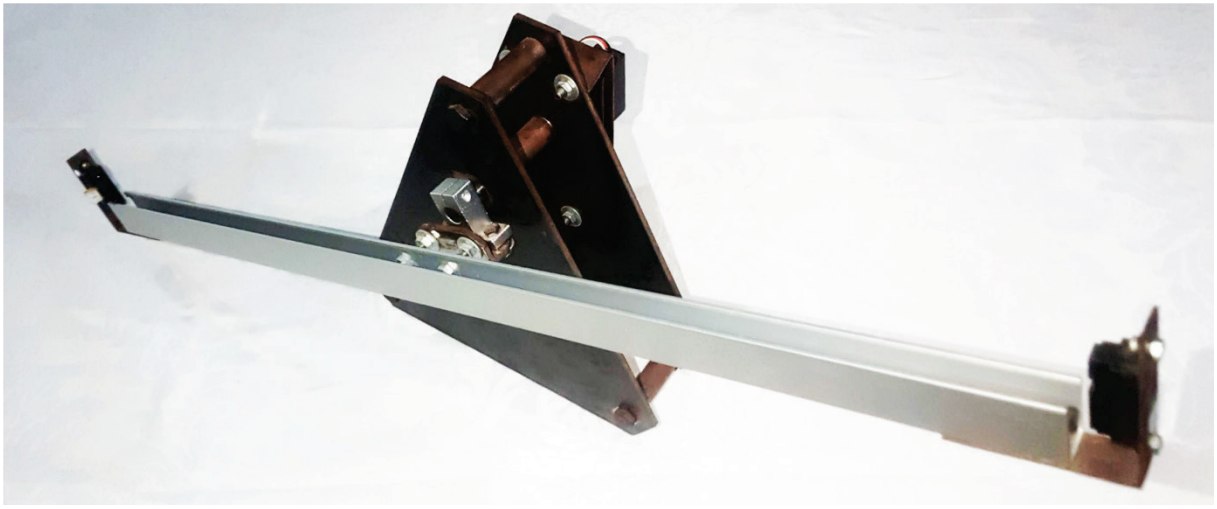
A bancada *ball beam* desenvolvida no presente trabalho é apresentada na Figura 40 e Figura 41.

Figura 40 – Vista frontal da bancada *ball beam*



Fonte: Elaborado pelo autor do trabalho.

Figura 41 – Vista em perspectiva da bancada *ball beam*



Fonte: Elaborado pelo autor do trabalho.

5.2 RESULTADOS EXPERIMENTAIS

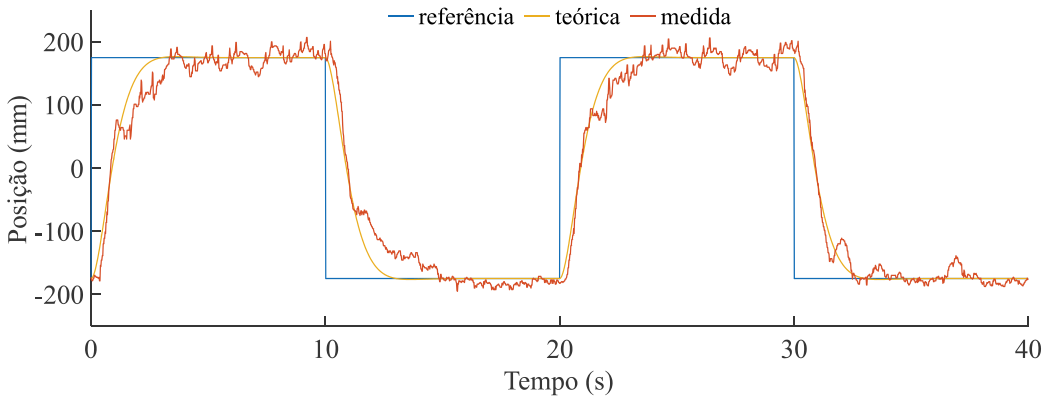
Após a implementação do código fonte no microcontrolador a bancada foi colocada em funcionamento. Foram executados testes de rastreamento de sinais periódicos.

Os resultados experimentais de resposta a onda quadrada e a onda senoidal são apresentados na Figura 42 e Figura 43, respectivamente. São apresentados a posição de referência, a posição teórica da esfera obtida a partir da função de transferência de malha fechada, Equação (22), e a posição medida a partir das leituras dos sensores.

5.2.1 RESPOSTA A ONDA QUADRADA

A bancada foi submetida a um sinal de referência periódico do tipo onda quadrada com período de 20 segundos e amplitude de 175mm. Observa-se que o rastreamento ocorreu de forma relativamente satisfatória. As respostas, logo após a mudança da posição de referência, iniciam-se de maneira bastante precisa, ocorrendo na maioria das vezes um pequeno desvio em relação a posição teórica logo após a passagem pela região central da barra, o que pode ser atribuído a transferência de leitura de posição entre os sensores bem como ao efeito de interferência, o qual é mais forte na região central da barra. Ocorreu também uma pequena oscilação em regime permanente, o que foi considerado aceitável devido à natureza instável do sistema e a falta de precisão tanto da medição da posição, quanto do atuador que tem a função de controlar a inclinação da barra.

Figura 42 – Resposta a onda quadrada

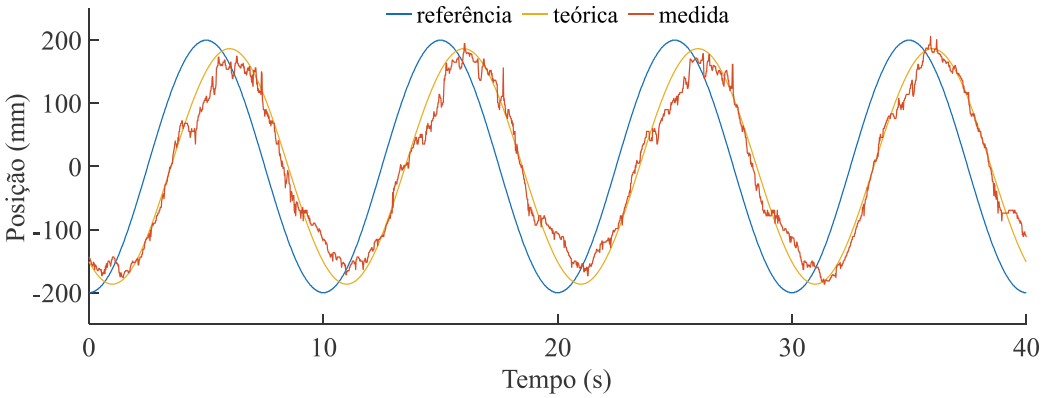


Fonte: Elaborado pelo autor do trabalho.

5.2.2 Resposta a Onda Senoidal

A bancada foi submetida a um sinal de referência periódico do tipo onda senoidal com período de 10 segundos e amplitude de 200mm. Observa-se que o rastreamento também ocorreu de forma relativamente satisfatória, com defasagem e amplitudes muito próximas ao esperado. Assim como na onda quadrada, ocorreu um pequeno desvio em relação a posição teórica logo após a passagem pela região central da barra.

Figura 43 – Resposta a onda senoidal



Fonte: Elaborado pelo autor do trabalho.

6 CONSIDERAÇÕES FINAIS

As limitações de projeto impostas tanto pelo atuador como também pelos transdutores, aliados a dinâmica não controlada do servomotor, impuseram um grande desafio à controlabilidade da bancada. As soluções implantadas mitigaram em boa parte estas limitações, permitindo um desempenho aceitável.

Contudo para se obter um melhor resultado, faz-se necessário a utilização de outro tipo de transdutor, tal que produza um sinal mais preciso e um nível de ruído mais atenuado, bem como pode-se optar por um atuador sem controle interno de posição, o que abriria uma janela de possibilidades totalmente nova para o sistema de controle.

REFERÊNCIAS

AGUIRRE, L. A. **Introdução a Identificação de Sistemas - Técnicas Lineares e Não Lineares:** Teoria e Aplicação. 4ª. ed. Belo Horizonte: UFMG, 2015.

ARDUINO UNO REV3. Disponivel em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Acesso em: 05 out. 2018.

ATMEL. 8-bit AVR Microcontrollers ATmega328/P DATASHEET COMPLETE, nov. 2016. Disponivel em: <http://ww1.microchip.com/downloads/en/devicedoc/atmel-42735-8-bit-avr-microcontroller-atmega328-328p_datasheet.pdf>. Acesso em: 9 out. 2018.

CASTRUCCI, P. D. L.; BITTAR, A.; SALES, R. M. **Controle Automático**. Rio de Janeiro: LTC, 2011.

CONTROLE DE UM BALL AND BEAM UTILIZANDO SISTEMA SEGUIDOR COM ATRIBUIÇÃO DE AUTO-ESTRUTURA COMPLETA. **Anais do XX Congresso Brasileiro de Automática**, Belo Horizonte, 20 a 24 Setembro 2014. 2587 a 2592.

COSTA, F. D. M. **Desenvolvimento e Controle do sistema "Ball and Beam"**. Universidade Federal de Ouro Preto. João Monlevade. 2018.

DORF, R. C.; BISHOP, R. H. **Sistemas de Controle Modernos**. 8ª. ed. Rio de Janeiro: LTC, 2001.

GOTECK RC INC. GS-5515MG.pdf, 27 ago. 2010. Disponivel em: <www.goteckrc.com/Download/STD Servo/GS-5515MG.pdf>. Acesso em: 18 set. 2018.

HALLIDAY, D.; RESNICK, R.; WALKER, J. **Fundamentos de Física**. 4ª. ed. Rio de Janeiro: LTC, v. 1, 1996.

LANDAU, I. D.; ZITO, G. **Digital Control Systems:** Designs, Identification and Implementation. London: Springer, 2006.

LIMA, B. M.; LIMA, D. M.; NORMEY-RICO, J. E. **A robust predictor for dead-time systems based on the Kalman filter**, Florianópolis, 2018. 9th IFAC Symposium on Robust Control Design.

LIMA, C. B. D.; VILLAÇA, M. V. M. **AVR e ARduino:** técnicas de projeto. 2ª. ed. Florianópolis: dos autores, 2012.

NASCIMENTO FILHO, P. S. **Construção de Protótipo de Bancada para Ensino e Pesquisa de Sistemas de Controle: "Ball & Beam"**. UFPA. Belém. 2008.

SHARP CORPORATION. GP2Y0A21YK0F.pdf, 01 dez. 2006. Disponivel em: <http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a21yk_e.pdf>. Acesso em: 21 set. 2018.

YU, W.; ORTIZ, F. Stability analysis of PD regulation for ball and beam system. **Proceedings of 2005 IEEE Conference on Control Applications**, Toronto, 2005. 517-522.

APÊNDICE A – Código-fonte

```
// constantes e grandezas físicas

const float
  e  = 2.7182818285, // número de Euler
  pi = 3.1415926536, // pi
  g  = 9.8, // aceleração da gravidade
  R  = 15e-3, // raio da esfera (15mm)
  W  = 11e-3, // largura da pista (11mm)
  K  = g * (20*pow(R,2) - 5*pow(W,2)) / (28*pow(R,2) - 5*pow(W,2)), // constante da
                                                                    função de transferência
  T  = 20e-3; // período de amostragem (20ms)

// pré cálculos úteis

const float
  KT24 = K * pow(T,2) / 4,
  pi180 = pi / 180;

// geração do sinal de referência

int wave = 2; // tipo do sinal de referência (0 constante, 1 senoidal, 2 quadrada,
                                                  3 triangular e 4 dente de serra)
unsigned long
  Tr = 20e6, // período do sinal de referência (µs)
  tr; // temporizador do sinal de referência (µs)
float
  tf, // fração de tempo do período atual
  xr = 150e-3, // posição de referência da esfera (m)
  peak = 150e-3; // amplitude de pico da onda de referência da esfera (m)

// mapeamento

const byte mapLevels = 14; // número de níveis mapeados
const int
  minRange = 65, // alcance mínimo (mm)
  setRange = 200, // alcance máximo exclusivo (mm)
  midRange = 290, // alcance com esfera na posição central da barra (mm)
  maxRange = 320, // alcance máximo (mm)
  rangeLevel[mapLevels] = // níveis de alcance (mm)
    {320, 290, 260, 230, 200, 170, 140, 110, 100,  90,  80,  75,  70,  65},
  voltLevel[2][mapLevels] = // níveis de tensão (LSB)
   {{ 30,  42,  53,  60,  67,  79,  91, 109, 117, 125, 137, 143, 148, 154},
                                                                  // sensor esquerdo
```

```
41      { 31,  36,  42,  48,  56,  65,  75,  93, 100, 109, 121, 126, 133, 140}};
                                                                        // sensor direito
42  float
43    invLevel[mapLevels], // inverso dos níveis de alcance
44    stepZone[2][mapLevels-1], // passo das zonas entre níveis de alcance
45    range[2]; // distância entre a esfera e os sensores
46
47  // validação das leituras dos sensores
48
49  const int validLimit = 50; // limite do contador de leituras válidas
50  int validCount = -1; // contador de leituras válidas
51
52  // filtro de média móvel
53
54  const byte N = 12; // número máximo de elementos no filtro
55  byte n = 5; // número de elementos ativos no filtro
56  float
57    xt[N], // vetor de leituras de posição
58    xtSum, // somatório dos elementos ativos no filtro
59    y; // posição da esfera na saída do filtro
60
61  // estimador
62
63  const byte D = 12; // limite de ciclos de atraso do processo
64  byte d = 4; // número de ciclos de atraso a serem considerados
65  float
66    xe, // posição estimada da esfera
67    x[3], // posição combinada da esfera
68    ke = 2.8e-2; // ganho do filtro estimador
69
70  // preditor
71
72  float xp[3]; // posição predita da esfera
73
74  // controlador
75
76  boolean control = false; // inicializa controle BallBeam desligado
77  const float uMax = 15 * pi180; // saturação do sinal de controle
                                                        (inclinação máxima de 15°)
78  float
79    up, // ação proporcional
80    ud[2], // ação derivativa
81    u[D+2], // sinal de controle
82    Kp = 0.45, // ganho proporcional
83    Kd = 0.45, // ganho derivativo
84    fd = 0.12, // filtro derivativo
85    kd = pow(e, - T / (Kd * fd)), // constante derivativa no mapeamento polo-zero
```

```
86    theta; // inclinação equivalente ao sinal de controle (°)
87
88  // compensadores
89
90  float
91    backlash, // compensação de folga (°)
92    backlashMax = 1.7, // compensação máxima de folga (°)
93    backlashBand = 30e-3, // banda de compensação parcial de folga (m)
94    torque, // compensação de torque (°)
95    torqueRatio = 4.0; // taxa de compensação de torque (°/m)
96
97  // geração do pulso
98
99  const int
100   servoTimeRatio = 2, // taxa de conversão de tempo em largura de pulso (2 LSB/µs)
101   servoDegreeRatio = 20; // taxa de conversão de graus em largura de pulso (20 LSB/°)
102 int
103   servoOffset = 1860 * servoTimeRatio, // largura do pulso para barra na
                                                    posição horizontal (µs * LSB/µs)
104   servoOut; // largura de pulso proporcional ao sinal de controle
                                                        e compensadores (° * LSB/°)
105
106 // geração do relatório
107
108 boolean report; // controle de relatório do processo
109
110 // interpretador de comandos
111
112 boolean commandError = false; // erro de comando
113 String commandBuffer; // buffer de comando
114
115 void setup() {
116
117   Serial.begin(115200); // inicia comunicação serial
118
119   // configuração do ADC para amostragem
120
121   DIDR0 = (1 << ADC0D) | (1 << ADC1D); // habilita ADC0 e ADC1 como entradas
122   ADMUX = (1 << REFS0) | (1 << ADLAR); // inicializa MUX no ADC0 alinhado à esquerda
                                                    com referência interna de 5V
123   ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // define prescaler do ADC (128)
124   ADCSRA |= (1 << ADEN) | (1 << ADSC) | (1 << ADATE); // habilita e inicia o ADC
                                                        no modo Free Running
125
126   // cálculo dos inversos e passos para mapeamento de posição
127
128   for (int i = 0; i < 2; i++) // loop para cálculo dos sensores esquerdo e direito
129     for (int j = 0; j < mapLevels; j++) { // loop para cálculo dos inversos e passos
```

```
130       invLevel[j] = (float) 1 / rangeLevel[j]; // inverso do nível
131       if (j > 0) // descarta a primeira iteração para cálculo do passo
                                                                  das zonas entre níveis
132         stepZone[i][j-1] = (invLevel[j] - invLevel[j-1]) / (voltLevel[i][j]
                                                - voltLevel[i][j-1]); // cálculo do passo
133     }
134
135   // configuração do TC1 para geração do pulso
136
137   DDRB = 1 << DDB1; // habilita pino OC1A (pino 9) como saída
138   PORTB = !DDRB; // habilita pull-up interno dos demais pinos do PORTB
139   ICR1 = 39999; // define contagem de overflow (62.5ns * prescaler 8 * 40000 = 20ms)
140   OCR1A = servoOffset; // inicializa servo na posição horizontal
141   TCCR1A = (1 << COM1A1) | (1 << WGM11); // configura TC1 para modo fast PWM
                                                        via ICR1 (WGM11, WGM12 e WGM13)
142   TCCR1B = (1 << WGM13) | (1<<WGM12) | (1 << CS11); // com prescaler 8 (CS11),
                                                            habilitando canal A (COM1A1)
143   TIMSK1 = 0; // desabilita as interrupções do TC1
144 }
145
146 void loop() {
147
148   // geração do sinal de referência
149
150   tr += T*1e6; // adiciona tempo entre leituras de posição ao temporizador
                                                                  do sinal de referência
151   if (tr >= Tr) // se temporizador ultrapassou o período
152     tr -= Tr; // ajusta temporizador
153   tf = (float) tr / Tr; // calcula fração de tempo do período atual
154   switch (wave) { // identifica tipo de onda a ser gerada
155     case 1: // onda senoidal
156       xr = sin(2 * pi * tf) * peak;
157     break;
158     case 2: // onda quadrada
159       if (tf < 0.5)
160         xr = peak;
161       else
162         xr = -peak;
163     break;
164     case 3: // onda triangular
165       if (tf < 0.5)
166         xr = (4 * tf - 1) * peak;
167       else
168         xr = (3 - 4 * tf) * peak;
169     break;
170     case 4: // onda dente de serra
171       xr = ( 2 * tf - 1) * peak;
172     break;
```

```
173   }
174
175   // amostragem
176
177   for (int i = 0; i < 2; i++) { // loop para seleção do canal de amostragem
178     byte dataSet[36]; // conjunto de dados da amostragem
179     for (int j = 0; j < 36; j++) { // loop de amostragem
180       ADCSRA |= (1 << ADIF); // limpa o sinalizador de converão concluída
181       while (!(ADCSRA & (1 << ADIF))); // aguarda término da conversão
182       dataSet[j] = ADCH; // armazena amostra
183     } // ao final da amostragem de um canal
184     ADMUX = (1 << REFS0) | (1 << ADLAR) | (1 - i); // muda o canal de amostragem
                                                       mantendo referência e alinhamento
185
186     // filtro de moda
187
188     byte modeCount = 0; // contagem da moda
189     byte modeValue = 0; // valor da moda
190     for (int j = 0; j < 36; j++) { // loop da amostragem
191       int count = 0; // inicializa contador
192       for (int k = 0; k < 36; k++) // loop da varredura
193         if (dataSet[j] == dataSet[k]) // se amostra coincide com varredura
194           count++; // incrementa contador
195       if (count > modeCount) { // se contagem atual maior que contagem da moda
196         modeCount = count; // atualiza contagem da moda
197         modeValue = dataSet[j]; // atualiza valor da moda
198       }
199     }
200
201     // mapeamento
202
203     if (modeValue >= voltLevel[i][mapLevels-1]) // se valor lido maior que
                                                                          limite máximo
204       range[i] = minRange; // define leitura para posição mínima
205     else // senão
206       if (modeValue < voltLevel[i][0]) // se valor lido menor que limite mínimo
207         range[i] = maxRange; // define litura para posição máxima
208       else // senão mapeia leitura
209         for (int j = 0; j < mapLevels - 1; j++) // loop para busca de intervalo
                                                                          de mapeamento
210           if ((modeValue >= voltLevel[i][j]) && (modeValue < voltLevel[i][j+1]))
                                                       // se leitura está no intervalo
211             range[i] = 1 / ((modeValue - voltLevel[i][j]) * stepZone[i][j]
                                                       + invLevel[j]); // mapeia leitura
212   }
213
214   // validação das leituras dos sensores
215
```

```
216   if ((range[0] == minRange) // se a esfera está muito próxima do sensor esquerdo
217   ||   (range[1] == minRange) // ou muito próxima do sensor direito
218   ||  ((range[0] <= setRange ) && (range[1] <= setRange)) // ou se há conflito
                                                                  entre os sensores
219   ||  ((range[0] == maxRange) && (range[1] == maxRange))) { // ou ainda se a esfera
                                                                  foi retirada da barra
220     control = false; // desliga controle
221     validCount = 0; // reinicia contagem de leituras válidas
222   }
223   else { // senão
224     if (validCount < validLimit) // se contagem de leituras válidas menor que limite
225       validCount++; // incrementa número de contagens válidas
226     if ((validCount == validLimit) && (!control)) { // se contagem atingiu limite
                                                          e controle está desligado
227       control = true; // liga o controle Ball Beam
228       report = true; // liga a geração de relatório
229       tr = 0; // reinicia temporizador do sinal de referência
230     }
231   }
232
233   // cálculo da leitura de posição da esfera
234
235   for (byte i = N - 1; i > 0; i--) xt[i] = xt[i-1]; // desloca vetor de
                                                          leituras de posição
236   xt[0] = 0; // inicializa leitura atual
237   if (validCount > 0) // se leitura atual é válida
238     if ((range[0] > setRange) && (range[1] > setRange)) // se a leitura de ambos
                                                  os sensores está na faixa de transição
239       for (byte i = 0; i < 2; i++) { // incia loop para cálculo dos pesos e posição
240         float weight = (maxRange - range[i]) / (2 * maxRange - range[0] - range[1]);
                                                          // peso de cada leitura
241         xt[0] += 2 * (0.5 - i) * (range[i] - midRange) * weight / 1e3; // posição
                                                  em função da leitura e do peso
242       }
243     else // senão verifica sensor mais próximo
244       if (range[0] < range[1]) // se a esfera está mais próxima do sensor esquerdo
245         xt[0] = (range[0] - midRange) / 1e3; // calcula posição pela
                                                          leitura do sensor esquerdo
246       else // senão está mais próxima do sensor direito
247         xt[0] = (midRange - range[1]) / 1e3; // calcula posição pela
                                                          leitura do sensor direito
248
249   // filtro de média móvel
250
251   xtSum = 0; // limpa somatório
252   for (int i = n - 1; i >= 0; i--) xtSum += xt[i]; // realiza somatório
253   y = xtSum / n; // calcula posição da esfera na saída do filtro
254
255   // estimador
256
```

```
257   for (byte i = 2; i > 0; i--) x[i] = x[i-1]; // desloca vetor de posições combinadas
258   if (control) { // se controle ativo
259     xe = KT24 * (u[d-1] + 2*u[d] + u[d+1]) + 2*x[1] - x[2];
                                                          // estimada posição da esfera
260     x[0] = xe + ke * (y - xe); // combina leitura e estimativa da posição da esfera
261   }
262   else { // senão
263     x[0] = y; // posição é determinada apenas pela leitura
264   }
265
266   // preditor
267
268   xp[1] = x[1]; xp[0] = x[0]; // inicializa vetor com posições combinadas
269   for (int i = 1; i < d; i++) { // loop de predição
270     for (byte i = 2; i > 0; i--) xp[i] = xp[i-1]; // desloca vetor de posições preditas
271     xp[0] = KT24 * (u[d-i-1] + 2*u[d-i] + u[d-i+1]) + 2*xp[1] - xp[2];
                                                                  // prediz posição
272   }
273
274   // controlador
275
276   for (byte i = D + 1; i > 0; i--) u[i] = u[i-1]; // desloca vetor de sinal de controle
277   ud[1] = ud[0]; // desloca vetor de ação derivativa
278   if (control) { // se controle está ativo
279     up = (xr - xp[0]) * Kp; // calcula ação proporcional
280     ud[0] = (xp[0] - xp[1]) / fd + kd * ud[1]; // e ação derivativa
281     u[0] = up - ud[0]; // combina ação proporcional e derivativa
282     u[0] = max(min(u[0], uMax), -uMax); // saturação do sinal de controle
283   }
284   else { // senão
285     u[0] = 0; // sinal de controle é nulo
286   }
287   theta = u[0] / pi180; // inclinação da barra equivalente ao sinal de controle (°)
288
289   // compensadores
290
291   if (control) {
292     backlash = - xp[0] * backlashMax / backlashBand;  // compensação de folga (°)
293     backlash = max(min(backlash, backlashMax), -backlashMax); // saturação da
                                                              compensação de folga
294     torque = - xp[0] * torqueRatio; // compensação de torque (°)
295   }
296   else {
297     backlash = 0;
298     torque = 0;
299   }
300
```

```
301   // geração do pulso
302
303   servoOut = (theta + backlash + torque) * servoDegreeRatio; // largura do
                                                                    pulso proporcional
304   OCR1A = servoOut + servoOffset; // combina largura de pulso proporcional e offset
305
306   // geração do relatório
307
308   if (report && control) { // se geração de relatório e controle ativos
309     Serial.print(xr*1e3,0); Serial.print(F(" "));
310     Serial.print(xt[0]*1e3,2); Serial.print(F(" "));
311     Serial.print(y*1e3,2); Serial.print(F(" "));
312     Serial.print(x[0]*1e3,1); Serial.print(F(" "));
313     Serial.print(xp[0]*1e3,1); Serial.print(F(" "));
314     Serial.print(225); Serial.print(F(" "));
315     Serial.print(-225); Serial.print(F(" "));
316     Serial.print(theta,1); Serial.print(F(" "));
317     Serial.println();
318   }
319
320   // interpretador de comandos
321
322   while (!(TIFR1 &= (1 << TOV1)));;; // enquanto aguarda novo pulso
323     while (Serial.available()) { // se houver dado disponível na porta serial
324       const byte commandBufferSize = 125; // limite do tamanho do buffer de comando
325       char c = Serial.read(); // lê byte do buffer serial
326       if (!((c == '\r') || (c == '\n'))) { // se não é caracter de fim de comando
327         if (commandBuffer.length() < commandBufferSize) // se tamanho do buffer
                                                              dentro do limite
328           commandBuffer += (String)c; // adiciona caracter ao buffer
329         else // senão
330           commandError = true; // ativa marcador de erro de comando
331       }
332       else { // mas se é caracter de fim de comando
333         const byte commandCount = 17; // número de comandos do interpretador
334         const String command[commandCount] = { "help", "status", "report", "offset",
                                                   "n", "d", "ke", "Kp", "Kd", "fd",
335           "bM", "bB", "tR", "xr", "wave", "peak", "period"}; // commandos
336         const String space = " "; // string de espaçamento entre comando e parâmetro
337         String commandString; // comando recebido
338         String commandParameter; // parâmetro recebido
339         int spaceIndex = commandBuffer.indexOf(space); // procura caracter de espaço
                                                            na string recebida
340         if (spaceIndex == -1) { // se não há espaço na string recebida
341           commandString = commandBuffer; // comando recebido é a prórpia string
342         }
343         else { // senão
344           commandString = commandBuffer.substring(0, spaceIndex); // separa comando
```

```
345           commandParameter = commandBuffer.substring(spaceIndex + 1); // do parâmetro
346         }
347         int commandIndex = -1; // índice do comando recebido
348         for (byte i = 0; i < commandCount; i++) // loop para identificação do
                                                                    índice do comando
349           if (commandString == command[i]) // compara string recebida com os comandos
350             commandIndex = i; // marca índice correspondente
351         float commandValue = commandParameter.toFloat(); // avalia numericamente
                                                                  o parâmetro recebido
352         switch (commandIndex) { // seleciona ação de acordo com o comando recebido
353
354           case 0: // help
355             report = false;
356             Serial.println();
357             Serial.println(F("Descrição dos comandos"));
358             Serial.println();
359             Serial.println(F("status: exibe os valores definidos para os parâmetros"));
360             Serial.println(F("report: liga ou desliga o relatório do processo"));
361             Serial.println(F("offset: define a largura de pulso para barra na
                                                          horizontal (µs)"));
362             Serial.println(F("n: define o número de elementos ativos no
                                                        filtro de média móvel"));
363             Serial.println(F("d: define o número de ciclos de atraso do sistema"));
364             Serial.println(F("ke: define o ganho do filtro estimador"));
365             Serial.println(F("Kp: define o ganho proporcional"));
366             Serial.println(F("Kd: define o ganho derivativo"));
367             Serial.println(F("fd: define a constante do filtro derivativo"));
368             Serial.println(F("bM: define a compensação máxima de folga
                                                          nas engrenagens (°)"));
369             Serial.println(F("bB: define a banda de compensação de folga
                                                        nas engrenagens (mm)"));
370             Serial.println(F("tR: define a taxa de compensação de torque (°/m)"));
371             Serial.println(F("xr: define uma posição de referência fixa
                                                          para esfera (mm)"));
372             Serial.println(F("wave: define um formato de onda para o sinal de
                                                        referência, sendo:"));
373             Serial.println(F("      1 - Senoidal"));
374             Serial.println(F("      2 - Quadrada"));
375             Serial.println(F("      3 - Triangular"));
376             Serial.println(F("      4 - Dente de serra"));
377             Serial.println(F("peak: define a amplitude de pico do sinal de
                                                          referência (mm)"));
378             Serial.println(F("period: define o período do sinal de referência (s)"));
379             Serial.println();
380           break;
381
382           case 1: // status
383             report = false;
384             Serial.println();
385             Serial.println(F("Status"));
```

```
386             Serial.println();
387             Serial.print(F("offset..: ")); Serial.print(int(servoOffset/2));
                                                          Serial.println(F("µs"));
388             Serial.print(F("n.......: ")); Serial.print(n);
                                                    Serial.println(F(" elementos"));
389             Serial.print(F("d.......: ")); Serial.print(d);
                                                       Serial.println(F(" ciclos"));
390             Serial.print(F("ke......: ")); Serial.print(ke*1e2,1);
                                                        Serial.println(F("%"));
391             Serial.print(F("Kp......: ")); Serial.println(Kp, 2);
392             Serial.print(F("Kd......: ")); Serial.println(Kd, 2);
393             Serial.print(F("fd......: ")); Serial.println(fd, 2);
394             Serial.print(F("bM......: ")); Serial.print(backlashMax, 1);
                                                        Serial.println(F("°"));
395             Serial.print(F("bB......: ")); Serial.print(backlashBand*1e3, 0);
                                                        Serial.println(F("mm"));
396             Serial.print(F("tR......: ")); Serial.print(torqueRatio, 2);
                                                        Serial.println(F("°/m"));
397             if (wave == 0) {
398               Serial.print(F("xr......: ")); Serial.print(xr*1e3, 0);
                                                        Serial.println(F("mm"));
399             }
400             else {
401               Serial.print(F("wave....: ")); Serial.print(wave);
402               if (wave == 1) Serial.println(F(" - senoidal"));
403               if (wave == 2) Serial.println(F(" - quadrada"));
404               if (wave == 3) Serial.println(F(" - triangular"));
405               if (wave == 4) Serial.println(F(" - dente de serra"));
406               Serial.print(F("peak....: ")); Serial.print(peak*1e3, 0);
                                                        Serial.println(F("mm"));
407               Serial.print(F("period..: ")); Serial.print(Tr/1e6, 0);
                                                        Serial.println(F("s"));
408             }
409             Serial.println();
410           break;
411
412           case 2: // report
413             report = !(report); // inverte status do relatório
414           break;
415
416           case 3: // offset: largura de pulso na horizontal (µs)
417             if ((commandValue >= 800) && (commandValue <= 2300)) {
                                                          // se parâmetro válido
418               servoOffset = (unsigned int)commandValue * 2; // define offset da barra
419               if ((!report) || (!control)) {
420                 Serial.print(F("offset..: ")); Serial.print(int(servoOffset/2));
                                                        Serial.println(F("µs"));
421               }
422             }
423             else // senão
424               commandError = true; // ativa marcador de erro de comando
```

```
425           break;
426
427           case 4: // n: número de elementos ativos no filtro de média móvel
428             if ((commandValue > 0) && (commandValue <= N)) { // se parâmetro válido
429               n = commandValue; // define número de elementos ativos
430               if ((!report) || (!control)) {
431                 Serial.print(F("n.......: ")); Serial.print(n);
                                                   Serial.println(F(" elementos"));
432               }
433             }
434             else // senão
435               commandError = true; // ativa marcador de erro de comando
436           break;
437
438           case 5: // d: número de ciclos de atraso do servo
439             if ((commandValue > 0) && (commandValue <= D)) { // se parâmetro válido
440               d = commandValue; // define ciclos de atraso
441               if ((!report) || (!control)) {
442                 Serial.print(F("d.......: ")); Serial.print(d);
                                                     Serial.println(F(" ciclos"));
443               }
444             }
445             else // senão
446               commandError = true; // ativa marcador de erro de comando
447           break;
448
449           case 6: // ke: ganho do filtro estimador
450             if ((commandValue > 0) && (commandValue <= 100)) { // se parâmetro válido
451               ke = commandValue / 1e2; // define ganho de filtragem
452               if ((!report) || (!control)) {
453                 Serial.print(F("ke......: ")); Serial.print(ke*1e2, 1);
                                                   Serial.println(F("%"));
454               }
455             }
456             else // senão
457               commandError = true; // ativa marcador de erro de comando
458           break;
459
460           case 7: // Kp: ganho proporcional
461             if ((commandValue >= 0.1) && (commandValue <= 10)) { // se parâmetro válido
462               Kp = commandValue; // define ganho
463               if ((!report) || (!control)) {
464                 Serial.print(F("Kp......: ")); Serial.println(Kp, 2);
465               }
466             }
467             else // senão
468               commandError = true; // ativa marcador de erro de comando
```

```
469             break;
470
471             case 8: // Kd: ganho derivative
472               if ((commandValue >= 0.1) && (commandValue <= 30)) { // se parâmetro válido
473                 Kd = commandValue; // define ganho
474                 if ((!report) || (!control)) {
475                   Serial.print(F("Kd......: ")); Serial.println(Kd, 2);
476                 }
477               }
478               else // senão
479                 commandError = true; // ativa marcador de erro de comando
480             break;
481
482             case 9: // fd: constante do filtro derivativo
483               if ((commandValue > 0) && (commandValue < 2)) { // se parâmetro válido
484                 fd = commandValue; // define constante do filtro
485                 kd = pow(e, - T / (Kd * fd)); // constante derivativa no
                                                                  mapeamento polo-zero
486                 if ((!report) || (!control)) {
487                   Serial.print(F("fd......: ")); Serial.println(fd, 2);
488                 }
489               }
490               else // senão
491                 commandError = true; // ativa marcador de erro de comando
492             break;
493
494             case 10: // bM: compensação máxima de folga nas engrenagens (°)
495               if ((commandValue >= 0) && (commandValue <= 3)) { // se parâmetro válido
496                 backlashMax = commandValue; // define compensação máxima de folga
497                 if ((!report) || (!control)) {
498                   Serial.print(F("bM......: ")); Serial.print(backlashMax, 1);
                                                          Serial.println(F("°"));
499                 }
500               }
501               else // senão
502                 commandError = true; // ativa marcador de erro de comando
503             break;
504
505             case 11: // bB: banda de compensação de folga nas engrenagens (mm)
506               if ((commandValue > 0) && (commandValue <= 225)) { // se parâmetro válido
507                 backlashBand = commandValue / 1e3; // define a banda de
                                                                  compensação de folga
508                 if ((!report) || (!control)) {
509                   Serial.print(F("bB......: ")); Serial.print(backlashBand*1e3, 0);
                                                        Serial.println(F("mm"));
510                 }
511               }
512               else // senão
```

```
513             commandError = true; // ativa marcador de erro de comando
514           break;
515
516           case 12: // tR: taxa de compensação de torque (°/m)
517             if ((commandValue >= 0) && (commandValue <= 20)) { // se parâmetro válido
518               torqueRatio = commandValue; // define a taxa de compensação de torque
519               if ((!report) || (!control)) {
520                 Serial.print(F("tR......: ")); Serial.print(torqueRatio, 2);
                                                    Serial.println(F("°/m"));
521               }
522             }
523             else // senão
524               commandError = true; // ativa marcador de erro de comando
525           break;
526
527           case 13: // xr: posição de referência fixa (mm)
528             if ((commandValue >= -200) && (commandValue <= 200)) { // se parâmetro
                                                                          válido
529               xr = commandValue / 1e3; // atualiza posição de referência
530               wave = 0; // define refrência fixa
531               if ((!report) || (!control)) {
532                 Serial.print(F("xr......: ")); Serial.print(xr*1e3, 0);
                                                    Serial.println(F("mm"));
533               }
534             }
535             else // senão
536               commandError = true; // ativa marcador de erro de comando
537           break;
538
539           case 14: // wave: tipo de onda do sinal de referência
                              (1 senoidal, 2 quadrada, 3 triangular e 4 dente de serra)
540             if ((int(commandValue) >= 1) && (int(commandValue) <= 4)) {
                                                    // se parâmetro válido
541               if (wave != int(commandValue) ) // se nova onda diferente da anterior
542                 tr = 0; // reinicia temporizador do sinal de referência
543               wave = int(commandValue); // define tipo de onda
544               if ((!report) || (!control)) {
545                 Serial.print(F("wave....: ")); Serial.print(wave);
546                 if (wave == 1) Serial.println(F(" - senoidal"));
547                 if (wave == 2) Serial.println(F(" - quadrada"));
548                 if (wave == 3) Serial.println(F(" - triangular"));
549                 if (wave == 4) Serial.println(F(" - dente de serra"));
550               }
551             }
552             else // senão
553               commandError = true; // ativa marcador de erro de comando
554           break;
555
```

```
556           case 15: // peak: amplitude de pico do sinal de referência (mm)
557             if ((commandValue > 0) && (commandValue <= 200)) { // se parâmetro válido
558               peak = commandValue / 1e3; // define amplitude de pico
559               if ((!report) || (!control)) {
560                 Serial.print(F("peak....: ")); Serial.print(peak*1e3, 0);
                                                    Serial.println(F("mm"));
561               }
562             }
563             else // senão
564               commandError = true; // ativa marcador de erro de comando
565           break;
566
567           case 16: // setPeriod: período do sinal de referência (s)
568             if ((commandValue >= 1) && (commandValue <= 30)) { // se parâmetro válido
569               float phase = (float) tr / Tr; // calcula a fase atual
570               Tr = commandValue * 1e6; // atualiza período do sinal de referência
571               tr = phase * Tr; // ajusta temporizador para manter a fase
572               if ((!report) || (!control)) {
573                 Serial.print(F("period..: ")); Serial.print(Tr/1e6, 0);
                                                    Serial.println(F("s"));
574               }
575             }
576             else // senão
577               commandError = true; // ativa marcador de erro de comando
578           break;
579         }
580         if (((!report) || (!control)) && (commandError)) { // se relatório desligado
                                                               e houve erro de comando
581           Serial.print(commandBuffer); // ecoa comando recebido e
582           Serial.println(F(" - Comando inválido!")); // envia mensagem de
                                                          comando inválido
583         }
584         commandError = false; // desliga marcador de erro de comando
585         commandBuffer = ""; // e apaga buffer
586       }
587     }
588 }
```