UNIVERSIDADE FEDERAL DA GRANDE DOURADOS
FACULDADE DE ENGENHARIA
CURSO DE ENGENHARIA MECÂNICA

Diogo Cunha José Karmouche

PROJETO DE OBSERVADOR DE ESTADO PARA UM SISTEMA PÊNDULO-CARRO

DOURADOS/MS
2018

Diogo Cunha José Karmouche

PROJETO DE OBSERVADOR DE ESTADO PARA UM SISTEMA PÊNDULO-CARRO

Trabalho de Conclusão de Curso apresentado à banca examinadora da Faculdade de Engenharia da Universidade Federal da Grande Dourados para a obtenção do título de Bacharel em Engenharia Mecânica.

Orientador:
Prof. Dr. Sanderson Manoel da Conceição

DOURADOS/MS
2018

Dados Internacionais de Catalogação na Publicação (CIP).

K18p Karmouche, Diogo Cunha José

PROJETO DE OBSERVADOR DE ESTADO PARA UM SISTEMA PÊNDULO-CARRO / Diogo Cunha José Karmouche -- Dourados: UFGD, 2018.

108f. : il. ; 30 cm.

Orientador: Sanderson Manoel Conceição

TCC (Graduação em Engenharia Mecânica)-Universidade Federal da Grande Dourados

Inchui bibliografia

1. Espaço de Estados. 2. Scilab. 3. Arduino. 4. Observadores. I. Título.

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

MINISTÉRIO DA EDUCAÇÃO
FUNDAÇÃO UNIVERSIDADE FEDERAL DA GRANDE DOURADOS

ANEXO D - AVALIAÇÃO FINAL DO TRABALHO DE CONCLUSÃO DE CURSO

Aluno: **Diogo Cunha José Karmouche**

Título do trabalho: **Projeto de observador de estado para um sistema pêndulo-carro**

BANCA EXAMINADORA

1. **Presidente (orientador):**
Prof. Dr. Sanderson Manoel da Conceição, Universidade Federal da Grande Dourados - UFGD

2. **Membro:**
Prof. Dr. Gerson Bessa Gibelli, Universidade Federal da Grande Dourados - UFGD

3. **Membro:**
Prof. Dr. Rodrigo Borges Santos, Universidade Federal da Grande Dourados - UFGD

De acordo com o grau final obtido pelo aluno, nós da banca examinadora, declaramos Aprovado (Aprovado/Reprovado) o aluno acima identificado, na componente curricular Trabalho de Conclusão de Curso (TCC-II) de Graduação no Curso de Engenharia Mecânica da Universidade Federal da Grande Dourados.

Dourados, 17 de dezembro de 2018.

Sanderson Manoel da Conceição
Prof. Dr. Sanderson Manoel da Conceição

Prof. Dr. Gerson Bessa Gibelli

Rodrigo Borges Santos
Prof. Dr. Rodrigo Borges Santos

Aos meus pais José Roberto e Sandra Lucia que nunca deixaram de acreditar em mim.

AGRADECIMENTOS

Primeiramente a Deus, pela minha vida, minha família e por todas as pessoas que foram colocadas em meu caminho me ajudando a moldar a pessoa que sou hoje.

Agradeço ao meus pais José Roberto Jorge Karmouche e Sandra Lucia Brandão Karmouche que nunca deixaram de acreditar em mim e por terem me ensinado tudo o que hoje prezo com muito carinho.

Agradeço por todos que tive contato, pois sem eles não possuiria a maturidade atual nem minhas experiências.

Agradeço ao meu orientador Sanderson Manoel da Conceição por toda paciência e por todo suporte para conclusão deste trabalho.

E por fim agradeço ao auxilio disponibilizado pela Universidade Federal da Grande Dourados.

“In theory, there is no difference between theory and practice.

But, in practice, there is.”

Jan L. A. van de Snepscheut

RESUMO

Desde a primeira revolução industrial, no século XVII é um fato a crescente necessidade de sistemas mais complexos que requerem processos automatizados, estáveis, que determinadas grandezas sejam controladas e, em caso de não puderem ser mensuradas ou é financeiramente inviável, que sejam estimadas. Logo, muitos sistemas estão sendo representados por espaço de estado e projetos de controladores e observadores estão cada vez mais comuns na indústria e em outras áreas. Em muitas áreas do conhecimento é necessário a mensuração de variáveis através de observadores, na área de análise de falhas é aplicado para compreender o sistema identificando falhas e avarias e na área de sistemas de controle para realimentar o sistema com todas as variáveis de estado controladas. Dentro deste contexto o presente trabalho apresenta um projeto de observação de estados utilizando o *software* livre Scilab em conjunto com o *hardware* Arduino, assim projetando um sistema de observador de estados de baixo custo e acessível a comunidade acadêmica.

Palavras-chave: Espaço de Estado. Scilab. Arduino. Observadores.

ABSTRACT

Since the first industrial revolution, in the XVIII century it's a fact that the growing need for more complex systems requiring automated process, stable processes, that certain quantities are controlled and, if they can't be measured or are financially infeasible, that they are estimated. Therefore, many systems are being represented by state space and controller designs and observers are increasingly common in industry and other areas. In many areas of knowledge it is necessary to measure variables through observers, in the area of fault analysis is applied to understand the system identifying faults and malfunctions and in the area of control systems to feed the system with all state variables controlled.In this context, the present work show a state observation project using Scilab free software in conjunction with Arduino hardware, thus designing a low cost state observer system accessible to the academic community.

Keywords: *Space of State. Scilab. Arduino. Observers.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E SIGLAS

$G(s)$	Função de Transferência
$\mathcal{L}$	Laplace
$\frac{Y(s)}{X(s)}$	Relação entre a saída e entrada de um sistema
$\boldsymbol{x_n}$	Vetor de estado
u_r	Entrada de um sistema
y_m	Saída de um sistema
$\dot{x}_n$	Sistema descrito em espaço de estados
$\boldsymbol{\dot{y}_m}$	Saída de um sistema
$\mathbf{A}$	Matriz de estado
$\mathbf{B}$	Matriz de entrada
$\mathbf{C}$	Matriz de saída
$\mathbf{D}$	Matriz de transmissão direta
C_{total}	Resposta total do sistema
$C_{natural}$	Resposta natural do sistema
$C_{forçada}$	Resposta forçada do sistema
$\det$	Determinante da matriz
adj	Adjunta da matriz
$\boldsymbol{C_{ctr}}$	Matriz de controlabilidade
$\boldsymbol{O_{obs}}$	Matriz de observabilidade
$\dot{\tilde{\boldsymbol{x}}}$	Sistema estimado descrito em espaço de estados
$\tilde{\boldsymbol{x}}$	Vetor de estado estimado
$\tilde{\boldsymbol{y}}$	Resposta estimada
$\boldsymbol{e}$	Erro de estimação
$\boldsymbol{L}$	Ganho do observador
X_g	Posição do centro de gravidade no eixo x
Y_g	Posição do centro de gravidade no eixo y
l	Comprimento efetivo da haste
I_{cm}	Momento de Inércia
τ	Torque
ω	Velocidade angular

$\frac{d\theta}{dt}$	Velocidade angular na forma diferencial
θ	Posição angular
$\ddot{\theta}$	Aceleração angular
x	Posição
$\ddot{x}$	Aceleração
H	Deslocamento horizontal
V	Deslocamento vertical
PWM	*Pulse With Modulation*

SUMÁRIO

Capítulo 1. INTRODUÇÃO

Desde a primeira revolução industrial, no século XVII, é possível observar a crescente necessidade da automatização dos processos de fabricação e controle de grandezas desejadas, seja por segurança ou conforto (OGATA, 2009). Por isso, a aplicação de projetos de controladores e observadores sofre uma crescente necessidade no mundo moderno.

No ramo industrial, há uma crescente preocupação com o desenvolvimento de novas técnicas para detecção de falhas ou suas localizações, pois evita paradas repentinas (KOROISHI, 2009). Atualmente as técnicas de observação de estados possuem forte presença nas áreas de controle e detecção de falhas em sistemas rotativos.

Em muitos sistemas é requisitado a mensuração de variáveis para a compreensão do estado do sistema, com objetivo de identificar falhas, dados importantes ou para aplicação de um controle. Contudo, adquirir essas variáveis pode se tornar uma tarefa custosa seja financeiramente ou por complexidade de mensuração (NISE, 2017). Devido a tal problema de mensuração, os sistemas estão sendo representados por modelos matemáticos e sendo aplicado técnicas de observação, para assim estimar os dados requisitados (FERNANDES, 2011). O que reduz da aquisição de dados, uma vez que o número de sensores pode ser reduzido.

Dentro deste contexto o presente trabalho apresenta um projeto de observação de estados utilizando o *software* livre Scilab em conjunto com o *hardware* Arduino, no qual projetado um sistema de observador de estados de baixo custo e acessível a comunidade estudantil.

1.1. REVISÃO BIBLIOGRÁFICA

1.1.1. Scilab

Scilab é um *software* científico livre em desenvolvimento, possui muitos módulos, que estão também crescendo e desenvolvendo, que fornece um poderoso ambiente computacional aberto para aplicações em diversas áreas, o que lhe torna semelhante ao *software* Matlab. Por ser um programa livre o scilab oferece grandes vantagens, para o usuário, no quesito de possibilidades de uso na área de educação e de um setor que comumente busca alternativas financeiramentes viáveis, à vista disso *softwares* livres são opções fortemente consideradas. O scilab em comparação a programas como Maple e Matlab se apresenta nivelado em quesito na área de cálculo numérico, sendo apenas inferior ao Maple na aplicação de soluções simbólicas

e problemas algébricos, entretanto por ser um *software* livre ainda ganha destaque entre os *softwares* citados (SOUZA, 2007).

1.1.1. Arduino

A crescente necessidade da automação de inúmeros processos, como no meio fabril, comercial ou residencial, é incontestável. O Arduino, uma plataforma de código aberto, proporciona soluções rápidas que podem ser embarcadas com agilidade e também aplicadas em processos simplificados (STEVAN e SILVA, 2015).

1.1.2. Scilab-Arduino

O scilab por ser um *software* de código aberto livre em desenvolvimento, possui muitos módulos em desenvolvimento que lhe permitem desempenhar tarefas como análises estatísticas e gerar interfaces para aquisição de dados de hardwares como o arduino. O scilab oferece um site de gestão para todas as bibliotecas externas, sendo estas atualizadas por usuários experientes em programação. A comunicação entre o scilab e a plataforma arduino é possível pelo módulo existente arduino do scilab, assim tal módulo pode ser utilizado em conjunto com técnicas de controle, assim retirando a teoria e aplicando na prática (CHAYA R., *et al*, 2016).

1.1.3. Observadores de estado

Observadores de estados se apresentam com grandes relevâncias nos setores industriais, seja na área de controle de processos ou na área de análise de falhas em sistemas rotativos (KOROISHI, 2009).

Na área de controle, observadores são imprescindíveis, principalmente em sistemas representado por equações de espaço de estados, pois há sistemas em que a mensuração de variáveis é impossível ou custosa, seja por necessitar de uma grande variedade de transdutores ou sensores com preço elevado (OGATA, 2009).

Na área de análise de falhas, observadores são ferramentas baratas que realizam a diagnose do sistema através de um modelo matemático da planta. Falha é compreendida como um lapso não permitido (MORAIS, 2006), assim o observador exerce sua função apresentando o comportamento em tempo real da planta e informando alguma alteração no sistema, em caso de falha ou alguma perturbação, para haver uma noção da situação do elemento analisado sem necessidade de cessar a operação sistema (FERNANDES, 2011).

1.2. OBJETIVO

Este trabalho tem como objetivo o estudo e implementação de um observador de estados em um sistema carro-pêndulo e a estimação dos estados do sistema. Também, explorar o uso do microcontrolador Arduino e seu uso junto ao *software* Scilab. Os aspectos práticos, como a montagem da estrutura carro-pêndulo e o processo de aquisição de dados, eram de interesses visto que este sistema pode ser adquirido para uso didático, porém com um alto custo.

1.3. ESTRUTURA DO TRABALHO

No Capítulo 2, será apresentado a importância de sistema de controle e observadores, como também a modelagem de sistemas dinâmicos em espaço de estados.

No Capítulo 3, será apresentado a modelagem do sistema pêndulo carro em espaço de estado e a construção do sistema para análise experimental.

No Capítulo 4, será validado a modelagem e analisados os resultados obtidos no ensaio experimental.

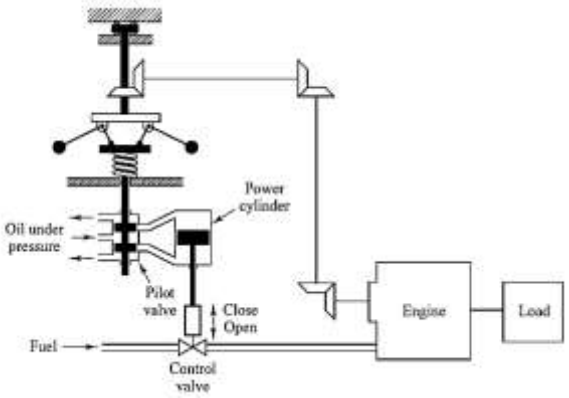
No Capítulo 5, são apresentadas as conclusões desse trabalho e sugestões para trabalhos futuros.

Capítulo 2. MODELAGEM

2.1. ESPAÇO DE ESTADOS

Os avanços na engenharia apresentam a tendência de aumentar gradativamente a complexidade dos sistemas dinâmicos e, com esta evolução, surgem novos meios de representar cada sistema, buscando representar a resposta do sistema com mais eficiência. Um dos primeiros grandes trabalhos da área de controle de acordo com Ogata (2009) foi o regulador centrífugo (Figura 1), também conhecido como governador centrífugo, de James Watt para controlar a velocidade de máquinas a vapor no século XVIII. Tal dispositivo regulador consistia em que a potência do sistema era transmitido ao regulador através do eixo de saída do motor, por meio de um elemento transmissor, como uma correia ou corrente, conectada à roda da correia inferior, sendo o governador conectado à válvula de controle de potência que regula o fluxo do fluido de trabalho, o vapor, que supre o propulsor da máquina a vapor.

Figura 1: Sistema de regulação de velocidade



FONTE: Ogata (2009)

A medida que à velocidade do propulsor aumenta, o eixo central do regulador gira em um ritmo acelerado, aumentando a energia cinética das bolas, permitindo a elevação das massas nos braços de alavanca, movendo-os para fora e para cima contra a gravidade. Assim, caso o movimento for longe o suficientemente os braços puxam um rolamento, para baixo, que por sua vez reduz a abertura da válvula borboleta, reduzindo a taxa de fluido do trabalho que entra no cilindro, logo, reduzindo a velocidade da máquina a vapor.

Como apresentado, um dos primeiros sistemas de controle que regulava e controlava alguma propriedade de acordo com as necessidades do projeto. Dando porta a outros trabalhos como Nicolas Minorsky em 1922 com controladores automáticos para pilotagem de embarcações, Harry Nyquist em 1932 determinando a estabilidade de sistemas dinâmicos, entre

outros. Tais trabalhos desencadearam novas possibilidades e, juntamente, novas necessidades. Requerendo melhores repostas do sistema em quesito de estabilidade, desempenho, custo, tempo e entre outras necessidades.

Várias abordagens surgiram desde a década de 40, de acordo com Nise (2013), duas abordagens grandes são destacadas dentro da área de controle de sistemas e análise de realimentação: o clássico e o moderno.

O método clássico ou também de análise de resposta em frequência, surgiu na década de 1940 e consistia em converter a equação diferencial (Eq. 1) que regia a dinâmica do sistema em uma função de transferência (Eq. 3) com a aplicação da transformada de Laplace (Eq. 2), assim formando uma relação algébrica entre a entrada e saída de um sistema de controle.

$$a_0 y^n + a_1 y^{n-1} + \cdots + a_{n-1}\dot{y} + a_n = b_0 x^m + b_1 x^{m-1} + \cdots + b_{m-1}\dot{x} + b_m x \qquad (1)$$

$$G(s) = \frac{\mathcal{L}(saída)}{\mathcal{L}(entrada)}\bigg|_{condições\ iniciais\ nulas} \qquad (2)$$

$$\frac{Y(s)}{X(s)} = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1} s b_m}{a_0 s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \qquad (3)$$

A vantagem em analisar o sistema através da resposta em frequência, modelo clássico, é a possibilidade de observar os diferentes efeitos de vários parâmetros no sistema por sua capacidade de prover respostas transitórias e de estabilidade. Entretanto, o modelo clássico é limitado para sistemas invariáveis ao tempo, lineares e com apenas um sinal de entrada e de saída, a função de transferência é uma propriedade inerente ao sistema sendo independente da magnitude e natureza da excitação do sistema e também o método clássico não fornece informações relativas a estrutura física do sistema relacionando apenas a entrada a saída do sistema.

Outro método na área de controle é o controle moderno. Um método que surgiu devida a necessidade de realizar tarefas mais complexas com alta precisão, também da ampliação de sistemas complexos (OGATA, 2009) considerando a estrutura física do sistema e tendo uma relação com a magnitude e natureza da excitação do sistema.

No modelo moderno, modelagem por equações no espaço de estados, além de permitir múltiplas entradas e saídas, lineares ou não, ela é essencialmente abordada no domínio do tempo e da frequência, enquanto a clássica é apenas no domínio da frequência.

Para entender um sistema representado em espaço de estado deve ser definido o que é estado, variável de estado, vetor de estado, espaço de estados:

Estado: Menor conjunto de variáveis, que em conjunto com as equações descrevendo a dinâmica do sistema analisado e um sinal de entrada, determinam o comportamento do sistema para qualquer instante.

Variáveis de estado: Menor conjunto de variáveis que determinam a dinâmica do sistema analisado.

"As variáveis de estado descrevem a configuração presente de um sistema e podem ser usadas para determinar a resposta futura, dadas as excitações de entrada e as equações que descrevem a dinâmica" (DORF; BISHOP, 2013)

Vetor de estado: Conjunto de n variáveis de estado descrevendo totalmente o comportamento do sistema. Assim, determinando o estado do sistema $x(t)$ para qualquer instante $t \geq t_0$, uma vez que é dado o estado em $t = t_0$ e a entrada $\boldsymbol{u(t)}$ para $t \geq t_0$ é especificada.

Espaço de estados: Espaço n-dimensional, cujos eixos coordenados são compostos por eixos que são as variáveis de estado.

Utilizando a técnica moderna de controle, o sistema pode ser modelado por equações no espaço de estados envolvendo "r" sinais de entrada, "m" sinais de resposta do sistema e "n" variáveis de estado pode ser descrito como:

Sendo $x_1(t), x_2(t), \dots, x_n(t)$, as variáveis de estado do sistema, $u_1(t), u_2(t), \dots, u_r(t)$ a entrada no sistema e $y_1(t), y_2(t), \dots, y_m(t)$ a resposta do sistema.

$$\begin{aligned} \boldsymbol{\dot{x}_1(t)} &= f_1(x_1, x_2, \dots, x_n; u_1, u_2, \dots, u_r; t) \\ \boldsymbol{\dot{x}_2(t)} &= f_2(x_1, x_2, \dots, x_n; u_1, u_2, \dots, u_r; t) \\ &\vdots \\ \boldsymbol{\dot{x}_n(t)} &= f_n(x_1, x_2, \dots, x_n; u_1, u_2, \dots, u_r; t) \end{aligned} \tag{4}$$

$$\begin{aligned} \boldsymbol{y_1(t)} &= g_1(x_1, x_2, \dots, x_n; u_1, u_2, \dots, u_r; t) \\ \boldsymbol{y_2(t)} &= g_2(x_1, x_2, \dots, x_n; u_1, u_2, \dots, u_r; t) \\ &\vdots \\ \boldsymbol{y_m(t)} &= g_m(x_1, x_2, \dots, x_n; u_1, u_2, \dots, u_r; t) \end{aligned} \tag{5}$$

Simplificando as Eq. (4) e Eq.(5):

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = f(x, u, t) \tag{6}$$

$$\boldsymbol{y}(\boldsymbol{t}) = g(x, u, t) \tag{7}$$

Onde as Eq. 6 e 7 representam, respectivamente, a equação de estado e a de saída. Para um sistema ser representado em equações de estado, é necessário a linearização do sistema. Um sistema é dado linear quando é verificado o chamado princípio de sobreposição, que é quando a resposta do sistema à soma de dois sinais é igual a soma das respostas do sistema a cada sinal individualmente.

Assim, linearizando as equações de saída e de estado em torno de um ponto de operação obtém-se:

$$\dot{\boldsymbol{x}}(\boldsymbol{t}) = \boldsymbol{A}(t)x(t) + \boldsymbol{B}(t)u(t) \tag{8}$$

$$\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{C}(t)x(t) + \boldsymbol{D}(t)u(t) \tag{9}$$

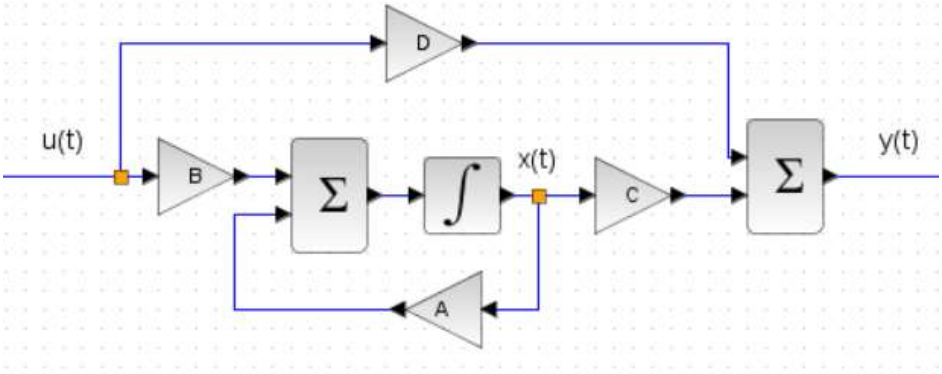
Sendo $\boldsymbol{A}(\boldsymbol{t}), \boldsymbol{B}(\boldsymbol{t}), \boldsymbol{C}(\boldsymbol{t})$ *e* $\boldsymbol{D}(\boldsymbol{t})$ matrizes que representam respectivamente a matriz de estados, matriz de entrada, matriz de saída e matriz de transmissão direta (OGATA, 2009). Sendo essas variantes no tempo, se as funções f e g, das Eq. (6) e (7), forem invariantes no tempo a equação no espaço de estados é simplificada para:

$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \tag{10}$$

$$y(t) = \boldsymbol{C}x(t) + \boldsymbol{D}u(t) \tag{11}$$

Para melhor visualização do sistema em espaço de estados é realizado o diagrama de blocos do mesmo (Figura 2) para coompreender quais matrizes recebem a exitação ou devolvem a resposta.

Figura 2: Diagrama de blocos do sistema em espaço de estados no *sofware* scilab



FONTE: Próprio Autor

2.2. PROPRIEDADES DO SISTEMA

2.2.1. ESTABILIDADE

A definição de estabilidade varia de acordo com o tipo do projeto, ponto de visão, neste trabalho o conceito estabilidade será trabalhado para sistemas lineares e invariantes no tempo.

Para compreender o que é estabilidade, de acordo com Nise (2013)**,** é necessário definir que a resposta total de um sistema é a soma da resposta natural, e a resposta forçada (ou transiente) de um sistema.

$$C_{total}(t) = C_{natural}(t) + C_{forçada}(t) \quad (12)$$

Sendo a reposta natural um modo que o sistema dissipa ou recebe energia, dependendo ou não da entrada do sistema. Já a resposta forçada é dependente apenas da entrada, em sistemas lineares.

Em alguns sistemas a reposta natural aproxima-se eventualmente para zero, ou oscila, apenas restando a reposta forçada. Em outros casos, como um pêndulo invertido, a reposta aumenta sem limites, assim tornando-a superior a reposta forçada resultando em um sistema não controlável. Esses sistemas são descritos como instáveis, sendo que, dependendo do projeto ou aplicação a instabilidade do sistema pode acarretar na destruição ou colocar a vida em risco das pessoas, em muitos projetos o sistema é configurado com limites para prevenir avarias,

como por exemplo um elevador é limitado em seu deslocamento para não ultrapassar o chão ou o telhado, tendo ainda um controle da velocidade para o conforto no transporte da carga.

Sistemas de controle devem ser estáveis, ou seja, a resposta natural deles deve tender a zero em um tempo infinito, caso o sistema oscilar é considerado relativamente ou criticamente estável.

Para sistemas de malha fechada a estabilidade é identificada através da posição dos polos no plano complexo.

2.2.1.1. Estabilidade para o método clássico

"[...]. sistemas estáveis possuem funções de transferência em malha fechada com polos apenas no meio plano esquerdo. [...]. Sistemas instáveis possuem funções de transferência com pelo menos um polo no meio plano direito e/ou polos de multiplicidade maior que 1 no eixo imaginário." (NISE, 2013, P.303)

Muitos critérios foram projetados para melhor determinar a estabilidade utilizando a informação acima. Para o método de controle clássico, além de encontrar as raízes através do denominador de uma função de transferência, é muito utilizado o critério de Routh-Hurwitz que através de um conjunto de operações matemáticas é possível determinar quantos polos estão em cada lado do plano imaginário.

Como segue a Eq.(13):

$$\frac{N(s)}{a_4 s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0} \tag{13}$$

Tabela 1: Determinação de estabilidade de Routh-Hurwitz

s^4	a_4	a_2	a_0
s^3	a_3	a_1	0
s^2	$-\frac{\begin{vmatrix} a_4 & a_2 \\ a_3 & a_1 \end{vmatrix}}{a_3} = b_1$	$-\frac{\begin{vmatrix} a_4 & a_0 \\ a_3 & 0 \end{vmatrix}}{a_3} = b_2$	$-\frac{\begin{vmatrix} a_4 & 0 \\ a_3 & 0 \end{vmatrix}}{a_3} = 0$
s^1	$-\frac{\begin{vmatrix} a_3 & a_1 \\ b_1 & b_2 \end{vmatrix}}{b_1} = c_1$	$-\frac{\begin{vmatrix} a_3 & 0 \\ b_1 & 0 \end{vmatrix}}{b_1} = 0$	$-\frac{\begin{vmatrix} a_3 & 0 \\ b_1 & 0 \end{vmatrix}}{b_1} = 0$
s^0	$-\frac{\begin{vmatrix} b_2 & b_2 \\ c_1 & 0 \end{vmatrix}}{c_1} = d_1$	$-\frac{\begin{vmatrix} b_1 & 0 \\ c_1 & 0 \end{vmatrix}}{c_1} = 0$	$-\frac{\begin{vmatrix} b_1 & 0 \\ c_1 & 0 \end{vmatrix}}{c_1} = 0$

Contudo este método não apresenta a localização exata dos polos do sistema dinâmico, assim possui maior foco no projeto que na análise do mesmo, pois caso houver alguma variável indeterminável este critério consegue determinar sua estabilidade.

2.2.1.2. Estabilidade para o método moderno

Para o método moderno a definição de estabilidade através dos polos é mantida, apenas a determinação dos polos é diferente. Os polos de um sistema, representado em equações de espaço de estado, são os autovalores da matriz $\boldsymbol{A}$ de acordo com a demonstração abaixo:

$$\boldsymbol{Ax} = \lambda \boldsymbol{x} \tag{14}$$

Realizando as devidas manipulações algébricas.

$$\boldsymbol{x} = \frac{adj(\lambda \boldsymbol{I} - \boldsymbol{A})}{\det(\lambda \boldsymbol{I} - \boldsymbol{A})}\boldsymbol{0} \tag{15}$$

Os valores de λ é calculado forçando o denominador para zero.

$$\det(\lambda \boldsymbol{I} - \boldsymbol{A}) = \boldsymbol{0} \tag{16}$$

Convertendo um sistema representado em equações de estado para uma função de transferência:

Utilizando a transformada de Laplace nas Eq. 8 e Eq. 9

$$s\boldsymbol{X}(s) = \boldsymbol{AX}(s) + \boldsymbol{BU}(s) \tag{17}$$

$$\boldsymbol{Y}(\boldsymbol{s}) = \boldsymbol{CX}(s) + \boldsymbol{DU}(s) \tag{18}$$

Resolvendo $\boldsymbol{X}(\boldsymbol{s})$:

$$(s\boldsymbol{I} - \boldsymbol{A})\boldsymbol{X}(s) = \boldsymbol{BU}(s)$$

$$\boldsymbol{X}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{BU}(s) \tag{19}$$

Substituindo a Eq. 19 na Eq.18:

$$\boldsymbol{Y(s) = C(sI - A)^{-1}BU}(s) + \boldsymbol{DU}(s) = [\boldsymbol{C(sI - A)^{-1}B + D}]\boldsymbol{U(s)} \quad (20)$$

Assim, sendo $Y(s)$ e $U(s)$ valores escalares.

$$\frac{\boldsymbol{Y(s)}}{\boldsymbol{U(s)}} = \boldsymbol{C(sI - A)^{-1}B + D} \quad (21)$$

Encontrando os polos da Eq. 21 através do determinante do denominador.

$$det(\boldsymbol{sI - A}) = 0 \quad (22)$$

Logo como as Eq. (22) e (16) são idênticas é correto afirmar que os polos de um sistema representado por equações de espaço de estados são iguais aos autovalores da matriz **A.**

2.2.2. CONTROLABILIDADE E OBSERVABILIDADE

De acordo com Gawronski (2004), analisando um sistema linear e não variante no tempo descrito pelas variáveis de estado (x) e provocado por uma entrada (u) e mensurada por uma saída (y), é possível que algumas variáveis de estado não sejam excitadas pela entrada ou as variando arbitrariamente e similarmente, há sistemas que possuem variáveis que não podem ser estimadas através dos dados mensuráveis. Em sistemas de controle estes casos são conhecidos como sistemas não controláveis ou parcialmente controláveis e não observáveis ou parcialmente observáveis.

A Observabilidade e controlabilidade são características de um sistema de grande importância ao buscar a estabilidade projetando o controlador e o observador. A definição destas caracteristicas para o controle e estimação de estados do sistema é de suma importância, sendo o conceito introduzido por Kalman (OGATA, 2009). Tais propriedades do sistema desempenham um grande papel na resolução de problemas de controle, sendo a grande maioria dos sistemas físicos controláveis e observáveis (OGATA, 2009), pois elas ditam a existência de uma solução para o problema sem grandes alterações do sistema analisado.

2.2.2.1. Controlabilidade

"Um sistema será dito controlável no instante t_0 se for possível, por meio de um vetor de controle não limitado, transferir o sistema de qualquer estado inicial $x(t_0)$ para qualquer outro estado, em um intervalo de tempo infinito" (OGATA, 2009, P.675)

Para determinar a controlabilidade é necessário fazer a análise da matriz de controlabilidade $\boldsymbol{C}_{ctr}$ que é formada pelos seguintes vetores coluna (n x 1):

$$\boldsymbol{C}_{ctr} = [\boldsymbol{B} \quad \boldsymbol{AB} \quad \boldsymbol{A^2B} \quad \boldsymbol{A^3B} \quad \ldots \quad \boldsymbol{A^{n-1}B}] \tag{23}$$

Um sistema é dito controlável se, e somente se, os vetores forem linearmente independentes, ou seja, o posto da matriz $\boldsymbol{C}_{ctr}$(n x n) ser igual a n. (SILVA e FONTES, 2017)

A controlabilidade é uma característica inerente ao sistema, que todo sistema é dito como completamente controlável quando com uma força de entrada $u(t)$ o sistema consegue controlar todas as variáveis de estado (NISE, 2013).

Entretanto por mais que o sistema seja completamente controlável em muitos sistemas não é possível controlar sem utilizar a matriz $\boldsymbol{C}$ como uma matriz identidade (n x n), ou seja, lendo todos os sinais do sistema, logo, se faz necessário a estimação dos estados para conseguir controlar o sistema.

2.2.2.2. Observabilidade

Muitos problemas de controle o usuário não possui acesso a todas as grandezas do sistema ou não possui transdutores para conseguir a leitura das mesmas, além de outras avarias que os sensores podem sofrer (NISE, 2013). Contudo, por mais que nem todas variáveis de estado estejam disponíveis, se faz necessário um observador que irá estimar as grandezas a partir de outras grandezas mensuráveis e realimentar las ao sistema.

Para determinar a possibilidade de todas as variáveis de estado serem estimadas, cada transição do estado deve influenciar cada elemento no vetor de saída (SILVA e FONTES, 2017).

Logo é considerado a matriz de Observabilidade $\boldsymbol{O}_{\boldsymbol{obs}}$(m x n):

$$\boldsymbol{O}_{\boldsymbol{obs}} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA^2} \\ \boldsymbol{CA^3} \\ \vdots \\ \boldsymbol{CA^{n-1}} \end{bmatrix} \tag{24}$$

Ou

$$\boldsymbol{O}_{\boldsymbol{obs}} = [\boldsymbol{C'} \quad \boldsymbol{C'A'} \quad \boldsymbol{C'A'^2} \quad \ldots \quad \boldsymbol{C'A'^{n-1}}] \tag{26}$$

Um sistema é dito completamente observável quando o posto da matriz de observabilidade $\boldsymbol{O}_{\boldsymbol{obs}}$ é igual a n. Assim, é possível estimar todas as variáveis do sistema a partir das grandezas mesuráveis do sistema.

2.3. PROJETO DE OBSERVADORES

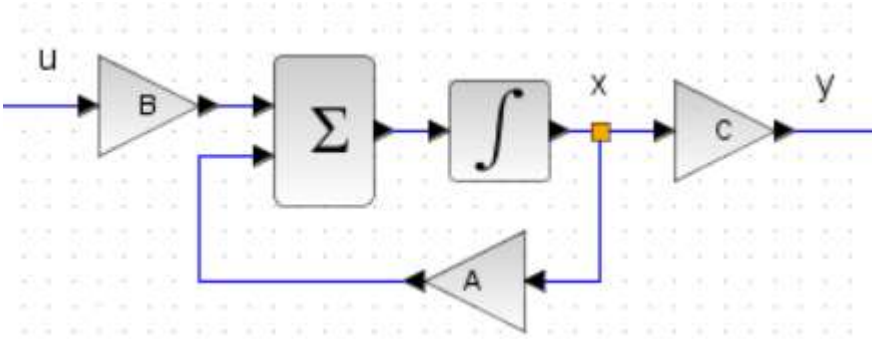
Em teoria todos os estados de um sistema dinâmico, representado em equações de espaço de estados, estão disponíveis para projetar o controlador necessário. Porém, na prática muitas grandezas não estão disponíveis para mensuração, requerem transdutores caros ou para obter o valor necessário será necessária uma grande quantidade de sensores. Em casos como esse é desejado projetar um observador para estimar as variáveis não mensuradas.

De acordo com Ogata (2009), em sistemas de controle a estimação de variáveis não mensuráveis é comumente denominada observação, sendo o programa que estimará ou observará é denominado observador de estado ou observador e o vetor de estados observados é representando por $\tilde{\boldsymbol{x}}$.

Em um sistema com n variáveis de estado, um observador que estima todas as variáveis de estado é denominado observador de ordem plena. Mas em alguns casos não é requerido observar todas as variáveis mensuradas, apenas as não mensuráveis, logo o sistema possui n variáveis de estado a necessidade do usuário requer a observação de m variáveis, sendo $n > m$, estes casos o observador é denominado de observador de ordem reduzida (OGATA, 2009).

Considerando a representação de um sistema em equações de espaço de estados em um diagrama de blocos sem perturbações externas (Figura 3):

Figura 3: Diagrama de blocos do sistema no *software* Scilab sem perturbações externas



FONTE:Próprio Autor

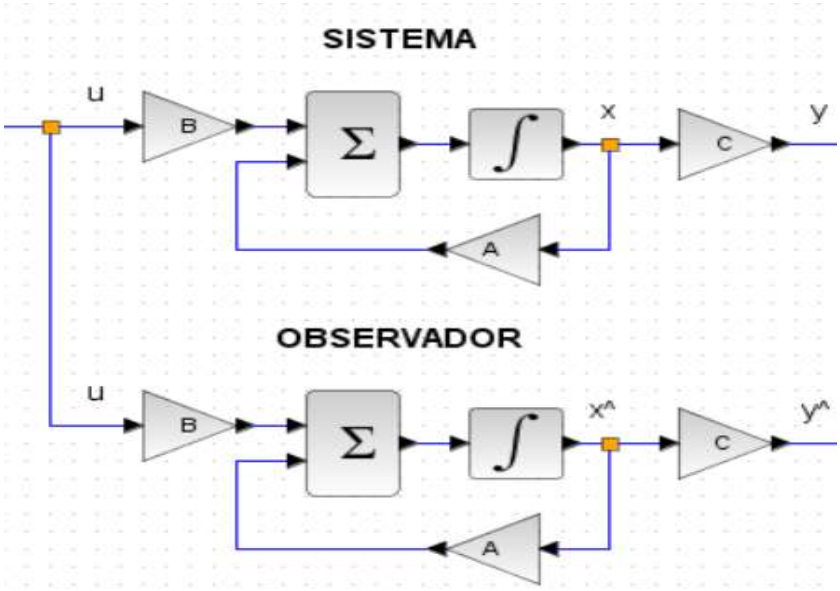
$$\dot{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{Bu}(\boldsymbol{t})$$
$$y(t) = \boldsymbol{C}\boldsymbol{x}(t) \qquad (27)$$

O observador estima as variáveis através do sinal de reposta $\boldsymbol{y}$ do sistema real, para que o observador possa desempenhar seu papel é essencial que a característica do sistema, observabilidade, seja completamente observável. O observador é um subsistema que reconstrói

o sistema em análise sendo excitado com a mesma perturbação se assemelhando com o sistema real.

A figura 4 ilustra um sistema de observação em malha aberta em relação ao sistema real (eq. 28), ou seja, os estados observados do sistema dinâmico podem estar atuando de um modo diferente ao real ou o observador possuirá uma velocidade vagarosa de convergência de resposta.

Figura 4: Diagrama de blocos do observador independente do sistema

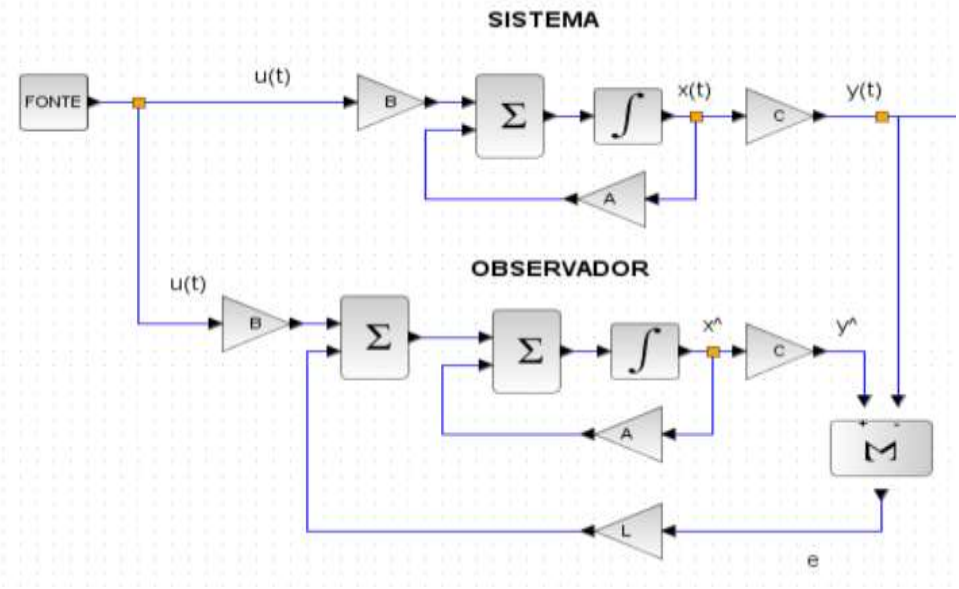


FONTE: Autor

$$\dot{\tilde{\boldsymbol{x}}} = \boldsymbol{A}\tilde{\boldsymbol{x}}(\boldsymbol{t}) + \boldsymbol{B}\boldsymbol{u}(\boldsymbol{t})$$
$$\tilde{\boldsymbol{y}} = \boldsymbol{C}(\boldsymbol{t}) \qquad (28)$$

Para o observador acompanhar a dinâmica do sistema, em uma velocidade razoável, é necessário que o mesmo seja alimentado com o erro de estimação $\boldsymbol{e}$ (eq. 29) do sistema retratado pela Figura 5, que consiste na diferença entre os estados reais e os estados estimados, e também um ganho $\boldsymbol{L}$ para determinar a rapidez que os estados estimados serão corrigidos.

Figura 5: Diagrama de blocos do observador dependente do sistema



FONTE: AUTOR

$$\boldsymbol{e} = \boldsymbol{x} - \tilde{\boldsymbol{x}} \tag{29}$$

O sistema da Figura 5 é descrito por:

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}} &= \boldsymbol{A}\tilde{\boldsymbol{x}} + \boldsymbol{B}u + \boldsymbol{L}(\boldsymbol{y} - \boldsymbol{C}\tilde{\boldsymbol{x}}) \\ \dot{\tilde{\boldsymbol{x}}} &= (\boldsymbol{A} - \boldsymbol{LC})\tilde{\boldsymbol{x}} + \boldsymbol{B}u + \boldsymbol{Ly} \end{aligned} \tag{30}$$

Subtraindo a primeira porção da eq.(28) com a eq.(30):

$$\begin{aligned} \dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}} &= \boldsymbol{Ax} - \boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{L}(\boldsymbol{Cx} - \boldsymbol{C}\tilde{\boldsymbol{x}}) \\ \dot{\boldsymbol{x}} - \dot{\tilde{\boldsymbol{x}}} &= (\boldsymbol{A} - \boldsymbol{LC})(\boldsymbol{x} - \tilde{\boldsymbol{x}}) \end{aligned} \tag{31}$$

Logo, substituindo a eq.(29) na eq.(31), é obtido a eq.29:

$$\dot{\boldsymbol{e}} = (\boldsymbol{A} - \boldsymbol{LC})\boldsymbol{e} \tag{32}$$

Analisando a eq.(32) é verídico determinar que a dinâmica do erro do observador é estabelecida pelos autovalores da matriz $\boldsymbol{A} - \boldsymbol{LC}$, e se caso a matriz for estável o erro do observador convergira para zero.

O ganho $\boldsymbol{L}$ do observador é determinado a partir da resolução da eq.(33):

$$\boldsymbol{det}(\lambda \boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{LC})) \tag{33}$$

2.3.1. Alocação de polos

Como técnicas de controle, há várias técnicas para determinar a velocidade que o observador precisa estimar considerando as características do sistema ou dos transdutores utilizados, utilizando filtros, baseado no controle ou determinando uma grande velocidade para o estimador alocando os polos ao extremo esquerdo do plano imaginário, sendo este fadado a erro em caso de transdutores ruidosos, para isso é necessário um equilíbrio nas velocidades para não processar muitos dados errôneos devido a perturbação causada pelo ruído, em casos de sistema ruidosos é recomendável o filtro de Kalman (NISE, 2013).

O projeto de observadores é similar ao projeto de um controlador, mas ele é realizado a parte do projeto do controlador (NISE, 2013). A técnica de alocação determina os polos do observador ou controlador para a posição desejada, assim controlando a velocidade que um sistema convergirá para a estabilidade ou a velocidade de estimação dos estados.

Entretanto para que esta técnica de controle e observação seja possível aplicar ela requer que certas condições sejam cumpridas como:

- O estado do sistema requer ser completamente controlável, em casos de projetos de controladores;
- O estado do sistema requer ser completamente observável, em casos de projetos de observadores.

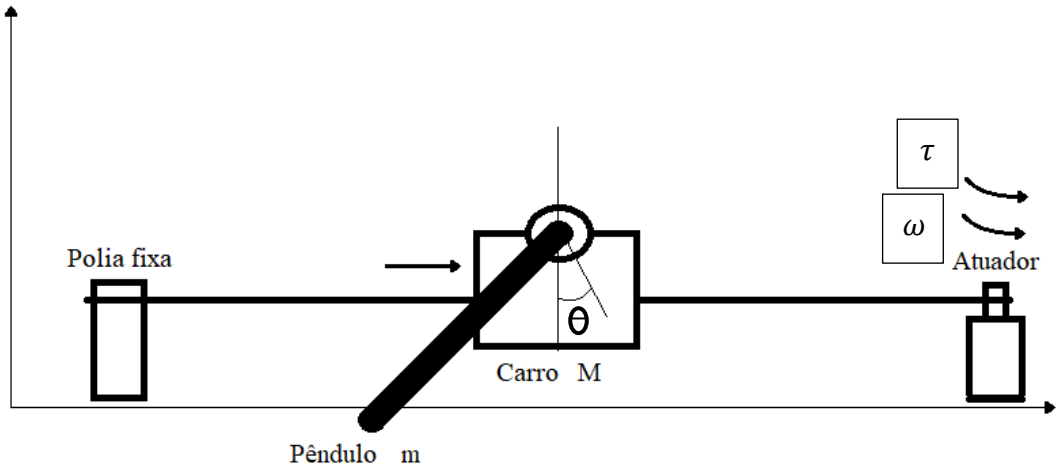
Capítulo 3. METODOLOGIA

Neste capítulo será apresentado o modelo matemático do sistema, considerando a dinâmica do atuador, como foi realizado a aquisição dos sinais e a própria construção do modelo em análise.

3.1. MODELO MATEMÁTICO

A Figura 6 representa esquematicamente o funcionamento da planta considerando seu atuador, neste caso o motor de corrente contínua.

Figura 6: Modelo matemático do conjunto pêndulo-carro



FONTE: AUTOR

Para este conjunto da Figura 6 inicialmente é definido o centro de gravidade da haste:

$$X_g = x + l\sin(\theta) \tag{34}$$

$$Y_g = l\cos(\theta) \tag{35}$$

Assim considerando que a haste possui movimentos, tanto na horizontal quanto na vertical e movimento rotacional.

$$I\ddot{\theta} = Vl\sin\theta - Hl\cos\theta \tag{36}$$

$$m\frac{d^2}{dt^2}(x + l\sin\theta) = H \tag{37}$$

$$m\frac{d^2x}{dt^2}(l\cos(\theta)) = V - mg \tag{38}$$

O movimento do carro é descrito pela eq.(39).

$$M\frac{d^2x}{dt^2} = f - H \tag{39}$$

Admitindo uma variação suficientemente pequena em sua posição angular e da velocidade angular é cabível admitir $\sin\theta_1 \cong \theta$, $\cos\theta_1 \cong 1$ e $\theta_1\dot{\theta}_1 = 0$, assim as equações de (36) a (38) podem ser linearizadas, resultando nas equações (40), (41) e (42):

$$I\ddot{\theta_1} = Vl\theta - Hl \tag{40}$$

$$m\frac{d^2}{dt^2}(x + l\theta) = H \tag{41}$$

$$0 = V - mg \tag{42}$$

Fazendo as devidas manipulações matemáticas com as eq.(39), (40), (41) e (42) é possível resumir a dinâmica do conjunto pêndulo-carro em duas equações (43) e (44).

$$(M + m)\ddot{x} + ml\ddot{\theta} = f \tag{43}$$

$$(I + ml^2)\ddot{\theta} + ml\ddot{x} = mlg\theta \tag{44}$$

Para este sistema o momento de inércia do pêndulo é desconsiderado por não possuir uma grande magnitude.

$$I_{cm} = \frac{1}{4}mR^2 + \frac{1}{12}ml^2 \tag{45}$$

$$I_{cm} = 0.0002984\ kg.m^2$$

A representação do sistema em espaço de estados é iniciada com as equações de movimento que regem o conjunto.

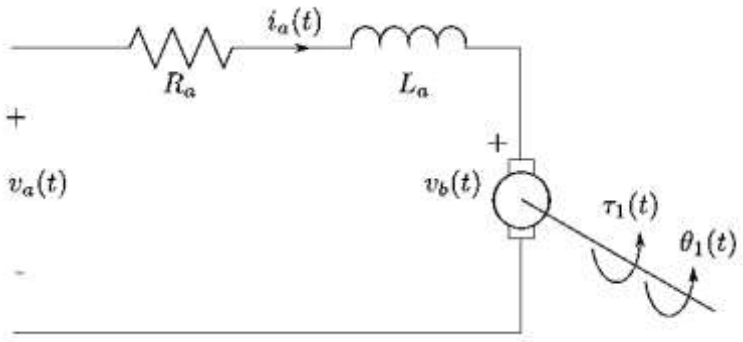
$$Ml\ddot{\theta} = (M + m)g\theta - f \tag{46}$$

$$M\ddot{x} = f - mg\theta \tag{47}$$

O atuador do conjunto pêndulo-carro é um motor de corrente continua, logo, é possível equacionar a força perturbando o sistema. Fazendo a análise do comportamento do motor de corrente contínua necessária para maior aproximação do modelo numérico e o experimental.

A Figura 7 acima representa o esquema elétrico de armadura de um motor de corrente continua.

Figura 7: Diagrama elétrico de um motor de corrente contínua



FONTE: Siradjuddin, Indrazno *et al*

Através da lei de Kirshoff é possível encontrar a equação (48):

$$i_a R_a + L_a \frac{di_a}{dt} + v_b = v_a \tag{48}$$

O indutor dentro do rotor é pequeno, logo é desconsiderado,(SIRADJUDDIN, 2017) assim a equação (48) é resumida para equação (49):

$$i_a R_a + + v_b = v_a \tag{49}$$

A força contra eletromotriz (emf), v_b, é proporcional a velocidade rotacional do atuador:

$$v_b = K_b \frac{d\theta}{dt} \tag{50}$$

Considerando que o torque de um motor de corrente contínua é proporcional a corrente de armadura, logo,

$$\tau = K_t i_a \tag{51}$$

Assim, substituindo a eq. (51) e eq. (50) na eq. (49) obtemos a eq.(52):

$$\frac{\tau}{K_t} R_a + K_b \frac{d\theta}{dt} = v_a \tag{52}$$

Realizando as devidas manipulações algébricas é possível afirmar que o torque do motor é representado pela equação (53):

$$\tau = -K_t \frac{K_b}{R_a} \omega + \frac{K_t}{R_a} v_a \tag{53}$$

Logo a força atuante no sistema é representada:

$$f = \frac{K_t}{R_a r} \left(-K_b \frac{\dot{x}}{r} + v_a\right) \tag{54}$$

As equações que representam a dinâmica da planta, considerando a força atuadora do motor, são alteradas substituindo a eq. (54) nas eq. (46) e eq. (47):

$$Ml\ddot{\theta} = (M+m)g\theta - \frac{K_t}{R_a r} \left(-K_b \frac{\dot{x}}{r} + v_a\right) \tag{55}$$

$$M\ddot{x} = \frac{K_t}{R_a r} \left(-K_b \frac{\dot{x}}{r} + v_a\right) - mg\theta \tag{56}$$

Definindo as variáveis de estado como $x_1, x_2, x_3\ e\ x_4$ sendo as mesmas representando as grandezas $\theta, \dot{\theta}, x\ e\ \dot{x}$ do sistema (OGATA, 2009). Pela definição de variáveis de estado obtemos que:

$$\dot{x}_1 = x_2 \tag{57}$$

$$\dot{x}_2 = \left(\frac{M+m}{Ml}\right) g x_1 + \frac{K_b K_t x_4}{R_a r^2 Ml} - \frac{K_t v_a}{R_a r Ml} \tag{58}$$

$$\dot{x}_3 = x_4 \tag{59}$$

$$\dot{x}_4 = -\frac{m}{M} g x_1 - \frac{K_b K_t x_4}{R_a r^2 M} + \frac{K_t v_a}{R_a r M} \tag{60}$$

Assim, o sistema pêndulo-carro pode ser representado pelas matrizes:

$$\boldsymbol{A} = \begin{vmatrix} 0 & 1 & 0 & 0 \\ \left(\frac{M+m}{Ml}\right) g & 0 & 0 & \frac{K_b K_t}{R_a r^2 Ml} \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M} g & 0 & 0 & -\frac{K_b K_t x_4}{R_a r^2 M} \end{vmatrix}$$

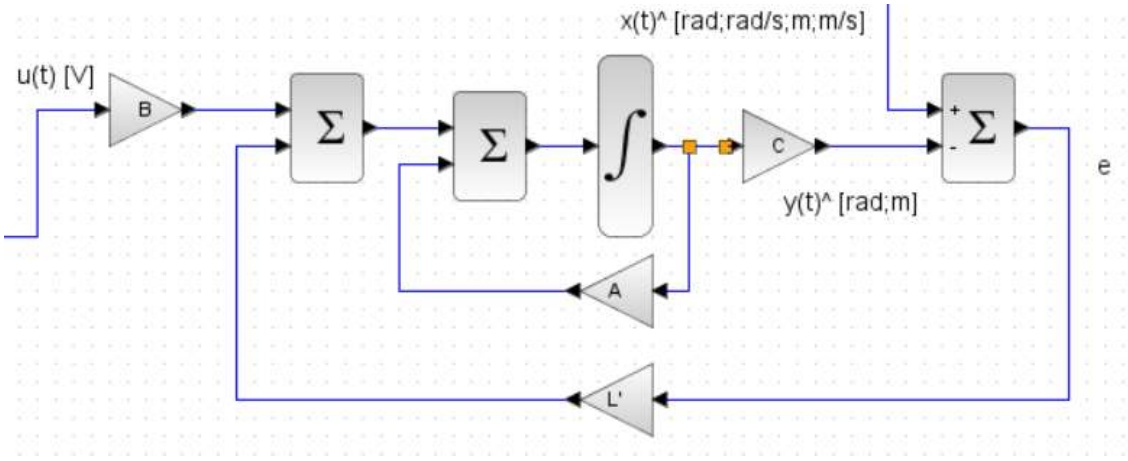
$$\boldsymbol{B} = \begin{vmatrix} 0 \\ -\dfrac{K_t}{R_a r M l} \\ 0 \\ \dfrac{K_t}{R_a r M} \end{vmatrix}$$

$$\boldsymbol{C} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

$$\boldsymbol{D} = 0$$

Admitindo um sistema sem perturbações externas a matriz **D** de saída transmissão direta é nula e considerando a dinâmica do atuador o observador do sistema é retratado pela **Erro! Autoreferência de indicador não válida.**.

Figura 8: Observador do conjunto pêndulo-carro



FONTE: Próprio Autor

3.2. AQUISIÇÃO DOS SINAIS E ATUADOR

Nesta sessão será explanado sobre como foi realizado a aquisição dos estados mensuráveis para elaborar o projeto do observador. O hardware utilizado foi o Arduino Uno representado pela Figura 9.

Figura 9: Arduino Uno

FONTE: *http://labdegaragem.com*

O Arduino é uma plataforma de *hardware* livre com suporte de entrada e saída embutido. A ligugem de programação utilizada nesse hardware é baseada na linguagem C/C++. Sendo a programação carregada ao Arduino, encontrada no APÊNDICE A, é específica para utilizar conjunto a biblioteca Arduino-Scilab, caso contrário o Scilab não reconhece programações diferente desta.

3.2.1. Ângulo

Para a aquisição angular do pêndulo é utilizado um potenciômetro linear de 100 $k\Omega$ como é ilustrado na Figura 10.

Figura 10: Potenciômetro linear de 100 kΩ

Fonte: www.filipeflop.com

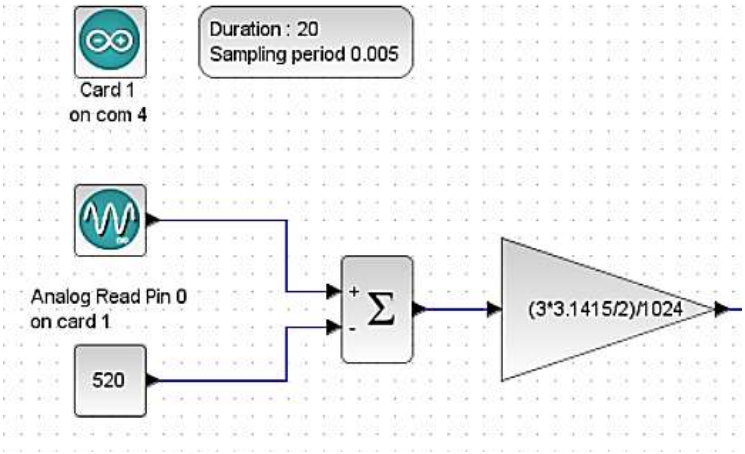
O potenciômetro possui um comportamento linear variando a resistência, assim variando a tensão de resposta. Utilizando a conversão de tensão para bits, é possível encontrar

uma função relacionando a reposta que o Arduino mensura com a rotação do potenciômetro, representado pela eq.(61).

$$\theta = \frac{\pi}{1023} Bit \tag{61}$$

A Figura 11 representa o diagrama de blocos para aquisição de dados e tradução de Bit para rad do pêndulo. O ganho de 520 bit representado juntamente na Figura 11 tem função de estabelecer o "zero" do pêndulo, este ganho pode ser alterado se necessário.

Figura 11: Diagrama de blocos no Scilab de aquisição do pêndulo

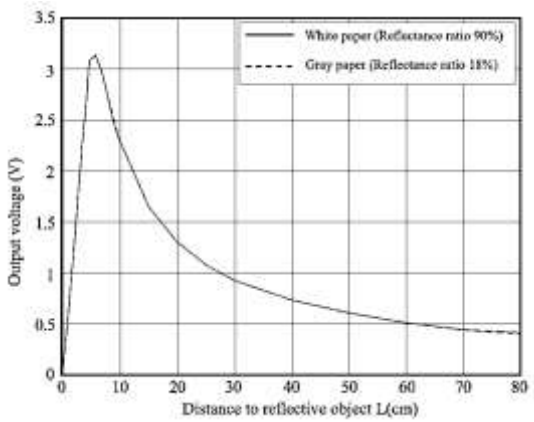


FONTE: Próprio Autor

3.2.2. Carro

Para a aquisição da posição do carro é utilizado o sensor Sharp do modelo GP2Y0A21YK0F. O transdutor converte a distância do carro para uma determinada voltagem. O sensor não é linear e segue o princípio da Figura 12.

Figura 12: Curva de Tensão de Saída por Distância do objeto refletido



FONTE: *datasheet*

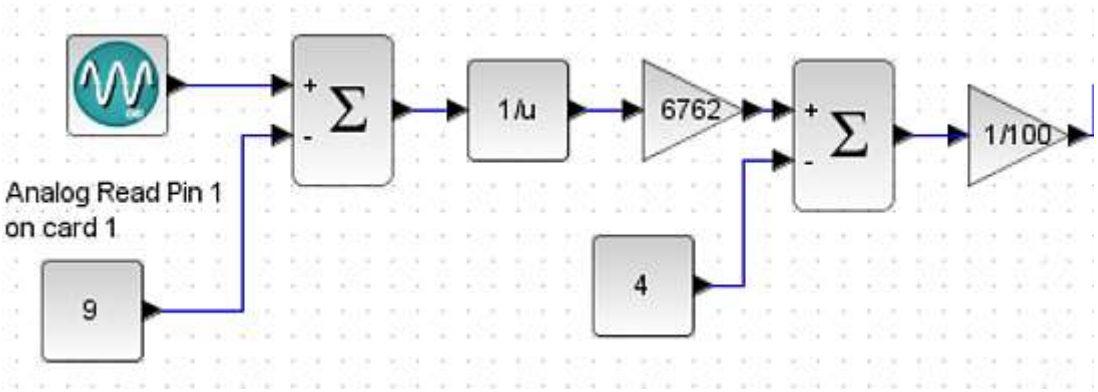
A figura 12 apresenta a voltagem de saída do sensor pela distância refletida. Para a conversão dos dados obtidos do sensor para metros é utilizada a equação (62).

$$L = \left(\frac{\frac{6762}{(bitD - 9)} - 4}{100} \right) \tag{62}$$

A eq. (62) foi obtida através de referência do site revendedor do transdutor e devolve um valor em cm.

A Figura 13 retrata o diagrama de bloco para a conversão da leitura que o transdutor de distância recebe para sistema internacional.

Figura 13: Diagrama de blocos no Scilab de aquisição do sensor Sharp



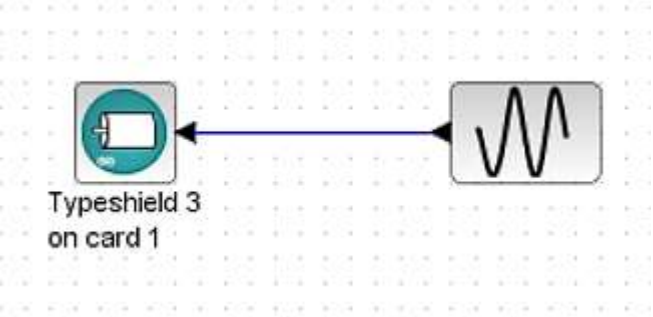
FONTE: Próprio Autor

3.2.3. Atuador

O sistema pêndulo-carro foi excitado por uma fonte senoidal, utilizando um motor de corrente conítnua como atuador.

O motor de corrente contínua funciona através dos campos magnéticos gerados pela corrente de armadura.

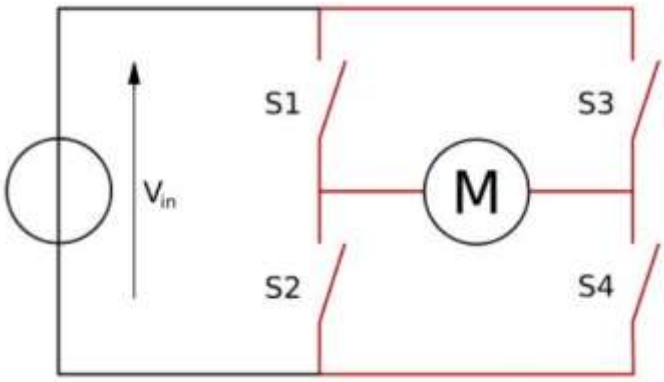
Figura 14: Diagrama de bloco de excitação do motor



FONTE: Próprio Autor

Para que o conjunto seja excitado por uma fonte é necessário que a direção e torque do motor sejam alterados. Assim, é necessário a utilização de *driver* que controle tanto a direção como a velocidade do atuador. Há inúmeros meios de saciar essa necessidade como o uso de pontes H.

Figura 15: Configuração de ponte H

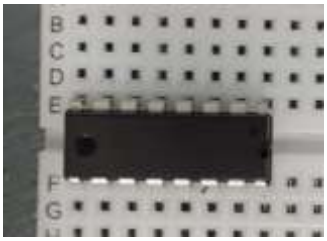


FONTE: *http://www.bosontreinamentos.com.br*

A ponte H é uma configuração de circuito que permite a inversão da direção de um motor de corrente contínua, através da inversão da polaridade da corrente que flui através de uma carga. Esta configuração é comumente encontrada em circuitos integrados como L293D, L298, LMD18200 e L9910.

Para a excitação do conjunto será utilizado o circuito integrado L293D (Figura 16), pois sua fácil configuração, pouco aquecimento do componente e por possuir quatro meias ponte H o tornam a escolha para esta aplicação.

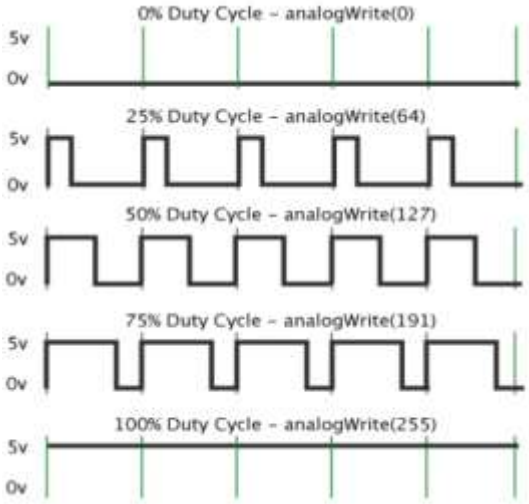
Figura 16: L293D encaixado na *protoboard*



FONTE: Próprio Autor

Para controle da velocidade é utilizada a técnica PWM (Modulação por Largura de Pulso), que consiste em controlar a velocidade através da variação do valor médio de uma forma de onda periódica. Para variar velocidade, a técnica mantém uma de frequência , de onda quadrada, fixa (Figura 17) e varia o tempo que o sinal permanece em nível lógico alto ($5V$). Está técnica é 8 bits, ou seja, varia de 0 a 255 *bit* para alterar a velocidade do motor.

Figura 17: Onda quadrada da técnica *PWM*



FONTE: www.arduino.cc

As características do motor foram descobertas através de informações do fabricante (mabuchi-motor):

Tensão:

- Faixa de operação: 9 – 30 V
- Nominal: 12 V

Sem carga:

- Velocidade: 4800 rad/min
- Corrente: 0.17 A

Eficiência máxima: tensão 12V

- Velocidade: 4240 rad/min
- Corrente: 1.3 A
- Torque: 25.6 mN.m
- Torque: 261 g.cm
- Potência: 11.4 W

Máximo:

- Torque: 221 mNm
- Torque: 2253 g.cm
- Corrente: 9.9 A

Utilizando torque máximo e corrente máxima com a eq. (51):

$$\tau = K_t i_a$$

$$0{,}221 = 9{,}9K_t$$

$$K_t = 0.0223232 \text{ Nm/A}$$

Utilizando a velocidade sem carga e a voltagem nominal com a eq. (50):

$$v_b = K_b \frac{d\theta}{dt}$$

$$12 = 4800 \times 2\pi \times K_b$$

$$K_b = 0.023873241 \text{ V.s/rad}$$

Utilizando a eq. (53):

$$\tau = -K_t \frac{K_b}{R_a}\omega + \frac{K_t}{R_a} v_a$$

$$0{,}0256 = \frac{(0{,}0223232 \times 0{,}23873241 \times 4240 \times \left({}^{2\pi}/_{60}\right) + 0{,}0223232)}{R_a}$$

$$R_a = 1{,}2208\ \Omega$$

3.3. CONSTRUÇÃO DO CONJUNTO PÊNDULO-CARRO

Neste trabalho para a construção da planta, a Figura 18 representa a impressora hp 3535 reciclada.

Figura 18: Impressora HP 3535

FONTE: Próprio Autor

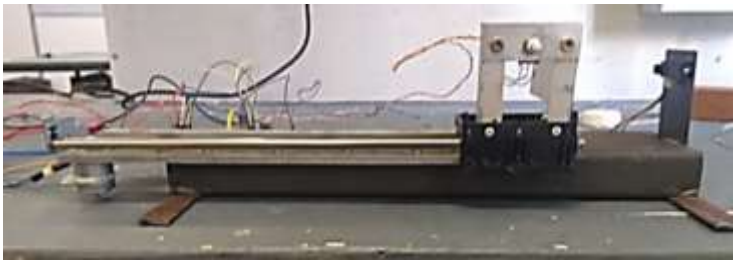
Onde da mesma foi aproveitado a haste Figura 19, que será utilizado como pêndulo com apenas algumas modificações para a fixação no suporte do carro.

Figura 19: Haste

FONTE: Próprio Autor

O suporte do carro da impressora que possui aproximadamente 0,39m, foi utilizado para limitar a trajetória do carro, e o motor de corrente contínua da impressora para excitar o pêndulo Figura 20.

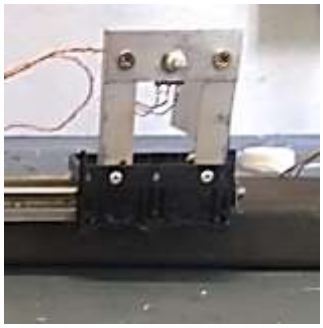
Figura 20: Suporte



FONTE: Próprio Autor

Para o suporte do conjunto pêndulo e potenciômetro, foram utilizadas placas de alumínio rebitadas ao carro como a Figura 21 representa.

Figura 21: Conjunto suporte do pêndulo-carro



FONTE: Próprio Autor

Para eliminação de ruídos devido vibrações excessivas foi utilizado uma viga H de aço, com aproximadamente 0,4m, e foi fixado, através de soldas, suportes para que caso necessário, com uso de sargentos, fixar a planta a mesa, como a Figura 22 representa, reduzindo os ruídos externos.

Figura 22: Visão superior do sistema

FONTE: Próprio Autor

Para evitar vibrações no transdutor Sharp, do modelo GP2Y0A21YK0F, o suporte do sensor foi soldado a viga H e considerando as especificações do sensor o mesmo está a uma distância de 10 cm do suporte do carro, representado na Figura 23, para melhor leitura do sinal, devido a confusão de valores que sua curva de mensuração apresenta (Figura 12).

Figura 23: Suporte do transdutor Sharp

FONTE: Próprio Autor

Capítulo 4. RESULTADOS E DISCUSSÕES

O experimento consistiu em excitar o carro com uma função senoidal em diferentes frequências e com diferentes taxas de coleta de dados, como a Tabela 2 mostra.

Tabela 2: Configurações do experimento

Taxa de Amostragem	Frequência 1 [rad/s]	Frequência 2 [rad/s]	Frequência 3 [rad/s]
200 [pontos/s]	π	$\frac{3}{2}\pi$	2π
20 [pontos/s]	π	$\frac{3}{2}\pi$	2π

4.1. VALIDAÇÃO DA SIMULAÇÃO

Para que a técnica de alocação de polos, utilizada para determinar a velocidade que o observador estimará os dados, seja válida é necessário confirmar a observabilidade total do sistema com a eq. (23).

$$\boldsymbol{O_{obs}} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA^2} \\ \boldsymbol{CA^3} \end{bmatrix} = \begin{bmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \\ \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} 0 & 1 & 0 & 0 \\ \left(\frac{M+m}{Ml}\right)g & 0 & 0 & \frac{K_b K_t}{R_a r^2 Ml} \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M}g & 0 & 0 & -\frac{K_b K_t x_4}{R_a r^2 M} \end{vmatrix} \\ \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} 0 & 1 & 0 & 0 \\ \left(\frac{M+m}{Ml}\right)g & 0 & 0 & \frac{K_b K_t}{R_a r^2 Ml} \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M}g & 0 & 0 & -\frac{K_b K_t x_4}{R_a r^2 M} \end{vmatrix}^{\mathbf{2}} \\ \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} 0 & 1 & 0 & 0 \\ \left(\frac{M+m}{Ml}\right)g & 0 & 0 & \frac{K_b K_t}{R_a r^2 Ml} \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M}g & 0 & 0 & -\frac{K_b K_t x_4}{R_a r^2 M} \end{vmatrix}^{\mathbf{3}} \end{bmatrix}$$

Realizando o posto da eq. (23) com os dados da planta é concluído que o sistema é completamente observável, sendo a $\boldsymbol{A}$ uma matriz 4x4.

$$posto(\boldsymbol{O_{obs}}) = 4$$

O ganho do observador foi calculado através da alocação de polos, os polos foram determinados pelo *ppol*, comando do scilab.

$$\mathbf{POLOS} = [\mathbf{-1000} \quad \mathbf{-1000} \quad \mathbf{-1000} \quad \mathbf{-1000}]$$

$$\boldsymbol{L} = \begin{bmatrix} \mathbf{2000} & \mathbf{1955702.9} & \mathbf{4415.8165} & \mathbf{4219907.2} \\ \mathbf{216.40701} & \mathbf{423213.46} & \mathbf{1955.6366} & \mathbf{913241.24} \end{bmatrix}$$

4.2. RESULTADO DAS SIMULAÇÕES

4.2.1. Taxa de amostragem 200 pontos/s

A seguir será apresentado os resultados obtidos através dos experimentos utilizando a taxa de aquisição em 200 pontos por segundo e variando a frequência da perturbação do atuador.

4.2.1.1. Frequência $\boldsymbol{\pi}$ $[\frac{\boldsymbol{rad}}{\boldsymbol{s}}]$

Figura 24: Deslocamento mensurado do carro

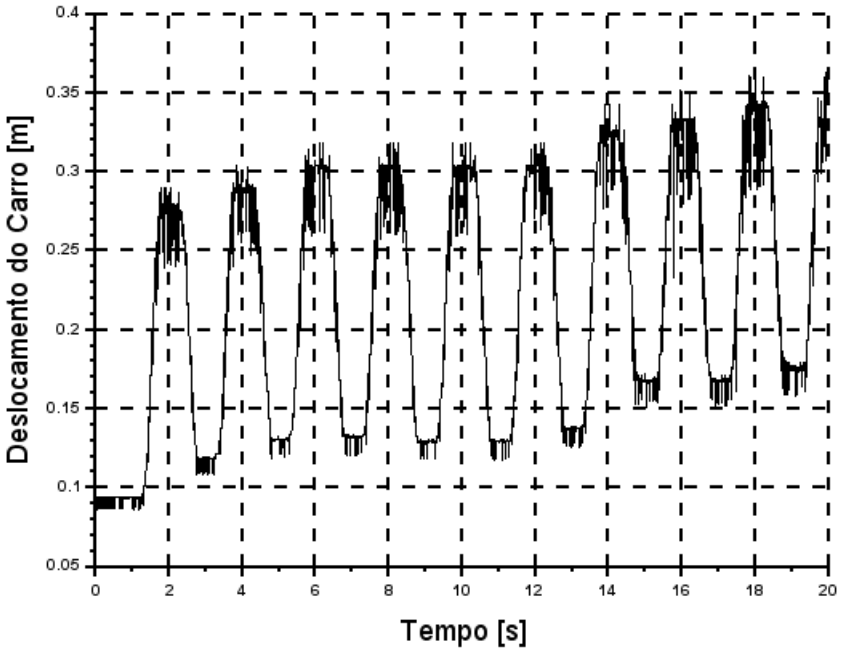


Figura 25: Deslocamento estimado do carro

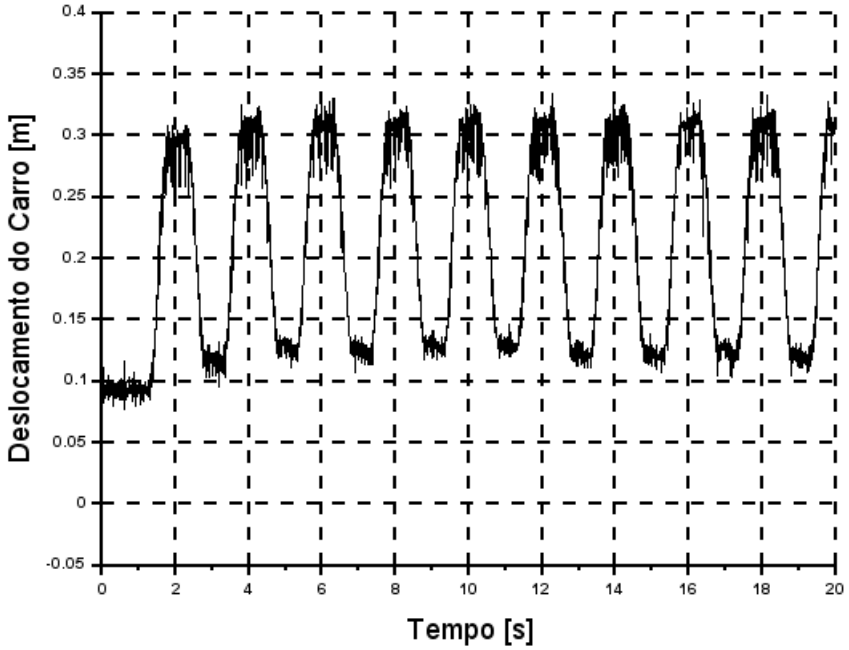


Figura 26: Deslocamento mensurado do pêndulo

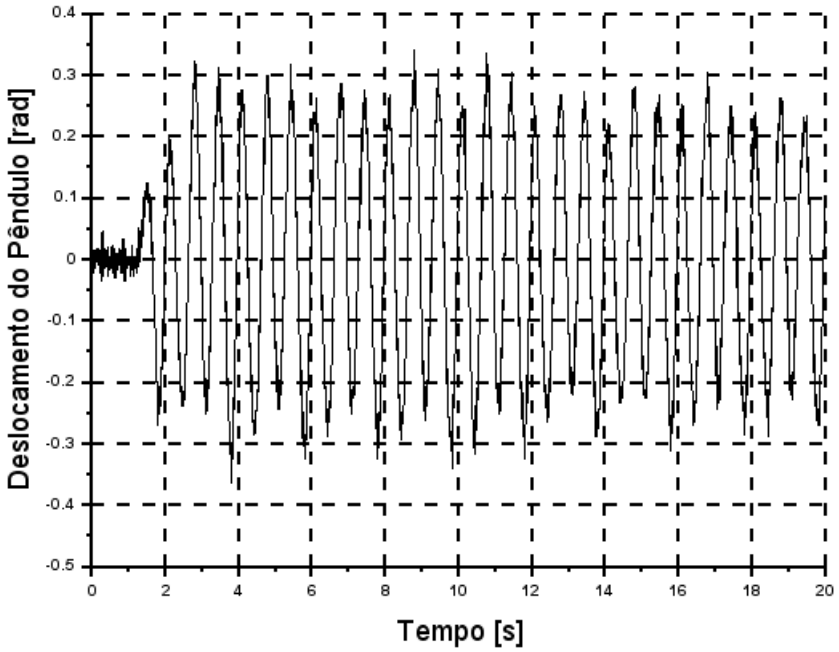
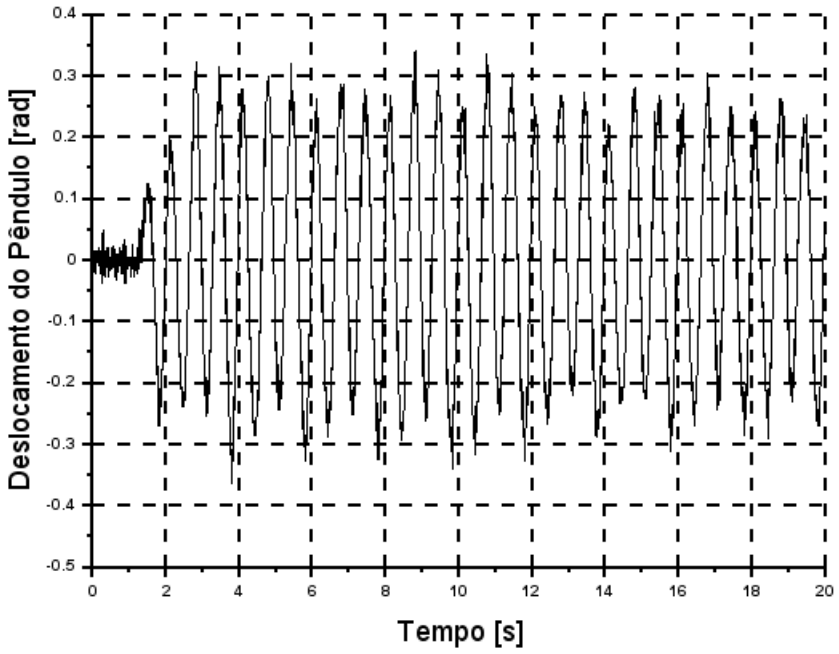


Figura 27: Deslocamento Estimado do pêndulo



4.2.1.2. Frequência $1.5\pi[\frac{rad}{s}]$

Figura 28: Deslocamento mensurado do carro

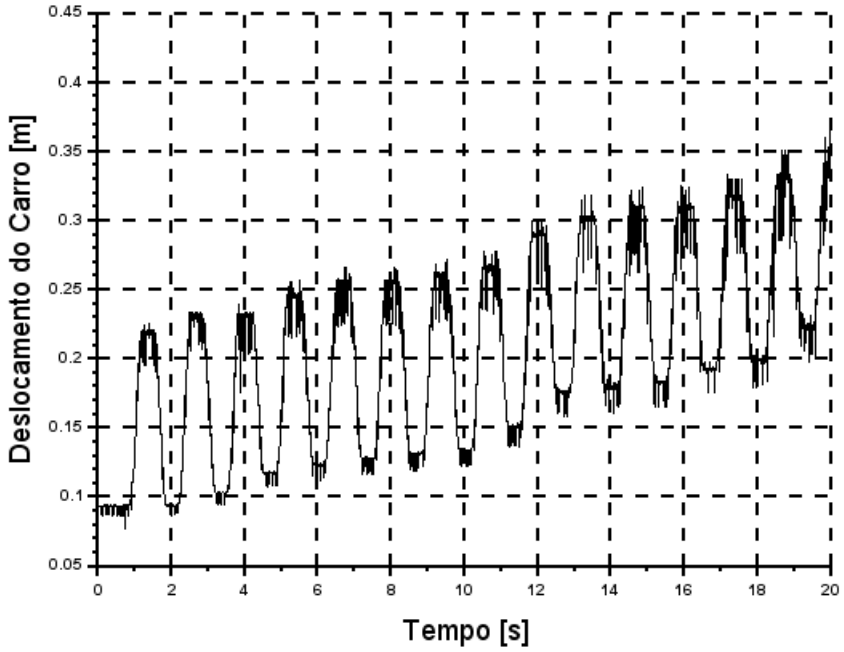


Figura 29: Deslocamento estimado do carro

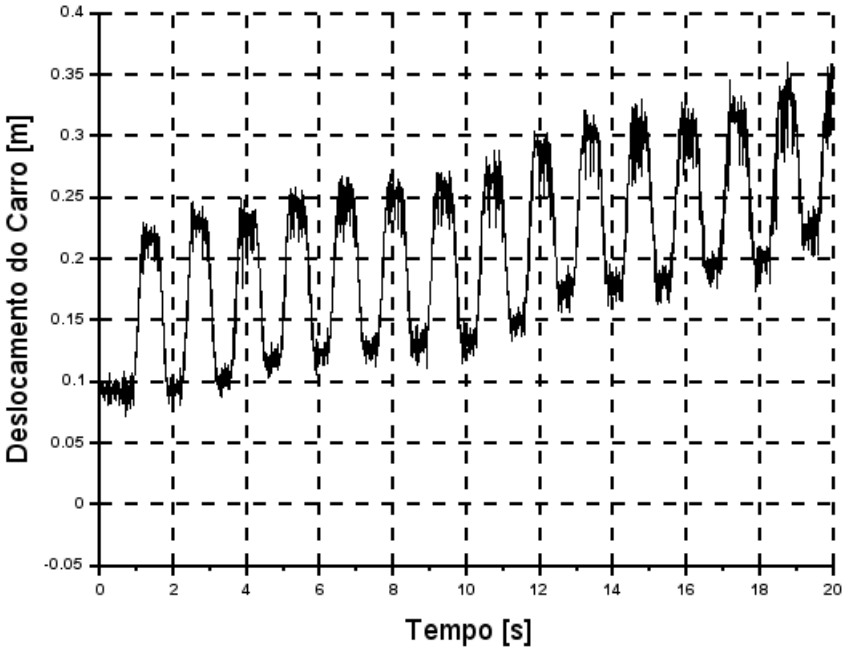


Figura 30: Deslocamento mensurado do pêndulo

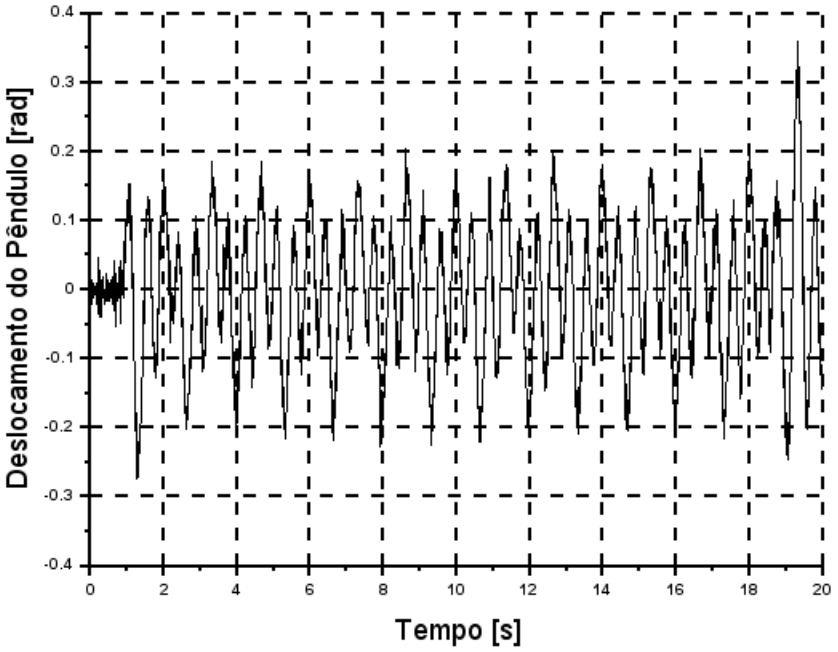
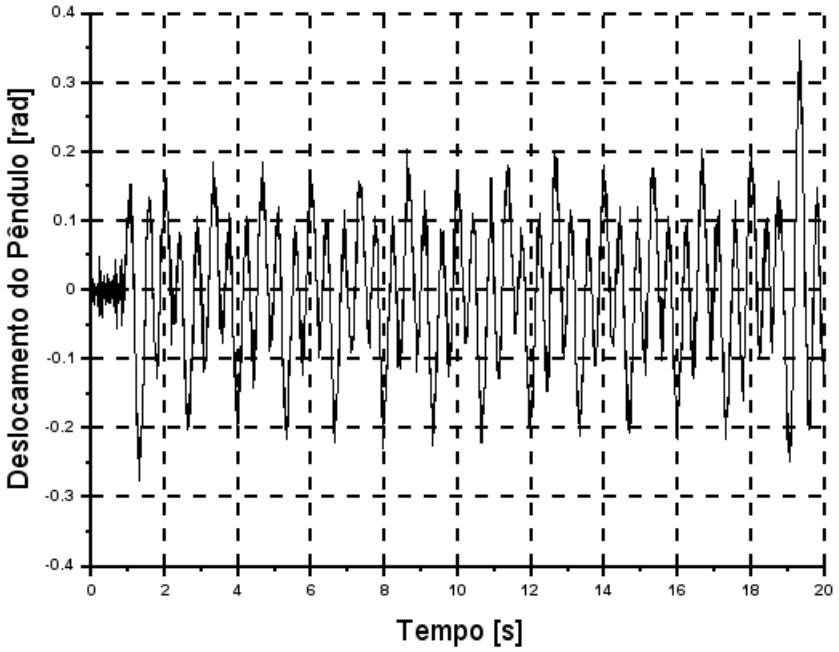


Figura 31: Deslocamento estimado do pêndulo



4.2.1.3. Frequência $2\pi[\frac{rad}{s}]$

Figura 32: Deslocamento mensurado do carro

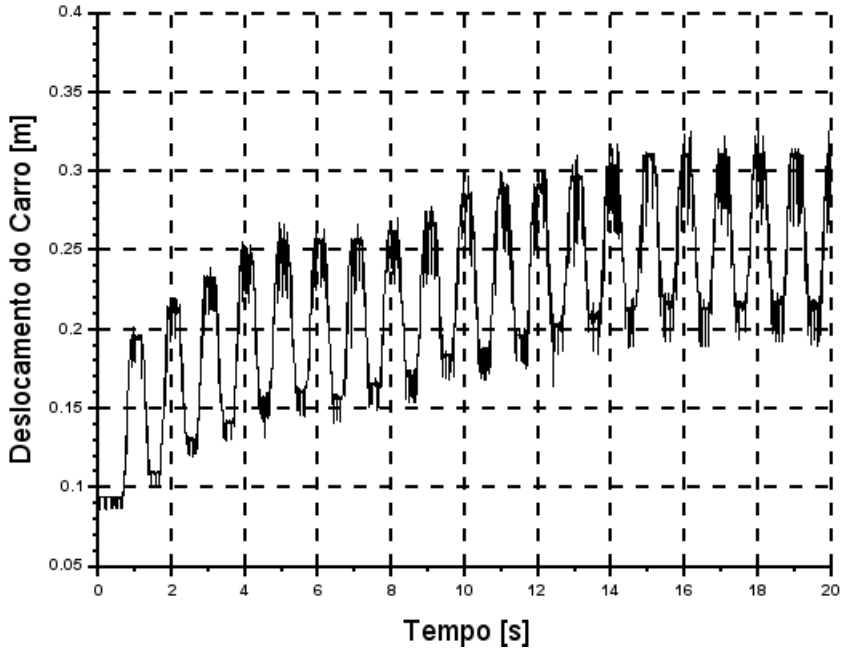


Figura 33: Deslocamento estimado do carro

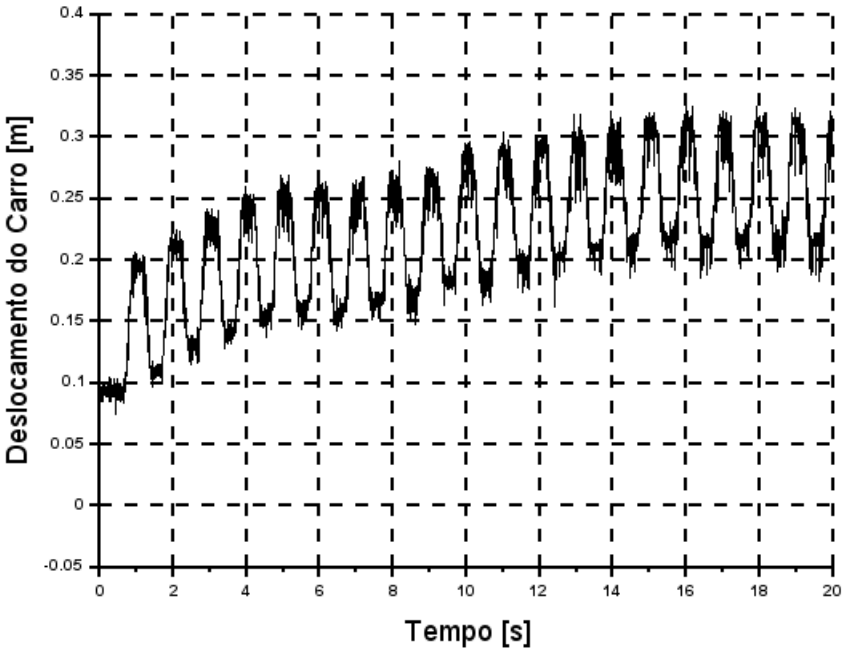


Figura 34: Deslocamento mensurado do pêndulo

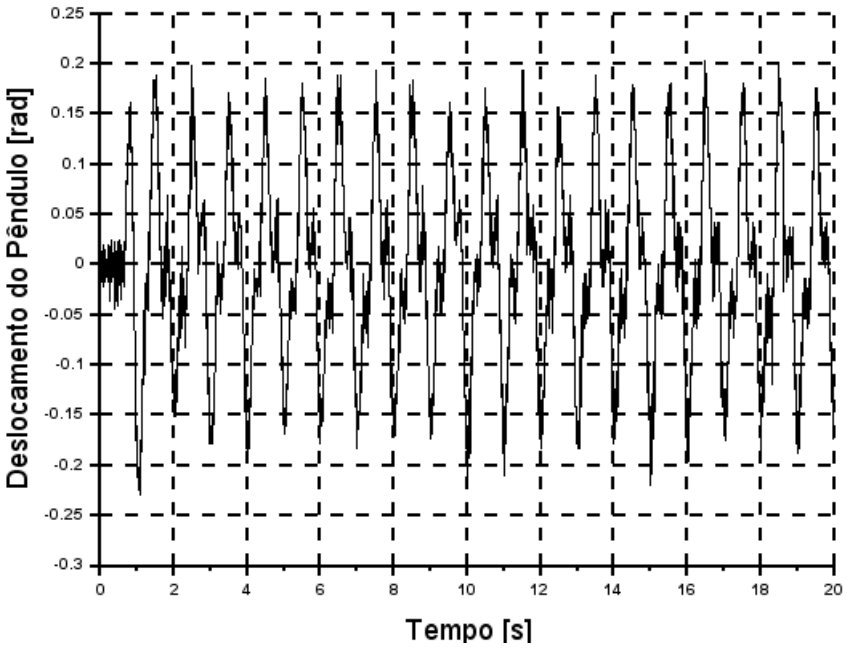
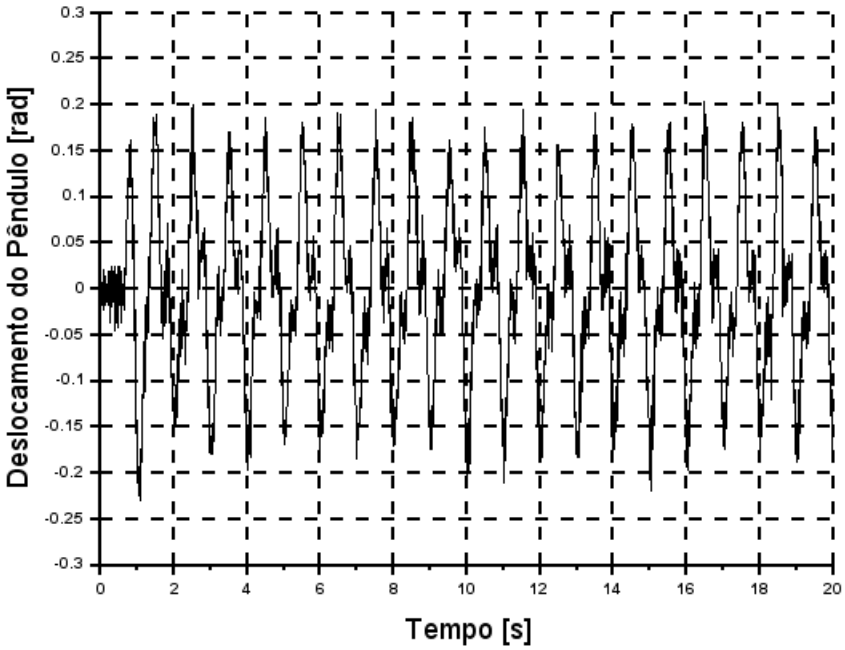


Figura 35: Deslocamento estimado do pêndulo



É percepitivel em todos os experimentos, com a taxa de aquisição de 200 pontos por segundo, a prença de ruído em ambos os sistemas de aquisição de dados.

4.2.1. Taxa de amostragem 20 pontos/s

A seguir será apresentado os resultados obtidos através dos experimentos utilizando a taxa de aquisição em 20 pontos por segundo e variando a frequência da perturbação do atuador.

4.2.1.1. Frequência $\pi[\frac{rad}{s}]$

Figura 36: Deslocamento mensurando do carro

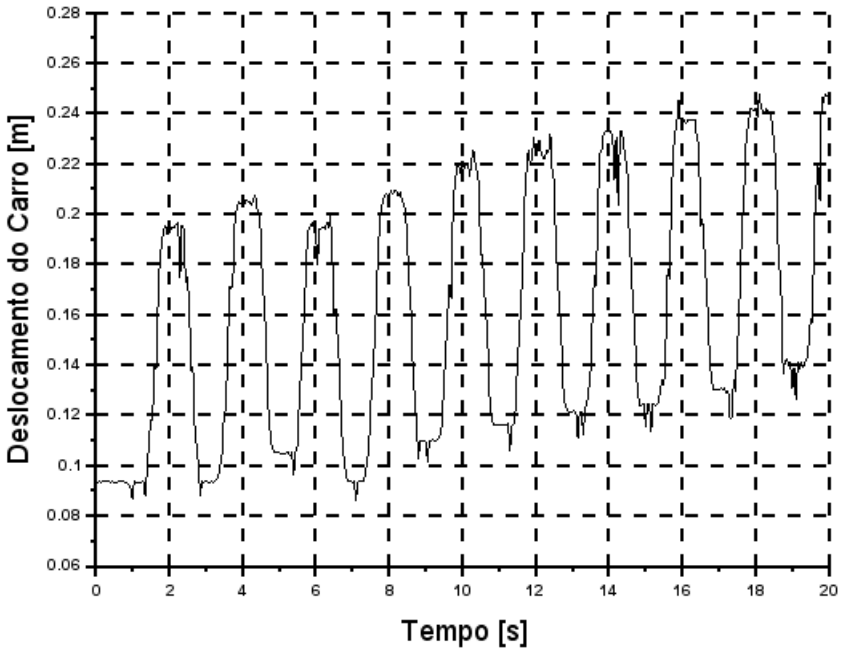


Figura 37: Deslocamento estimado do carro

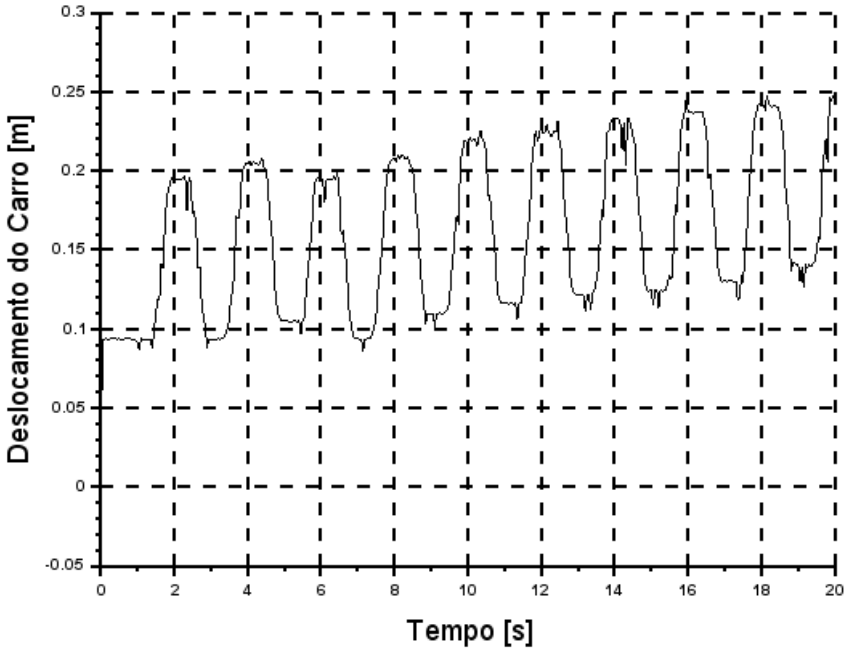


Figura 38: Deslocamento mensurado do pêndulo

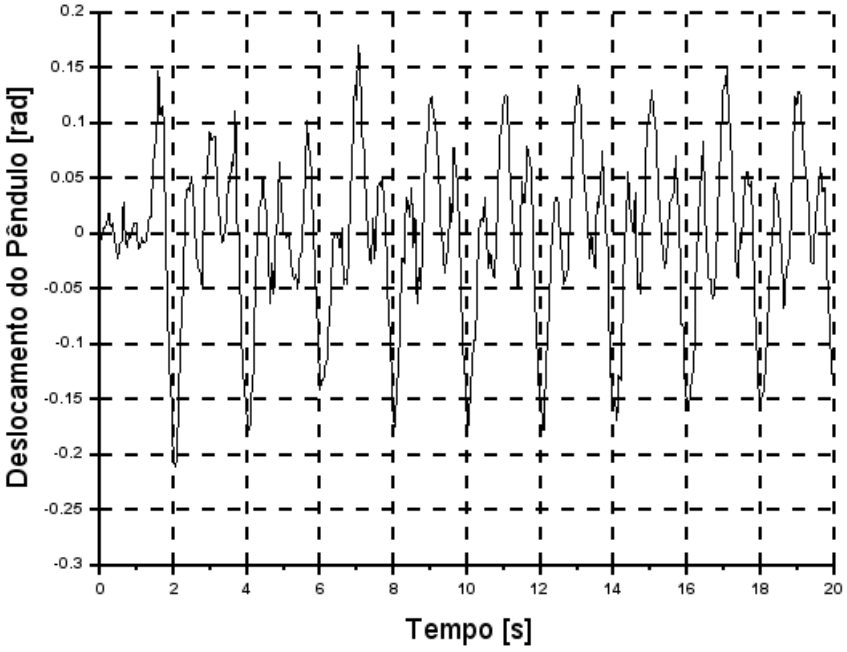
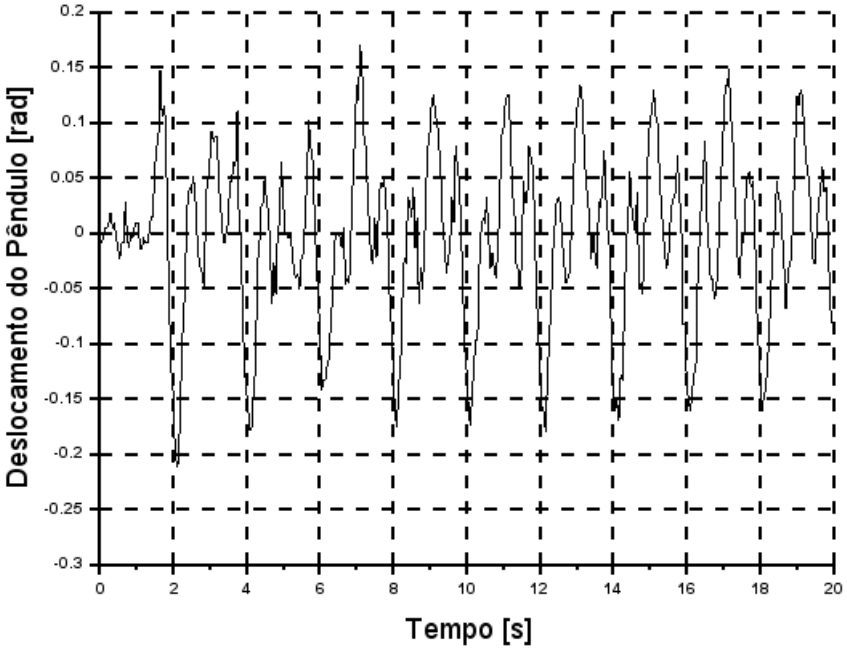


Figura 39: Deslocamento estimado do pêndulo



4.2.1.2. Frequência $1,5\pi[\frac{rad}{s}]$

Figura 40: Deslocamento mensurado do carro

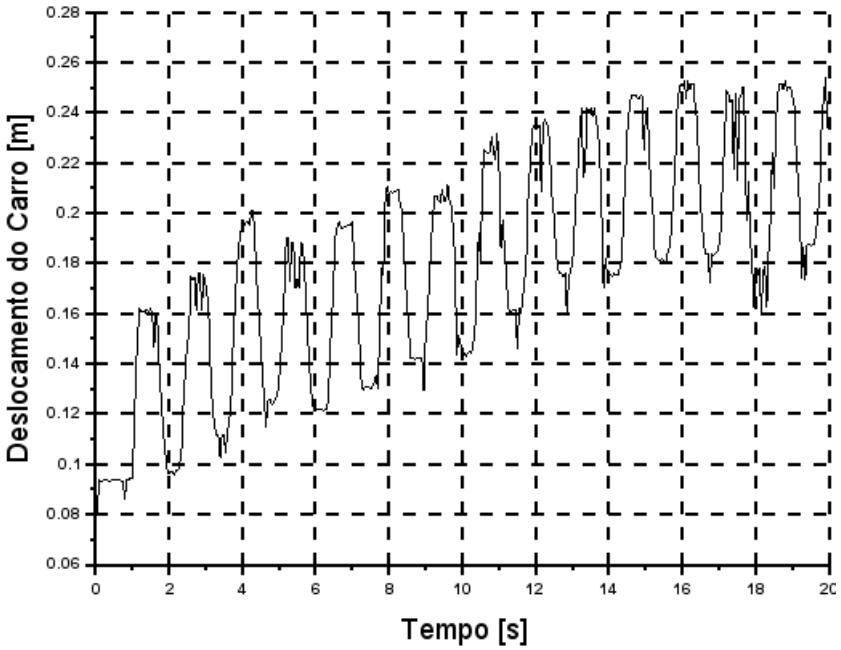


Figura 41: Deslocamento estimado do carro

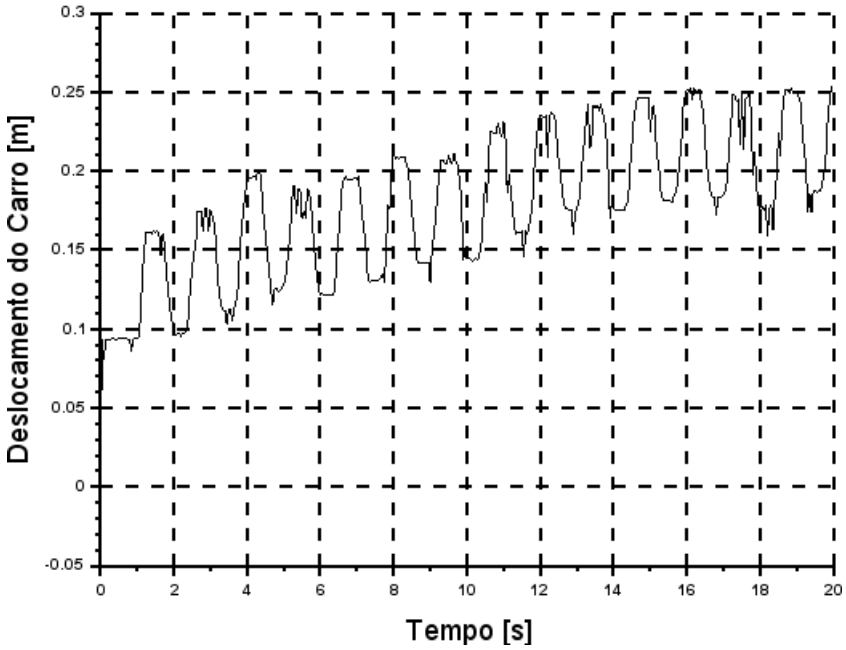


Figura 42: Deslocamento mensurado do pêndulo

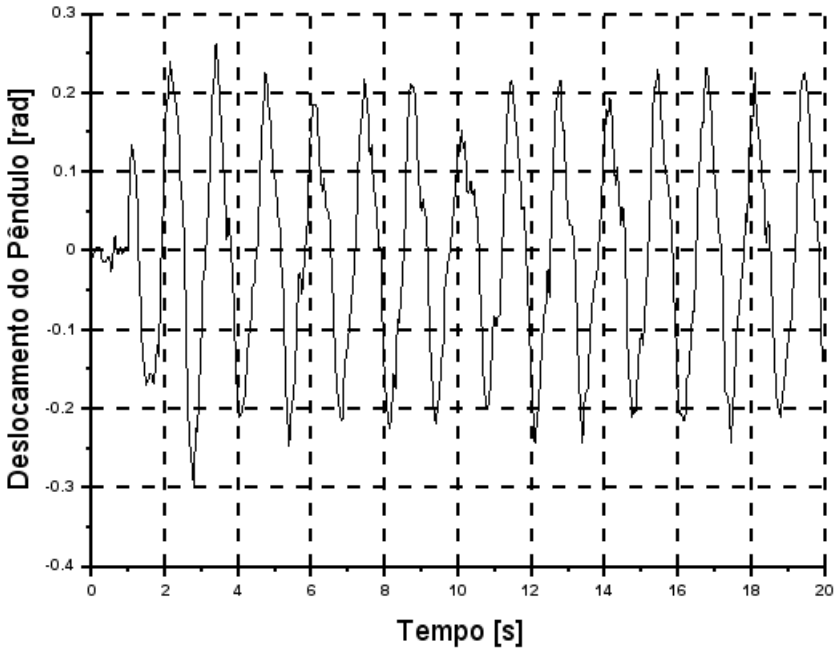
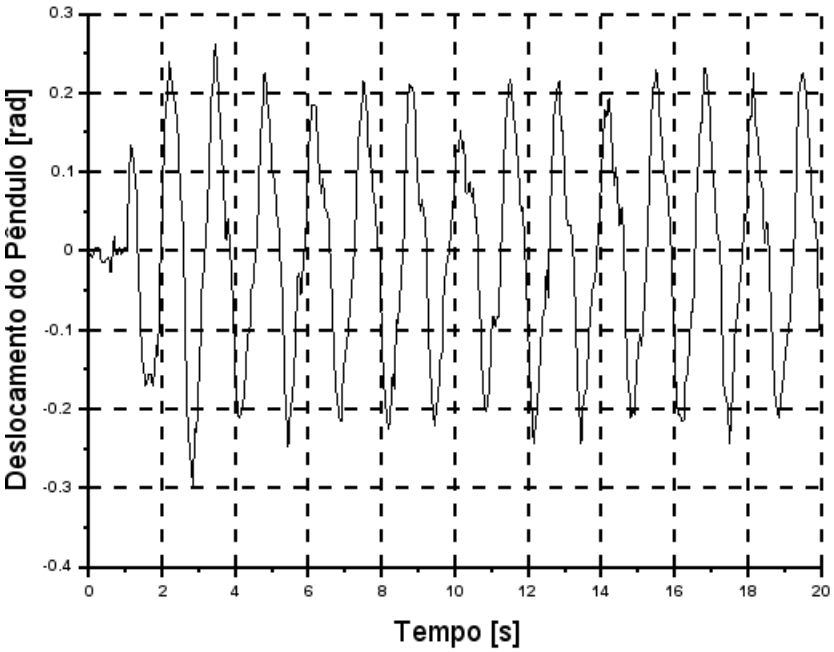


Figura 43: Deslocamento estimado do pêndulo



4.2.1.3. Frequência $2\pi[\frac{rad}{s}]$

Figura 44: Deslocamento mensurado do carro

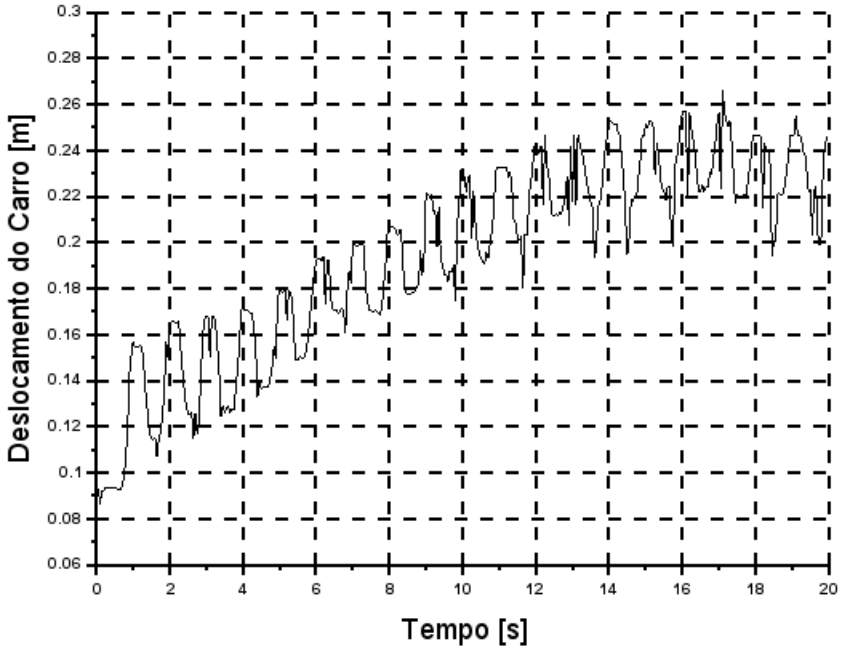


Figura 45: Deslocamento estimado do carro

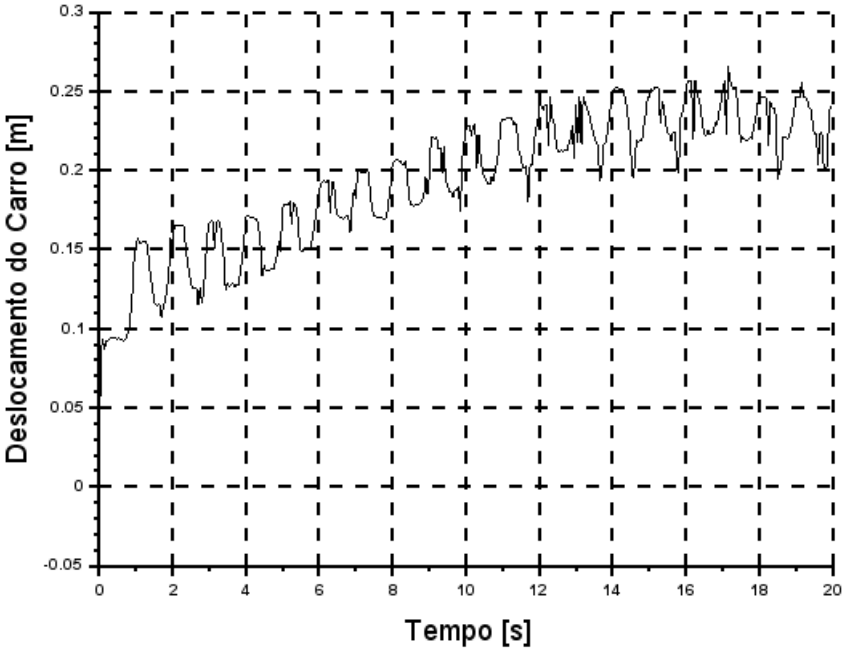


Figura 46: Deslocamento mensurado do pêndulo

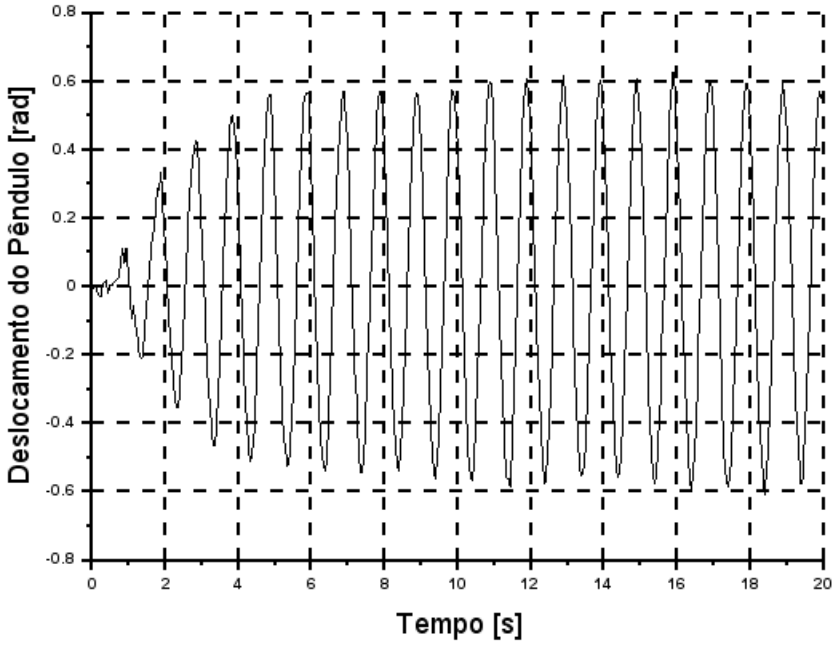
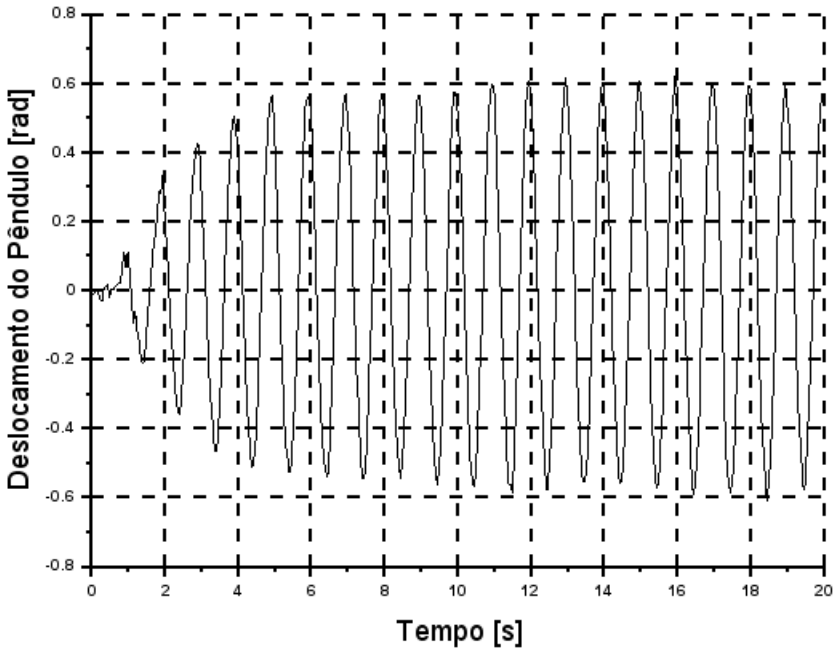


Figura 47: Deslocamento estimado do pêndulo



Com esta de taxa de aquisição, de 20 pontos por segundo, os dados aparentam sofrer menos devido aos ruídos inerentes ao sistema de medição.

Figura 48: Deslocamento estimado e real do carro para uma frequência de 2π $[\frac{rad}{s}]$ com taxa de amostragem de 200 pontos por segundo

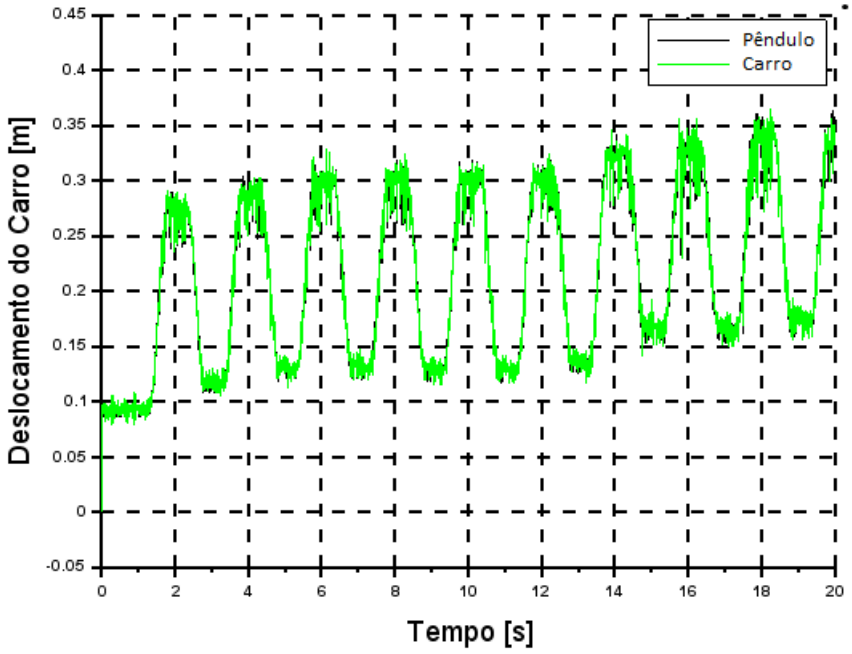
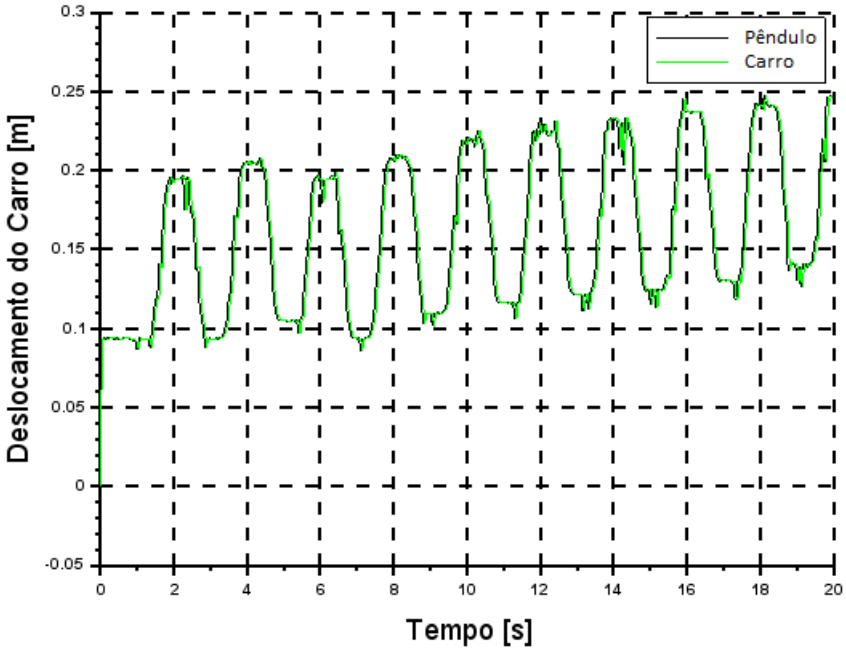


Figura 49: Deslocamento estimado e real do carro para uma frequência de 2π $[\frac{rad}{s}]$ com taxa de amostragem de 20 pontos por segundo



Comparando os gráficos das duas taxas de aquisição, com ênfase nas Figura 48 e Figura 1, é perceptível que a aquisição com 200 pontos por segundo prejudica a leitura da distância, devido a grande quantidade de ruído no sensor (Figura 23). Entretanto o comportamento da estimação do pêndulo é prejudicada, pois o erro do pêndulo (Figura 50) é elevado, uma vez que ,em uma taxa de aquisição menor, a curvatura dos dados recolhidos começa a ficar quadrada.

Figura 50: Erro da estimação com taxa de amostragem de 20 pontos por segundo para uma frequência de 2π $[\frac{rad}{s}]$

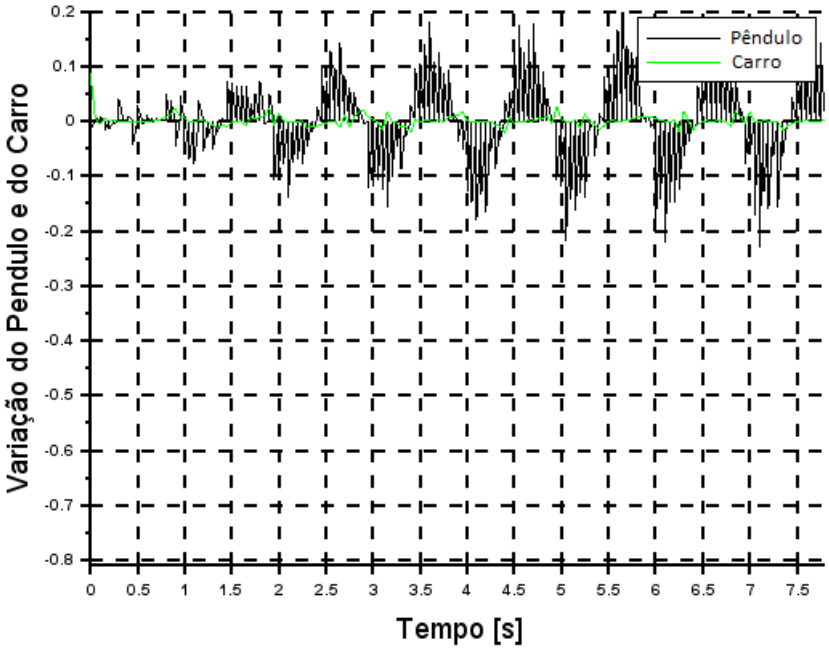
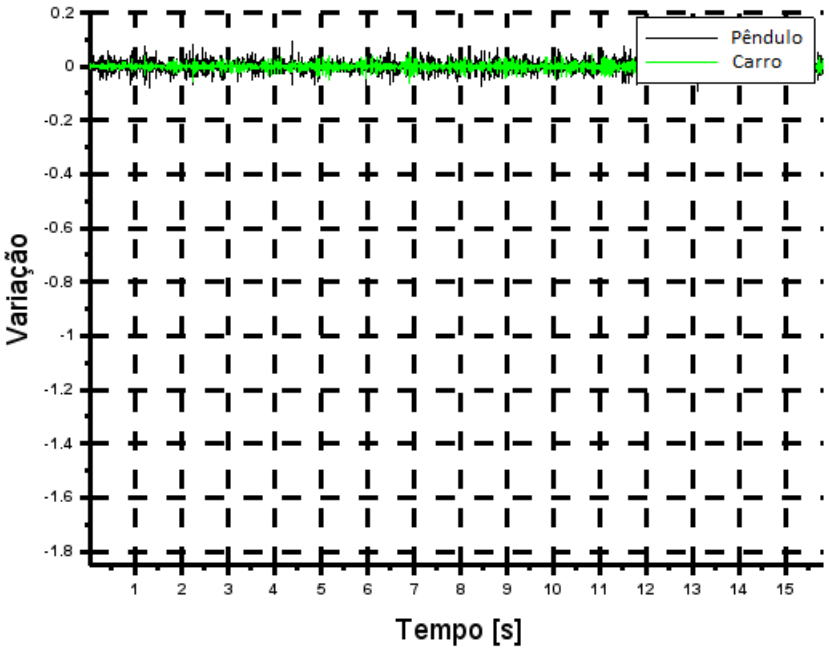


Figura 51:Erro da estimação com taxa de amostragem de 200 pontos por segundo para uma frequência de 2π $[\frac{rad}{s}]$



Já na taxa de aquisição com 20 pontos por segundo, o ruído no transdutor de distância é melhor trabalhado no estimador, porque a quantidade de dados trabalhados pelo estimador e pelo próprio sensor é menor, assim tornando o mais eficiente para a estimação de dados do carro. Entretanto de acordo com a Figura 50, o erro de estimação do pêndulo aumenta.

Capítulo 5. CONCLUSÃO

Encontrar transdutores específicos para certa mensuração é um esforço imensuravelmente cansativo, logo, um projeto de observadores é efetivo apresentando uma opção para a aquisição dos dados inacessíveis ou por via de transdutores financeiramente caros, utilizando sensores comuns do dia-a-dia. Neste trabalho foi construído um sistema pêndulo-carro sendo excitado por uma fonte senoidal e projetado um observador. Os dados foram obtidos e trabalhados através do *hardware* Arduino e o *software* Scilab-Arduino, os quais validaram a teoria estudada.

O projeto de observador aplicado ao sistema construído possui confiabilidade em suas estimações dos estados. Portanto, é possível afirmar que os estados não mensurados que estão sendo estimados, como velocidade angular do pêndulo e a velocidade do carro, são próximas as verdadeiras com um pequeno erro.

5.1. TRABALHOS FUTUROS

- Aplicar o filtro de Kalman para trabalhar com uma taxa de amostragem mais elevada sem prejudicar a leitura dos demais transdutores;
- Aplicar em sistemas rotativos para análise de falhas;
- Aplicar este projeto de observador em um sistema de controle com uma planta similar, como a de um pêndulo invertido;

BIBLIOGRAFIA

ARDUINO. ***Tutorial/PWM***. Disponível em:< https://www.arduino.cc>. Acessado em 27/11/2018.

BARŸO, M. J. S. Sistemas Dinâmicos em Espaço de Estados (Teoria). 2004.

CHAYA, Ravindra et al. Open Source, Real-Time Temperature Monitoring & Control using Scilab & Arduino. 2016.

DEMOSCIENCES. ***Projects scilab-arduino.*** Disponível em :<http://www.demosciences.fr>. Acessado em: 10/7/2018.

DORF, Richard C.; BISHOP, Robert H. Sistemas de Controle Modernos. 12. ed. LTC, 2013.

GAWRONSKI, Wodek. ***Advanced structural dynamics and active control of structures***. *Springer Science & Business Media*, 2004.

FERNANDES JÚNIOR, Jesus Antônio. Diagnose de trincas em sistemas rotativos, utilizando modelos de falhas através da metodologia dos observadores de estados. 2011.

FILIPEFLOP. **Categoria/componentes-eletronicos/resistores**. Disponível em:<www.filipeflop.com>. Acessado em: 27/11/2018.

PRECISIONMICRODRIVES. ***Reading the motor constants from typical performance characteristics*****.** Disponível em:<https://www.precisionmicrodrives.com>. Acessado em: 27/11/2018.

KOROISHI, Edson Hideki. Diagnose de falhas em sistemas rotativos com excitações desconhecidas, através da metodologia dos observadores de estado. 2009.

MORAIS, Tobias Souza. Diagnóstico de falhas via observadores de estado com excitações desconhecidas, identificadas via funções ortogonais. 2006.

NISE, Norman S. Engenharia de Sistemas de Controle. 6. ed. LTC, 2012.

OGATA, K. "Modern Control Engineering", 5th edition, Prentice Hall, 2009.

ROBOTSHOP. ***How do i interpret dc motor specifications***. Disponível em: <https://www.robotshop.com>. Acessado em: 27/11/2018.

SIRADJUDDIN, Indrazno et al. State space control using LQR method for a cart-inverted pendulum linearised model.

SOUZA, Davi Leonardo de et al. Análise do desempenho de sistemas de controle. 2007.

STEVAN, Sergio Luiz; SILVA, Rodrigo Adamshuk. **Automação e Instrumentação Industrial com Arduino: Teoria e Projetos**. Saraiva Educação SA.

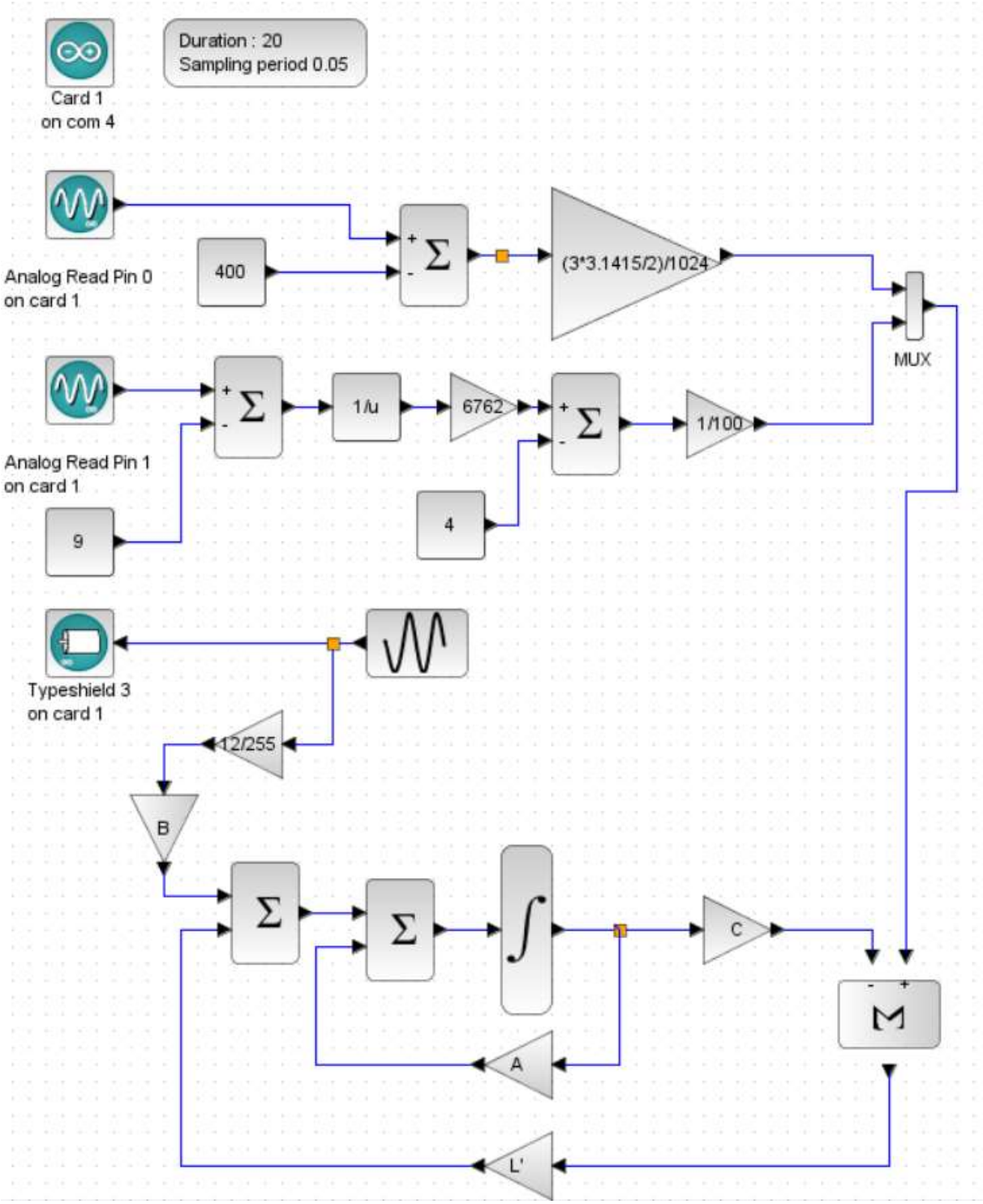
ANEXO A

Programação utilizada para calcular o ganho do observador:

```
//TCC_SCILAB
m=85/1000; //kg massa da haste
M=98/1000; //kg massa do carrinho
l=41/200; //m haste
g=9.8; //m/s² gravidade
Kt = 0.0222323; // constante de torque do motor
Kb = 0.023873241; // constante de voltagem do motor

Ra = 1.2208; // ohm resistência da armadura do motor
r = 10/1000; //m raio da engrenagem do motor
Kr = Kt/Ra;
c1 = Kr*Kb/(r1)^2;
c2 = Kr/r;
A = [0 1 0 0;(M+m)*g/(M*l) 0 0 c1/(M*l);0 0 0 1;-(m/M)*g 0 0 -
c1/M];//[x1;x2;x3;x4] respectivamente [theta; theta.; x; v]

B = [0;-c2/(M*l);0;c2/M];//[x1;x2;x3;x4][theta;theta.;x;v]
C=[1 0 0 0;0 0 1 0];
D=0;
//Confirmando a observabilidade do sistema
Obs = [C' A'*C' A'^2*C' A'^3*C']
posto = rank(Obs)

// Observador por alocação de polos
polos = 10*[-100 -100 -100 -100]
//polos = 5*spec(A2)
L = ppol(A',C',polos)
```

ANEXO B

Diagrama de bloco do observador no xcos com a biblioteca Arduino.



ANEXO C

Utilizando a biblioteca ARDUINO SCILAB

ARDUINO_SETUP:

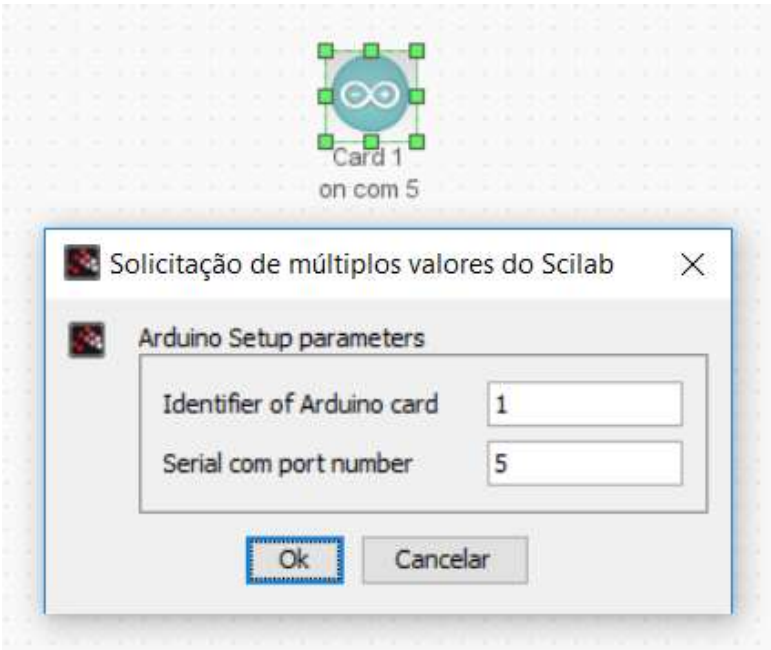
Bloco da biblioteca Arduino no xcos para configurar a porta serial onde haverá a comunicação entre o Arduino e o Scilab (Figura 52).

Figura 52: Cartão Arduino

FONTE: Próprio Autor

Com exceção do parâmetro identificador do Arduino que deve ser sempre um ao abrir a caixa de diálogo(Figura 53), para configurar os parâmetros do bloco de qualquer bloco do xcos, entra apenas com o número da porta serial em que o Arduino é conectado ao computador.

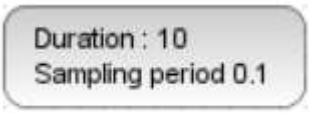
Figura 53: Caixa de Diálogo do cartão arduino



TIME_SAMPLE:

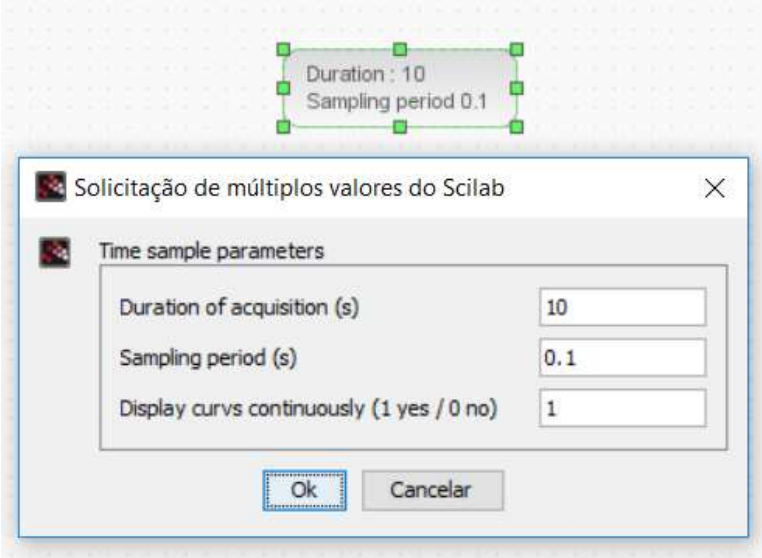
Este bloco da biblioteca Arduino no xcos (Figura 54) define a duração da simulação e o período de amostragem.

Figura 54: Bloco de duração da simulação



Os parâmetros de entrada para este bloco (Figura 55) são respectivamente a duração da aquisição de dados em segundos, o período de amostragem que deve ser maior ou igual a 0.005 segundos e também para exibir as curvas continuamente.

Figura 55: Caixa de Diálogo duração da simulação



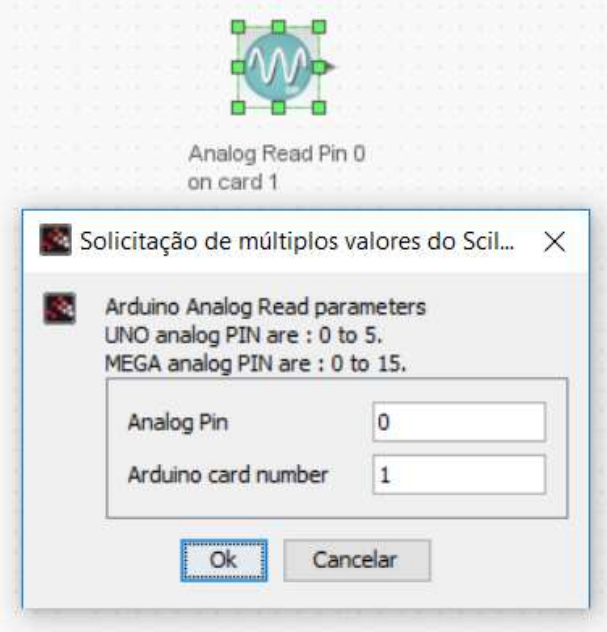
ANALOG_READ_SB:

Este bloco permite a aquisição dados analógicos da entrada analógica do Arduino (Figura 56). O canal de 10 bits converte a entrada analógica de 0 a 5 volts, para o valor digital entre 0 e 1023. A taxa de amostragem é de 8 ms para esse bloco.

Figura 56: Bloco de Entrada Analógica

Os parâmetros deste bloco requerem a entrada do pino analógico utilizado pelo Arduino, caso seja UNO os pinos vão de 0 a 5 e MEGA de 0 a 15, a carta Arduino deve ser sempre 1 (Figura 57). Caso seja necessário o uso de mais sinais analógicos é necessário utilizar mais blocos especificando o pino utilizado.

Figura 57: Caixa de diálogo do bloco de entrada analógica



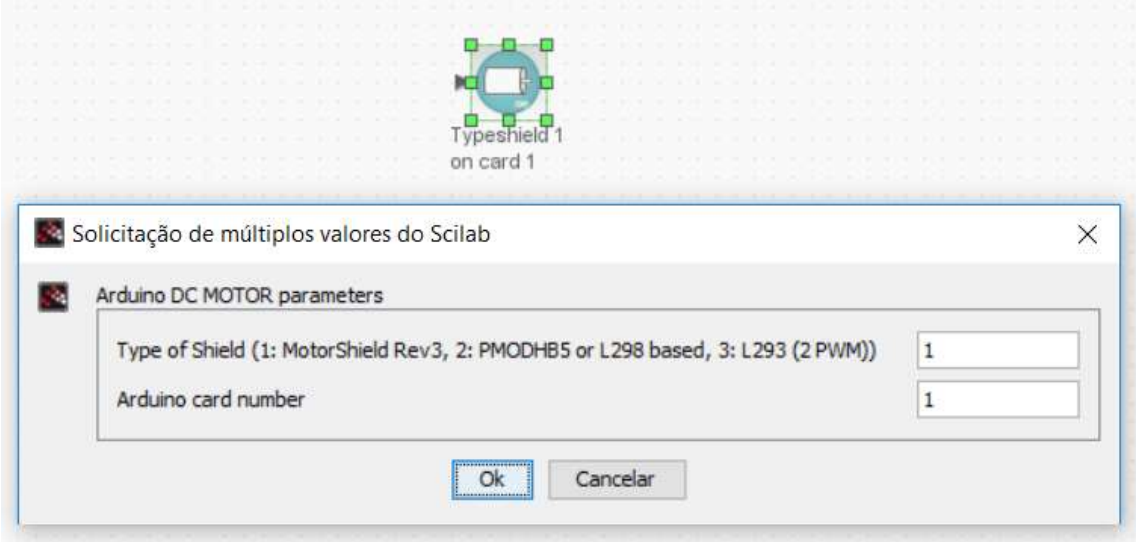
DCMOTOR_SB:

Este bloco (Figura 58) controla um ou mais motores DC. A placa Arduino não transmite energia suficiente, então é necessário o uso de uma ponte H circuito/IC para controlar o motor. A vários tipos de pontes H IC que não operam pelo mesmo princípio. Como por exemplo, o L298 requer o uso do sinal pwm com sensor de corrente. O L293 usa duas pwm para estabelecer a velocidade e a direção. Também shilds prontos estão disponíveis.

Figura 58: Bloco de *driver*

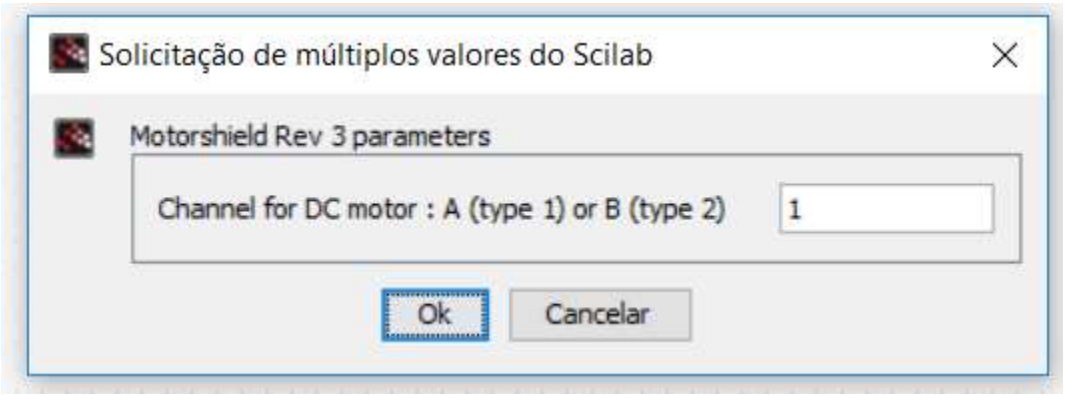
Ao abrir a caixa de diálogo para configurar os parâmetros deste bloco (figuraX) deve ser entrado com o tipo de Shield para controlar o motor.

Figura 59:Caixa de diálogo 1



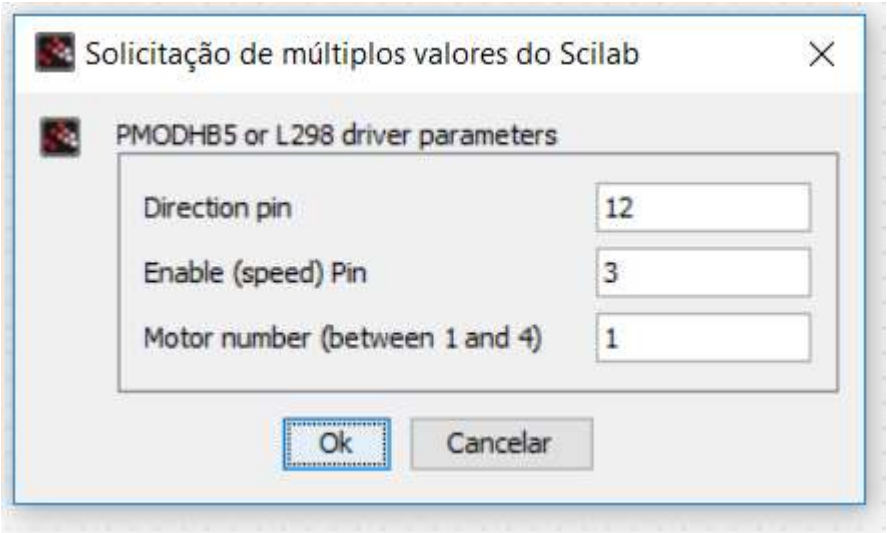
A Figura 60 mostra os parâmetro para o Motorshield Rev 3, que são os canais do motor DC(A ou B).

Figura 60: Caixa de diálogo 2 para tipo de *shild* 1



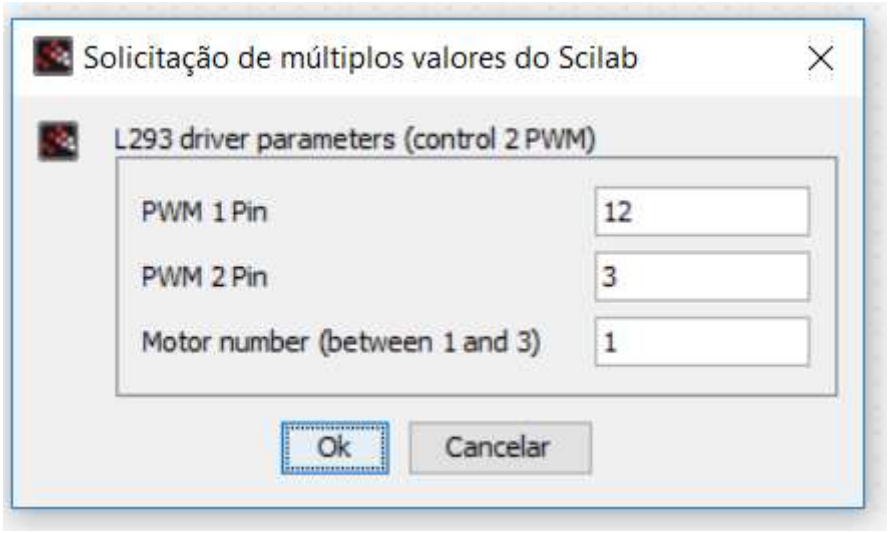
A Figura 61 mostra os parâmetros para a ponte H L298 ou PMODHB5, que são um pino para controlar a velocidade, um pino para direção que o motor rotacionará e o número do motor.

Figura 61: Caixa de diálogo 2 para tipo de shild 2



A Figura 62 mostra os parâmetros para a ponte H, L293, que são dois pinos para sinal *PWM*, este controlador controla a rotação e velocidade do motor por sinal *PWM*.

Figura 62:Caixa de diálogo 2 para tipo de shild 3



APÊNDICE A

Programação carregada ao arduino

```
/* This file is meant to be used with the SCILAB arduino
   toolbox, however, it can be used from the IDE environment
   (or any other serial terminal) by typing commands like:

   Conversion ascii -> number
   48->'0' ... 57->'9' 58->':' 59->';' 60->'<' 61->'=' 62->'>' 63->'?' 64->'@'
   65->'A' ... 90->'Z' 91->'[' 92->'\' 93->']' 94->'^' 95->'_' 96->'`'
   97->'a' ... 122->'z'

   Dan0 or Dan1 : attach digital pin n (ascii from 2 to b) to input (0) or output (1)
   Drn : read digital value (0 or 1) on pin n (ascii from 2 to b)
   Dwn0 or Dwn1 : write 1 or 0 on pin n
   An    : reads analog pin n (ascii from 0 to 19)
   Wnm  : write analog value m (ascii from 0 to 255) on pin n (ascii from 0 to 19)
   Sa1 or Sa2 : Attach servo 1 (digital pin 9) or 2 (digital pin 10)
   Sw1n or Sw2n : moves servo 1 or servo 2 to position n (from ascii(0) to ascii(180))
   Sd1 or Sd2 : Detach servo 1 or 2

   Generic DC_Motor
   Cijkl : setup for generic DCmotor number i (1 to 4), PW1 on pin number j, PWM2 or direction
on pin number k, mode=l
         l=0 for L293 (2 PWM) and l=1 for L298 (1PWM + 1 bit for direction)
   Mijk  : sets speed for generic DCmotor number i, j=0/1 for direction, k=ascii(0) .. ascii(255)
   Mir   : releases motor i (r=release)

   Generic Interrupt counter
   Iai   : activate counter on INT number i (i=ascii(2 or 3 or 18 or 19 or 20 or 21)
   Iri   : release counter on INT number i
   Ipi   : read counter on INT number i
   Izi   : reset counter on INT number i
```

```
  Generic Encoder
  Eajkl: activate encoder on channelA on INT number j (j=ascii(2 or 3 or 18 or 19 or 20 or 21)
et channelB on pin k or INT number k (k=ascii(0)..ascii(53))
        and l=1 or 2 or 4 for 1x mode (count every rising of chA) or 2x mode (count every
change statement of chA)
        or  4x mode (every change statement of chA et chB)
  Eri   : release encoder on INTi
  Epi   : read position of encoder on INTi
  Ezi   : reset value of encoder on INTi position

  R0    : sets analog reference to DEFAULT
  R1    : sets analog reference to INTERNAL
  R2    : sets analog reference to EXTERNAL

  */

#include <Servo.h>

/* define internal for the MEGA as 1.1V (as as for the 328)  */
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#define INTERNAL INTERNAL1V1
#endif

/* create and initialize servos                          */
Servo servo1;
Servo servo2;

/* Generic motors */
int dcm1_pin1,dcm1_pin2,dcm1_mode;
int dcm2_pin1,dcm2_pin2,dcm2_mode;
int dcm3_pin1,dcm3_pin2,dcm3_mode;
int dcm4_pin1,dcm4_pin2,dcm4_mode;
```

```
// Generic encoder
/* Encoders initialisation */
// volatile declare as those variables will change in interrupts
volatile long int encoder_0_position = 0,encoder_1_position = 0, encoder_2_position = 0, encoder_3_position = 0, encoder_4_position = 0, encoder_5_position = 0;
int encoder_0_int2 ;        // Pin used for encoder0 chanel B : define from scilab
int encoder_1_int2 ;        // Pin used for encoder1 chanel B : define from scilab
int encoder_2_int2 ;        // Pin used for encoder2 chanel B : define from scilab
int encoder_3_int2 ;        // Pin used for encoder3 chanel B : define from scilab
int encoder_4_int2 ;        // Pin used for encoder4 chanel B : define from scilab
int encoder_5_int2 ;        // Pin used for encoder5 chanel B : define from scilab
int encoder_num, encoder_int2;
int corresp[6]={2,3,21,20,19,18}; //Correspondance beetween interrupt number and pin number

//Generic counter
volatile long int counter_0=0,counter_1=0,counter_2=0,counter_3=0,counter_4=0,counter_5=0;

int initiat=1;

void setup() {
  /* initialize serial                */
  Serial.begin(115200);

}

void loop() {

  /* variables declaration and initialization        */

  static int  s   = -1;    /* state                  */
  static int  pin = 13;    /* generic pin number     */
```

```
  static int  dcm =  4;    /* generic dc motor number        */

  int  val =  0;          /* generic value read from serial */
  int  agv =  0;          /* generic analog value           */
  int  dgv =  0;          /* generic digital value          */
  static int  enc   = 1;    /* encoder number 1 (or 2 for Arduino mega)     */

  while (Serial.available()==0) {}; // Waiting char
  val = Serial.read();

//  if (val==0){// version
//    Serial.print('v3');
//    val=-1;
//  }
  //case A -> Analog
  if (val==65){//A -> Analog read
    while (Serial.available()==0) {}; // Waiting char
//    val=Serial.read();
//    if (val==114){ //'r'-> read pin
//       while (Serial.available()==0) {}; // Waiting char
       val=Serial.read();
       if (val>47 && val<67) { //from pin 0, to pin 19
          pin=val-48; //number of the pin
          agv=analogRead(pin);
          //Serial.println(agv);
          Serial.write((uint8_t*)&agv,2); /* send binary value via serial  */
       }
       val=-1;
  }
  else if (val==87){//W -> Analog write
     while (Serial.available()==0) {}; // Waiting char
     val=Serial.read();
       if (val>47 && val<67) { //from pin 0 to pin 19
```

```
        pin=val-48; //number of the pin
        while (Serial.available()==0) {}; // Waiting char
        val=Serial.read();
        analogWrite(pin,val);
      }
      val=-1;
    }
//}

//case D -> Digital
else if (val==68){//D -> Digital pins
  while (Serial.available()==0) {}; // Waiting char
  val=Serial.read();
  if (val==97){ //'a'-> declare pin
    while (Serial.available()==0) {}; // Waiting char
    val=Serial.read();
    if (val>49 && val<102) {
      pin=val-48;
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val==48 || val==49) {
        if (val==48){//'0' -> input
          pinMode(pin,INPUT);
        }
        else if (val==49){//'1' -> output
          pinMode(pin,OUTPUT);
        }
      }
    }
  }
  if (val==114){ //'r'-> read pin
    while (Serial.available()==0) {}; // Waiting char
    val=Serial.read();
```

```
      if (val>49 && val<102) {
        pin=val-48; //number of the digital pin
        dgv=digitalRead(pin);
//        Serial.println(dgv);
        Serial.print(dgv);
      }
    }
    if (val==119){ //'w'-> write pin
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val>49 && val<102) {
        pin=val-48; //number of the digital pin
        while (Serial.available()==0) {}; // Waiting char
        val=Serial.read();
        if (val==48 || val==49) { // 0 or 1
          dgv=val-48;
          digitalWrite(pin,dgv);
//          Serial.println(dgv);
        }
      }
    }
   val=-1;

  }
  //case S -> servomotor
  else if (val==83){
    while (Serial.available()==0) {}; // Waiting char
    val=Serial.read();
    if (val==97){ //'a'-> declare servo
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val==49 || val==50) { //servo 1 or 2
        pin=val-48; //number of the servo
```

```
        if (pin==1) {
          servo1.attach(9);
          servo1.write(0);
//          agv=servo1.read();
//          Serial.println(agv);
        }
        if (pin==2) {
          servo2.attach(10);
          servo2.write(0);
//          agv=servo2.read();
//          Serial.println(agv);
        }

      }
    }
    if (val==100){ //'d'-> detach servo
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val==49 || val==50) { //servo 1 or 2
        pin=val-48; //number of the servo
        if (pin==1) {servo1.detach(); }
        if (pin==2) {servo2.detach(); }
      }
    }
    if (val==119){ //'w'-> write pin
      while (Serial.available()==0) {}; // Waiting char
      val=Serial.read();
      if (val==49 || val==50) { //servo 1 or 2
        pin=val-48; //number of the servo
        while (Serial.available()==0) {}; // Waiting char
        val=Serial.read();
        if (val>=0 && val<=180){
          if (pin==1) {
```

```
            servo1.write(val);
//              agv=servo1.read();
//              Serial.println(agv);
            }
            if (pin==2) {
             servo2.write(val);
//              agv=servo2.read();
//              Serial.println(agv);
            }
         }
       }
    }
   val=-1;

  }

  //case I -> Interrupt
  else if (val==73){
     /* ASKING ACTIVATION OF AN COUNTER     */
     while (Serial.available()==0) {}; // Waiting char
     val=Serial.read();
     if (val==97) { //a = activation
       while (Serial.available()==0) {};              // Waiting char
       val=Serial.read();                          // Read int_number (must be 0 or 1 on UNO / 1 to 5
on MEGA) : int_number set to encoder number
       pinMode(corresp[val],INPUT);             // set interrupt pin as input
       if           (val  ==  0)  {attachInterrupt(val,  counter_0_change,  RISING);counter_0=0;}
//counter INT0
       else  if    (val  ==  1)  {attachInterrupt(val,  counter_1_change,  RISING);counter_1=0;}
//counter INT1
       else  if    (val  ==  2)  {attachInterrupt(val,  counter_2_change,  RISING);counter_2=0;}
//counter INT2
```

```
      else if   (val == 3) {attachInterrupt(val, counter_3_change, RISING);counter_3=0;}
//counter INT3
      else if   (val == 4) {attachInterrupt(val, counter_4_change, RISING);counter_4=0;}
//counter INT4
      else if   (val == 5) {attachInterrupt(val, counter_5_change, RISING);counter_5=0;}
//counter INT5
    }
    /* ASKING POSITION OF A COUNTER     */
    if (val==112) { //p = sending counting value
      while (Serial.available()==0) {};   // Waiting char
      val = Serial.read()   ;            //reading next value = counter number
      if     (val==0){ Serial.write((uint8_t*)&counter_0,4); }// asking counter 0
      else if (val==1){ Serial.write((uint8_t*)&counter_1,4); }// asking counter 1
      else if (val==2){ Serial.write((uint8_t*)&counter_2,4); }// asking counter 2
      else if (val==3){ Serial.write((uint8_t*)&counter_3,4); }// asking counter 3
      else if (val==4){ Serial.write((uint8_t*)&counter_4,4); }// asking counter 4
      else if (val==5){ Serial.write((uint8_t*)&counter_5,4); }// asking counter 5
    }
    /* ASKING RELEASE OF AN INTERRUPT   */
    if (val==114) { //r = release counter
      while (Serial.available()==0) {};       // Waiting char
      val = Serial.read();                  //reading next value = counter number
      detachInterrupt(val);                  // Detach interrupt on chanel A of counter num=val
      if (val==0)       { counter_0=0;}       // Reset counter
      else if (val==1)  { counter_1=0;}       // Reset counter
      else if (val==2)  { counter_2=0;}       // Reset counter
      else if (val==3)  { counter_3=0;}       // Reset counter
      else if (val==4)  { counter_4=0;}       // Reset counter
      else if (val==5)  { counter_5=0;}       // Reset counter
    }
    /* ASKING RESET VALUE OF AN COUNTER     */
    if (val==122) { //z set to zero
      while (Serial.available()==0) {};       // Waiting char
```

```
      val = Serial.read();                  //reading next value = counter number
      if (val==0)       { counter_0=0;}       // Reset counter
      else if (val==1)  { counter_1=0;}       // Reset counter
      else if (val==2)  { counter_2=0;}       // Reset counter
      else if (val==3)  { counter_3=0;}       // Reset counter
      else if (val==4)  { counter_4=0;}       // Reset counter
      else if (val==5)  { counter_5=0;}       // Reset counter
     }
   val=-1;

 }

 //case E -> Encoder
 else if (val==69){
    /*Generic encoder functions */
    while (Serial.available()==0) {};
    val=Serial.read();
    /* ASKING ACTIVATION OF AN ENCODER     */
    if (val==97) {                          //activation
      while (Serial.available()==0) {};              // Waiting char
      encoder_num=Serial.read();                     // Read int_number (must be 0 or 1 on UNO /
1 to 5 on MEGA) : int_number set to encoer number
      pinMode(corresp[encoder_num],INPUT);           // set interrupt pin as input
      while (Serial.available()==0) {};              // Waiting char
      encoder_int2=Serial.read();                   // Read int2 (must be a digital PIN with interrupt
or not : depends on mode)
                                          // no declaration for the moment : wait for encoder mode
      while (Serial.available()==0) {};              // Waiting char
      int mode = Serial.read()-48;                  // Read mode 1 ou 2 (1 counting only rising of
chA, 2 counting rising and falling)
      if (mode == 4) {                             // mode 4x : 2 cases : chA=pin2 / chB=pin3 or
chA=pin3/chB=pin2 [Uno retriction]
        pinMode(corresp[encoder_int2],INPUT);        // set interrupt number as input
```

```
} else {
  pinMode(encoder_int2,INPUT);                // set pin as input
}

if (encoder_num == 0) {                       //encoder INT0
 encoder_0_position=0;                        // Reset position
 if (mode==4) {
  encoder_0_int2=corresp[encoder_int2];       // Save pin of second interruption
  attachInterrupt(encoder_num , encoder_change_m4_A0, CHANGE); // Attach interrupt on chanel A change
  attachInterrupt(encoder_int2, encoder_change_m4_B0, CHANGE); // Attach interrupt on chanel B change
 } else if (mode==2) {
  encoder_0_int2=encoder_int2;
  attachInterrupt(encoder_num, encoder_0_change_m2, CHANGE); // Attach interrupt on chanel A change
 } else if (mode==1) {
  encoder_0_int2=encoder_int2;
  attachInterrupt(encoder_num, encoder_0_change_m1, RISING); // Attach interrupt on chanel A rising
 }
} else if (encoder_num == 1) {                //encoder INT1
 encoder_1_position=0;                        // Reset position
 if (mode==4) {
  encoder_1_int2=corresp[encoder_int2];       // Save pin of second interruption
  attachInterrupt(encoder_num , encoder_change_m4_A1, CHANGE); // Attach interrupt on chanel A change
  attachInterrupt(encoder_int2, encoder_change_m4_B1, CHANGE); // Attach interrupt on chanel B change
 } else if (mode==2) {
  encoder_1_int2=encoder_int2;
  attachInterrupt(encoder_num, encoder_1_change_m2, CHANGE); // Attach interrupt on chanel A change
```

```
        } else if (mode==1) {
          encoder_1_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_1_change_m1, RISING); // Attach interrupt on
chanel A rising
        }
      } else if (encoder_num == 2) {                  //encoder INT2
        encoder_2_position=0;                         // Reset position
        if (mode==4) {
          encoder_2_int2=corresp[encoder_int2];          // Save pin of second interruption
          attachInterrupt(encoder_num , encoder_change_m4_A2, CHANGE); // Attach
interrupt on chanel A change
          attachInterrupt(encoder_int2, encoder_change_m4_B2, CHANGE); // Attach interrupt
on chanel B change
        } else if (mode==2) {
          encoder_2_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_2_change_m2, CHANGE); // Attach interrupt
on chanel A change
        } else if (mode==1) {
          encoder_2_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_2_change_m1, RISING); // Attach interrupt on
chanel A rising
        }
      } else if (encoder_num == 3) {                  //encoder INT3
        encoder_3_position=0;                         // Reset position
        if (mode==4) {
          encoder_3_int2=corresp[encoder_int2];          // Save pin of second interruption
          attachInterrupt(encoder_num , encoder_change_m4_A3, CHANGE); // Attach
interrupt on chanel A change
          attachInterrupt(encoder_int2, encoder_change_m4_B3, CHANGE); // Attach interrupt
on chanel B change
        } else if (mode==2) {
          encoder_3_int2=encoder_int2;
```

```
        attachInterrupt(encoder_num, encoder_3_change_m2, CHANGE); // Attach interrupt on chanel A change
      } else if (mode==1) {
        encoder_3_int2=encoder_int2;
        attachInterrupt(encoder_num, encoder_3_change_m1, RISING); // Attach interrupt on chanel A rising
      }
    } else if (encoder_num == 4) {                  //encoder INT4
      encoder_4_position=0;                         // Reset position
      if (mode==4) {
        encoder_4_int2=corresp[encoder_int2];       // Save pin of second interruption
        attachInterrupt(encoder_num , encoder_change_m4_A4, CHANGE); // Attach interrupt on chanel A change
        attachInterrupt(encoder_int2, encoder_change_m4_B4, CHANGE); // Attach interrupt on chanel B change
      } else if (mode==2) {
        encoder_4_int2=encoder_int2;
        attachInterrupt(encoder_num, encoder_4_change_m2, CHANGE); // Attach interrupt on chanel A change
      } else if (mode==1) {
        encoder_4_int2=encoder_int2;
        attachInterrupt(encoder_num, encoder_4_change_m1, RISING); // Attach interrupt on chanel A rising
      }
    } else if (encoder_num == 5) {                  //encoder INT5
      encoder_5_position=0;                         // Reset position
      if (mode==4) {
        encoder_5_int2=corresp[encoder_int2];       // Save pin of second interruption
        attachInterrupt(encoder_num , encoder_change_m4_A5, CHANGE); // Attach interrupt on chanel A change
        attachInterrupt(encoder_int2, encoder_change_m4_B5, CHANGE); // Attach interrupt on chanel B change
      } else if (mode==2) {
```

```
          encoder_5_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_5_change_m2, CHANGE); // Attach interrupt on chanel A change
         } else if (mode==1) {
          encoder_5_int2=encoder_int2;
          attachInterrupt(encoder_num, encoder_5_change_m1, RISING); // Attach interrupt on chanel A rising
         }
        }
      }
      /* ASKING POSITION OF AN ENCODER     */
      if (val==112) { //p = sending encoder position
        while (Serial.available()==0) {};   // Waiting char
        val = Serial.read()   ;            //reading next value = encoder number
        if         (val==0){  Serial.write((uint8_t*)&encoder_0_position,4);  }// asking encoder 0 position
        else  if  (val==1){  Serial.write((uint8_t*)&encoder_1_position,4);  }// asking encoder 1 position
        else  if  (val==2){  Serial.write((uint8_t*)&encoder_2_position,4);  }// asking encoder 2 position
        else  if  (val==3){  Serial.write((uint8_t*)&encoder_3_position,4);  }// asking encoder 3 position
        else  if  (val==4){  Serial.write((uint8_t*)&encoder_4_position,4);  }// asking encoder 4 position
        else  if  (val==5){  Serial.write((uint8_t*)&encoder_5_position,4);  }// asking encoder 5 position
      }
      /* ASKING RELEASE OF AN ENCODER     */
      if (val==114) { //r = release encoder
        while (Serial.available()==0) {};                // Waiting char
        val = Serial.read();                             //reading next value = encoder number
        detachInterrupt(val);                            // Detach interrupt on chanel A of encoder num=val
        if (val==0)       { encoder_0_position=0;encoder_0_int2=-1;}        // Reset position
```

```
      else if (val==1)  { encoder_1_position=0;encoder_1_int2=-1;}       // Reset position
      else if (val==2)  { encoder_2_position=0;encoder_2_int2=-1;}       // Reset position
      else if (val==3)  { encoder_3_position=0;encoder_3_int2=-1;}       // Reset position
      else if (val==4)  { encoder_4_position=0;encoder_4_int2=-1;}       // Reset position
      else if (val==5)  { encoder_5_position=0;encoder_5_int2=-1;}       // Reset position
      while (Serial.available()==0) {};             // Waiting char
      val = Serial.read();                          // reading next value = encoder number
      detachInterrupt(val);                         // Detach interrupt on chanel B of encoder num=val
(may be the same if mode=1 or 2)
    }
    /* ASKING RESET POSITION OF AN ENCODER    */
    if (val==122) {  // z = encoder position to zero
      while (Serial.available()==0) {};             // Waiting char
      val = Serial.read();                          //reading next value = encoder number
      if (val==0)      { encoder_0_position=0;}       // Reset position
      else if (val==1)  { encoder_1_position=0;}      // Reset position
      else if (val==2)  { encoder_2_position=0;}      // Reset position
      else if (val==3)  { encoder_3_position=0;}      // Reset position
      else if (val==4)  { encoder_4_position=0;}      // Reset position
      else if (val==5)  { encoder_5_position=0;}      // Reset position
    }
   val=-1;

 }

 //case C -> DCmotor init
 else if(val==67){
   while (Serial.available()==0) {};
   val = Serial.read();
   /* 2nd char = motor number */
   if (val>48 && val<53) {
      dcm=val-48;
      while (Serial.available()==0) {};
```

```
      val = Serial.read();
      /* the third received value indicates the pin1 number from ascii(2)=50 to ascii(e)=101  */
      if (val>49 && val<102) {
        if (dcm==1) dcm1_pin1=val-48;/* calculate motor pin1 */
        if (dcm==2) dcm2_pin1=val-48;/* calculate motor pin1 */
        if (dcm==3) dcm3_pin1=val-48;/* calculate motor pin1 */
        if (dcm==4) dcm4_pin1=val-48;/* calculate motor pin1 */
        pinMode(val-48, OUTPUT); //set pin as output
        analogWrite(val-48,0);   /* DUTY CYCLE */
        while (Serial.available()==0) {};
        val = Serial.read();
        /* the fourth received value indicates the pin2 number from ascii(2)=50 to ascii(e)=101
*/
        if (val>49 && val<102) {
          if (dcm==1) dcm1_pin2=val-48;/* calculate motor pin2 */
          if (dcm==2) dcm2_pin2=val-48;/* calculate motor pin2 */
          if (dcm==3) dcm3_pin2=val-48;/* calculate motor pin2 */
          if (dcm==4) dcm4_pin2=val-48;/* calculate motor pin2 */
          pinMode(val-48, OUTPUT); //set pin as output
          while (Serial.available()==0) {};
          val = Serial.read();
          /* the fifth received value indicates the pin2 number from ascii(2)=50 to ascii(e)=101
*/
          if (val>47 && val<50) {
            int mode = val-48;
            if (dcm==1) dcm1_mode=mode;/* calculate motor mode */
            if (dcm==2) dcm2_mode=mode;/* calculate motor mode */
            if (dcm==3) dcm3_mode=mode;/* calculate motor mode */
            if (dcm==4) dcm4_mode=mode;/* calculate motor mode */
            //initialization of port
            if(mode==0){//L293
              if (dcm==1) analogWrite(dcm1_pin2,0);   /* DUTY CYCLE */
              if (dcm==2) analogWrite(dcm2_pin2,0);   /* DUTY CYCLE */
```

```
            if (dcm==3) analogWrite(dcm3_pin2,0);   /* DUTY CYCLE */
            if (dcm==4) analogWrite(dcm4_pin2,0);   /* DUTY CYCLE */
          } else if (mode==1) {//L297
            if (dcm==1) digitalWrite(dcm1_pin2, LOW);  /* DIRECTION */
            if (dcm==2) digitalWrite(dcm2_pin2, LOW);  /* DIRECTION */
            if (dcm==3) digitalWrite(dcm3_pin2, LOW);  /* DIRECTION */
            if (dcm==4) digitalWrite(dcm4_pin2, LOW);  /* DIRECTION */
          }
          Serial.print("OK"); // tell Scilab that motor s initialization finished
                        // Cette commande sert à rien dans la toolbox de base,
                        // sauf si on prévoit d'ajouter des actions à l'init des moteurs
                        // par exemple chercher la position d'origine !
        }
       }
      }
  }
  val=-1;

}

//case M -> DC motor
else if(val==77){
    while (Serial.available()==0) {};
    val = Serial.read();
    /* the second received value indicates the motor number
       from abs('1')=49, motor1, to abs('4')=52, motor4    */
    if (val>48 && val<53) {
      dcm=val-48;               /* calculate motor number */
      while (Serial.available()==0) {};       // Waiting char
      val = Serial.read();
      /* the third received value indicates the sens direction or release*/
      if (val==48 || val ==49){
         int direction=val-48;
```

```
while (Serial.available()==0) {};       // Waiting char
val = Serial.read();                    //reading next value = 0..255
if (dcm==1){
 if(dcm1_mode==0){//L293
  if(direction==1){
    analogWrite(dcm1_pin1,val);
    analogWrite(dcm1_pin2,0);
  } else {
    analogWrite(dcm1_pin2,val);
    analogWrite(dcm1_pin1,0);
  }
 } else {//L298
  if (direction==0) digitalWrite(dcm1_pin2,LOW);
  if (direction==1) digitalWrite(dcm1_pin2,HIGH);
  analogWrite(dcm1_pin1,val);
 }
}
if (dcm==2){
 if(dcm2_mode==0){//L293
  if(direction==1){
    analogWrite(dcm2_pin1,val);
    analogWrite(dcm2_pin2,0);
  } else {
    analogWrite(dcm2_pin2,val);
    analogWrite(dcm2_pin1,0);
  }
 } else {//L298
  if (direction==0) digitalWrite(dcm2_pin2,LOW);
  if (direction==1) digitalWrite(dcm2_pin2,HIGH);
  analogWrite(dcm2_pin1,val);
 }
}
if (dcm==3){
```

```
   if(dcm3_mode==0){//L293
    if(direction==1){
      analogWrite(dcm3_pin1,val);
      analogWrite(dcm3_pin2,0);
    } else {
      analogWrite(dcm3_pin2,val);
      analogWrite(dcm3_pin1,0);
    }
   } else {//L298
    if (direction==0) digitalWrite(dcm3_pin2,LOW);
    if (direction==1) digitalWrite(dcm3_pin2,HIGH);
    analogWrite(dcm3_pin1,val);
   }
  }
  if (dcm==4){
   if(dcm4_mode==0){//L293
    if(direction==1){
      analogWrite(dcm4_pin1,val);
      analogWrite(dcm4_pin2,0);
    } else {
      analogWrite(dcm4_pin2,val);
      analogWrite(dcm4_pin1,0);
    }
   } else {//L298
    if (direction==0) digitalWrite(dcm4_pin2,LOW);
    if (direction==1) digitalWrite(dcm4_pin2,HIGH);
    analogWrite(dcm4_pin1,val);
   }
  }
}
if (val==114){//release motor
 if(dcm==1) {
 analogWrite(dcm1_pin1,0);
```

```
        if(dcm1_mode==0)  analogWrite(dcm1_pin2,0);
         }
       if(dcm==2) {
        analogWrite(dcm2_pin1,0);
        if(dcm2_mode==0)  analogWrite(dcm2_pin2,0);
       }
       if(dcm==3) {
        analogWrite(dcm3_pin1,0);
        if(dcm3_mode==0)  analogWrite(dcm3_pin2,0);
       }
       if(dcm==4) {
        analogWrite(dcm4_pin1,0);
        if(dcm4_mode==0)  analogWrite(dcm4_pin2,0);
       }
      }

   }
 val=-1;

}


//case R -> Analog reference
if(val==82){
  while (Serial.available()==0) {};
  val = Serial.read();
  if (val==48) analogReference(DEFAULT);
  if (val==49) analogReference(INTERNAL);
  if (val==50) analogReference(EXTERNAL);
  if (val==51) Serial.print("v3");
  val=-1;
}
```

```
} /* end loop statement                    */


/***************************************/
// Generic interrupt encoder functions//
/***************************************/
//Encoder on INT0
void encoder_0_change_m1() { //encoder0 mode 1x
  int chB=digitalRead(encoder_0_int2);
  if (!chB) { encoder_0_position++;}
  else { encoder_0_position--; }
}
void encoder_0_change_m2() { //encoder0 mode 2x
  int chB=digitalRead(encoder_0_int2);
  int chA=digitalRead(corresp[0]);
  if ((chA & !chB)|(!chA & chB)) { encoder_0_position++; }
  else { encoder_0_position--; }
}
void encoder_change_m4_A0(){//encoder0 mode 4x chA
  int chA=digitalRead(corresp[0]);
  int chB=digitalRead(encoder_0_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_0_position++; }
  else { encoder_0_position--; }
}
void encoder_change_m4_B0(){//encoder0 mode 4x chB
  int chA=digitalRead(corresp[0]);
  int chB=digitalRead(encoder_0_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_0_position++; }
  else { encoder_0_position--; }
}
//Encoder on INT1
void encoder_1_change_m1() { //encoder1 mode 1x
  int chB=digitalRead(encoder_1_int2);
```

```
  if (!chB) { encoder_1_position++;}
  else { encoder_1_position--; }
}
void encoder_1_change_m2() { //encoder1 mode 2x
  int chB=digitalRead(encoder_1_int2);
  int chA=digitalRead(corresp[1]);
  if ((chA & !chB)|(!chA & chB)) { encoder_1_position++; }
  else { encoder_1_position--; }
}
void encoder_change_m4_A1(){//encoder1 mode 4x chA
  int chA=digitalRead(corresp[1]);
  int chB=digitalRead(encoder_1_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_1_position++; }
  else { encoder_1_position--; }
}
void encoder_change_m4_B1(){//encoder1 mode 4x chB
  int chA=digitalRead(corresp[1]);
  int chB=digitalRead(encoder_1_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_1_position++; }
  else { encoder_1_position--; }
}
//Encoder on INT2
void encoder_2_change_m1() { //encoder2 mode 1x
  int chB=digitalRead(encoder_2_int2);
  if (!chB) { encoder_2_position++;}
  else { encoder_2_position--; }
}
void encoder_2_change_m2() { //encoder2 mode 2x
  int chB=digitalRead(encoder_2_int2);
  int chA=digitalRead(corresp[2]);
  if ((chA & !chB)|(!chA & chB)) { encoder_2_position++; }
  else { encoder_2_position--; }
}
```

```
void encoder_change_m4_A2(){//encoder2 mode 4x chA
  int chA=digitalRead(corresp[2]);
  int chB=digitalRead(encoder_2_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_2_position++; }
  else { encoder_2_position--; }
}
void encoder_change_m4_B2(){//encoder2 mode 4x chB
  int chA=digitalRead(corresp[2]);
  int chB=digitalRead(encoder_2_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_2_position++; }
  else { encoder_2_position--; }
}
//Encoder on INT3
void encoder_3_change_m1() { //encoder3 mode 1x
  int chB=digitalRead(encoder_3_int2);
  if (!chB) { encoder_3_position++;}
  else { encoder_3_position--; }
}
void encoder_3_change_m2() { //encoder3 mode 2x
  int chB=digitalRead(encoder_3_int2);
  int chA=digitalRead(corresp[3]);
  if ((chA & !chB)|(!chA & chB)) { encoder_3_position++; }
  else { encoder_3_position--; }
}
void encoder_change_m4_A3(){//encoder3 mode 4x chA
  int chA=digitalRead(corresp[3]);
  int chB=digitalRead(encoder_3_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_3_position++; }
  else { encoder_3_position--; }
}
void encoder_change_m4_B3(){//encoder3 mode 4x chB
  int chA=digitalRead(corresp[3]);
  int chB=digitalRead(encoder_3_int2);
```

```
  if ((!chA & !chB)|(chA & chB)) { encoder_3_position++; }
  else { encoder_3_position--; }
}
//Encoder on INT4
void encoder_4_change_m1() { //encoder4 mode 1x
  int chB=digitalRead(encoder_4_int2);
  if (!chB) { encoder_4_position++;}
  else { encoder_4_position--; }
}
void encoder_4_change_m2() { //encoder4 mode 2x
  int chB=digitalRead(encoder_4_int2);
  int chA=digitalRead(corresp[4]);
  if ((chA & !chB)|(!chA & chB)) { encoder_4_position++; }
  else { encoder_4_position--; }
}
void encoder_change_m4_A4(){//encoder4 mode 4x chA
  int chA=digitalRead(corresp[4]);
  int chB=digitalRead(encoder_4_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_4_position++; }
  else { encoder_4_position--; }
}
void encoder_change_m4_B4(){//encoder4 mode 4x chB
  int chA=digitalRead(corresp[4]);
  int chB=digitalRead(encoder_4_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_4_position++; }
  else { encoder_4_position--; }
}
//Encoder on INT5
void encoder_5_change_m1() { //encoder5 mode 1x
  int chB=digitalRead(encoder_5_int2);
  if (!chB) { encoder_5_position++;}
  else { encoder_5_position--; }
}
```

```
void encoder_5_change_m2() { //encoder5 mode 2x
  int chB=digitalRead(encoder_5_int2);
  int chA=digitalRead(corresp[5]);
  if ((chA & !chB)|(!chA & chB)) { encoder_5_position++; }
  else { encoder_5_position--; }
}
void encoder_change_m4_A5(){//encoder5 mode 4x chA
  int chA=digitalRead(corresp[5]);
  int chB=digitalRead(encoder_5_int2);
  if ((chA & !chB)|(!chA & chB)) { encoder_5_position++; }
  else { encoder_5_position--; }
}
void encoder_change_m4_B5(){//encoder5 mode 4x chB
  int chA=digitalRead(corresp[5]);
  int chB=digitalRead(encoder_5_int2);
  if ((!chA & !chB)|(chA & chB)) { encoder_5_position++; }
  else { encoder_5_position--; }
}

/*************************************/
// Generic interrupt counter functions//
/*************************************/
//Counter on INT0
void counter_0_change() { //counter 0
  counter_0++;
}
//Counter on INT1
void counter_1_change() { //counter 1
  counter_1++;
}
//Counter on INT2
void counter_2_change() { //counter 2
  counter_2++;
```

```
}
//Counter on INT3
void counter_3_change() { //counter 3
  counter_3++;
}
//Counter on INT4
void counter_4_change() { //counter 4
  counter_4++;
}
//Counter on INT5
void counter_5_change() { //counter 5
  counter_5++;
}
```