



TRABALHO DE GRADUAÇÃO

PROJETO E IMPLEMENTAÇÃO EM CONTROLADOR INDUSTRIAL PARA POSICIONAMENTO DE RISERS COM VALIDAÇÃO EXPERIMENTAL

Por,
Ataias Pereira Reis
Emanuel Pereira Barroso Neto

Brasília, julho de 2016



**ENGENHARIA
MECATRÔNICA**
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

PROJETO E IMPLEMENTAÇÃO EM CONTROLADOR INDUSTRIAL PARA POSICIONAMENTO DE RISERS COM VALIDAÇÃO EXPERIMENTAL

Por,
Ataias Pereira Reis
Emanuel Pereira Barroso Neto

*Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação*

Banca Examinadora

Prof. Eugênio Libório Feitosa Fortaleza, _____
ENM/UnB
Orientador

Prof. Eduardo Stockler Tognetti, ENE/UnB _____
Co-orientador

Prof. Guilherme Caribé de Carvalho, _____
ENM/UnB

Prof. Geovany Araújo Borges, ENE/UnB _____

Brasília, julho de 2016

FICHA CATALOGRÁFICA

REIS, ATAIAS PEREIRA; NETO, EMANUEL PEREIRA BARROSO;
Projeto e Implementação em Controlador Industrial para Posicionamento de Risers com Validação Experimental ,
[Distrito Federal] 2015.
x, 47p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2015). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Controle em Malha Fechada 2. Controlador Lógico Programável
3. Sistemas Offshore
I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

REIS, A. P., NETO, E. P. B. (2015). Projeto e Implementação em Controlador Industrial para Posicionamento de Risers com Validação Experimental. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº0XX, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 109p.

CESSÃO DE DIREITOS

AUTORES: Ataias Pereira Reis e Emanuel Pereira Barroso Neto

TÍTULO DO TRABALHO DE GRADUAÇÃO: Projeto e Implementação em Controlador Industrial para Posicionamento de Risers com Validação Experimental.

GRAU: Engenheiro

ANO: 2015

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito dos autores.

Ataias Pereira Reis

72926-048 Águas Lindas – GO – Brasil

Emanuel Pereira Barroso Neto

72130-450 Taguatinga – DF – Brasil

Dedicatórias

A Emanuel B., meu avô, fonte de inspiração eterna

Emanuel Pereira Barroso Neto

Dedico à minha família, que me apoiou ao longo desta jornada.

Ataias Pereira Reis

Agradecimentos

Eu agradeço a Deus, aos professores Eugênio Fortaleza e Eduardo Tognetti, orientadores deste trabalho, e a todos que realizaram trabalhos prévios que foram utilizados aqui. Agradeço também ao solícito Rédytton Brenner que nos ajudou muito no início.

Ataias Pereira Reis

Agradeço a Deus, aos professores Eugênio Fortaleza e Eduardo Tognetti, orientadores deste trabalho, sempre disponíveis quando mais precisamos; ao grande amigo Rédytton Brenner, que utilizou a bancada antes de nós e nos deu várias dicas úteis; a todos os meus amigos e familiares, que me deram forças nessa jornada; por fim, agradeço à minha dupla, Ataias, pela paciência, cordialidade e principalmente pela ajuda.

Emanuel Pereira Barroso Neto

RESUMO

A extração de petróleo em águas profundas requer operações bastante complexas. Em especial, a operação de entrada reentrada ocorre quando um *riser* é conectado de sua plataforma a um poço de petróleo. O deslocamento do *riser* até o poço deve ser feito de forma rápida e precisa, mas, atualmente, o controle de posição é manual, o que pode ser ineficiente. O presente trabalho busca validar técnicas de controle por meio de experimentos em uma planta de laboratório representativa de um *riser* por meio de controle em malha aberta e fechada para operar o deslocamento do *riser* de forma automática, com resultados rápidos e precisos, além de oferecer resistência a perturbações causadas pelo oceano, para que a ponta do *riser* seja corretamente conectada ao poço.

Palavras Chave: *riser*, controle, deslocamento

ABSTRACT

Deep sea petroleum exploration requires very complex operations. Particularly, the re-entry operation occurs when a riser is connected from the platform to a wellhead. The riser's displacement must be done quickly and precisely but, presently, the position control is manual, which may be inefficient. The present paper seeks to validate control techniques with experiments in a *riser* representative lab plant using open and closed loop control to operate the riser's displacement automatically, with fast and precise results, in addition to offering resistance against disturbances caused by the ocean, so that riser's tip is correctly connected to the wellhead.

Keywords: *riser*, control, displacement

SUMÁRIO

1	Introdução	1
1.1	CONTEXTUALIZAÇÃO	1
1.2	OBJETIVOS DO PROJETO	1
2	Fundamentos	4
2.1	MODELAGEM	4
2.1.1	EQUAÇÃO GOVERNANTES	4
2.1.2	DISCRETIZAÇÃO	6
2.1.3	ESTRATÉGIA DE REDUÇÃO DA ORDEM DO MODELO	8
2.2	CONTROLE	10
2.2.1	TÉCNICAS SIMPLES DE CONTROLE	10
2.2.2	CONTROLE DISCRETO NO ESPAÇO DE ESTADOS	11
2.2.3	PREDITOR DE SMITH	12
2.2.4	FILTRO DE KALMAN	15
2.3	BANCADA	17
2.3.1	CONTROLADOR LÓGICO-PROGRAMÁVEL	17
2.3.2	SERCOS INTERFACE	19
2.3.3	LINE INTERFACE MODULE 2094-AL09	19
2.3.4	DRIVE KINETIX 6000 DA ALLEN BRADLEY	19
2.3.5	SERVOMOTOR	19
2.3.6	SENSORES INDUTIVOS	21
2.3.7	CÂMERA PRESENCEPLUS	21
2.4	REDES UTILIZADAS	22
2.4.1	DEVICENET	22
2.4.2	ETHERNET/IP	22
2.4.3	OPC - <i>OLE for Process Control</i>	23
2.5	PROGRAMAÇÃO DO CLP	23
2.5.1	VISÃO GERAL	23
2.5.2	LINGUAGEM <i>ladder</i>	24
2.5.3	TEXTO ESTRUTURADO	26
2.6	PARÂMETROS PARA GERAR TRAJETÓRIAS	27
3	Resultados	28

3.1	CÂMERA	28
3.1.1	CONFIGURAÇÃO DA REDE	28
3.1.2	CALIBRAÇÃO	29
3.1.3	PROGRAMAÇÃO	30
3.2	CALIBRAÇÃO DO SERVOMOTOR	30
3.3	MODELO NO ESPAÇO DE ESTADOS	31
3.4	MALHA ABERTA	32
3.4.1	EXCURSÃO DE 30CM	33
3.4.2	EXCURSÃO DE 20CM	34
3.5	MALHA FECHADA.....	35
3.5.1	CONTROLADOR P	35
4	Conclusões.....	37
4.1	PERSPECTIVAS FUTURAS.....	37
REFERÊNCIAS BIBLIOGRÁFICAS		38
Anexos.....		41
I	Programas utilizados.....	42
I.1	REDUÇÃO MODAL	42
I.2	TEXTO ESTRUTURADO	47
I.2.1	INICIALIZAÇÃO DO PRIMEIRO TESTE DE MALHA ABERTA	47
I.2.2	INICIALIZAÇÃO DO SEGUNDO TESTE DE MALHA ABERTA.....	51
I.2.3	INICIALIZAÇÃO DOS TESTES DE MALHA FECHADA	53
I.2.4	INICIALIZAÇÃO DA REDE DEVICENET	54
I.2.5	PROGRAMA DE CONTROLE EM MALHA ABERTA	54
I.2.6	PROGRAMA DE CONTROLE PROPORCIONAL - MALHA FECHADA	54
I.3	LINGUAGEM <i>ladder</i>	55
I.3.1	EXECUÇÃO DE <i>trigger</i> DA CÂMERA	55
I.3.2	ROTINA DE PARADA DE EMERGÊNCIA	55
I.4	PROGRAMAS DA CÂMERA	55
I.4.1	DETECÇÃO DA POSIÇÃO HORIZONTAL DA BOLINHA.....	55

LISTA DE FIGURAS

1.1	Operação de reentrada [1]	2
1.2	Método atual para reconexão no poço [2]	2
1.3	Método proposto para reconexão no poço [2]	2
2.1	Malha aberta de controle	10
2.2	Malha fechada de controle.....	11
2.3	Malha fechada de controle, espaço de estados discreto.....	12
2.4	Estrutura básica do preditor de Smith [3].	13
2.5	Estrutura do preditor de Smith com filtro de Kalman e referências de topo e fundo [3].	15
2.6	Exemplo de estrutura de sistema discreto no tempo [4].	15
2.7	Exemplo de estrutura de filtro de Kalman [4].	16
2.8	Esquemático da Bancada Utilizada para o Experimento [2].....	17
2.9	CLP com identificação de elementos.....	17
2.10	Line Interface Module modelo 2094-AL09 da Allen Bradley [2]	20
2.11	Drive Kinetix 6000 da Allen Bradley	20
2.12	Servomotor modelo MPL-A310F-SJ22AA.....	21
2.13	Sensor indutivo 871T-R8B18 [2]	21
2.14	Câmera da Banner Engineering utilizada no projeto [2].....	22
2.15	Exemplo de programa <i>ladder</i> com instruções de movimentação.....	25
3.1	RSLinx com câmera reconhecida	29
3.2	Programa PresencePLUS para calibração da câmera	30
3.3	Resposta ao degrau para modelos reduzidos com atraso e sem atraso	32
3.4	Resposta ao degrau para modelos reduzidos com atraso e sem atraso, 25 segundos ..	33
3.5	Referência de Posição para Excursão de 30cm	33
3.6	Referência de Velocidade para Excursão de 30cm.....	33
3.7	Resultado com Velocidade Modelada para Excursão de 30cm	34
3.8	Resultado com Velocidade Constante para Excursão de 30cm.....	34
3.9	Referência de Posição para Excursão de 20cm	35
3.10	Referência de Velocidade para Excursão de 20cm	35
3.11	Resultado com Velocidade Modelada para Excursão de 20cm	35
3.12	Resultado com Velocidade Constante para Excursão de 20cm.....	35
3.13	Malha fechada de controle.....	36

3.14 Resultado com Malha Fechada Proporcional e $K_p = 0.0025 \frac{u}{mm \cdot s}$	36
3.15 Resultado com Malha Fechada Proporcional e $K_p = 0.0050 \frac{u}{mm \cdot s}$	36
I.1 <i>Trigger</i> da câmera	55
I.2 Parada de emergência	55
I.3 Detecção da posição horizontal da bolinha	56

LISTA DE TABELAS

2.1	Constantes do barbante	5
2.2	Constantes da bolinha de isopor	6
2.3	Principais instruções <i>ladder</i>	25
2.4	Principais instruções de controle de movimento em <i>ladder</i>	25
2.5	Dados para simulação em escala real [2]	27
2.6	Dados para simulação em escala laboratorial	27
3.1	Relações mm/px para diferentes seções da barra de alumínio	30
3.2	Dados de calibração do servomotor, média obtida é de 71.32 mm/unidade	31

LISTA DE SÍMBOLOS

Símbolos Latinos

v	Velocidade linear	[m/s]
r	Referência de posição	[m]

Símbolos Gregos

ω	Velocidade angular	[rad/s]
----------	--------------------	---------

Subscritos

M	Sistema Modal
R	Sistema Reduzido
D	Sistema Reduzido considerando Atraso

Sobrescritos

.	Variação temporal
-	Valor médio
$\hat{\cdot}$	Estimação

Siglas

PCI	<i>Peripheral Component Interconnect</i>
CPU	Unidade Central de Processamento - <i>Central Processing Unit</i>
CAN	<i>Control Area Networking</i>
CIP	Protocolo Industrial Comum - <i>Common Industrial Protocol</i>
CLP	Controlador Lógico Programável
PWM	Modulação por Largura de Pulso - <i>Pulse Width Modulation</i>
SI	Sistema Internacional de Unidades
EDS	<i>Electronic Data Sheet</i>
RSLogix	<i>Rockwell Software Logix 5000</i>
RSLogix5000	<i>Rockwell Software Logix 5000</i>
OLE	<i>Object Linking and Embedding</i>
OPC	<i>OLE for Process Control</i>

Capítulo 1

Introdução

A motivação do presente trabalho aparece na área petrolífera, nas operações de reentrada de risers; o objetivo é a validação de um sistema de controle em malha fechada em escala laboratorial como uma alternativa ao modelo manual empregado atualmente.

1.1 Contextualização

O petróleo tem importância econômica global. Uma das formas de extração do mesmo é aquela feita em águas profundas, a qual o Brasil tem feito importantes avanços desde a descoberta do Pré-Sal em 2006. Em abril de 2015, chegou-se à produção de mais de 800 mil barris por dia no pré-sal, com campos situados em águas profundas e ultraprofundas [5]. Os desafios nesta área da engenharia são enormes, pois as operações são muito complexas.

Na Figura 1.1, observa-se uma das operações necessárias para a extração no Pré-Sal, especificamente a operação de reentrada. Nesta operação, um *riser* deve ser conectado da plataforma até o poço de petróleo no leito oceânico. O comprimento do *riser* chega a 2km.

Devido à complexidade inerente das diversas operações *offshore*, propulsores e sensores de localização e orientação - GPS, giroscópios, câmeras, etc - são requisitos essenciais para se poder posicionar a embarcação e os *risers* [2].

1.2 Objetivos do projeto

O foco deste trabalho está na operação de reentrada, conforme apresentada na Figura 1.1. Atualmente, é uma operação feita manualmente por um operador na plataforma que observa remotamente as imagens capturas por *ROVs* (*Remotely Operated Vehicles*) da região do *riser* próxima ao poço e controla a plataforma através de um *joystick* com o auxílio do sistema de posicionamento dinâmico. O custo envolvido na operação é enorme e os riscos para o equipamento

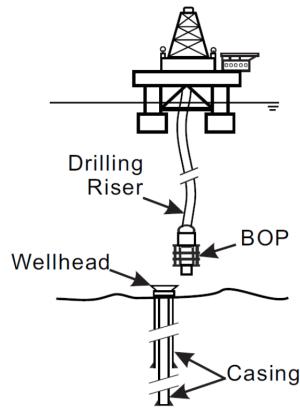


Figura 1.1: Operação de reentrada [1]

também, já que os próprios *ROVs* e o *riser* ficam sujeitos às perturbações das ondas, correntezas e variações ambientais no fundo do mar. A Figura 1.2 apresenta o esquema descrito.

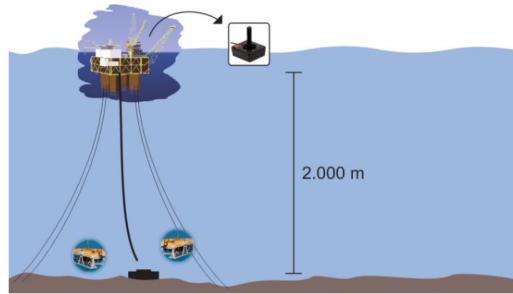


Figura 1.2: Método atual para reconexão no poço [2]

O objetivo deste trabalho é apresentar uma forma mais eficiente de realizar a operação de reentrada, economizando tempo e recursos, evitando riscos para pessoal e equipamento. Rédytton [2] validou o controle em malha aberta; neste presente trabalho, deseja-se projetar um controle em malha fechada, utilizando técnicas mais robustas, que atenda aos requisitos de trajetória da operação de reentrada do *riser*.

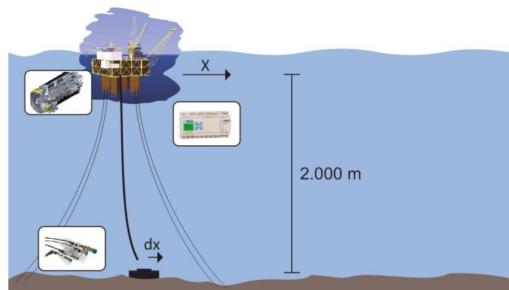


Figura 1.3: Método proposto para reconexão no poço [2]

Para fechar a malha, necessita-se de um sensor que realmente os dados de posição, o qual, na presente bancada, é uma câmera industrial. Passos necessários incluem:

- Alimentação da câmera;
- Atualização do Firmware da Câmera
- Conexão na rede Ethernet/IP da câmera e CLP;
- Configuração da câmera no software RSLogix;
- Alimentação da bancada com fontes dedicadas de 24V¹

Além de ser necessário o uso do sensor, para o presente trabalho será utilizada uma estrutura conhecida como Preditor de Smith; o sistema será representado por um modelo reduzido, e haverá um atraso intrínseco ao sistema. A combinação entre controlador, planta e modelo reduzido, em conjunção com o projeto das trajetórias de topo e de fundo do *riser* e a presença de um filtro de Kalman é a base para se compreender a estrutura do sistema de controle a ser desenvolvido.

¹Anteriormente, duas fontes de tensão de saída ajustável eram utilizadas, mas foram trocadas por fontes de saída de tensão única.

Capítulo 2

Fundamentos

Este capítulo apresenta equações básicas do sistema que se deseja validar, explica a redução modal utilizada a técnica de controle em malha fechada. Também são apresentadas informações sobre a bancada laboratorial e sobre a programação do CLP.

2.1 Modelagem

2.1.1 Equação Governantes

Estruturas submarinas tais como os *risers* são esbeltas e tem um alto módulo de cisalhamento. Portanto, a simplificação de Euler-Bernoulli para vigas é utilizada para propósitos de modelagem. O deslocamento de interesse é o horizontal e o *riser* está sob a ação de forças hidrodinâmicas externas e de tração. A equação diferencial parcial para a variável deslocamento, Υ , é dada por

$$m_s \frac{\partial^2 \Upsilon}{\partial t^2} = -EJ \frac{\partial^4 \Upsilon}{\partial z^4} + \frac{\partial}{\partial z} \left(T(z) \frac{\partial \Upsilon}{\partial z} \right) + F_n(z, t), \quad (2.1)$$

na qual m_s é a densidade linear do tubo, E é o módulo de Young e J é o segundo momento de inércia do *riser*. $T(z)$ descreve as forças de tração ao longo do comprimento do *riser*. $F_n(z, t)$ é a força resultante externa — força linear, unidade N/m [6].

No caso do problema em escala laboratorial que foi trabalhado, o tubo é representado por um barbante e daqui em diante serão discutidas e apresentadas as variáveis necessárias para se trabalhar com esse problema:

- m_s é a massa linear do barbante (densidade linear, kg/m);
- E é o módulo de Young do barbante e ele é desconhecido;
- J é o segundo momento de área e representa a resistência do barbante à flexão. O barbante não apresenta tal resistência, daí $J = 0$;
- $T(z)$ é a força de tração e é dada por

$$T(z) = (m_b + zm_s) g,$$

sendo m_b a massa da bolinha (kg), $m_s = m_{\text{barbante,kg}}/L$, sendo L o comprimento do barbante, z a posição vertical a partir do carrinho e g é a força da gravidade.

As únicas forças externas atuando no *riser* são hidrodinâmicas, exceto nas extremidades do topo e do fundo, nas quais forças de reação seguem condições de contorno. A equação de Morison descreve a força externa resultante:

$$F_n(z, t) = -m_{fbar} \frac{\partial^2 \Upsilon}{\partial t^2} - \mu \left| \frac{\partial \Upsilon}{\partial t} \right| \frac{\partial \Upsilon}{\partial t}, \quad (2.2)$$

na qual μ é o coeficiente de arrasto (unidade 1/s) e m_{fbar} é a massa do fluido adicionado, que será posteriormente pormenorizada. Fazendo $m = m_s + m_{fbar}$ e substituindo a Equação 2.2 na 2.1, obtém-se:

$$\frac{\partial^2 \Upsilon}{\partial t^2} = -\frac{EJ}{m} \frac{\partial^4 \Upsilon}{\partial z^4} + \frac{\partial}{\partial z} \left(\frac{T(z)}{m} \frac{\partial \Upsilon}{\partial z} \right) - \frac{\mu}{m} \left| \frac{\partial \Upsilon}{\partial t} \right| \frac{\partial \Upsilon}{\partial t} \quad (2.3)$$

Em relação à massa do fluido adicionado, m_{fbar} é dada por

$$\begin{aligned} m_{fbar} &= 2\pi r_{bar}^2 \rho_{ar} \\ &= 0.00770 \text{ g/m}. \end{aligned} \quad (2.4)$$

Já que $m_{fbar} \ll m_s$ conforme a Tabela 2.1, consideraremos $m = m_s$ nos cálculos. Em relação à massa m_{fb} do fluido adicionado ao redor da bolinha de isopor, ela é dada por

$$\begin{aligned} m_{fb} &= 1.2V_b \rho_{ar} \\ &= 1.2 \left(\frac{4}{3}\pi r_b^3 \right) \rho_{ar} \\ &= 0.0220 \text{ g}, \end{aligned} \quad (2.5)$$

de onde se pode observar que $m_{fb} \ll m_b$. Assim, os cálculos consideraram $m' \approx m_b$.

Significado	Símbolo	Valor	Unidade
Massa	m_{bar}	0.492	g
Comprimento	L	0.82	m
Massa linear	m_s	0.6	g/m
Raio	r_{bar}	1	mm
Densidade	ρ_{bar}	191	kg/m ³

Tabela 2.1: Constantes do barbante

Um ponto importante de se notar é que o barbante pesa mais do que o isopor, o que faz com que a tração não seja principalmente devida pela bolinha, mas sim pelo barbante. Neste caso, não se utiliza um valor médio para $T(z)$ como em [6], mas ainda se pode usar um valor médio para as constantes τ e τ' , que substituem o termo $\frac{\mu}{m} \left| \frac{\partial \Upsilon}{\partial t} \right|$ para o barbante e para a bolinha, respectivamente. Essas constantes são definidas de acordo com a trajetória prevista, uma vez que

Significado	Símbolo	Valor	Unidade
Massa	m_b	0.492	g
Raio	r_b	15.3	mm
Coeficiente de inércia	C_m	1.2	-
Coeficiente de arrasto	C_d	0.6	-
Volume	V_b	$\frac{4}{3}\pi r_b^3$	m^3
Área da seção transversal	A_b	πr_b^2	m^2

Tabela 2.2: Constantes da bolinha de isopor

a velocidade média depende dessa trajetória. Já levando em conta um valor médio para $\left| \frac{\partial \Upsilon}{\partial t} \right|$, tem-se

$$\frac{\partial^2 \Upsilon}{\partial t^2} = -\frac{EJ}{m} \frac{\partial^4 \Upsilon}{\partial z^4} + \frac{\partial}{\partial z} \left(\frac{T(z)}{m} \frac{\partial \Upsilon}{\partial z} \right) - \tau \frac{\partial \Upsilon}{\partial t} \quad (2.6)$$

Antes de se prosseguir para a discretização e obtenção das matrizes em espaço de estados, é importante pensar nas condições de contorno. No topo, $z = L$, tem-se $\Upsilon(L, t) = u(t)$, ou seja, o carrinho se move conforme uma trajetória $u(t)$ definida. Neste mesmo ponto, $\frac{\partial \Upsilon}{\partial z}(L, t) = 0$. Para a ponta na qual a carga está situada, $z = 0$, tem-se $\frac{\partial \Upsilon}{\partial z}(0, t) = \frac{F_L}{T}$, sendo F_L a força aplicada pela ponta do riser na carga.

2.1.2 Discretização

O objetivo desta seção é representar a Equação 2.6 em um espaço de estados finito discreto. Para isso, aplica-se o método de diferenças finitas na coordenada z de maneira a se aproximar a EDP governante em um número finito de EDOs. No espaço discreto, a equação do k -ésimo elemento é dada por

$$\begin{aligned} \frac{d^2 \Upsilon_k}{dt^2} &= -\frac{EJ}{ml^4} (\Upsilon_{k-2} - 4\Upsilon_{k-1} + 6\Upsilon_k - 4\Upsilon_{k+1} + \Upsilon_{k+2}) \\ &\quad + \frac{T_0 + mg(k-1)l}{ml^2} (\Upsilon_{k-1} - 2\Upsilon_k + \Upsilon_{k+1}) + g \frac{-\Upsilon_{k-1} + \Upsilon_{k+1}}{2l} - \tau \frac{d\Upsilon_k}{dt}, \end{aligned} \quad (2.7)$$

sendo N o número de pontos de discretização e l a distância entre dois pontos vizinhos ($l = L/N$).

Deve-se notar que $k \in \mathbb{N} : 2 \leq k \leq N - 1$, pois um dos extremos é a bolinha e a equação do pêndulo rege seu movimento, enquanto que na outra ponta se aplica uma condição de contorno. O que aconteceria quando $k = 2$ e se precisasse de Υ_{k-2} ? Para o presente experimento, $J = 0$ e esse problema não ocorre. Caso se façam testes com um valor de $J \neq 0$, tem-se de resolver esse problema primeiro. Uma solução seria utilizar diferenças finitas unilaterais para esses pontos próximos à fronteira.

Para simplificar, definem-se as constantes

$$a = -\frac{EJ}{ml^4} \quad (2.8)$$

$$b_k = \frac{T_0 + mg(k-1)l}{ml^2}, \quad k \geq 2 \quad (2.9)$$

$$c = \frac{g}{2l} \quad (2.10)$$

$$d_k = b_k - c, \quad k \geq 2 \quad (2.11)$$

$$e_k = b_k + c, \quad k \geq 2 \quad (2.12)$$

Uma estratégia para se analisar como as matrizes do espaço de estados do sistema ficarão é escolher um número de pontos N pequeno e escrever todas as equações. Observa-se que $a = 0$ para o barbante, pois $J = 0$, como apresentado anteriormente, o que simplifica os cálculos.

Para o caso $N = 6$, por exemplo, tem-se

$$\mathbf{x} = (\Upsilon_1 \ \Upsilon_2 \ \Upsilon_3 \ \Upsilon_4 \ \Upsilon_5 \ \Upsilon_6 \ \dot{\Upsilon}_1 \ \dot{\Upsilon}_2 \ \dot{\Upsilon}_3 \ \dot{\Upsilon}_4 \ \dot{\Upsilon}_5 \ \dot{\Upsilon}_6)^T \quad (2.13)$$

$$u = \Upsilon(L, t) = \Upsilon_7 \quad (2.14)$$

$$y = \Upsilon(0, t) = \Upsilon_1 \quad (2.15)$$

e as equações são

$$\begin{aligned} \ddot{\Upsilon}_2 &= b_2(\Upsilon_1 - 2\Upsilon_2 + \Upsilon_3) + c(-\Upsilon_1 + \Upsilon_3) - \tau\dot{\Upsilon}_2 \\ &= d_2\Upsilon_1 - 2b_2\Upsilon_2 + e_2\Upsilon_3 - \tau\dot{\Upsilon}_2 \end{aligned} \quad (2.16)$$

$$\begin{aligned} \ddot{\Upsilon}_3 &= b_3(\Upsilon_2 - 2\Upsilon_3 + \Upsilon_4) + c(-\Upsilon_2 + \Upsilon_4) - \tau\dot{\Upsilon}_3 \\ &= d_3\Upsilon_2 - 2b_3\Upsilon_3 + e_3\Upsilon_4 - \tau\dot{\Upsilon}_3 \end{aligned} \quad (2.17)$$

$$\begin{aligned} \ddot{\Upsilon}_4 &= b_4(\Upsilon_3 - 2\Upsilon_4 + \Upsilon_5) + c(-\Upsilon_3 + \Upsilon_5) - \tau\dot{\Upsilon}_4 \\ &= d_4\Upsilon_3 - 2b_4\Upsilon_4 + e_4\Upsilon_5 - \tau\dot{\Upsilon}_4 \end{aligned} \quad (2.18)$$

$$\begin{aligned} \ddot{\Upsilon}_5 &= b_5(\Upsilon_4 - 2\Upsilon_5 + \Upsilon_6) + c(-\Upsilon_4 + \Upsilon_6) - \tau\dot{\Upsilon}_5 \\ &= d_5\Upsilon_4 - 2b_5\Upsilon_5 + e_5\Upsilon_6 - \tau\dot{\Upsilon}_5 \end{aligned} \quad (2.19)$$

$$\begin{aligned} \ddot{\Upsilon}_6 &= b_6(\Upsilon_5 - 2\Upsilon_6 + u) + c(-\Upsilon_5 + u) - \tau\dot{\Upsilon}_6 \\ &= d_6\Upsilon_5 - 2b_6\Upsilon_6 + e_6u - \tau\dot{\Upsilon}_6 \end{aligned} \quad (2.20)$$

A equação para a posição da carga Υ_1 leva em conta a massa da bolinha e a força de Morison e é dada por

$$m_b\ddot{\Upsilon}_1 = \frac{m_bg}{l}(\Upsilon_2 - \Upsilon_1) + \rho_{\text{ar}}C_mV_b\ddot{\Upsilon}_1 - \frac{1}{2}\rho_{\text{ar}}C_dA_b\dot{\Upsilon}_1 |\dot{\Upsilon}_1|. \quad (2.21)$$

Isolando $\ddot{\Upsilon}_1$ na Equação 2.21, tem-se

$$\ddot{\Upsilon}_1 = \frac{m_bg}{m'l}(\Upsilon_2 - \Upsilon_1) - \frac{1}{2m'}\rho C_d A_b |\dot{\Upsilon}_1|. \quad (2.22)$$

Nota-se que $m' = m_b + \rho_{\text{ar}}C_mV_b = m_b + m_{fb} \approx m_b$. Assim, assume-se $m' = m_b$ para os cálculos. Anteriormente, foi apresentada a linearização τ para o termo $\frac{1}{2m}\rho C_d A |\dot{\Upsilon}_k|$ do cabo. Para o caso

da bolinha de isopor, essa constante é diferente e é denotada por τ' , resultando na equação final para Υ_1 :

$$\ddot{\Upsilon}_1 = b_1 (-\Upsilon_1 + \Upsilon_2) - \tau' \dot{\Upsilon}_1, \quad (2.23)$$

sendo

$$b_1 = \frac{m_b g}{m'l} = \frac{g}{l}. \quad (2.24)$$

Desta forma, a partir das Equações 2.23 e 2.16-2.20, pode-se definir o sistema linear em forma matricial

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -b_1 & b_1 & 0 & 0 & 0 & 0 & -\tau' & 0 & 0 & 0 & 0 & 0 \\ d_2 & -2d_2 & e_2 & 0 & 0 & 0 & 0 & -\tau & 0 & 0 & 0 & 0 \\ 0 & d_3 & -2d_3 & e_3 & 0 & 0 & 0 & 0 & -\tau & 0 & 0 & 0 \\ 0 & 0 & d_4 & -2d_4 & e_4 & 0 & 0 & 0 & 0 & -\tau & 0 & 0 \\ 0 & 0 & 0 & d_5 & -2d_5 & e_5 & 0 & 0 & 0 & 0 & -\tau & 0 \\ 0 & 0 & 0 & 0 & d_6 & -2d_6 & 0 & 0 & 0 & 0 & 0 & -\tau \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ e_6 \end{bmatrix} u \quad (2.25)$$

que pode ser representado concisamente como

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_{6 \times 6} & \mathbf{I}_{6 \times 6} \\ \mathbf{M}_{6 \times 6} & \mathbf{L}_{6 \times 6} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{0}_{11 \times 1} \\ e_6 \end{bmatrix} u. \quad (2.26)$$

Para o caso de uma discretização com N pontos, tem-se

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0}_{N \times N} & \mathbf{I}_{N \times N} \\ \mathbf{M}_{N \times N} & \mathbf{L}_{N \times N} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{0}_{2N-1 \times 1} \\ e_N \end{bmatrix} u, \quad (2.27)$$

$$y = \begin{bmatrix} 1 & \mathbf{0}_{1 \times 2N-1} \end{bmatrix} \mathbf{x}. \quad (2.28)$$

2.1.3 Estratégia de Redução da ordem do modelo

A maior parte da teoria clássica de controle lida com sistemas representados por um pequeno número de variáveis de estado. Portanto, uma forma de aplicar métodos clássicos de controle da literatura para sistemas de parâmetros distribuídos discretos é por meio de uma redução da ordem do modelo [6].

Tal redução do modelo será feita em duas etapas: primeiro, uma transformação modal é aplicada nas equações originais do espaço de estados, resultando em uma nova representação em variáveis modais. Nesta forma, o sistema pode ser visto como um conjunto de subsistemas

dissociados em paralelo, cuja influência na saída pode ser calculada individualmente. Então, os subsistemas com os maiores ganhos estáticos são escolhidos para criar um modelo de ordem reduzida [6].

Primeiro, deve-se obter os autovalores do espaço de estados do *riser*. Observa-se que eles são sempre distintos entre si, uma condição suficiente para a diagonalização da matriz do espaço de estados. Assim, calcula-se a matriz modal \mathbf{T} , cuja i -ésima coluna é o i -ésimo autovetor do sistema:

$$\mathbf{T} = (\mathbf{v}_1 \mid \mathbf{v}_2 \mid \dots \mid \mathbf{v}_{2N})_{1 \times 2N} \quad (2.29)$$

A matriz \mathbf{T} provavelmente tem valores complexos. Isso é um problema para a representação em espaço de estados e sua simulação. A solução é criar uma matriz $\tilde{\mathbf{T}}$ que tenha só números reais. Antes de explicar como criá-la, deve-se lembrar que os autovalores complexos sempre aparecem em pares conjugados, já que a matriz \mathbf{A} só tem valores reais. Quando a primeira coluna de um autovetor de um par complexo conjugado for encontrada, a coluna respectiva de $\tilde{\mathbf{T}}$ será sua parte real. A segunda coluna desse par complexo conjugado será a parte imaginária da coluna de \mathbf{T} .

A matriz $\tilde{\mathbf{T}}$ é utilizada para uma transformação de similaridade no sistema original:

$$\mathbf{A}_M = \tilde{\mathbf{T}}^{-1} \mathbf{A} \tilde{\mathbf{T}}, \quad (2.30)$$

$$\mathbf{x}_M = \tilde{\mathbf{T}}^{-1} \mathbf{x}, \quad (2.31)$$

$$\mathbf{B}_M = \tilde{\mathbf{T}}^{-1} \mathbf{B}, \text{ e} \quad (2.32)$$

$$\mathbf{C}_M = \mathbf{C} \tilde{\mathbf{T}}. \quad (2.33)$$

O sistema transformado, denotado pelo subscrito M , é mais adequado à análise. \mathbf{A}_M é uma matriz diagonal, com seus autovalores explícitos, e permitindo o desacoplamento do sistema original em N subsistemas formados por pares de autovalores reais ou complexo-conjugados. Nota-se que cada subsistema é de ordem 1 ou 2 dependendo se o autovetor é real ou um par complexo conjugado.

Neste estágio, procura-se determinar quais dos subsistemas são mais adequados para aproximar o modelo original por meio do ganho estático de cada um. Este método depende da predominância de uns poucos autovalores na resposta do sistema, já que altas frequências são muito atenuadas pelas forças hidrodinâmicas e pela suavidade da entrada.

Os subsistemas selecionados são combinados em um modelo reduzido

$$\dot{\mathbf{z}} = \mathbf{A}_R \mathbf{z} + \mathbf{B}_R u \quad (2.34)$$

$$y = \mathbf{C}_R \mathbf{z} + \mathbf{D}_R u \quad (2.35)$$

cuja ordem é escolhida considerando o custo-benefício entre a acurácia da dinâmica reduzida e a simplicidade da estrutura de controle exigida. Além disso, o sistema reduzido deve compensar o ganho estático perdido nos autovalores desconsiderados. Isto é feito por meio de uma matriz de transferência direta \mathbf{D}_R , que é a diferença dos ganhos dos sistemas original e reduzido:

$$\mathbf{D}_R = \mathbf{C} \mathbf{A}^{-1} \mathbf{B} - \mathbf{C}_R \mathbf{A}_R^{-1} \mathbf{B}_R. \quad (2.36)$$

A matriz de transferência direta \mathbf{D}_R introduz novas dinâmicas: uma saída não-nula que não leva em conta o atraso de propagação da entrada e um ganho em altas frequências. Conforme mostrado por Fortaleza [7], pode-se refinar o modelo reduzido introduzindo um atraso de entrada ϵ que minimiza a transferência direta e garante dinâmica nula para $t < \epsilon$:

$$\begin{aligned}\dot{\mathbf{z}} &= \mathbf{A}_R \mathbf{z} + \mathbf{B}_D u(t - \epsilon), \\ y &= \mathbf{C}_R \mathbf{z} + \mathbf{D}_D u(t - \epsilon);\end{aligned}\quad (2.37)$$

sendo

$$\mathbf{B}_D = \mathbf{A}_R \left(e^{\epsilon \mathbf{A}_R} \right) \mathbf{A}_R^{-1} \mathbf{B}_R \text{ e} \quad (2.38)$$

$$\mathbf{D}_D = \mathbf{C}_R \left(e^{\epsilon \mathbf{A}_R} - \mathbf{I} \right) \mathbf{A}_R^{-1} \mathbf{B}_R + \mathbf{D}_R. \quad (2.39)$$

O novo modelo reduzido (2.37) é tal que, para uma entrada degrau no instante t' , a saída mantém seu valor inicial enquanto $t < t' + \epsilon$. Para $t \geq t' + \epsilon$, ambos os modelos reduzidos produzem a mesma saída. O atraso ϵ pode ser visto como uma aproximação para o atraso natural de propagação da estrutura.

2.2 Controle

2.2.1 Técnicas Simples de Controle

Técnicas de controle simples devem ser introduzidas de forma a se compreender o objetivo deste trabalho, que é do posicionamento do *riser* por meio de controle em malha fechada. Primeiramente, apresenta-se o controle em malha aberta, cujo diagrama pode ser observado na Figura 2.1. A variável $r = r(t)$ é a referência do sistema. Neste tipo de controle, a saída não é realimentada na entrada. Desta forma, este tipo de controle não requer sensores, pois somente dá uma referência de entrada que a planta deve seguir. Caso haja erros para seguir a trajetória, eles não poderão ser compensados e é um controle mais recomendado quando o sistema é preciso e há pouca ou nenhuma perturbação. No entanto, este não é o caso do *riser*, pois o movimento das águas no leito oceânico perturba o tubo, causando erros na posição final desejada. O controle malha aberta foi anteriormente verificado na referência [2].

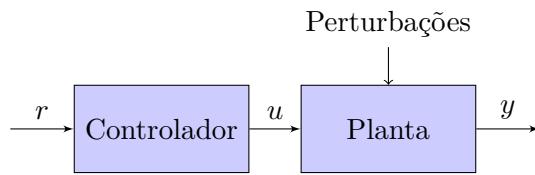


Figura 2.1: Malha aberta de controle

Uma forma de se compensar as perturbações do ambiente é realimentando a saída na entrada, calculando a diferença entre a referência e o valor medido. Assim, um valor de erro $e = e(t)$ é obtido e o sistema calcula o sinal u conforme o erro evolui. A Figura 2.2 mostra um esquema básico deste sistema, evidenciando a presença de uma malha fechada de controle.

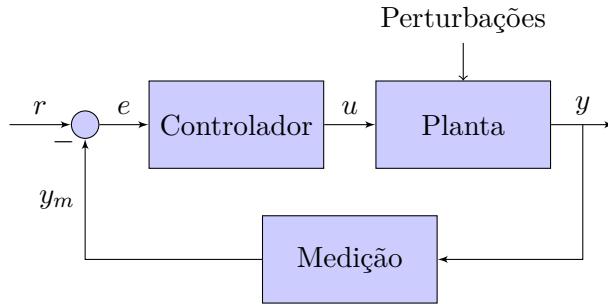


Figura 2.2: Malha fechada de controle

2.2.2 Controle Discreto no Espaço de Estados

Na seção 2.1, obtiveram-se equações do sistema contínuo no tempo, mas discretizado espacialmente. A discretização espacial transformou a equação diferencial parcial, um sistema de ordem infinita [6], em um sistema em espaço de estados finito. Mais informações sobre o espaço de estados estão disponíveis em [8] e [9].

Neste trabalho, uma câmera é utilizada para leitura da posição da bolinha, conforme é mostrado na seção 2.3. A câmera leva entre 20 e 50ms para terminar uma leitura de posição, dependendo das ferramentas utilizadas. Portanto, faz sentido trabalhar de forma discreta. O próximo CLP tem um limite de processamento que deve ser respeitado, daí quanto maior o tempo de amostragem melhor. O tempo de amostragem não pode ser arbitrariamente grande, já que é importante seguir o teorema de Nyquist [8] que requer que a frequência de amostragem seja pelo menos duas vezes a frequência da maior oscilação. Para o presente trabalho, $T_s = 0.1\text{s}$ já atende esse requerimento. o que economizaria processamento.

Definido o período de amostragem, o sistema contínuo da Equação 2.37 deve ser convertido para espaço discreto. A função `c2d` do MATLAB [10] faz a conversão para o espaço discreto, bastando fornecer o período de amostragem T_s . O modo padrão é a conversão utilizando um segurador de ordem zero [9]. Uma vez tendo o modelo discreto, deve-se fazer o projeto do controlador, alocando os pólos no plano z . Uma referência para este tipo de controle é [9].

O controle que será utilizado será realimentação de estados com um canal integral. O canal integral adiciona um estado a mais no sistema em malha fechada. Considerando um sistema definido pelas matrizes \mathbf{A} , \mathbf{B} e \mathbf{C} , o sistema aumentado passa ter matrizes $\hat{\mathbf{A}}$ e $\hat{\mathbf{B}}$. O projeto considera essas duas matrizes para a definição dos pólos.

Considerando a Figura 2.3, têm-se as matrizes aumentadas

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \text{e} \quad (2.40)$$

$$\hat{\mathbf{B}} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}. \quad (2.41)$$

$$(2.42)$$

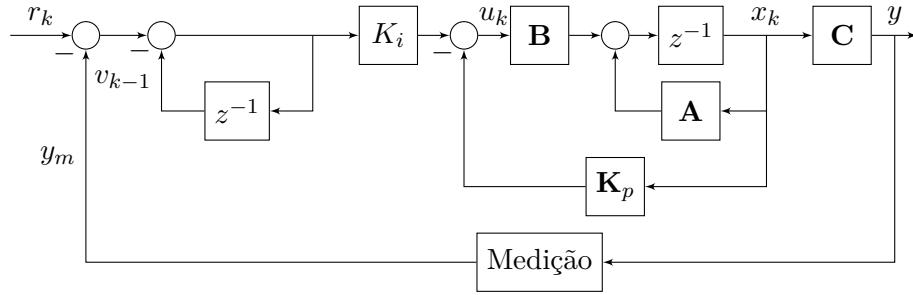


Figura 2.3: Malha fechada de controle, espaço de estados discreto

O projeto do controlador resume-se a definir o ganho $\hat{\mathbf{K}}$ seguindo a fórmula de Acker, explicada em [9],

$$\hat{\mathbf{K}} = [0, 0, 0, 0, 1] [\hat{\mathbf{B}}, \hat{\mathbf{A}}\hat{\mathbf{B}}, \hat{\mathbf{A}}^2\hat{\mathbf{B}}, \hat{\mathbf{A}}^3\hat{\mathbf{B}}, \hat{\mathbf{A}}^4\hat{\mathbf{B}}] \phi(\hat{\mathbf{A}}), \quad (2.43)$$

sendo $\phi(\hat{\mathbf{G}})$ dado por

$$\phi(\hat{\mathbf{A}}) = \hat{\mathbf{A}}^5 + \alpha_1 \hat{\mathbf{A}}^4 + \alpha_2 \hat{\mathbf{A}}^3 + \alpha_3 \hat{\mathbf{A}}^2 + \alpha_4 \hat{\mathbf{A}} + \alpha_5 \mathbf{I}, \quad (2.44)$$

no qual os coeficientes α_i são coeficientes do polinômio característico desejado, tal como

$$z^5 + \alpha_1 z^4 + \alpha_2 z^3 + \alpha_3 z^2 + \alpha_4 z + \alpha_5 = 0. \quad (2.45)$$

O MATLAB já tem um comando que permite obter $\hat{\mathbf{K}}$ a partir dos pólos do sistema:

```
1 Khat = acker(Ahat, Bhat, p);
```

As matrizes aumentadas apresentadas anteriormente consideram uma forma mais simples para obtenção do vetor de ganhos $\hat{\mathbf{K}}$, que deve ser então convertido para um vetor \mathbf{K} que pode ser utilizado no projeto, por meio de

$$\mathbf{K} = (\hat{\mathbf{K}} + [\mathbf{0} \ 1]) \begin{bmatrix} \mathbf{A} - \mathbf{I}_n & \mathbf{B} \\ \mathbf{C}\mathbf{A} & \mathbf{C}\mathbf{B} \end{bmatrix}^{-1}. \quad (2.46)$$

Na Equação 2.46, note que n é a ordem da matriz \mathbf{A} . O vetor \mathbf{K} contém os ganhos proporcionais, vetor \mathbf{K}_p , e o ganho integral, escalar K_i , que serão utilizados. Esses elementos podem ser extraídos de \mathbf{K} , sendo sua estrutura

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_p \\ K_i \end{bmatrix}. \quad (2.47)$$

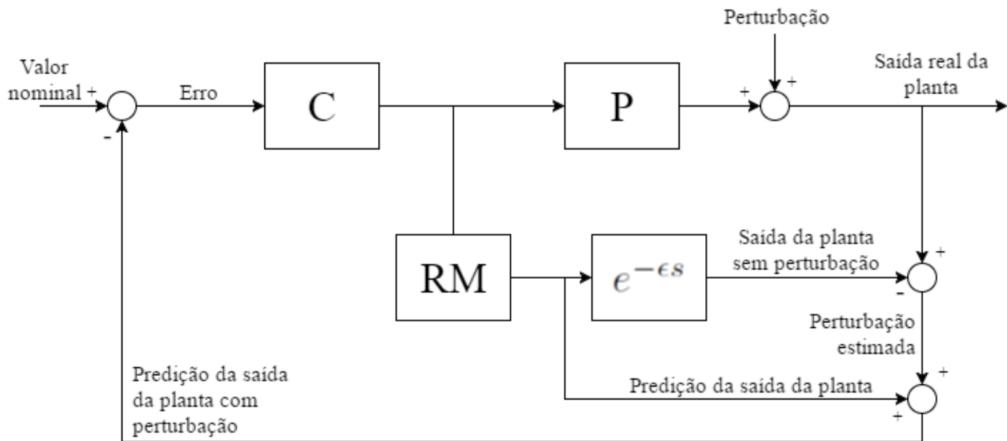
2.2.3 Predictor de Smith

Segundo Rafael [3], não é trivial desenvolver um controlador comum com realimentação de estados caso o sistema a ser considerado possua atraso, devido aos seguintes fatos:

- Os efeitos de uma perturbação externa demoram a ser detectados pelo controlador;
- As ações de controle demoram a surtir efeito nas variáveis controladas;
- O controlador, na realidade, tenta corrigir estados que já passaram.

De forma a se lidar com os problemas envolvidos em sistemas com atraso, é possível facilitar o projeto do controlador utilizando uma estrutura conhecida como preditor de Smith. Esse preditor pode ser visto como um compensador de atraso, tornando possível projetar um controlador para o sistema equivalente sem atraso, facilitando o desenvolvimento de uma estratégia de controle. A Figura 2.4 mostra uma estrutura básica do preditor de Smith, em que **P** é a planta, **C** é um controlador desenvolvido com realimentação, **RM** é o modelo obtido por redução modal, sem atraso e $e^{-\epsilon s}$ é o atraso puro.

Figura 2.4: Estrutura básica do preditor de Smith [3].



Nota-se que a saída do modelo sem atraso é uma estimativa da saída da planta desconsiderando-se efeitos de perturbação, ou seja, é uma predição da saída da planta sem perturbação após decorrido o atraso. Subtraindo-se essa saída sem atraso da saída real da planta, tem-se uma estimativa de perturbação, que somada à saída sem atraso, resulta em uma predição do modelo com perturbação; assim, a realimentação do preditor considera a perturbação sem ser influenciada pelo atraso.

A grande vantagem de se utilizar o preditor de Smith é que, dado o fato que o projeto pode ser feito sobre o modelo reduzido sem atraso, podem ser utilizadas técnicas elementares de controle em espaço de estados, apresentadas na Seção 2.2.1. Conhecendo-se o atraso ϵ , e sabendo que o modelo reduzido é um sistema SISO invariante no tempo e controlável, ele pode ser representado em forma canônica de controle. Assumindo-se as novas variáveis de estado nessa forma como o vetor $\bar{x} = [\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4]^T$ (sendo o sistema reduzido de ordem 4), pode-se considerar a saída plana do sistema F como

$$F = \bar{x}_1 \quad (2.48)$$

Como o sistema reduzido considerado é de ordem 4, deriva-se a saída F até a quarta ordem:

$$\begin{cases} \dot{F} &= \dot{\bar{x}}_1 = \bar{x}_2 \\ \ddot{F} &= \dot{\bar{x}}_2 = \bar{x}_3 \\ \dddot{F} &= \dot{\bar{x}}_3 = \bar{x}_4 \\ F^{(4)} &= \dot{\bar{x}}_4 = -a_4\bar{x}_1 - a_3\bar{x}_2 - a_2\bar{x}_3 - a_1\bar{x}_4 + u \end{cases} \quad (2.49)$$

Na Equação 2.49, as constantes a_i correspondem à última linha da matriz de estados do sistema sem atraso, na forma canônica controlável. Portanto, o sistema pode ser parametrizado em função de F e suas derivadas temporais:

$$\begin{cases} \bar{x}_1 &= F \\ \bar{x}_2 &= \dot{F} \\ \bar{x}_3 &= \ddot{F} \\ \bar{x}_4 &= \dddot{F} \\ u &= a_4F + a_3\dot{F} + a_2\ddot{F} + a_1\dddot{F} + F^{(4)} \end{cases} \quad (2.50)$$

Redefinindo-se a entrada do sistema considerado tomando-se $\nu = F^{(4)} = -a_4F - a_3\dot{F} - a_2\ddot{F} - a_1\dddot{F} + u$, pode-se reescrever o modelo reduzido utilizando-se a fórmula canônica de Brunovsky:

$$\begin{bmatrix} \dot{F} \\ \ddot{F} \\ \dddot{F} \\ F^{(4)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} F \\ \dot{F} \\ \ddot{F} \\ \dddot{F} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \nu \quad (2.51)$$

De acordo com Rafael [3], a entrada ν , em termos de acompanhamento da trajetória, também é dada por

$$\nu = \nu^* - \sum_{i=0}^{r-1} k_i e^{(i)} = \nu^* - \sum_{i=0}^{r-1} k_i (F^{(i)} - F^{*(i)}) \quad (2.52)$$

em que $e^{(i)}$ representa o erro de F e suas derivadas temporais. Pela Equação 2.52, a variável ν passa a ser entendida como um fator de correção para o controlador em malha fechada

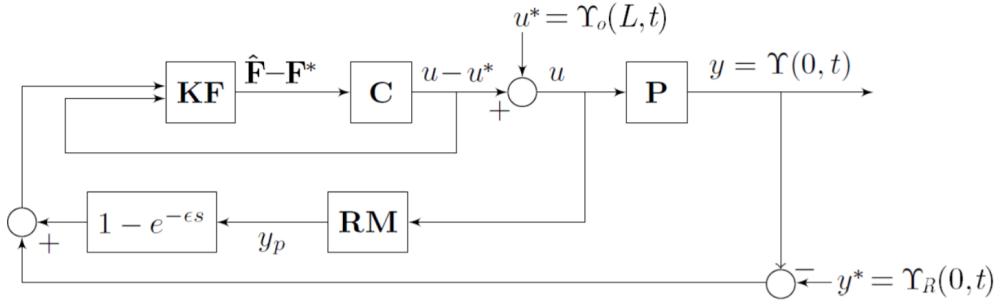
$$\nu = F^{(4)} - \sum_{i=0}^3 k_i (F^{(i)} - F^{*(i)}) \quad (2.53)$$

em que os ganhos k_i são escolhidos de forma que o polinômio característico do sistema possua raízes com parte real estritamente negativa (em termos discretos, tais raízes devem estar no interior do círculo unitário).

Com o desenvolvimento apresentado, pode-se montar uma nova estrutura para o preditor de Smith, conforme a Figura 2.5. Nessa estrutura, **P** é a planta, **C** é o controlador em malha fechada, **RM** é o modelo sem atraso e **KF** é um Filtro de Kalman, que será explicado adiante.

Com essa nova estrutura, dada a linearidade do sistema, o princípio de superposição é válido; portanto, os problemas de planejamento e acompanhamento de trajetória podem ser tratados de forma separada. O preditor leva em consideração as referências de trajetória de topo, $u^* = \Upsilon_0(L, t)$, e de trajetória de fundo, $y^* = \Upsilon_R(0, t)$. Desta forma, garante-se que a trajetória de fundo seja seguida, mesmo com a presença de perturbações.

Figura 2.5: Estrutura do preditor de Smith com filtro de Kalman e referências de topo e fundo [3].

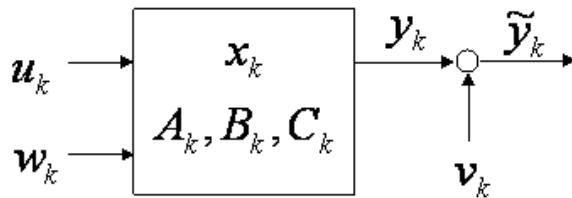


2.2.4 Filtro de Kalman

Uma parte importante do preditor de Smith é uma estrutura que deve ser capaz de estimar, dado que as variáveis de estado e demais sinais possuem ruídos associados, valores tanto para o processo quanto para o resultado. Assumindo-se que os estados do sistema sejam observáveis, é possível desenvolver, para o processo, uma estrutura denominada **Filtro de Kalman**, que leva o nome do seu criador, Rudolf E. Kalman.

O Filtro de Kalman, também conhecido como observador ótimo, é responsável por estimar, utilizando uma estrutura de previsão e correção, variáveis de estado não medidas de forma a se utilizar os resultados obtidos em uma dada estratégia de controle [4]. O Filtro de Kalman original é aplicado, principalmente, em sistemas discretos no tempo, como o que foi obtido para este projeto por meio da redução modal. A Figura 2.6 mostra um exemplo de processo no qual o filtro pode ser aplicado.

Figura 2.6: Exemplo de estrutura de sistema discreto no tempo [4].



O sistema representado pela Figura 2.6 pode então ser representado por equações lineares, como se segue:

$$x_k = \mathbf{A}_k x_{k-1} + \mathbf{B}_k u_k + w_k \quad (2.54)$$

$$y_k = \mathbf{C}_k x_k \quad (2.55)$$

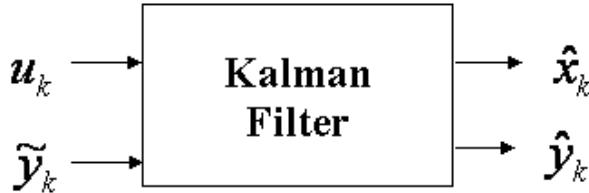
$$\tilde{y}_k = y_k + v_k \quad (2.56)$$

No sistema representado pelas Equações 2.54-2.56, as matrizes \mathbf{A}_k , \mathbf{B}_k e \mathbf{C}_k são as matrizes próprias do espaço de estados do sistema discreto; x_k é o vetor de variáveis de estado, y_k é a

saída do processo e \tilde{y}_k é uma medida da referida saída. As variáveis w_k e v_k são ruídos associados respectivamente ao processo e à medida. Suas covariâncias podem ser representadas por \mathbf{Q}_k para o processo e \mathbf{R}_k para a medida.

Dada a estrutura do sistema, o objetivo do uso do filtro de Kalman é estimar os dados do processo e a medida das saídas, dadas as entradas e as medições das saídas. A Figura 2.7 mostra um esquema de uso do filtro, em que \hat{x}_k e \hat{y}_k são, respectivamente, as estimativas das variáveis de estado e da saída.

Figura 2.7: Exemplo de estrutura de filtro de Kalman [4].



O Filtro de Kalman é um algoritmo de dois passos: predição, em que o estado mais recente e a estimativa do erro de covariância são projetados de forma a se computar o estado atual das variáveis de estado, e correção, onde o estado predito pelo primeiro passo é corrigido incorporando-se a medida mais recente do processo para se gerar uma nova estimativa do estado [4]. Matematicamente, a predição é dada por

$$\hat{x}_k^- = \mathbf{A}_k \hat{x}_{k-1} + \mathbf{B}_k u_k \quad (2.57)$$

$$\hat{\mathbf{P}}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_k \quad (2.58)$$

enquanto que a correção pode ser descrita por

$$\mathbf{K}_k = \hat{\mathbf{P}}_k^- \mathbf{C}_k^T \left(\mathbf{C}_k \hat{\mathbf{P}}_k^- \mathbf{C}_k^T + \mathbf{R}_k \right)^{-1} \quad (2.59)$$

$$\hat{x}_k = \hat{x}_k^- + \mathbf{K}_k (\tilde{y}_k - \mathbf{C}_k \hat{x}_k^-) \quad (2.60)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \hat{\mathbf{P}}_k^- \quad (2.61)$$

Nas Equações 2.57-2.61, \mathbf{P}_k é uma estimativa da covariância de erro da medição, enquanto \mathbf{K}_k é denominado ganho de Kalman. Após os dois passos do algoritmo, \hat{x}_k e \mathbf{P}_k são preservados continuamente, de forma a serem utilizados na parte de predição no próximo instante de tempo [4].

No presente projeto, o filtro de Kalman é utilizado em conjunção com o controlador. Porém, é admitido um caso especial, em que as matrizes passadas para o filtro são discretizadas a partir do sistema contínuo escrito em forma canônica controlável. Nesse caso, \hat{y}_k é diretamente dependente de uma das variáveis de estado, tornando o conjunto controlador-filtro mais simples de ser implementado, uma vez que é tomada a integral de uma variável de estado, apenas.

2.3 Bancada

A bancada da ponte rolante presente no Laboratório de Automação e Controle está esquematizada na Figura 2.8. Serão detalhados os componentes desta bancada, de modo a se entender o papel de cada um deles. Observe que no esquemático falta a câmera, que é um sensor que fica de frente para a bancada.

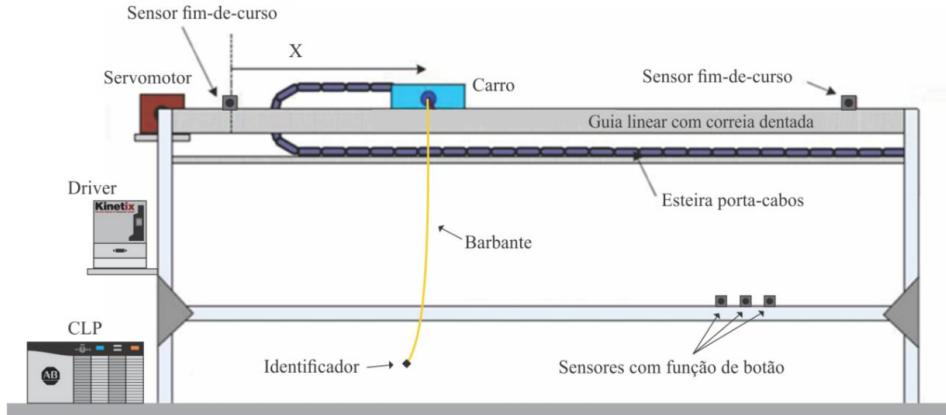


Figura 2.8: Esquemático da Bancada Utilizada para o Experimento [2]

2.3.1 Controlador Lógico-Programável

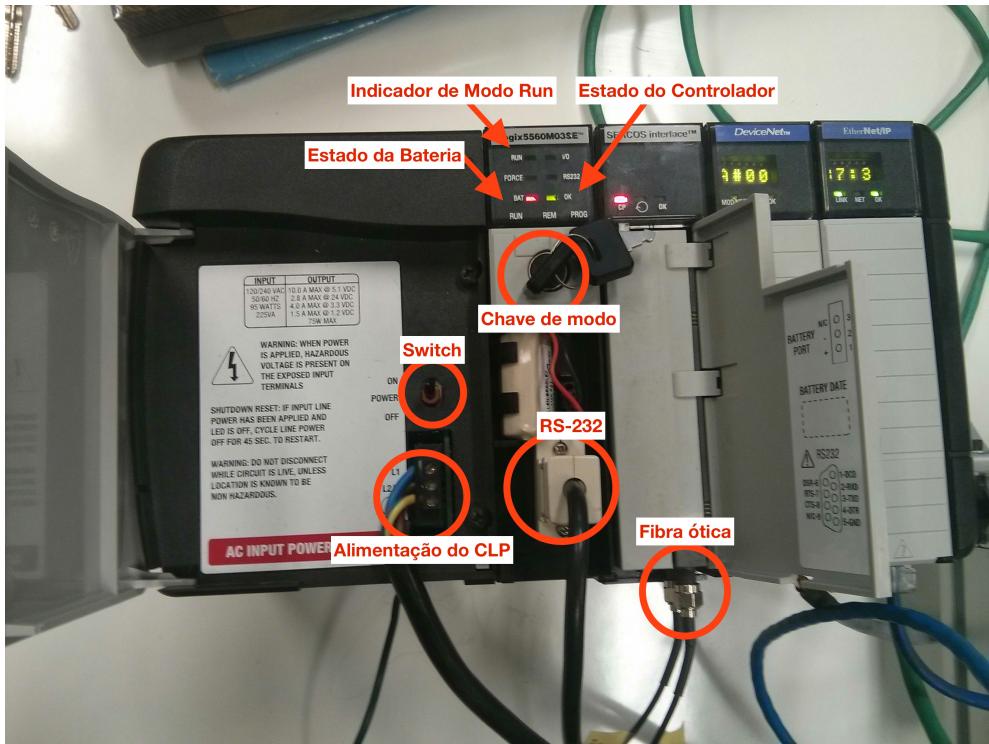


Figura 2.9: CLP com identificação de elementos

O controlador lógico-programável (CLP - ver Figura 2.9) é a espinha dorsal da bancada. Ele é responsável por executar os comandos de controle sobre todos os elementos que estão conectados a ele. A programação do CLP é realizada pelo computador; porém, uma vez feito o *download* do programa ao CLP, a execução ocorre independentemente do computador, contanto que o CLP esteja em modo de execução.

O CLP utilizado é fabricado pela *Allen Bradley*, modelo Logix5560M03SE. Tal modelo possui memória lógica e de dados de 750 KiB, e memória de *I/O* de 494 KiB. Há quatro módulos no *chassis* do controlador:

- O próprio controlador;
- *SERCOS Interface*;
- DeviceNET;
- EtherNet/IP.

Cada um desses módulos será apresentado posteriormente com mais detalhes. Além dos módulos, o controlador ainda possui um *switch liga/desliga* presente no *chassis*. No módulo Logix, há uma chave responsável por alterar o modo de funcionamento do mesmo. As posições possíveis dessa chave são:

- RUN (*Run mode*);
- REM (*Remote mode*);
- PROG (*Program mode*);

Na prática, o modo REM se divide em dois modos: REM RUN e REM PROG. A maneira de se diferenciar os dois é observar, no módulo Logix, o estado do LED indicador de modo RUN quando a chave estiver na posição REM.

No modo RUN, o controlador apenas roda o programa presente em sua memória; não há qualquer comunicação remota. No modo PROG, o controlador não roda nenhum programa; ele apenas pode receber um novo código. Nos modos REM, há a comunicação com o computador, permitindo verificar valores de variáveis de interesse e alterar, se necessário, o programa a ser rodado pelo controlador. O programa presente no controlador, em modo REM, só roda o código se estiver no modo REM RUN; se for necessário atualizar o programa, o modo deve ser o REM PROG.

Para fins de se utilizar melhor o CLP, será utilizado sempre o modo REM e suas derivações. O modo REM RUM permite a execução do programa; já o modo REM PROG permite a atualização do programa dentro da memória do CLP.

2.3.2 SERCOS Interface

Sercos é um barramento digital de automação que interconecta controladores, *drives*, dispositivos de entrada/saída e atuadores para máquinas e sistemas controlados numericamente. Foi projetado para comunicação serial de alta velocidade de dados em sistemas de tempo real por meio de fibra ótica (Sercos I & II) ou um cabo Ethernet Industrial (Sercos III). Sercos é um padrão internacional [11].

Num sistema Sercos, todas as malhas que contém servomotores são normalmente fechadas no *drive*. Isto reduz a carga computacional no CLP, permitindo-o sincronizar mais eixos do que conseguiria caso contrário. Além disso, fechar a malha dos servomotores com o *driver* ajuda a reduzir o efeito do atraso de transporte entre o controle de movimento e o *driver* [11].

Nesta bancada, o CLP deve se comunicar com o servomotor (MPL-A310F-SJ22AA) através do *drive* Kinetix (2094-AC05MP5), de forma a movimentar o carrinho segundo uma trajetória planejada ou segundo uma lei de controle em malha fechada executando no CLP. Essa comunicação se dá por um par de fibras óticas full-duplex, conforme se observa na Figura 2.9, o que caracteriza uma rede Sercos I ou II.

2.3.3 Line Interface Module 2094-AL09

Este módulo não apareceu no esquemático da Figura 2.8, mas ele tem a função essencial de interfacear a rede trifásica com o servo *drive*, permitindo o acionamento do motor. Na Figura 2.10, se observa que há três conjuntos de disjuntores nesse módulo: CB1 – liga ou desliga a rede trifásica do *drive* –, CB2 – fornece tensão monofásica ao servo *drive* – e CB3 – liga as duas fontes de tensão DC de 24V –, responsáveis pelas entradas e saídas digitais do módulo e alimentação do freio do motor [2].

2.3.4 Drive Kinetix 6000 da Allen Bradley

Por meio da Interface Sercos, o CLP comanda o drive que então fornece potência ao motor. O drive controla o servomotor por meio de pulsos PWM. A Figura 2.11 apresenta um Drive Kinetix 6000 da Allen Bradley similar ao utilizado no laboratório. Mais detalhes sobre o funcionamento do drive estão disponíveis em [2] e [12].

2.3.5 Servomotor

Um servomotor é um atuador rotatório que permite controle preciso da posição angular. O motor consiste de um motor acoplado a um sensor para a realimentação de posição/velocidade. Também é necessário um drive servo para completar o sistema. O *drive* usa o sensor de realimentação para controlar precisamente a posição angular do motor, ou seja, é uma operação em malha fechada. Assim, usando servomotores em malha fechada, tem-se uma alternativa de alto desempenho aos motores de passo e de indução [13].

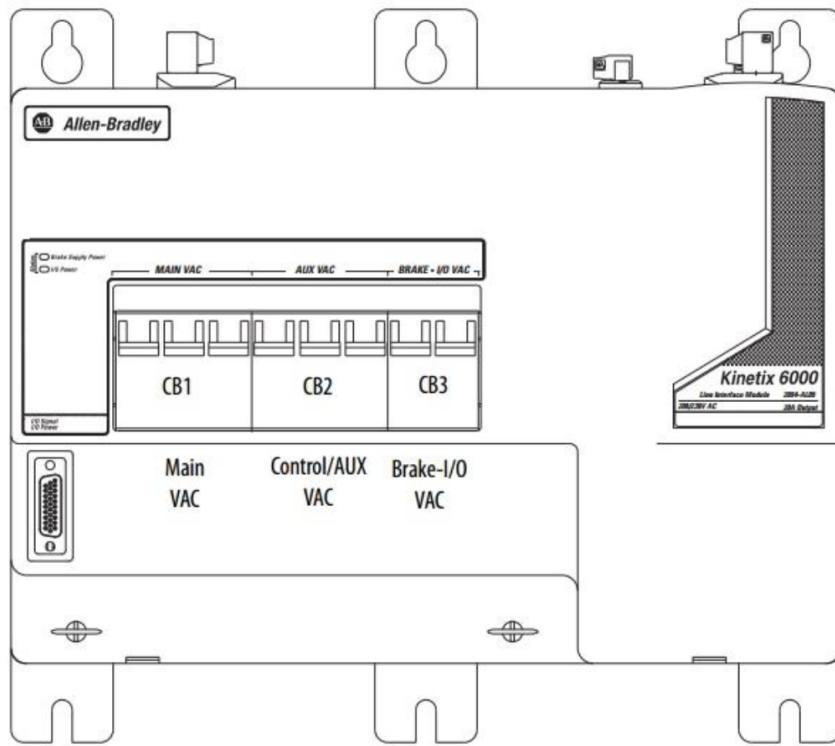


Figura 2.10: Line Interface Module modelo 2094-AL09 da Allen Bradley [2]



Figura 2.11: Drive Kinetix 6000 da Allen Bradley

O servomotor MPL-A310F-SJ22AA da *Allen-BRADLEY* foi utilizado e está representado na Figura 2.12. O mesmo é composto por um motor indutivo de tensão nominal $230V_{ac}$ e um encoder do tipo StegmanHiperface, que mede posição de forma absoluta e velocidade de forma incremental [2].



Figura 2.12: Servomotor modelo MPL-A310F-SJ22AA

2.3.6 Sensores indutivos

Os sensores indutivos são elementos detectores de presença, particularmente de objetos metálicos. Eles funcionam através da variação de campo magnético ocasionada pela presença do objeto a ser identificado. Tal variação de campo magnético provoca uma variação de corrente dentro do sensor, alterando seu estado.

Na presente bancada, há 6 sensores indutivos da família 871TM, similares ao da Figura 2.13, fabricados pela *Allen-Bradley*. Eles são alimentados com tensão de 24 V, que está dentro dos limites padrão. São sensores feitos de aço, adaptados a ambientes industriais.



Figura 2.13: Sensor indutivo 871T-R8B18 [2]

A importância desses sensores é enorme. O motivo é que a câmera às vezes falha para obter a posição atual e tem um limite de frequência devido ao processamento interno que ela tem de realizar. Com os sensores indutivos, tem-se um processamento rápido para identificar se o motor está na região próxima dos limites. Assim, a rotina de segurança que trava o motor depende diretamente desses sensores indutivos, como mostra o Anexo I.3.2.

2.3.7 Câmera PresencePlus

A câmera PresencePlus P4 GEO da Banner Engineering foi utilizada no projeto - veja Figura 2.14. Esta câmera é um sensor robusto utilizado em ambientes industriais com fácil utilização. Sua programação é feita em software próprio da Banner e é visual.

Dentre as capacidades da câmera, nota-se que ela pode capturar até 24 imagens por segundo. No entanto, além da captura das imagens há o processamento das imagens, que pode ser feito na própria câmera ou em um módulo externo da própria Banner, que também consome certo tempo. O programa utilizado no projeto está disponível no Anexo I.4.1.

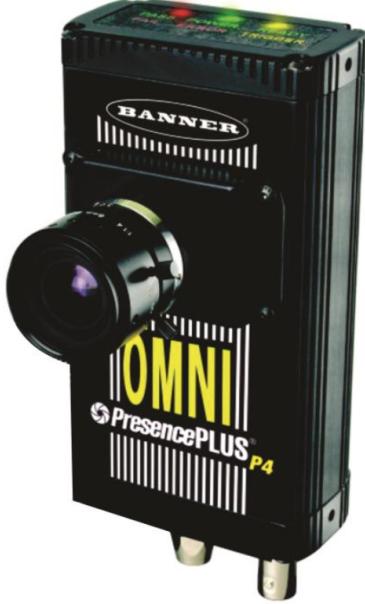


Figura 2.14: Câmera da Banner Engineering utilizada no projeto [2]

2.4 Redes Utilizadas

2.4.1 DeviceNET

DeviceNET é um protocolo de baixo nível da camada de aplicação, voltado a ambientes industriais [14]. Tal rede suporta comunicação entre dispositivos de baixo nível, como sensores e atuadores, e dispositivos de alto nível, como o computador e o CLP [15]. Permite uma rápida configuração entre dispositivos a *byte*, suportando tanto dispositivos analógicos quanto digitais [2]. Permite velocidades de transmissão de até 500 kbps, sendo bem mais lenta que uma rede Ethernet.

No experimento, a rede DeviceNET é utilizada para a conexão entre o CLP e os sensores indutivos presentes na planta [2]. Para fins de detecção de fim-de-curso, será utilizado o modo digital dos sensores, uma vez que apenas é necessário verificar se o carrinho está na área de detecção do sensor. O módulo responsável pelo gerenciamento da rede DeviceNET é o 1756-DNB.

2.4.2 Ethernet/IP

A rede Ethernet/IP, introduzida em meados de 2001 pela ODVA (*Open DeviceNet Vendor Association*) [16], é um tipo de rede Ethernet voltada ao ambiente industrial, seguindo o CIP, assim como o DeviceNET [17]. É uma rede robusta, organizada segundo o modelo OSI de 7 camadas, que permite conexão com dispositivos conectados a redes Ethernet padrão; permite a passagem de dados via pacotes TCP ou UDP; é indicada em aplicações que exigem uma transferência rápida e confiável de dados, principalmente em aplicações de tempo real) [18].

No presente experimento, a rede Ethernet/IP é utilizada para receber dados de inspeção da câmera, notadamente a posição horizontal da bola de isopor presa ao barbante. Além disso, ela é

utilizada na transferência de programas entre o computador e o CLP (através do módulo Ethernet 1756-ENBT/A), uma vez que ela provê uma comunicação mais rápida do que o RS-232.

2.4.3 OPC - *OLE for Process Control*

A rede OPC, cujo primeira versão foi lançada em 1996, é uma rede voltada ao ambiente industrial. Essa rede é responsável, em termos gerais, pela troca confiável de dados entre o CLP, computadores e outros dispositivos. O objetivo do uso de tal rede é abstrair o protocolo de rede utilizado no CLP (seja ModBus, Profibus, Ethernet/IP, etc.) em uma interface padrão de forma que sistemas HMI/SCADA possam se comunicar com um "intermediário" que converte as operações de dados de OPC em operações específicas de dispositivo e vice-versa [19].

Uma das características da rede OPC é permitir a troca de dados entre o CLP e o computador por meio de um sistema *server/client*. No presente projeto, o *server* é disponibilizado pelo *software* RSLinx, enquanto que o cliente é um programa desenvolvido externamente ao CLP. Para esse programa, foi escolhida a linguagem Python; a razão da escolha de tal linguagem foi, além da existência de módulos programados por terceiros para lidar com a rede OPC, o fato de Python ter outros módulos que facilitam a implementação de controladores e ser razoavelmente rápido (por exemplo, Python é mais rápido que o *software* MATLAB).

Para se programar o *client* da rede OPC em Python, foi utilizado o módulo OpenOPC. Esse módulo é bem simples de se utilizar; as funções de trocas de dados OPC são abstraídas de modo a utilizar elementos simples da linguagem, como o uso de dicionários. Além de trabalhar em Windows, Linux e MAC OS X, o módulo OpenOPC faz com que seja possível utilizar elementos de programação funcional, reduzindo o tamanho do programa resultado [20].

2.5 Programação do CLP

2.5.1 Visão geral

Programação, em termos gerais, é aplicada na resolução de problemas. Em particular, a programação de CLPs busca resolver, no âmbito industrial, problemas relacionados à automação de processos. Essa resolução de problemas segue uma metodologia, de forma a se direcionar o projeto. Antes de se proceder à programação, faz-se necessário seguir os seguintes passos [21]:

1. Descrição do problema;
2. Detalhamentos e melhoria do processo;
3. Especificação dos atuadores e sensores da planta;
4. Elaboração do algoritmo;
5. Representação gráfica do algoritmo, quando aplicável;

6. Esquema funcional, quando aplicável;
7. Seleção dos módulos do controlador;
8. Programação, utilizando linguagens suportadas.

Ao se programar o CLP, deve-se ter em conta que o controlador executa sempre três ciclos [21]:

1. *Scan* de Entrada: Ciclo em que o controlador recebe todos os dados de seus módulos de entrada;
2. *Scan* do Programa: Ciclo em que o controlador processa as entradas recebidas e gera as saídas;
3. *Scan* de Saída: Ciclo em que o controlador envia os dados de saída para os módulos de saída.

Para o presente experimento, entre as várias linguagens disponíveis para CLPs, foram selecionadas duas: uma linguagem gráfica, o *ladder*, e uma linguagem textual, o **texto estruturado**. Ambas foram selecionadas por serem linguagens muito utilizadas, normalmente rápidas e que ocupam pouca memória, ao contrário de linguagens como SFC (*Sequential Flow Chart*) e FBD (*Function Block Diagram*), que também poderiam ser utilizadas.

2.5.2 Linguagem *ladder*

2.5.2.1 Introdução ao *ladder*

A linguagem *ladder* é uma linguagem de programação gráfica, e uma das primeiras a ser utilizada na programação de CLPs. Seu nome vem do inglês *ladder*, que significa escada; nome dado em razão dos programas, ao serem feitos, assumirem a forma de uma escada, e lidos de cima para baixo (movimento de descida). Essa linguagem foi estruturada de forma a ter uma simbologia semelhante à de um diagrama de conexão de relés, que eram utilizados em indústrias antes do CLP.

Todo programa *ladder* possui duas linhas verticais e, entre elas, uma ou mais linhas horizontais nas quais são especificados os comportamentos do programa. A linguagem possui várias instruções, em sua maioria comuns entre diferentes tipos de CLP. A Tabela 2.3 mostra as principais instruções utilizadas em *ladder*, utilizando a sintaxe presente no software RSLogix5000.

2.5.2.2 Instruções específicas

No presente trabalho, a presença do servomotor exige uma lógica e entradas que não são booleanas, ou seja, que não podem ser tratadas pelos operadores descritos na Tabela 2.3. Portanto, faz-se necessário o uso de novas instruções, diretamente relacionadas com o movimento do atuador. Tais instruções são conhecidas, dentro do RSLogix5000, como *Motion Control Instructions*, ou

Tabela 2.3: Principais instruções *ladder*

Desenho da Instrução	Nome da Instrução	Descrição
-(-)	OTE	Atualiza variável booleana de acordo com a condição da linha
-(L)-	OTL	Atribui valor verdadeiro à variável booleana se a linha for alimentada
-(U)-	OTU	Atribui valor falso à variável booleana se a linha for alimentada
-[]-	XIC	Examina se a variável possui valor verdadeiro
-[/]-	XIO	Examina se a variável possui valor falso

instruções de controle de movimento. A Tabela 2.4 mostra as instruções desse tipo utilizadas no experimento.

Tabela 2.4: Principais instruções de controle de movimento em *ladder*

Nome da Instrução	Sigla	Função
<i>Motion Servo On</i>	MSO	Inicializa o servomotor, travando a esteira.
<i>Motion Axis Jog</i>	MAJ	Envia comandos de velocidade e sentido de rotação ao motor, executando sua movimentação.
<i>Motion Axis Stop</i>	MAS	Interrompe a movimentação do motor; a esteira permanece travada.
<i>Motion Servo Off</i>	MSF	Desativa o servomotor, destravando a esteira e permitindo a movimentação manual do carrinho.

As funções descritas pela Tabela 2.4 lidam com outros tipos de variáveis, como números reais, para a velocidade do motor, e até mesmo estruturas que o *software RSLogix* cria para armazenar informações sobre os eixos e o controle feito pelas instruções. Tais parâmetros podem ser alterados dentro dos blocos das funções. A Figura 2.15 mostra um exemplo de tais instruções.



Figura 2.15: Exemplo de programa *ladder* com instruções de movimentação.

2.5.3 Texto Estruturado

2.5.3.1 Visão geral do texto estruturado

Além de *ladder*, que é uma linguagem gráfica, o CLP utilizado no experimento também suporta uma linguagem puramente textual, o texto estruturado. Essa linguagem é uma linguagem sequencial, que lembra muito linguagens como BASIC, C ou Pascal. Cada comando é executado em sequência, salvo em caso de desvios (estruturas do tipo *if-then-else*) ou repetições (*loops*, estruturas do tipo *while*).

Em relação ao *ladder*, o texto estruturado possui vantagens e desvantagens. A principal vantagem é o fato do texto estruturado se assemelhar às linguagens de programação mais utilizadas; isso torna tal linguagem de fácil aprendizado. Além disso, por ser texto, caso seja necessário editar muitos parâmetros cuja criação seja de fácil automação, um programa auxiliar escrito em linguagens de programação genéricas pode simplificar a geração do texto estruturado. Porém, uma desvantagem do texto em relação ao *ladder* reside nas funções de movimentação, uma vez que, ao contrário do *ladder*, o texto estruturado não dá uma indicação clara associando um valor e a variável que ele representa, além da ordem dos parâmetros. Isso torna o uso de funções como MAJ (vide Tabela 2.4) um pouco mais complicado em texto estruturado.

2.5.3.2 Instruções de movimentação do motor para o texto estruturado

Assim como o *ladder*, o texto estruturado permite o uso das funções descritas na Tabela 2.4. A sintaxe de uso dessas funções lembra muito a sintaxe da linguagem C, com o nome da função e uma lista de parâmetros válidos separados por vírgulas. A seguir, um exemplo de uso:

```
1  /**
2  * MS0: ativa o servo cujo eixo eh descrito
3  * por drive_axis; informacoes de controle
4  * sao gravadas em MS0_1
5  */
6  MS0(drive_axis,MS0_1);
7  /* Atribui o valor 0.0 ao primeiro elemento do array speed */
8  speed[0] := 0.0;
9  /* Atribui 1 para dataInitialized */
10 dataInitialized := 1;
```

Nota-se que o código apresentado, a menos da operação de atribuição, possui uma sintaxe muito semelhante à de C. A função MS0 acima recebe dois parâmetros, assim como em *ladder*; porém, a única indicação de como os parâmetros se comportam é a ordem com que eles são colocados; no caso de MS0, o 1º parâmetro é o eixo que será inicializado, e o 2º parâmetro é uma *tag* que guarda as informações de controle executadas por MS0.

2.6 Parâmetros para gerar trajetórias

Fabrício et al [6] desenvolveram um método para gerar trajetórias *offline* (controle malha aberta) assim como *online* (controle malha fechada). Para se utilizar do método, é necessário ter parâmetros do sistema. Se fosse necessário simular ou controlar o sistema real, utilizariam-se os dados da Tabela 2.5, conforme cita Rédyton [2].

Tabela 2.5: Dados para simulação em escala real [2]

Dados do Riser	
Diâmetro externo	0.55m
Diâmetro interno	0.5m
Comprimento	2000m
Módulo de Elasticidade	200GPa
Densidade	7860kg/m ³
Dados do fluido (Água)	
Densidade do fluido	1000kg/m ³
Viscosidade dinâmica	$10^3 \text{ Pa} \cdot \text{s}$

Tabela 2.6: Dados para simulação em escala laboratorial

Dados da Massa na Ponta - Isopor	
Diâmetro Externo	30.6mm
Densidade	10kg/m ³
Dados do Riser (Barbante)	
Diâmetro externo	2mm
Diâmetro interno	0mm
Comprimento	82cm
Módulo de Elasticidade	2.1MPa
Densidade	191kg/m ³
Dados do fluido (Ar)	
Densidade do fluido	1.2754kg/m ³
Viscosidade dinâmica	$17.2 \cdot 10^6 \text{ Pa} \cdot \text{s}$

Como é importante se validar o sistema, mas nem sempre isso é possível em escala real devido ao altíssimo custo e aos riscos envolvidos, é necessário um conjunto equivalente de parâmetros tais que permita validar o sistema em escala laboratorial. O *riser* é representado por um barbante e a água é trocada pelo ar. Uma massa na ponta foi adicionada e os dados resultantes estão na Tabela 2.6.

Capítulo 3

Resultados

Este capítulo apresenta os procedimentos experimentais realizados e seus resultados.

3.1 Câmera

Os problemas com a câmera se resumiram a configuração da rede, calibração e programação, sendo o mais difícil esse último.

3.1.1 Configuração da Rede

A câmera estava com um *firmware* antigo e foi necessário obter o arquivo 2014R1B PresencePLUS Firmware¹ que é o programa de atualização da *Banner Engineering* para esta câmera. Bastou executá-lo no computador, estando a câmera conectada ao *switch* do laboratório assim como o computador estava, ambos por cabos Ethernet. Antes desta atualização, o *software* da câmera não permitia selecionar a opção Ethernet/IP de forma a permitir utilizar o módulo Ethernet/IP do CLP.

Uma vez atualizado o *firmware*, foi obtido o arquivo EDS² da câmera que foi então integrado ao *software* da Rockwell no computador por meio do programa *EDS Hardware Installation Tool* da própria Rockwell, parte do RSLinx.

Após os procedimentos anteriores, adiciona-se a câmera como um módulo genérico usando o *software* RSLogix, conforme instruções da Banner Engineering [22]. Dentro das configurações realizadas no RSLogix Para se efetuar a configuração da câmera, é necessário utilizar o RSLinx para verificar se a mesma está conectada ao computador e o *software* da PresencePlus, para identificar a câmera pelo endereço IP, uma vez que a rede utilizada é Ethernet/IP. A Figura 3.1 mostra a janela do RSLinx com a câmera reconhecida.

¹ Atualização de Firmware da Câmera - http://info.bannerengineering.com/_dav/cs/idcplg?IdcService=GET_FILE&RevisionSelectionMethod=LatestReleased&dDocName=B_4170042. Acesso em 29/11/2015.

² Arquivo EDS disponível em <http://www.bannerengineering.com/en-US/products/sub/78#ui-tabs-37>. Acesso em 29/11/2015.

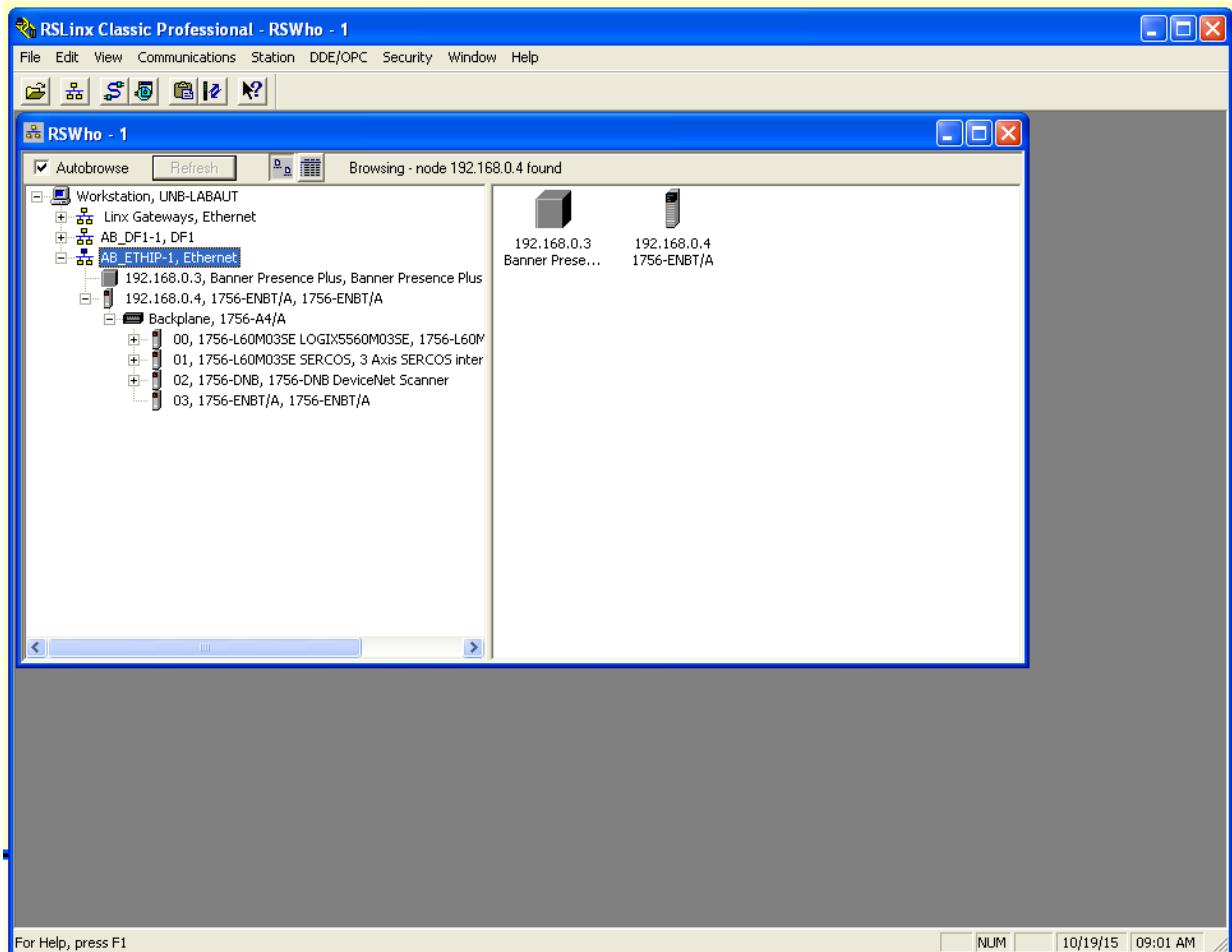


Figura 3.1: RSLinx com câmera reconhecida

3.1.2 Calibração

A câmera permite fazer medidas de distâncias em *pixels*. De forma a se converter essa distância para milímetros, uma barra de alumínio com marcas e tamanho conhecido é utilizada. É importante primeiro calibrar o sistema, para se saber se há deformação de pixels significante ao longo da distância de interesse, nomeadamente o tamanho da barra de alumínio sendo utilizado, cerca de 532mm. No PresencePlus P4 GEO 1.3, um programa é feito, com imagem de referência conforme Figura 3.2, que usa várias ferramentas de detecção de borda (ferramenta **Locate**) para identificar as posições de cada uma das marcas pretas da barra. Seis marcas foram feitas e a Tabela 3.1 apresenta os resultados para cada seção. A distância entre duas marcas é de 10cm, com exceção da distância entre P0 e PEND que é o comprimento total da barra.

O maior desvio da quantidade de milímetros por *pixel* das seções em relação à da barra inteira é de aproximadamente 4.93%. Há algumas imprecisões na maneira como os traços foram desenhados e é possível que o erro seja menor.

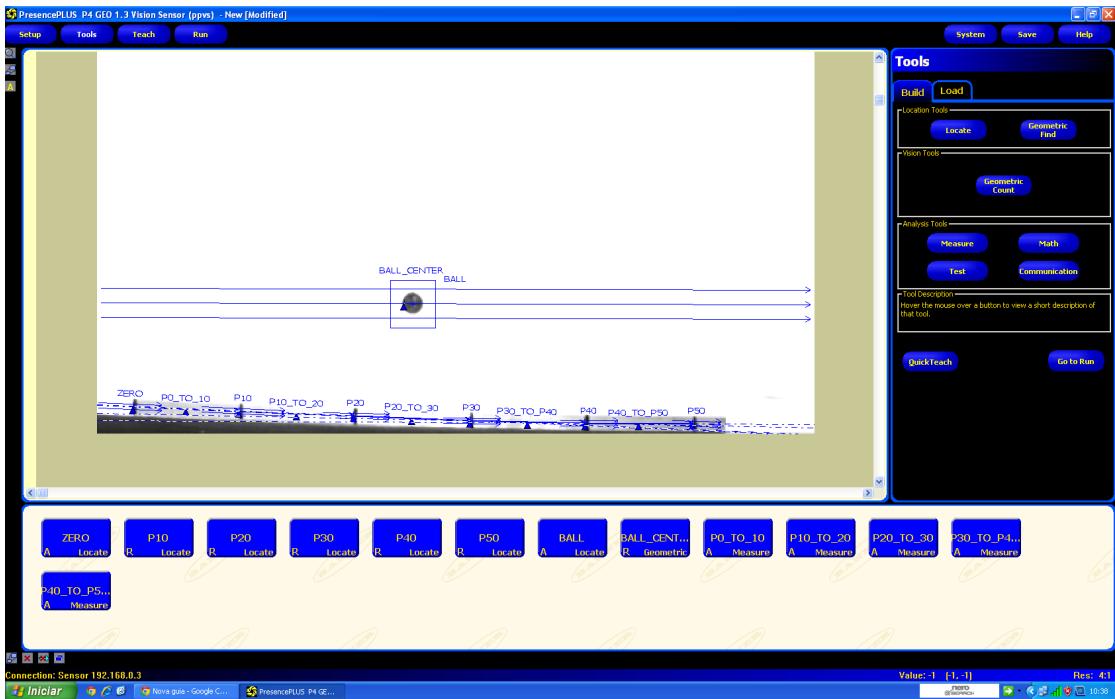


Figura 3.2: Programa PresencePLUS para calibração da câmera

Tabela 3.1: Relações mm/px para diferentes seções da barra de alumínio

Seção 1	Seção 2	Distância (px)	mm/px
P0	P10	160	0.625
P10	P20	173	0.578
P20	P30	176	0.568
P30	P40	173	0.578
P40	P50	163	0.613
P0	PEND	893	0.596

3.1.3 Programação

A programação da câmera inicia obtendo uma imagem de referência e adicionando-se ferramentas de detecção de pontos de interesse, como mostra a Figura 3.2. A ferramenta **Locate** é responsável por detectar as bordas da bolinha, e a ferramenta **Geometric** detecta o centro da mesma. Também é possível adicionar ferramentas que fazem operações matemáticas (ferramenta **Math**), executam medições (ferramenta **Measure**), assim como as que enviam dados pela rede (ferramenta **Communication**), o que é essencial para comunicar com o CLP.

3.2 Calibração do Servomotor

O RSLogix tem o bloco **MAJ – Motion Axis Jog** – que permite alterar a velocidade do motor enquanto ele se movimenta. No entanto, o bloco espera que a entrada seja do tipo [u/s] ao invés de

alguma unidade no SI tal como [mm/s]. Devido a isso, foi necessária uma calibração do sistema. Nela, anotou-se a posição inicial x_0 e a posição final x_f , ambas em milímetros, e definia-se um tempo Δt no qual o carrinho se movimentaria a uma velocidade v em [u/s]. Daí, calculava-se a velocidade em [mm/s] com esses dados e tirou a média de alguns ensaios para se obter o valor de uma unidade, que é aproximadamente 71.32mm. Os dados de calibração estão na Tabela 3.2.

Tabela 3.2: Dados de calibração do servomotor, média obtida é de 71.32 mm/unidade

x_0 - [mm]	x_f - [mm]	Δt - [s]	Velocidade - [u/s]	Velocidade - [mm/s]	mm/u
2	71.8	2	0.5	34.9	69.8
6	76.1	2	0.5	35.05	70.1
6	188	5	0.5	36.4	72.8
6	185	2.5	1	71.6	71.6
6	77	10	0.1	7.1	71
6	296.5	20	0.2	14.525	72.625

3.3 Modelo no Espaço de Estados

A partir da teoria apresentada na subseção 2.1.3, desenvolveram-se algumas rotinas em linguagem Julia [23] para obter as matrizes reduzidas. A ordem da matriz de saída é um parâmetro. A Seção I.1 dos anexos apresenta o código completo.

As matrizes dos modelo reduzidos são apresentadas nas Equações 3.1 e 3.2. Antes da compensação do atraso, o sistema é representado pela Equação 3.3. Quando se analisa a resposta ao degrau em malha aberta para este caso, Figura 3.3, nota-se que o atraso é cerca de 0.3s. Para ser preciso, $\epsilon = 0.313$ s. Utilizando esse atraso, a transferência direta se tornaria aproximadamente zero. No entanto, esses modelos no espaço contínuo serão discretizados com período $T_s = 0.1$ s. Daí, o ϵ mais próximo é 0.3s. Apesar de próximo, esse atraso não reduz muito a transferência direta, como se pode observar a diferença entre D_R , Equação 3.1, e D_D , Equação 3.2.

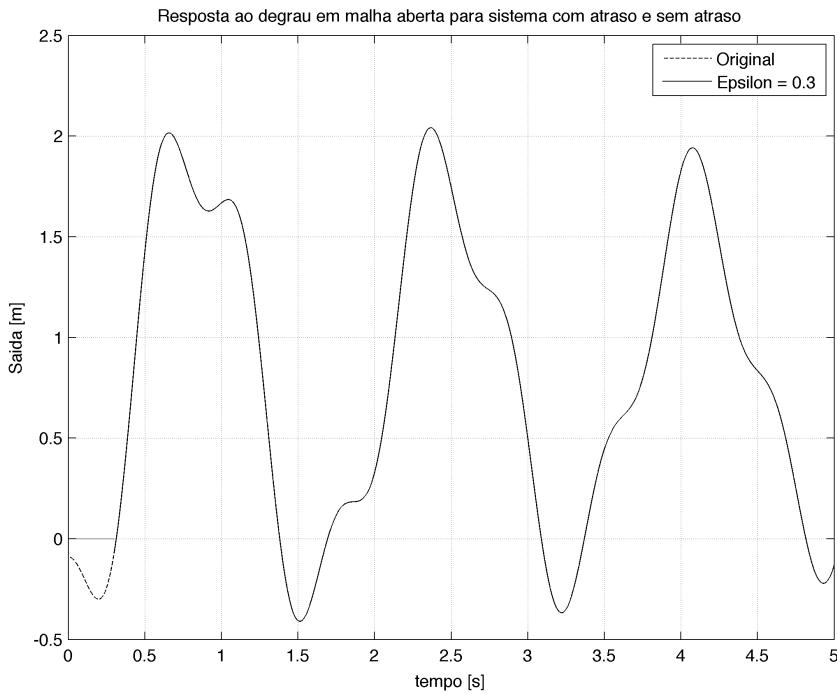
$$\left. \begin{aligned} \mathbf{A}_R &= \left[\begin{array}{cccc} -0.0881 & -3.8389 & 0 & 0 \\ 3.8389 & -0.0881 & 0 & 0 \\ 0 & 0 & -0.1061 & -10.8145 \\ 0 & 0 & 10.8145 & -0.1061 \end{array} \right] \\ \mathbf{B}_R &= 10^3 [0.3313, 0.0066, 1.3549, 0.0163]^T \\ \mathbf{C}_R &= [-0.0003, 0.0148, 0.0001, -0.0029] \\ D_R &= -0.0906 \end{aligned} \right\} \quad (3.1)$$

$$\left. \begin{aligned} \mathbf{B}_D &= 10^3 [0.1255, 0.2974, -1.3039, -0.1504]^T \\ D_D &= -0.0685 \end{aligned} \right\} \quad (3.2)$$

$$\left. \begin{array}{l} \dot{\mathbf{z}} = \mathbf{A}_R \mathbf{z} + \mathbf{B}_R u(t) \\ y = \mathbf{C}_R \mathbf{z} + \mathbf{D}_R u(t) \end{array} \right\} \quad (3.3)$$

$$\left. \begin{array}{l} \dot{\mathbf{z}} = \mathbf{A}_R \mathbf{z} + \mathbf{B}_D u(t - \epsilon) \\ y = \mathbf{C}_R \mathbf{z} + \mathbf{D}_D u(t - \epsilon) \end{array} \right\} \quad (3.4)$$

Figura 3.3: Resposta ao degrau para modelos reduzidos com atraso e sem atraso

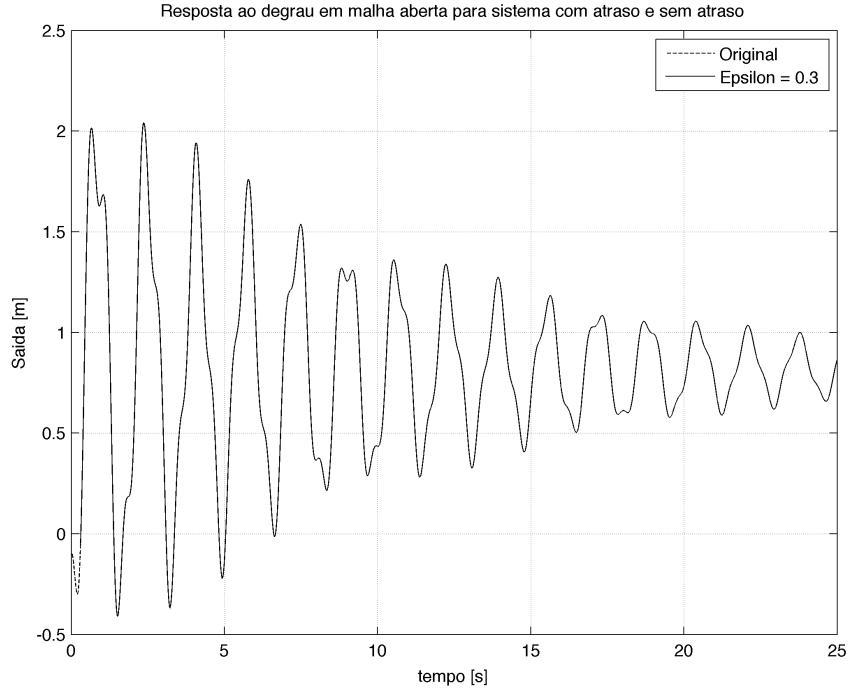


O sistema em malha aberta oscila muito, mas é estável. Observa-se que a resposta decai conforme o tempo passa, conforme Figura 3.4. O objetivo do controle é reduzir ao máximo essas oscilações, tendo uma trajetória o mais suave possível de um ponto inicial a um ponto final.

3.4 Malha aberta

O planejamento de trajetória em malha aberta e malha fechada para o *riser* que está sendo utilizado neste trabalho foi desenvolvido por Fabrício et al [6], conforme mencionado anteriormente. Um programa em MATLAB foi escrito e gerou trajetórias de excursão pré-definida. Rédyton [2] testou o sistema para uma excursão de cerca de 1m, que é maior que o tamanho do barbante (82cm) conforme apresentado na Tabela 2.6. No entanto, seu trabalho não utilizou uma massa de isopor na ponta, daí existirá uma diferença entre os resultados. Observe que, em malha aberta, deixou-se o ar condicionado da sala desligado, pois isto seria uma perturbação.

Figura 3.4: Resposta ao degrau para modelos reduzidos com atraso e sem atraso, 25 segundos



3.4.1 Excursão de 30cm

O primeiro teste em malha aberta testou a trajetória de posição da Figura 3.5. Conforme Rédyton [2] mencionou em seu trabalho, utilizar os módulos de posição é mais lento do que utilizar módulos de velocidade para seguir essa trajetória. Desta forma, optou-se por usar diferenças finitas para derivar essa trajetória, resultando na Figura 3.6.

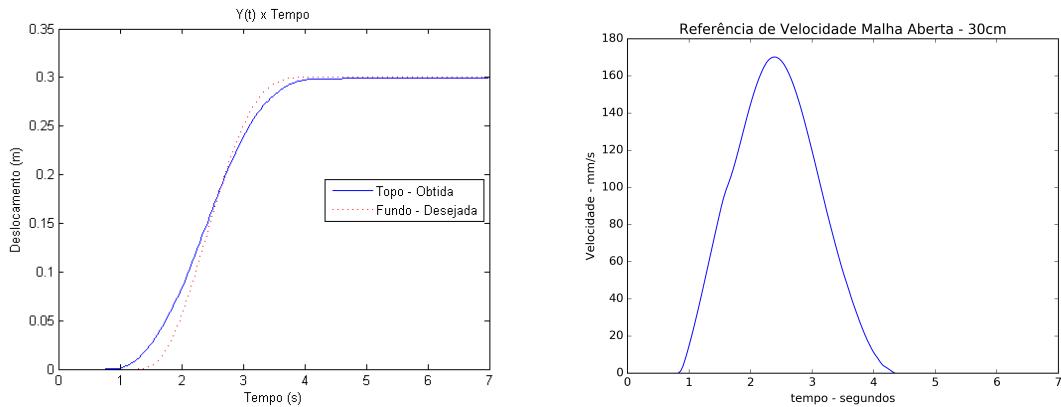


Figura 3.5: Referência de Posição para Ex- cursão de 30cm

Figura 3.6: Referência de Velocidade para Excursão de 30cm

Com um período de cerca de 41ms, uma tarefa eventual executava no CLP, aplicando um certo valor de velocidade ao carrinho. A câmera leu os valores de posição cada vez que a tarefa iniciava e o software RSLogix desenhou os dados, que podem ser observados na Figura 3.7. Observa-se que

a posição evolui de forma bem suave. Para se realizar uma comparação, calculou-se o tempo que o carrinho se move na trajetória (Δt , tempo no qual a velocidade não é nula) e calculou-se $\bar{v} = \frac{\Delta x}{\Delta t}$. No caso presente, $\Delta x = 30\text{cm}$ e $\Delta t \approx 3\text{s}$ (veja Figura 3.6), resultando em $\bar{v} = 100\text{mm/s}$. A Figura 3.8 apresenta esses resultados e observa-se várias oscilações quando o carrinho para, diferente do caso anterior.

Vídeos foram criados para cada um destes casos e estão disponíveis no YouTube®, tanto para a trajetória modelada³ quanto para a trajetória com velocidade constante⁴.

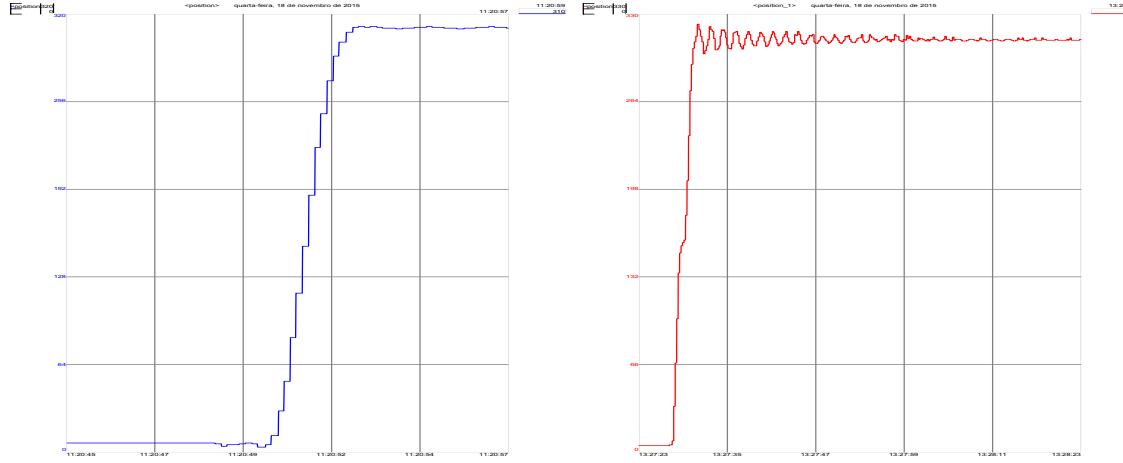


Figura 3.7: Resultado com Velocidade Modelada para Excursão de 30cm

Figura 3.8: Resultado com Velocidade Constante para Excursão de 30cm

3.4.2 Excursão de 20cm

Na prática, o diâmetro do *riser* e de sua massa é bem menor que o comprimento do mesmo. Daí, é importante testar uma menor excursão. Além disso, esse teste terá menor tempo de movimentação e resultará numa velocidade média maior que o caso anterior. Desta forma, maiores oscilações são esperadas e o controle se torna mais difícil.

A trajetória de posição obtida é apresentada na Figura 3.9 e a sua derivada é apresentada na Figura 3.10. O período da tarefa eventual foi escolhido em 100ms. O resultado experimental⁵ apresentou mais oscilações que o caso anterior, mas está bem controlado, conforme se vê na Figura 3.11. O resultado com velocidade constante⁶ está na Figura 3.12 e nota-se que o resultado ficou bem pior. A velocidade média foi cerca de $\bar{v} \approx \frac{200\text{mm}}{1.5\text{s}} = 133.3\text{mm/s}$.

³Controle Malha Aberta Modelado, 30cm - <https://youtu.be/lKajz6LyauE>. Acesso em 29/11/2015.

⁴Controle Malha Aberta a Velocidade Constante, 30cm - <https://youtu.be/tB0TsmBcfVg>. Acesso em 29/11/2015.

⁵Controle Malha Aberta Modelado, 20cm - https://youtu.be/wg1Wq_6VRSG. Acesso em 29/11/2015.

⁶Controle Malha Aberta a Velocidade Constante, 20cm - <https://youtu.be/Ges2-eYy69k>. Acesso em 29/11/2015.

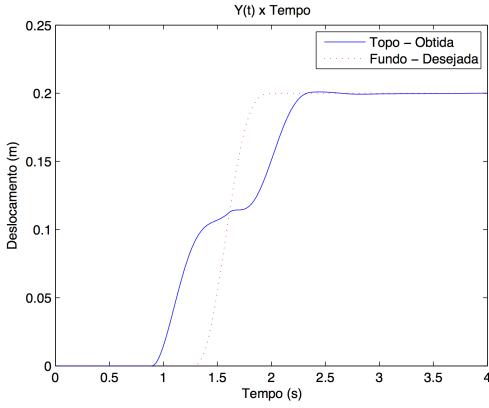


Figura 3.9: Referência de Posição para Exercício de 20cm

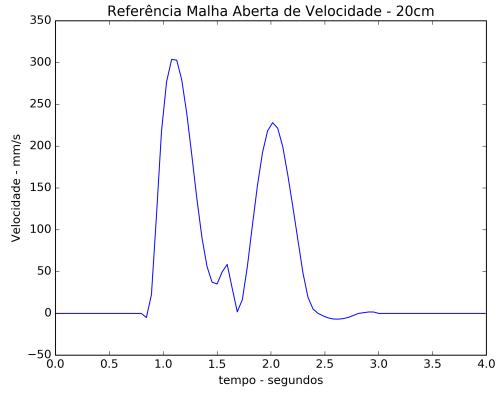


Figura 3.10: Referência de Velocidade para Excursão de 20cm

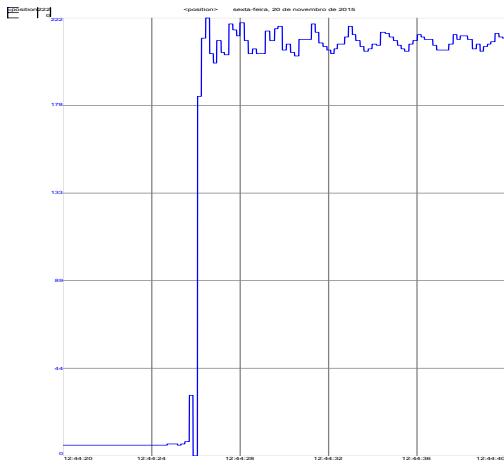


Figura 3.11: Resultado com Velocidade Modulada para Excursão de 20cm

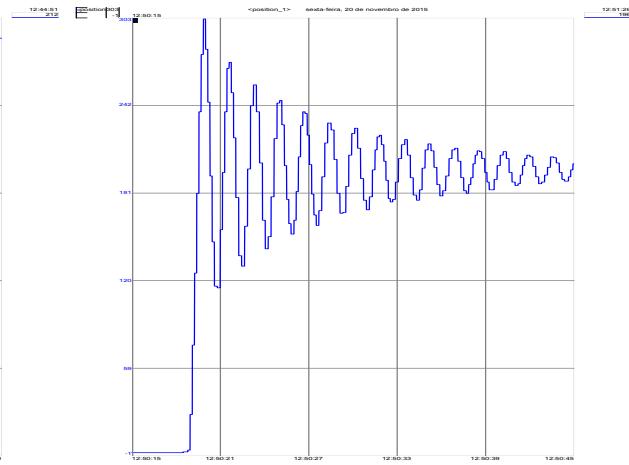


Figura 3.12: Resultado com Velocidade Constante para Excursão de 20cm

3.5 Malha fechada

O controle em malha fechada deve ser capaz de compensar por perturbações, pois o sensor identifica o resultado da atuação e realimenta a informação no sistema. Neste trabalho, conseguiu-se configurar a câmera adequadamente e faz-se um teste em malha fechada de um controlador bem simples: um controlador proporcional.

3.5.1 Controlador P

O controlador proporcional utilizado tem o esquema conforme Figura 3.13. A referência de posição $r = r(t)$ é a entrada do sistema e a saída medida pela câmera, y_m , é subtraída da referência para resultar no erro e , cuja unidade é dada em milímetros. A entrada da planta, v , é dada em $[u/s]$, conforme discutido anteriormente na subseção 3.2. Desta forma, a unidade de K_p é $[\frac{u}{mm \cdot s}]$.

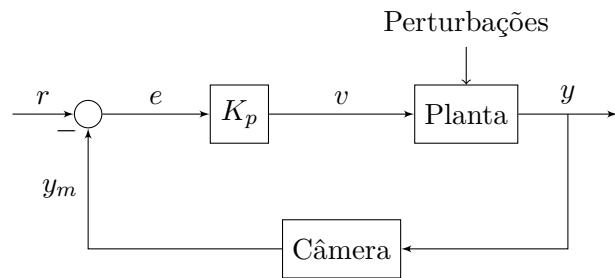


Figura 3.13: Malha fechada de controle

Valores de K_p foram escolhidos empiricamente e o resultado para $K_p = 0.0025$ ⁷ está na Figura 3.14 enquanto para $K_p = 0.0050$ ⁸ está na Figura 3.15.

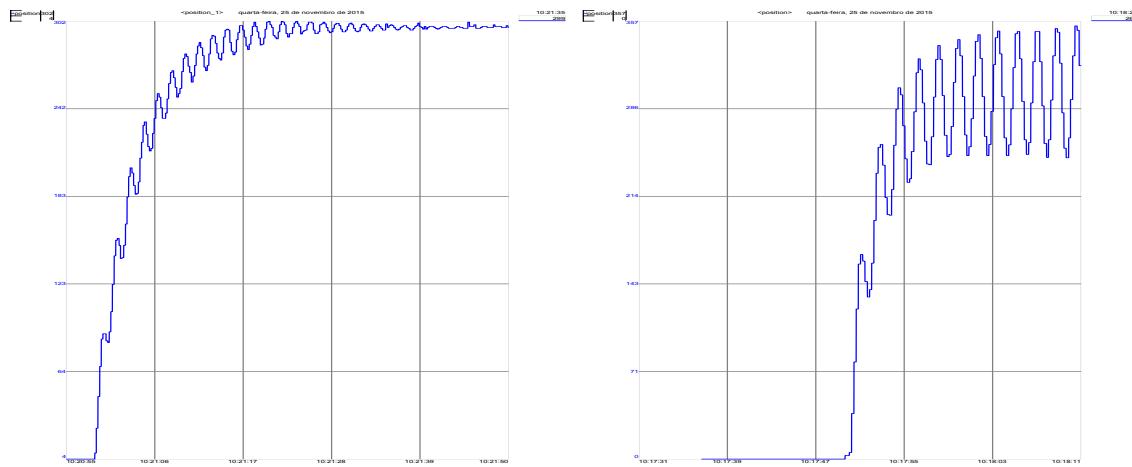


Figura 3.14: Resultado com Malha Fechada Proporcional e $K_p = 0.0025 \frac{u}{mm \cdot s}$

Figura 3.15: Resultado com Malha Fechada Proporcional e $K_p = 0.0050 \frac{u}{mm \cdot s}$

O período de amostragem utilizado foi de 100ms e o ar condicionado estava ligado nas duas situações apresentadas e nota-se que o primeiro resultado é bem melhor que o segundo, tendo menos oscilações. No entanto, este resultado está aquém do obtido em malha aberta.

⁷Controle Malha Fechada com $K_p = 0.0025$ - <https://youtu.be/mmT1ZwFBJ4s>. Acesso em 29/11/2015.

⁸Controle Malha Fechada com $K_p = 0.0050$ - <https://youtu.be/sHh3yvBrek4>. Acesso em 29/11/2015.

Capítulo 4

Conclusões

No início do trabalho, não havia nenhum conhecimento prévio dos autores sobre programação em tempo real utilizando CLPs, nem sobre programação CLP em geral. Nenhum conhecimento prévio da bancada existia também. No estágio atual, pode-se dizer que há relevante facilidade em se trabalhar com a bancada, apesar de não se ter entrado em detalhes de como funciona o servomotor, *drive* e outros componentes.

O foco principal era fechar a malha, o que foi possível, conforme se observa na seção 3.5. Agora, pode-se testar mais técnicas de controle em malha fechada com a câmera, o que não foi feito antes nesta bancada. Apesar do resultado de um controlador P não ser ideal, é o verdadeiro início da validação de controladores para *risers* com esta bancada, pois na realidade *ROVs* com câmeras são utilizados, o que justifica também ser utilizada uma câmera aqui.

4.1 Perspectivas Futuras

Pra o Trabalho de Graduação 2, objetiva-se utilizar uma técnica de controle mais rebuscada, utilizando-se o projeto de Fabrício et al [6]. Nele, o modelo é levado em consideração através de redução modal, que o deixa mais simples em questão de custo computacional. Após verificar-se se a mencionada técnica de controle funciona bem mesmo com perturbações, pretende-se analisar outras literaturas na área de controle preditivo e tentar projetar um controlador novo que funcione melhor que o anterior, em algum aspecto.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] FORTALEZA, E.; ALBUQUERQUE, D.; YAMAMOTO, M. An investigation about the trajectory control during the subsea equipment installation using cable. In: *Volume 1: Offshore Technology*. ASME International, 2012. Disponível em: <<http://dx.doi.org/10.1115/OMAE2012-83798>>.
- [2] SOUSA, R. B. *Implementação de Controle de Riser, Validação Experimental e Análise Através de Processamento de Imagens*. Brasília, DF: Faculdade de Tecnologia, Universidade de Brasília, 2015. Trabalho de Graduação em Engenharia de Controle e Automação.
- [3] SIMÕES, R. D. P. *Sistema de Posicionamento Dinâmico para Instalações Submarinas*. Brasília, DF: Universidade de Brasília. Faculdade de Tecnologia, Departamento de Engenharia Mecânica, 2016. Dissertação de Mestrado.
- [4] GODDARD CONSULTING. *An Introduction to the Kalman Filter*. Disponível em: <http://www.goddardconsulting.ca/kalman-filter.html>. Acesso em: 28/06/2016.
- [5] PETROBRAS. *Pré-Sal: Exploração e Produção de Petróleo*. Disponível em: <http://www.petrobras.com.br/pt/nossas-atividades/areas-de-atuacao/exploracao-e-producao-de-petroleo-e-gas/pre-sal/>. Acesso em: 27/11/2015.
- [6] MONTEIROS, F. R.; FILHO, J. O. de A. L.; FORTALEZA, E. Modal reduction based tracking control for installation of subsea equipments*. *IFAC-PapersOnLine*, v. 48, n. 6, p. 15 – 20, 2015. ISSN 2405-8963. 2nd {IFAC} Workshop on Automatic Control in Offshore Oil and Gas Production {OOGP} 2015 Florianópolis, Brazil, 27–29 May 2015. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S2405896315008708>>.
- [7] FORTALEZA, E. *Active Control Applied to Offshore Structures: Positioning and Attenuation of Vortex Induced Vibrations*. Tese (Doutorado) — École Nationale Supérieure des Mines de Paris, 2009. Ph.D. thesis.
- [8] OGATA, K. *Modern Control Engineering*. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2010. ISBN 0-13-615673-8.
- [9] OGATA, K. *Discrete-Time Control Systems*. 2nd. ed. [S.l.]: Pearson, 1995. ISBN 978-0130342812.

- [10] MATHWORKS. *Convert model from continuous to discrete time - MATLAB c2d*. Disponível em: <http://www.mathworks.com/help/control/ref/c2d.html>. Acesso em: 26/06/2016.
- [11] SERCOS INTERNATIONAL. *Introduction to Sercos*. Disponível em: <http://www.sercos.com/technology/index.htm>. Acesso em: 28/11/2015.
- [12] ALLEN-BRADLEY. *User Manual Kinetix 6000 Multi-axis Servo Drives*. July 2015. Disponível em: http://literature.rockwellautomation.com/idc/groups/literature/documents/um/2094-um001_en-p.pdf. Rockwell Automation Publication 2094-UM001I-EN-P. Acesso em: 28/11/2015.
- [13] KOLLMORGEN. *Servomotors*. Disponível em: <http://www.kollmorgen.com/en-us/products/motors/servo/servomotors/>. Acesso em: 28/11/2015.
- [14] RTA AUTOMATION. *DeviceNet™Unplugged – A View “Under the Hood” for End Users*. Disponível em: <http://www.rtautomation.com/technologies/devicenet/>. Acesso em: 30/11/2015.
- [15] ROCKWELL AUTOMATION. *DeviceNet Network*. Disponível em: <http://ab.rockwellautomation.com/Networks-and-Communications/DeviceNet-Network>. Acesso em: 30/11/2015.
- [16] ODVA. *Ethernet/IP - Quick Start for Vendors Handbook*. Disponível em: https://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00213R0_EtherNetIP_Developers_Guide.pdf. Acesso em: 30/11/2015.
- [17] RINALDI, J. *The 6.5 Things You Must Know about EtherNet/IP*. Disponível em: <http://www.rtautomation.com/technologies/ethernetip/>. Acesso em: 30/11/2015.
- [18] ROCKWELL AUTOMATION. *A Single IT-Friendly Network for Enterprise and Industrial Applications*. Disponível em: <http://www.rockwellautomation.com/global/products-technologies/integrated-architecture/ethernet-ip.page>. Acesso em: 30/11/2015.
- [19] OPC FOUNDATION. *What is OPC?* Disponível em: <https://opcfoundation.org/about/what-is-opc/>. Acesso em: 26/06/2016.
- [20] SOURCEFORGE. *OpenOPC for Python*. Disponível em: <http://openopc.sourceforge.net/>. Acesso em: 26/06/2016.
- [21] EPUSP. *Automação elétrica de processos industriais*. Disponível em: http://lara.unb.br/~gaborges/disciplinas/ca/teoria_final_1_prova.pdf. Acesso em: 29/11/2015.
- [22] BANNER ENGINEERING. *PresencePLUS Industrial Ethernet User’s Guide Vol 2*. September 2015. Disponível em: <http://www.bannerengineering.com/en-US/products/sub/78#ui-tabs-37>. Acesso em: 29/11/2015.

[23] NUMFOCUS FOUNDATION. *The Julia Language*. Disponível em: <http://julialang.org/>. Acesso em: 29/06/2016.

ANEXOS

I. PROGRAMAS UTILIZADOS

I.1 Redução modal

codes/ModalReduction.jl

```
1 module ModalReduction
2   export generateA, generateB, generateC
3   export generateABC, getABC_M, getABCD_R
4   export manuscript_p48, simulation
5   export generateMATLABSimulationScript
6
7
8   function simulation(n, original=false, nout=4)
9     A, B, C = generateABC(n)
10    A_M, B_M, C_M = getABC_M(n, A, B, C)
11    A_R, B_R, C_R, D_R = getABCD_R(n, A_M, B_M, C_M, nout)
12    if original
13      generateMATLABSimulationScriptToCompare("simulacaoN" * string(n) *
14        "Compare.m", A, B, C, A_R, B_R, C_R, D_R)
15      generateMATLABSimulationScript(n, "simulacaoN" * string(n) *
16        "Reduced.m", A_R, B_R, C_R, D_R)
17    else
18      generateMATLABSimulationScript(n, "simulacaoN" * string(n) *
19        "Reduced.m", A_R, B_R, C_R, D_R)
20    end
21  end
22
23 #Gera A, B, C to sistema completo
24 function generateABC(n)
25   tau = 0.2426      # tau do barbante (1/s) para excursão de 30cm
26   tauL = 0.1133     # tau da bolinha (1/s) para excursão de 30cm
27   ms = 0.0006       # massa linear do barbante (kg/m)
28   mb = 0.00015      # massa da bolinha (kg)
29   g = 9.80665       # aceleração da gravidade (m/s^2)
30   L = 0.82          # Comprimento total do barbante (m)
31   l = L/n           # distância entre dois pontos de discretização (m)
32   T0 = mb*g         # Tração no ponto 0 (logo acima da bolinha) - considerando
33   peso da bolinha (N)
34
35   b = zeros(n)
36   c = g/(2l)
37   d = zeros(n)
38   e = zeros(n)
```

```

36 b[1] = g/l
37 for k = 2:n
38     b[k] = (T0 + ms*g*(k-1)*l)/(ms*l^2)
39     d[k] = b[k] - c
40     e[k] = b[k] + c
41 end
42
43 A = generateA(n, b, d, e, tau, taul)
44 B = generateB(n,e[n])
45 C = generateC(n)
46
47 return A, B, C
48 end
49
50 function generateA(n, b, d, e, tau, taul)
51 M = zeros(n,n)
52 #Primeira linha de M
53 M[1,1] = -b[1]
54 M[1,2] = b[1]
55
56 #Linhas 2 ate n-1
57 for i = 2:n-1
58     M[i,i-1] = d[i]
59     M[i,i] = -2*b[i]
60     M[i,i+1] = e[i]
61 end
62
63 #Linha n
64 M[n,n-1] = d[n]
65 M[n,n] = -2*b[n]
66
67 L = eye(n)
68 for i = 1:n
69     L[i,i] = i == 1 ? -taul : -tau
70 end
71
72 #Concatenar matrizes, gerando matriz (2n,2n)
73 A = [[zeros(n,n) eye(n)]; [M L]]
74
75 return A
76 end
77
78 function generateB(n, eN)
79 B = zeros(2*n)
80 B[2*n] = eN
81
82 return B

```

```

83 end
84
85 function generateC(n)
86     C = zeros(1,2*n)
87     C[1,1] = 1
88
89     return C
90 end
91
92 function getT(n, A)
93     eig_A = eigvals(A)
94     Tcomplex = eigvecs(A)
95
96     T = zeros(2*n, 2*n)
97     i = 1
98     while i <= 2*n #será que tem algo errado? tem de testar
99         if abs(imag(eig_A[i])) > 1e-10
100            T[:,i] = real(Tcomplex[:,i])
101            T[:,i+1] = -imag(Tcomplex[:,i])
102            i = i + 2
103        else
104            T[:,i] = Tcomplex[:,i]
105            i = i + 1
106        end
107    end
108
109    return T
110 end
111
112 function getABC_M(n, A, B, C)
113     T = getT(n,A)
114
115     A_M = T \ A * T
116     B_M = T \ B
117     C_M = C * T
118
119     return A_M, B_M, C_M
120 end
121
122 function getABCD_R(n, A_M, B_M, C_M,n_out=4)
123     C_M_diag = diagm(vec(C_M)) #matriz diagonal
124     G = C_M_diag / A_M * B_M #ganhos
125     subsystems = getSubsystems(n, eigvals(A_M), G)
126     A_R = zeros(n_out,n_out)
127     B_R = zeros(n_out)
128     C_R = zeros(1,n_out)
129

```

```

130 #Construir matrizes
131 i = 1
132 j = 1
133 while i <= n_out
134     index = subsystems[j][2]
135     if length(index) == 2 #complexo
136         A_R[i:i+1,i:i+1] = A_M[index, index]
137         B_R[[i,i+1]] = B_M[index]
138         C_R[1,[i,i+1]] = C_M[index]
139         i = i + 2
140     else
141         A_R[i,i] = A_M[index[1],index[1]]
142         B_R[i] = B_M[index[1]]
143         C_R[1,i] = C_M[index[1]]
144         i = i + 1
145     end
146     j = j + 1
147 end
148
149 D_R = C_M / A_M * B_M - C_R / A_R * B_R
150 return A_R, B_R, C_R, D_R
151 end
152
153 function getSubsystems(n, eig_A, G)
154     i = 1
155     subsystems = []
156     while i <= 2*n
157         if abs(imag(eig_A[i])) > 1e-10 #complexo
158             gain = abs(G[i] + G[i+1]) #considera sinal na soma como aqui ou soma
159             os módulos?
160             append!(subsystems, [(gain, [i,i+1])])
161             i = i + 2
162         else #real
163             gain = abs(G[i])
164             append!(subsystems, [(gain, [i])])
165             i = i + 1
166         end
167     end
168     #ordena pelo primeiro elemento da tupla
169     #ordem descendente
170     sort!(subsystems, rev=true)
171     return subsystems
172 end
173
174 function manuscript_p48()
175     n = 2
176     M = [[-1 1]; [1 -3]]

```

```

176 L = -2 * eye(n)
177 A = [[zeros(n,n) eye(n)];[M L]]
178 B = generateB(2,2)
179 C = generateC(n)
180
181 A_M, B_M, C_M = getABC_M(n,A,B,C)
182 A_R, B_R, C_R, D_R = getABCD_R(n, A_M, B_M, C_M, 4)
183
184 return A_R, B_R, C_R, D_R
185 end
186
187 function generateMATLABSimulationScript(n, filename, A, B, C, D)
188     output = "A = " * string(A) * ";\n\n"
189     output = output * "B = " * string(B) * "';\n\n"
190     output = output * "C = " * string(C) * ";\n\n"
191     output = output * "D = " * string(D) * ";\n\n"
192     output = output * "sys = ss(A, B, C, D);\n"
193     output = output * "opt = stepDataOptions;" 
194     output = output * "opt.InputOffset = 0;\n"
195     output = output * "opt.StepAmplitude = 0.3;\n"
196     output = output * "t = (0:0.01:50)';\n"
197     output = output * "y = step(sys, t, opt);"
198
199     output = output * "fig = figure;\n"
200     output = output * "hold on;\n"
201     output = output * "plot(t,y,'k-');\n"
202     output = output * "xlabel('Time (s)'), ylabel('Position (m));\n"
203     output = output * "title('Sistema original N = " * string(n) * ", "
204         simulacao reduzida para ordem 4');\n"
205     output = output * "print('SimulationReduceN" * string(n) * "', '-dpng',
206         '-r300');\n"
207     output = output * "close(fig);\n"
208
209     file = open(filename, "w")
210     write(file, output)
211     close(file)
212 end
213
214 function generateMATLABSimulationScriptToCompare(filename, A, B, C, A_R,
215     B_R, C_R, D_R)
216     output = "A_R = " * string(A_R) * ";\n\n"
217     output = output * "B_R = " * string(B_R) * "';\n\n"
218     output = output * "C_R = " * string(C_R) * ";\n\n"
219     output = output * "D_R = " * string(D_R) * ";\n\n"
220     output = output * "sysR = ss(A_R, B_R, C_R, D_R);\n\n"
221
222     output = output * "A = " * string(A) * ";\n\n"

```

```

220 output = output * "B = " * string(B) * "';\n\n"
221 output = output * "C = " * string(C) * "';\n\n"
222 output = output * "sys0 = ss(A, B, C, [0]);\n\n"
223
224 output = output * "opt = stepDataOptions;" 
225 output = output * "opt.InputOffset = 0;\n"
226 output = output * "opt.StepAmplitude = 0.3;\n\n"
227
228 output = output * "t = (0:0.01:50);\n"
229 output = output * "yR = step(sysR, t, opt);\n"
230 output = output * "y0 = step(sys0, t, opt);\n\n"
231
232 output = output * "fig = figure;\n"
233 output = output * "hold on;\n"
234 output = output * "plot(t,yR,'k--');\n"
235 output = output * "plot(t,y0,'k')\n"
236 output = output * "legend('Reduzido', 'Original');\n"
237 output = output * "xlabel('Time (s)'), ylabel('Position (m)');\n"
238 n = round(Int,size(A)[1]/2)
239 output = output * "title('Simulacao com sistema original N = " * string(n)
240     * " e reduzido N = 4');\n"
240 output = output * "print('SimulationN" * string(n) * "', '-dpng',
241     '-r300');\n"
241 output = output * "close(fig);\n"
242
243 file = open(filename, "w")
244 write(file, output)
245 close(file)
246 end
247
248
249 end

```

I.2 Texto Estruturado

I.2.1 Inicialização do primeiro teste de malha aberta

codes/init1.st

```

1 /* Universidade de Brasilia
2 Trabalho de Graduacao em Engenharia Mecatronica
3 Alunos:
4 Ataias Pereira Reis 10/0093817
5 Emanuel Pereira Barroso Neto 11/0115716
6 This code must be executed only once. */
7

```

```
8 MS0(drive_axis,MS0_1);
9
10 speed[0] := 0.0;
11 speed[1] := 0.0;
12 speed[2] := 0.0;
13 speed[3] := 0.0;
14 speed[4] := 0.0;
15 speed[5] := 0.0;
16 speed[6] := 0.0;
17 speed[7] := 0.0;
18 speed[8] := 0.0;
19 speed[9] := 0.0;
20 speed[10] := 0.0;
21 speed[11] := 0.0;
22 speed[12] := 0.0;
23 speed[13] := 0.0;
24 speed[14] := 0.0;
25 speed[15] := 0.0;
26 speed[16] := 0.0;
27 speed[17] := 0.0;
28 speed[18] := 0.0;
29 speed[19] := 0.0;
30 speed[20] := 0.0;
31 speed[21] := 0.0147026554205;
32 speed[22] := 0.0547055820855;
33 speed[23] := 0.114361720023;
34 speed[24] := 0.18111091036;
35 speed[25] := 0.251809893147;
36 speed[26] := 0.32607467121;
37 speed[27] := 0.403453064793;
38 speed[28] := 0.483471620607;
39 speed[29] := 0.565658044965;
40 speed[30] := 0.649536805548;
41 speed[31] := 0.734630464537;
42 speed[32] := 0.820458741642;
43 speed[33] := 0.906543473223;
44 speed[34] := 0.992424061587;
45 speed[35] := 1.07765619092;
46 speed[36] := 1.16189824235;
47 speed[37] := 1.24456943015;
48 speed[38] := 1.32036725857;
49 speed[39] := 1.3812552175;
50 speed[40] := 1.43231065067;
51 speed[41] := 1.48663214948;
52 speed[42] := 1.54839415747;
53 speed[43] := 1.61599008767;
54 speed[44] := 1.68757723373;
```

```

55 speed[45] := 1.7613579514;
56 speed[46] := 1.83569169336;
57 speed[47] := 1.90907847349;
58 speed[48] := 1.9801747816;
59 speed[49] := 2.04784594244;
60 speed[50] := 2.11113619377;
61 speed[51] := 2.16944638193;
62 speed[52] := 2.22182598313;
63 speed[53] := 2.26825099412;
64 speed[54] := 2.30985215012;
65 speed[55] := 2.34320587986;
66 speed[56] := 2.36620871737;
67 speed[57] := 2.38071882377;
68 speed[58] := 2.38672834648;
69 speed[59] := 2.38430263817;
70 speed[60] := 2.37407699776;
71 speed[61] := 2.35619724752;
72 speed[62] := 2.33055863496;
73 speed[63] := 2.29754175796;
74 speed[64] := 2.25797832185;
75 speed[65] := 2.21262636405;
76 speed[66] := 2.16123527696;
77 speed[67] := 2.10467481035;
78 speed[68] := 2.04388225087;
79 speed[69] := 1.97863997419;
80 speed[70] := 1.90950133568;
81 speed[71] := 1.83693203375;
82 speed[72] := 1.7619821587;
83 speed[73] := 1.686412295;
84 speed[74] := 1.60932227957;
85 speed[75] := 1.53031257908;
86 speed[76] := 1.45140199977;
87 speed[77] := 1.37297157979;
88 speed[78] := 1.29394180319;
89 speed[79] := 1.2157528212;
90 speed[80] := 1.1408355106;
91 speed[81] := 1.06779007087;
92 speed[82] := 0.995420551475;
93 speed[83] := 0.924581999131;
94 speed[84] := 0.855323010933;
95 speed[85] := 0.788477897412;
96 speed[86] := 0.726538190521;
97 speed[87] := 0.66801109369;
98 speed[88] := 0.607741032753;
99 speed[89] := 0.547567758592;
100 speed[90] := 0.493301637758;
101 speed[91] := 0.440419162243;

```

```
102 speed[92] := 0.387507265216;
103 speed[93] := 0.338400915621;
104 speed[94] := 0.291658541896;
105 speed[95] := 0.248512542153;
106 speed[96] := 0.209796973955;
107 speed[97] := 0.172530018125;
108 speed[98] := 0.141664432169;
109 speed[99] := 0.11599366767;
110 speed[100] := 0.087107967923;
111 speed[101] := 0.0627295944751;
112 speed[102] := 0.0490272896769;
113 speed[103] := 0.0374349286012;
114 speed[104] := 0.0234408281723;
115 speed[105] := 0.0103613796074;
116 speed[106] := 0.0;
117 speed[107] := 0.0;
118 speed[108] := 0.0;
119 speed[109] := 0.0;
120 speed[110] := 0.0;
121 speed[111] := 0.0;
122 speed[112] := 0.0;
123 speed[113] := 0.0;
124 speed[114] := 0.0;
125 speed[115] := 0.0;
126 speed[116] := 0.0;
127 speed[117] := 0.0;
128 speed[118] := 0.0;
129 speed[119] := 0.0;
130 speed[120] := 0.0;
131 speed[121] := 0.0;
132 speed[122] := 0.0;
133 speed[123] := 0.0;
134 speed[124] := 0.0;
135 speed[125] := 0.0;
136 speed[126] := 0.0;
137 speed[127] := 0.0;
138 speed[128] := 0.0;
139 speed[129] := 0.0;
140 speed[130] := 0.0;
141 speed[131] := 0.0;
142 speed[132] := 0.0;
143 speed[133] := 0.0;
144 speed[134] := 0.0;
145 speed[135] := 0.0;
146 speed[136] := 0.0;
147 speed[137] := 0.0;
148 speed[138] := 0.0;
```

```

149 speed[139] := 0.0;
150 speed[140] := 0.0;
151 speed[141] := 0.0;
152 speed[142] := 0.0;
153 speed[143] := 0.0;
154 speed[144] := 0.0;
155 speed[145] := 0.0;
156 speed[146] := 0.0;
157 speed[147] := 0.0;
158 speed[148] := 0.0;
159 speed[149] := 0.0;
160 speed[150] := 0.0;
161 speed[151] := 0.0;
162 speed[152] := 0.0;
163 speed[153] := 0.0;
164 speed[154] := 0.0;
165 speed[155] := 0.0;
166 speed[156] := 0.0;
167 speed[157] := 0.0;
168 speed[158] := 0.0;
169 speed[159] := 0.0;
170 speed[160] := 0.0;
171 speed[161] := 0.0;
172 speed[162] := 0.0;
173 speed[163] := 0.0;
174 speed[164] := 0.0;
175 speed[165] := 0.0;
176 speed[166] := 0.0;
177 speed[167] := 0.0;
178 speed[168] := 0.0;
179 speed[169] := 0.0;
180 speed[170] := 0.0;
181
182 dataInitialized := 1;

```

I.2.2 Inicialização do segundo teste de malha aberta

codes/init2.st

```

1 /* Universidade de Brasilia
2 Trabalho de Graduacao em Engenharia Mecatronica
3 Alunos:
4 Ataias Pereira Reis 10/0093817
5 Emanuel Pereira Barroso Neto 11/0115716
6 This code must be executed only once. */
7
8 MS0(drive_axis,MS0_1);

```

```

9
10 speed[0] := 0.0;
11 speed[1] := 0.0;
12 speed[2] := 0.0;
13 speed[3] := 0.0;
14 speed[4] := 0.0;
15 speed[5] := 0.0;
16 speed[6] := 0.0;
17 speed[7] := 0.0;
18 speed[8] := 0.0;
19 speed[9] := 0.0;
20 speed[10] := 0.0;
21 speed[11] := 0.0;
22 speed[12] := 0.0;
23 speed[13] := 0.0;
24 speed[14] := 0.0;
25 speed[15] := 0.0;
26 speed[16] := 0.0;
27 speed[17] := 0.0;
28 speed[18] := -0.0692113524459;
29 speed[19] := 0.314942000684;
30 speed[20] := 1.64847179452;
31 speed[21] := 3.06203932423;
32 speed[22] := 3.88556305076;
33 speed[23] := 4.25984881416;
34 speed[24] := 4.24772217569;
35 speed[25] := 3.9166637819;
36 speed[26] := 3.35019221407;
37 speed[27] := 2.64853848716;
38 speed[28] := 1.92086693496;
39 speed[29] := 1.27214274157;
40 speed[30] := 0.788466188462;
41 speed[31] := 0.524290908189;
42 speed[32] := 0.493490156489;
43 speed[33] := 0.694837969193;
44 speed[34] := 0.822419246039;
45 speed[35] := 0.418206857379;
46 speed[36] := 0.0262247526643;
47 speed[37] := 0.230482402967;
48 speed[38] := 0.789536874806;
49 speed[39] := 1.48057490162;
50 speed[40] := 2.15097967763;
51 speed[41] := 2.69969451667;
52 speed[42] := 3.0612889055;
53 speed[43] := 3.19935866684;
54 speed[44] := 3.10564873155;
55 speed[45] := 2.7984308071;

```

```

56 speed[46] := 2.32194132411;
57 speed[47] := 1.78925311378;
58 speed[48] := 1.23475193349;
59 speed[49] := 0.680977178876;
60 speed[50] := 0.269581981875;
61 speed[51] := 0.0735006589406;
62 speed[52] := 0.0;
63 speed[53] := -0.0411084579085;
64 speed[54] := -0.0752386045019;
65 speed[55] := -0.0939490510793;
66 speed[56] := -0.0958419020537;
67 speed[57] := -0.0879379646741;
68 speed[58] := -0.0665110466718;
69 speed[59] := -0.0340396907113;
70 speed[60] := 0.0;
71 speed[61] := 0.0116207365686;
72 speed[62] := 0.0239437089848;
73 speed[63] := 0.0232352765649;
74 speed[64] := 0.0;
75 speed[65] := 0.0;
76 speed[66] := 0.0;
77 speed[67] := 0.0;
78 speed[68] := 0.0;
79 speed[69] := 0.0;
80 speed[70] := 0.0;
81 speed[71] := 0.0;
82 speed[72] := 0.0;
83 speed[73] := 0.0;
84 speed[74] := 0.0;
85 speed[75] := 0.0;
86 speed[76] := 0.0;
87 speed[77] := 0.0;
88 speed[78] := 0.0;
89 speed[79] := 0.0;
90 speed[80] := 0.0;
91 speed[81] := 0.0;
92 speed[82] := 0.0;
93 speed[83] := 0.0;
94 speed[84] := 0.0;
95 speed[85] := 0.0;
96
97 dataInitialized := 1;

```

I.2.3 Inicialização dos testes de malha fechada

codes/initP.st

```

1 /* Universidade de Brasilia
2 Trabalho de Graduacao em Engenharia Mecatronica
3 Alunos:
4 Ataias Pereira Reis 10/0093817
5 Emanuel Pereira Barroso Neto 11/0115716
6 This code must be executed only once. */
7
8 MS0(drive_axis,MS0_1);
9 Kp := 0.0075;
10
11 dataInitialized := 1;

```

I.2.4 Inicialização da rede DeviceNET

codes/InitDNetST.st

```

1 Local:2:0.CommandRegister.Run [:=] 1;
2 if NOT dataInitialized then
3   EVENT(Initialize_speed);
4 end_if;

```

I.2.5 Programa de controle em malha aberta

codes/malhaAbertaT2.st

```

1 // k is the index of the speed vector
2 if dataInitialized AND k < 85 then
3   MAJ(drive_axis,MAJ_1,0,speed[k],0,50.0,1,50.0,1,0,50.0,50.0,1,0,0);
4   k:=k + 1;
5 end_if;
6
7 if k >= 85 then
8   MAS(drive_axis,MAS_1,0,0,100,1,0,100,1);
9   if MAS_1.DN then
10    MSF(drive_axis,MSF_0);
11   end_if;
12 end_if;

```

I.2.6 Programa de controle proporcional - malha fechada

codes/malhaFechadaP.st

```

1 // k is the index of the speed vector
2

```

```

3 if dataInitialized AND k < 300 then
4   error := 300.0 - position;
5   speed := error*Kp;
6   MAJ(drive_axis,MAJ_1,0,speed,0,50.0,1,50.0,1,0,50.0,50.0,1,0,0);
7   k:=k + 1;
8 end_if;
9
10 if k >= 300 then
11   MAS(drive_axis,MAS_1,0,0,100,1,0,100,1);
12   if MAS_1.DN then
13     MSF(drive_axis,MSF_0);
14   end_if;
15 end_if;

```

I.3 Linguagem *ladder*

I.3.1 Execução de *trigger* da câmera

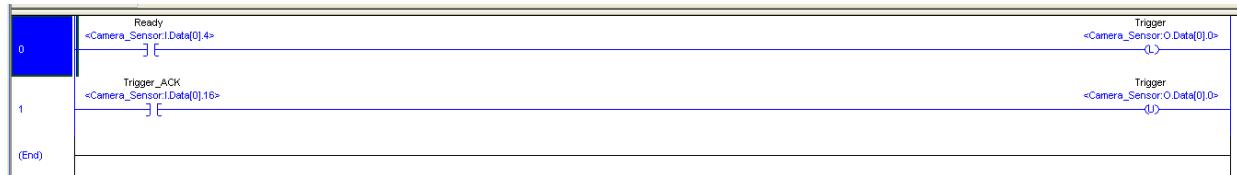


Figura I.1: *Trigger* da câmera

I.3.2 Rotina de parada de emergência

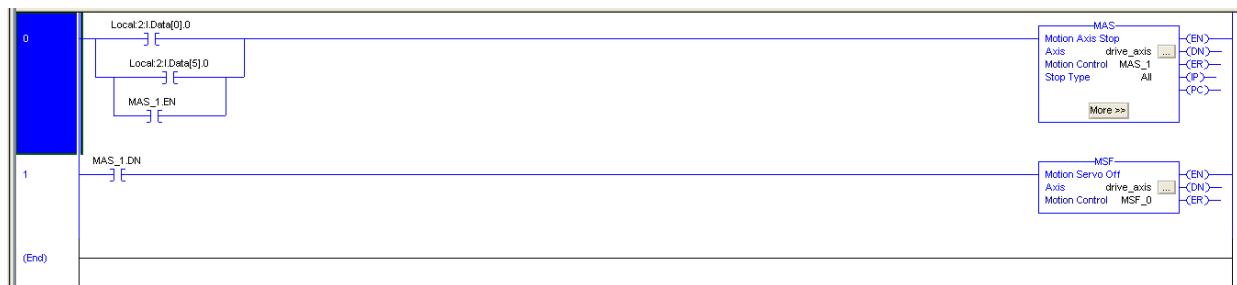


Figura I.2: Parada de emergência

I.4 Programas da câmera

I.4.1 Detecção da posição horizontal da bolinha

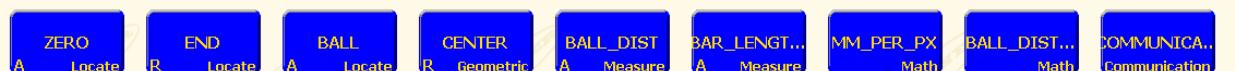
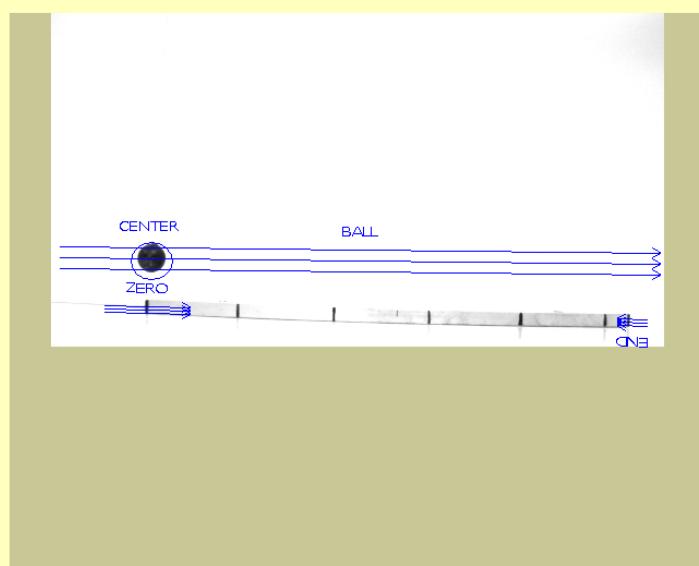


Figura I.3: Detecção da posição horizontal da bolinha