São Paulo 19 de Novembro de 2015

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
DEPARTAMENTO DE ENGENHARIA MECATRÔNICA

TRABALHO DE CONCLUSÃO DE CURSO
MONOGRAFIA

Modelagem, controle e prototipação de veículo baseado em Pêndulo Invertido.

Orientandos:
Diego Delgado Colombo de Oliveira Nusp.: 7206221
Leonardo Novak Rossi Nusp.: 6846425

Orientador:
Prof. Dr. Roberto Spinola Barbosa

São Paulo 19 de Novembro de 2015

Monografia apresentada no Departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro.

Área de Concentração: Engenharia Mecatrônica

Catalogação-na-publicação

Rossi, Leonardo Novak
Modelagem, controle e prototipação de um veículo baseado em Pêndulo Invertido / L. N. Rossi, D. D. C. de Oliveira -- São Paulo, 2015.
68 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Controle Automático Digital 2.Modelagem Mecânica 3.Projeto de Hardware e Software embarcados 4.Filtragem de sinais 5.Sistemas de medição inercial (IMU) I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t. III.de Oliveira, Diego Delgado Colombo

São Paulo 19 de Novembro de 2015

DECLARAÇÃO DE ORIGINALIDADE

“Este relatório é apresentado como requisito parcial para obtenção do grau de Engenheiro Mecatrônico na Escola Politécnica da Universidade de São Paulo. É o produto do meu próprio trabalho, exceto onde indicado no texto. O relatório pode ser livremente copiado e distribuído desde que a fonte seja citada.”

Diego Delgado Colombo de Oliveira

Leonardo Novak Rossi

Resumo

Este relatório versa a respeito das etapas de modelagem mecânica e matemática de um protótipo veicular baseado em um pêndulo invertido, além de contemplar o projeto e implementação de todo o hardware e softwares necessários para alcançar o estado de equilíbrio vertical daquele.

Introduzem-se as motivações e o escopo do trabalho nas seções iniciais e apresentam-se os resultados no desenvolvimento.

O leitor acompanha no texto desde a fase de desenho do protótipo inicial em ambiente *CAD* até a fabricação do mesmo, e a validação entre as propriedades inerciais levantadas em *software* com aquelas encontradas no chassi construído.

Segue-se com o equacionamento da planta com o intuito de projetar-se um controlador capaz de equilibrar o veículo na posição ereta. Após a definição das funções de transferência numéricas da planta, levantadas tanto com modelos virtuais quanto por vias de métodos experimentais, finalmente, projeta-se um controlador capaz de satisfazer resposta dinâmica da planta predeterminada.

O problema clássico de controlar uma planta instável por vias da realimentação da saída, amplamente abordado e de caráter pouco desafiador, mascara uma complicação intrínseca à disposição da planta em si, que é a correta observação do ângulo do pêndulo com a vertical, tarefa não trivial provendo-se indisponibilidade de um referencial inercial no caso da planta pendular invertida veicular.

Métodos de filtragem e aquisição de dados inerciais são discutidos e uma solução de observação de ângulo é implementada. Resultados experimentais são exibidos ao final do texto e considerações finais a respeito da relação entre a modelagem teórica de sistemas mecatrônicos e o comportamento real de plantas desta natureza são realizados na conclusão.

O Protótipo final realizou com sucesso a tarefa de se equilibrar na posição vertical.

Sumário

1. Introdução

Desde o início da era humana no planeta Terra, o ser humano tem tentado manipular e dominar o meio ao seu redor. Nos primórdios buscava-se dominar e controlar as forças da natureza, de maneira simples e reduzida, a fim de obter vantagens e benefícios. Com a evolução do homem e o desenvolvimento das sociedades, a tecnologia passou a ser parte integrante do nosso dia a dia e desta maneira, cada vez mais, a humanidade busca métodos superiores e de maior eficiência, de forma geral, o homem procura controlar de forma completa e segura os processos e físicos e naturais.

O tema de sistemas pendulares, que são recorrentes na natureza e em muitos sistemas mecânicos produzidos pelos homens despertou muita atenção de estudiosos. Sistemas pendulares, caracterizados por realizarem movimentos circulares oscilatórios em torno de um ponto fixo, tem seu controle almejado por cientistas, matemáticos e engenheiros desde os tempos antigos.

O grande astrônomo e matemático italiano Galileu Galilei foi responsável por iniciar os estudos destes sistemas mecânicos nos primeiros anos do século XVII, inaugurando o estudo do que se chamou de movimento harmônico simples. O desenvolvimento de conhecimento e de técnicas de controle de sistemas pendulares permitiram então o surgimento dos relógios pendulares que foram inventados e aperfeiçoados por Christian Huygens, proeminente cientista e pensador holandês, e são descritos em sua obra *Horologium Oscillatorium* do ano de 1673. Os relógios de pêndulo implicaram grande impacto na história do homem, em vista de que foram contundentes no desenvolvimento da astronomia e por consequência nas áreas de geolocalização e navegação.

No mundo atual, muitos sistemas e fenômenos que são objetos de controle podem ser modelados matematicamente através do modelo de pêndulo invertido e suas variações como o pêndulo duplo invertido. Exemplos práticos que podem ser modelados dessa maneira são os braços e pernas de um ser humano (Hurst, 2005), o comportamento de navios sujeitos a oscilações forçadas provocadas pelo movimento de ondas (A. H. Nayfeh, 2007) e manipuladores robóticos (Sadin, 2003).

1.1. Objetivos e Motivações

Este trabalho objetiva desenvolver e implementar um processo de controle capaz de estabilizar um pêndulo invertido sobre rodas, sujeito a ruídos aleatórios, em torno de sua posição superior baseando-se num controlador clássico digital.

O projeto ainda ambiciona o desenvolvimento de um algoritmo de análise e predição de estados em tempo real através de um processamento de filtragem e observação dos sinais gerados pela rede de sensoriamento, que posteriormente possa ser utilizado na realimentação da planta controlada.

Cumpridos tais objetivos, visa-se deixar abertas as portas para a extensão do projeto, o controle da mesma planta com um número maior de graus de liberdade, de forma progressiva, até atingirmos o controle integral do pêndulo invertido sobre rodas, podendo controlar não só o seu equilíbrio em uma posição genérica, mas também sua capacidade de transladar e até realizar movimento plano, ao seguir uma trajetória no solo.

Move-nos também a vontade de expandir os limites de nosso conhecimento relacionado à implementação de observadores de estado e filtros e à fabricação e modelagem física de um protótipo previamente projetado. Principalmente ambiciona-se realizar uma tarefa plena de

engenharia que venha por à prova nossos conhecimentos da teoria e a capacidade de emprega-la na resolução de um problema de controle em uma planta real naturalmente instável.

Objetiva-se também legar à academia um trabalho que possa fomentar e auxiliar o estudo e desenvolvimento de técnicas de controle por novos estudantes de engenharia e interessados, levando-se em conta que o controlador PID e suas variações são comumente a primeira forma de controle explorada por aqueles que abordam as teorias de controle (Ogata, 2010).

1.2. Metodologia de projeto e Definição do Problema

O projeto segue linearmente, porém algumas de suas etapas estão atreladas, e são refinadas com um viés de interdependência. A seguir algumas etapas bem definidas do trabalho:

- Primeiramente será projetado um sistema mecânico na configuração de pêndulo invertido sobre rodas, com o uso de ferramentas de *design* computacional. Esta etapa fornecerá parâmetros inerciais e de massa do sistema, provida de propriedades geométricas e de densidade das peças envolvidas.
- Um modelo matemático da planta é então equacionado e linearizado, adotando-se hipóteses pertinentes. As equações resultantes são levadas ao domínio da frequência para revelar funções de transferência da planta.
- Inicia-se então o projeto de um controlador em ambiente de simulação usando-se os as funções de transferência encontradas, o tempo de amostragem é escolhido de acordo com a frequência natural da planta. Iterativamente, seleciona-se um controlador que satisfaça aos requisitos de projeto.
- O protótipo mecânico é construído e validado conforme as propriedades de massa encontradas no modelo em *CAD*, alterações na planta real e simulada são realizadas conforme necessário.
- Projeta-se um observador de estados e programa-se este e o controlador escolhido em sua forma discreta (equação de diferenças), por vias de software em um microcontrolador. Hardware eletrônico é projetado, dimensionado e fabricado, a planta final é montada e inicia-se a fase de testes.
- O controlador final é ajustado para melhoria de desempenho e hipóteses são levantadas em relação a possíveis discrepâncias entre a planta final e a simulada. Parâmetros de desempenho são observados experimentalmente.

A interdependência de tarefas pode ser notada, por exemplo, na etapa de construção do protótipo e dimensionamento do controle. Pode-se projetar a planta em CAD e um controlador que se adeque aos requisitos de desempenho, tudo em ambiente puramente computacional. Isto muda de figura quando se constrói um protótipo, pois a planta real pode divergir consideravelmente da planta projetada, fazendo assim com que as simulações devam ser recorrigidas para se adequar à situação real. Isto exemplifica um fluxo de trabalho em espiral, que engloba progressos e revisões em mais de um *front* de pesquisa diferente, convergindo para uma solução ideal por vias do estudo simultâneo de todos os *fronts* envolvidos.

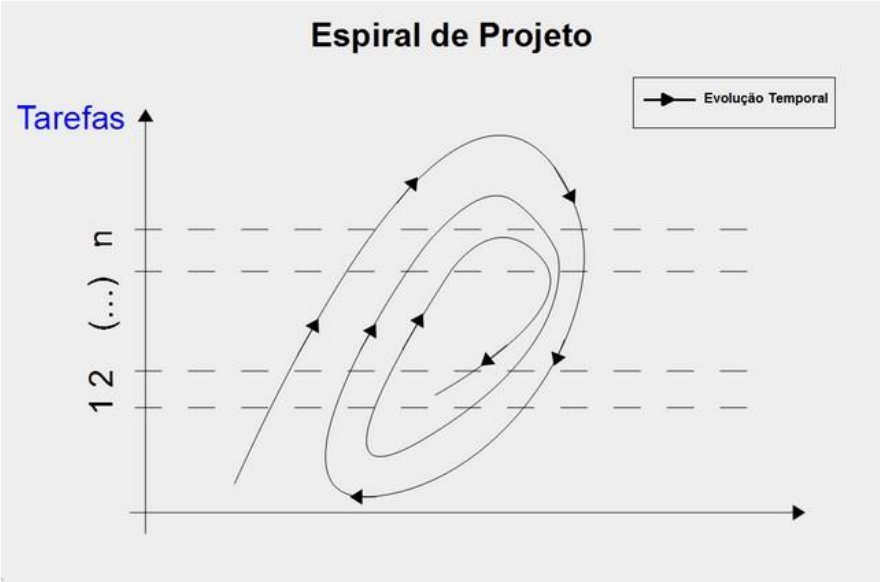

Figura 1: Espiral de projeto clássica. Os eixos representam o progresso em atividades mutuamente dependentes

2. Desenvolvimento

2.1. Revisão e estado da arte

A temática do controle do pêndulo invertido sobre rodas fornece um vastíssimo horizonte para desenvolvimento e testes de novas técnicas e abordagens de controle, pois contempla inúmeros conceitos, dos mais básicos aos mais complexos, da engenharia de controle. Se tratando de um sistema inerentemente instável requer um controlador robusto e de alta precisão e desempenho, portanto, configura-se em uma problemática valorosa para por à prova o desempenho de novas teorias e abordagens de controladores.

No início dos anos 50 do século passado surgiram os primeiros e fundamentais trabalhos publicados sobre o controle de pêndulos invertidos e suas variações. Estes trabalhos eram fundamentalmente utilizados para o ensino de técnicas de controle com realimentação linear e estabilização de sistemas instáveis em malha aberta (Kailath, 1980).

A primeira vez que um trabalho sobre o controle de um pêndulo invertido em torno de uma posição de equilíbrio instável foi publicada foi em 1960, com a tese de bacharelado de J. K. Roberge no instituto de tecnologia de Massachusetts (Roberge, 1960) Logo se seguiu a publicação de outro trabalho com a mesma temática, produzido em 1966 por J. F. Schaefer e R. H. Cannon (Shaefer, et al., 1966).

Estas obras se configuraram como pedra fundamental no processo documental do controle de sistemas inerentemente instáveis e técnicas de controle para sistemas deste tipo como o pêndulo invertido, sendo citados em diversas literaturas de ensino de técnicas de controle utilizadas em universidades pelo mundo como em (Khalil, 2002)

Em (Do, et al., 2010) é abordado o controle de um veículo sobre rodas que é modelado mecanicamente como um pêndulo invertido sobre rodas. Diferencia-se este trabalho, por conter uma problemática que envolve maior número de graus de liberdade, ainda visa o equilíbrio do

sistema em torno de uma posição de equilíbrio instável. A solução apontada é baseada em técnicas de controle através da observação e realimentação de estados do sistema.

Já (Juang, et al., 2013) opera de forma muito próxima dos objetivos e metas do presente trabalho, pois aborda o problema de controle de posição de um robô sobre duas rodas utilizando-se um sistema embarcado que implementa um controlador PID alimentado por um sinal de erro obtido através de um filtro complementar que processa os dados de acelerômetros e giroscópio.

2.2. Alternativas de projeto consideradas

O processo de definição da problemática a ser abordada incluiu a exploração de diversos tipos de sistemas mecânicos na configuração de pêndulo invertido.

Quanto ao sistema mecânico, foi aventada a possibilidade de se construir um pêndulo invertido sobre um carro transladante - PICT ou sobre trilhos - PIT. Ambos os modelos são sistemas com complexidade de controle inferior ao sistema tema desta monografia.
O modelo PICT, por possuir um carro transladante, possibilita a instalação de um sensor fixo em relação ao polo de rotação, garantindo um referencial inercial e permitindo a detecção de posição do pêndulo invertido de maneira fácil e precisa.

O modelo PIT garante a determinação imediata da posição espacial do sistema e sua velocidade, possibilitando inclusive ótimo controle de posição translacional do carro dependendo do acionamento utilizado. Esta configuração é a mais utilizada quando se ambiciona o controle MIMO da planta, não só a posição angular do pêndulo, mas também a translacional da base rotante.

Mesmo que estes modelos facilitem o trabalho de controle ao dispensarem a implementação de um observador de estados, eles não possuem aplicação tão flexível quanto ao pêndulo invertido sobre rodas.

Quanto ao sistema de controle, analisou-se a possibilidade de execução do projeto de um controlador em tempo contínuo, porém, com o grande avanço da velocidade e poder de processamento de dados dos sistemas embarcados, aliados a sua miniaturização física e a preços cada vez menores destes dispositivos, essa alternativa foi refutada, optando-se por uma implementação digital, que não só vislumbra as técnicas mais modernas de controle como também aborda a temática do controle em si de maneira mais completa.

Sobre as técnicas de controle, durante a fase de fundamentação teórica e revisão bibliográfica foram conhecidas diversas abordagens para sistemas não lineares e instáveis e com realimentação de variáveis desconhecidas e/ou de difícil determinação, entre elas a abordagem de controle por observadores Grey (Huang Shiuh-Jer, 2000), abordagem por lógica Fuzzy (V. Lopes Jr., 2010) e através da análise e estimação de estados usando-se filtros das mais diversas formas. Adotamos esta última por estarmos mais familiarizados com os conceitos e técnicas, além de ser altamente difundida e aplicada nos sistemas mecatrônicos encontrados no mercado de engenharia atual.

2.3. Requisitos de projeto

O protótipo final deve:

- Ser capaz de equilibrar-se em regime permanente (leia-se por um tempo maior do que 30 vezes o período natural da planta).
- Quando em equilíbrio, ser minimamente robusto a distúrbios externos, e.g. toques no mastro, pequenas inclinações e acidentes no solo, etc.
- Ser minimamente resistente a possíveis quedas, caso estas venham a acontecer.

- O hardware mecânico deve configurar uma planta simples, porém funcional, e de fácil fabricação. É objetivada a construção do protótipo minimizando-se o uso de máquinas ferramenta ou especiais, com o intuito de diminuir tempo gasto em usinagem ou tarefas de construção mecânica.
- O hardware eletrônico deve ser implementado em placa de circuito impresso, minimizando-se jumpers e elementos construtivos que comprometam a robustez do mesmo.
- O software deve ser embarcado e desenvolvido em IDE única e de fácil acesso ao público, com o intuito de facilitar o uso do código para terceiros sob licença aberta de uso comum.
- O protótipo deve ser capaz de ser ajustado de forma *on the fly* ou durante sua operação, sem a necessidade de atualização do software.

2.4. Modelagem Mecânica

A modelagem mecânica da estrutura foi separada em duas etapas diferentes, sendo a primeira relacionada à obtenção de uma função de transferência para a parte predominantemente inercial da planta, e.g. chassi e peças de massas não desprezíveis relacionados ao hardware.

2.4.1. Projeto estrutural em ambiente CAD

Nesta etapa foi desenhado em ambiente computacional o modelo do pêndulo invertido sobre rodas. Para tanto foi utilizado um software CAD (*SOLIDWORKS* 2014) e todas as peças foram modeladas com as cotas estabelecidas em projeto. Desenhos de fabricação podem ser encontrados no primeiro anexo.

Foram modelados o Corpo (Semelhante à haste do pêndulo), rodas, suporte da bateria e dos motores e suporte para o controlador embarcado, sendo que inércias oriundas de placas de circuitos, fios e outros componentes serão desprezados em uma primeira análise.

Os mastros, inicialmente não instalados, possibilitam possível calibragem inercial da planta por adição de pesos.

Após cada peça modelada realizou-se a etapa de montagem das peças, obtendo-se assim um modelo 3D final. Figuras ilustrativas encontram-se no Anexo 1.

Os mastros, inicialmente não instalados, possibilitam possível calibragem inercial da planta por adição de pesos.

Com o modelo 3D projetado, foram atribuídas então as características físicas dos materiais empregados a fim de se obterem as matrizes de Inércia do modelo e outras propriedades fundamentais para realização da função de transferência da planta de controle, no caso o pêndulo invertido sobre rodas. O material mais abundante na planta será madeira, e assim selecionamos uma madeira extremamente comum e de fácil manuseio e usinagem, chamada Tauari (IPT).

Conhecidas as propriedades físicas típicas do Tauari, basta inserir a densidade em ambiente de simulação que o *software* se encarrega de definir a posição do CG de cada peça bem como suas propriedades de inércia.

Para materiais de densidade variável ou desconhecida, as peças foram todas pesadas, e adotou-se a hipótese de densidade uniforme ao longo do volume. Tal aproximação é válida para a bateria e fixadores metálicos, talvez não sendo ótima para os motores em si. No entanto, o problema é sanado devido à própria configuração do sistema, que pressupõe dois motores montados simetricamente em relação ao plano direito do sistema, assim fazendo com que quaisquer erros axiais de posicionamento do CG dos motores sejam anulados.

Finalmente, definidas as propriedades de todas as peças bem como da montagem como um todo, calculamos a posição exata do CG do sistema em função de um sistema de coordenadas definido na intersecção do plano direito da planta com o eixo de giro dos motores (É ao redor deste sistema que a planta revolucionará, caracterizando este ponto como o ponto de articulação do modelo da haste do pêndulo).

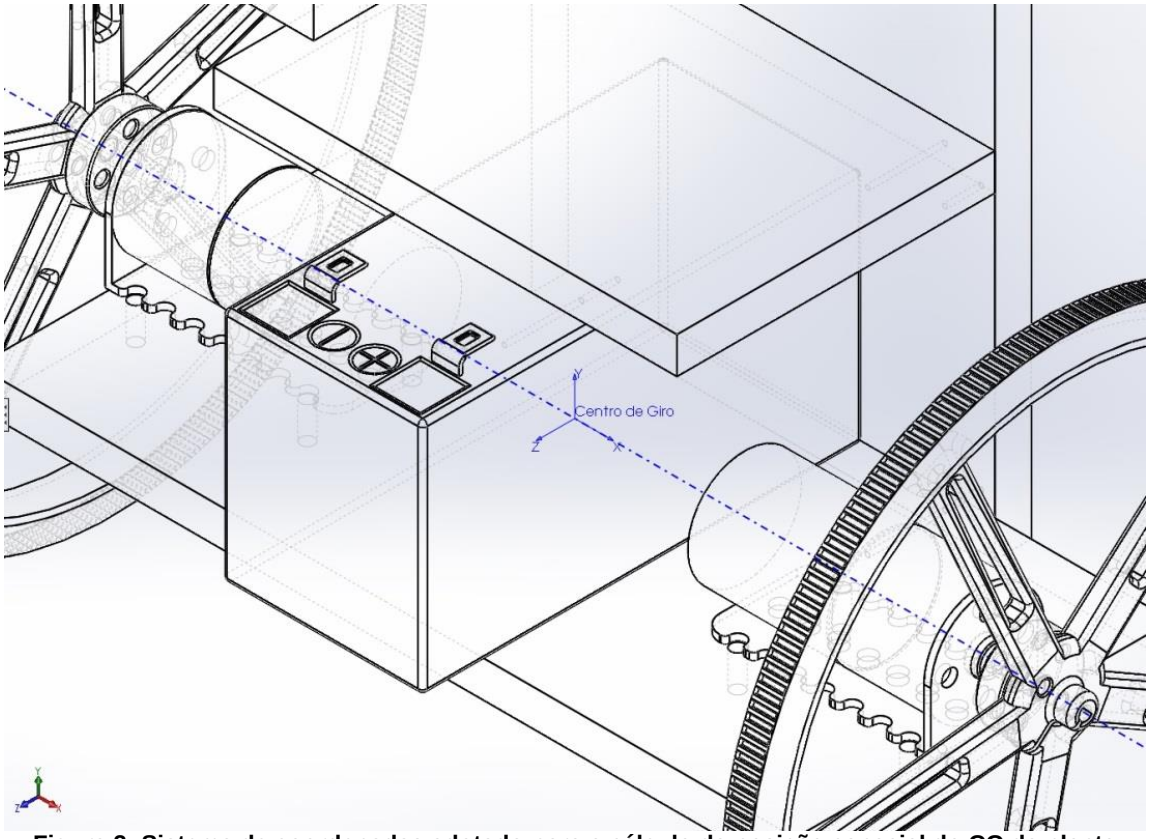

Figura 2: Sistema de coordenadas adotado para o cálculo da posição espacial do CG da planta

Levantam-se as seguintes propriedades do chassi oscilante:

```
Mass properties of Segway-MONTAGEM
   Configuration: Valor predeterminado
   Coordinate system: Centro de Giro

Mass = 1.027 kilograms

Volume = 0.001 cubic meters

Surface area = 0.199 square meters

Center of mass: ( meters )
     X = -0.000
     Y = 0.043
     Z = -0.001
```

Figura 3: Propriedades inerciais e geométricas numéricas do chassi

Do modelo em CAD, retiramos os parâmetros apresentados na *Tabela 1* a serem substituídos no equacionamento matemático da planta. Ainda foram medidos

experimentalmente os valores de massa do protótipo fabricado, as medidas experimentais e teóricas coincidiram satisfatoriamente.

Tabela 1: Parâmetros geométricos de massa utilizados na função de transferência da planta

m_r	0.0268 kg	Massa do conjunto de rodas e rotores
m_p	1.027 kg	Massa do pêndulo, bateria e estatores
l	0.040 m	Distância vertical do CG até o centro de giro
R	0.035 m	Raio das rodas
g	9,8 m/s²	Aceleração da gravidade

2.4.2. Equacionamento matemático da planta mecânica

O chassi foi modelado usando-se equacionamento matemático obtido pelo uso de do Método de Lagrange. A *figura 3* ilustra parâmetros e disposição da planta.

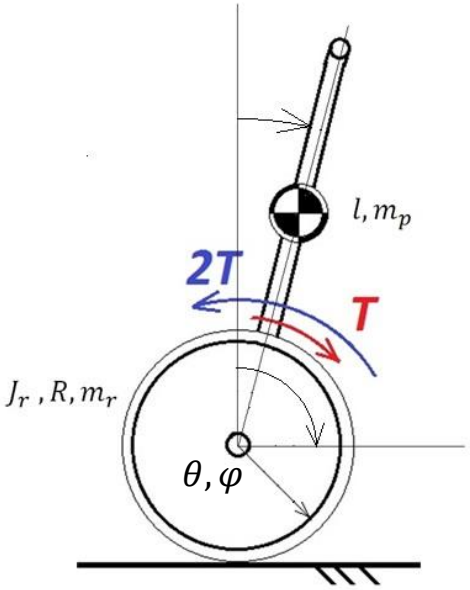

Figura 4 - Planta dinâmica usada para modelar o sistema

O sistema do pêndulo invertido é separado em dois corpos, um composto pelo chassi do veículo, baterias, hardware e armaduras dos motores elétricos, e outro composto por dois rotores, um de cada motor, girando supostamente síncronos, como se compondo um único corpo rígido juntamente às rodas e pneus.

Desprezam-se todas as dissipações de Rayleigh associadas, começando por calcular as energias cinética e potencial do sistema. Considera-se θ o ângulo de giro do chassi e φ o ângulo de giro das duas rodas.

$$E_c = 2\left(\frac{J_r\dot{\varphi}^2}{2} + \frac{m_r\dot{x}^2}{2}\right) + \frac{m_p\dot{v}_p^2}{2}$$

$$E_p = mgl(1-\cos\theta)$$

Onde $\dot{v}_p$ é a velocidade translacional do baricentro do pêndulo, seguindo a relação:

$$\begin{cases}\dot{x}_p = \dot{x} + \dot{\theta}l\cos\theta \\ \dot{y}_p = \dot{\theta}l\sin\theta\end{cases}$$

Ainda, temos a relação entre a rotação das rodas e o deslocamento translacional do centro de giro:

$$x = \varphi R$$

Assim, temos nosso lagrangeano:

$$L = E_c - E_p$$

$$L = 2\left(\frac{J_r\dot{\varphi}^2}{2} + \frac{m_r\dot{x}^2}{2}\right) + \frac{m_p((\dot{x}+\dot{\theta}l\cos\theta)^2 + (\dot{\theta}l\sin\theta)^2)}{2} - m_pgl + m_pgl\cos\theta$$

Escrevemos o langrangeano exclusivamente em função dos dois giros:

$$L = \left(\frac{3m_rR^2}{2} + \frac{m_pR^2}{2}\right)\dot{\varphi}^2 + \frac{m_p\dot{\theta}^2l^2}{2} + m_p\dot{\varphi}\dot{\theta}Rl\cos\theta - m_pgl + m_pgl\cos\theta$$

A forma acima que explicita os termos de massa nas inércias rotativas envolvidas assume modelos de massa concentrada para o corpo do pêndulo em seu centro de gravidade, e um modelo de disco de massa distribuída para as rodas.

E agora encontra-se as duas equações de movimento; vamos partir para a primeira, levando em conta o primeiro grau de liberdade como φ:

$$\frac{\partial L}{\partial\dot{\varphi}} = (3m_rR^2 + m_pR^2)\dot{\varphi} + m_p\dot{\theta}Rl\cos\theta \xrightarrow{d/dt} (3m_rR^2 + m_pR^2)\ddot{\varphi} + m_pRl[\ddot{\theta}\cos\theta + \dot{\theta}^2\cos\theta]$$

$$\frac{\partial L}{\partial\varphi} = 0$$

A primeira equação segue para a dinâmica das rodas:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}}\right) - \frac{\partial L}{\partial \varphi} = \sum F_{ext}$$

$$\left(3m_r R^2 + m_p R^2\right)\ddot{\varphi} + m_p R l\ddot{\theta}\cos\theta + m_p R l\dot{\theta}^2\cos\theta = -2T \qquad \cdots \qquad (1)$$

Vale notar que a somatória das forças externas relacionadas às rodas é *2T* e não *T,* pois modelamos ambas as rodas como sendo um único corpo rígido.

Continuamos para a dedução algébrica da segunda equação dinâmica, desta vez, acoplada ao grau de liberdade θ.

$$\frac{\partial L}{\partial \dot{\theta}} = m_p l^2\dot{\theta} + m_p\dot{\varphi}Rl\cos\theta \xrightarrow{d/dt} m_p l^2\ddot{\theta} + m_p Rl\left[\ddot{\varphi}\cos\theta - \dot{\theta}\dot{\varphi}\sin\theta\right]$$

$$\frac{\partial L}{\partial \theta} = -m_p Rl\dot{\varphi}\dot{\theta}\sin\theta - m_p gl\sin\theta$$

A segunda equação segue para a dinâmica do chassi:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = \sum F_{ext}$$

$$m_p l^2\ddot{\theta} + m_p Rl\ddot{\varphi}\cos\theta + m_p gl\sin\theta = 2T \qquad \cdots \qquad (2)$$

Prosseguimos com a linearização em torno da posição de equilíbrio instável, com o pêndulo de pé. Para tal, podemos considerar as seguintes aproximações:

$$\begin{cases} sin\theta \cong \theta \\ cos\theta \cong 1 \end{cases}$$

Estas considerações fazem com que termos de segunda ordem, como θ^2 ou $\dot{\theta}^2$ possam ser desprezados, revelando as seguintes equações linearizadas:

$$\left(3m_r R^2 + m_p R^2\right)\ddot{\varphi} + m_p Rl\ddot{\theta} = -2T \qquad (3)$$

$$m_p l^2\ddot{\theta} + m_p Rl\ddot{\varphi} + m_p gl\theta = 2T \qquad (4)$$

Podemos explicitar os termos que multiplicam as os giros para facilitar o tratamento algébrico:

$$\begin{cases} \left(3m_r R^2 + m_p R^2\right) = A \\ m_p Rl = B \\ m_p l^2 = C \\ m_p gl = D \end{cases}$$

Substituindo as relações acima na eq. (4) e isolando o termo $\ddot{\varphi}$, obtemos:

$$\ddot{\varphi} = \frac{2T - C\ddot{\theta} - D\theta}{B} \tag{5}$$

Substituindo (5) em (3), explicita-se a dinâmica entre o torque externo (provido pelo motor) e a variável θ.

$$A\left(\frac{2T - C\ddot{\theta} - D\theta}{B}\right) + B\ddot{\theta} = -2T \tag{6}$$

Preparamos a equação para a transformação de Laplace.

$$\left(B - \frac{AC}{B}\right)\ddot{\theta} - \frac{AD}{B}\theta = -\left(\frac{2A}{B} + 2\right)T \tag{7}$$

E obtemos pela transformada, a seguinte função de transferência:

$$\frac{\Theta(s)}{T(s)} = \frac{-2\left({}^{A}/_{B} + 1\right)}{\left(B - \frac{AC}{B}\right)s^2 - \frac{AD}{B}} = \frac{2\left({}^{A}/_{B} + 1\right)}{\left(\frac{AC}{B} - B\right)s^2 + \frac{AD}{B}} \tag{8}$$

Substituindo os parâmetros inerciais encontrados na seção anterior, obtemos a função de transferência final numérica para a planta.

$$A = 0.00136$$
$$B = 0.00155$$
$$C = 0.00190$$
$$D = 0.43278$$

$$\frac{\Theta(s)}{T(s)} = \frac{2\left(\frac{A}{B} + 1\right)}{\left(\frac{AC}{B} - B\right)s^2 + \frac{AD}{B}} = \frac{3.75535}{0.000121s^2 + 0.379838} \tag{9}$$

2.4.3. Equacionamento matemático dos motores

Para modelar a dinâmica dos motores de acionamento, inicialmente adotamos um modelo de segunda ordem amplamente utilizado (Ricci).

$$\frac{\Omega(s)}{V(s)} = \frac{K_t}{R_a B(1 + \tau_e s)(1 + \tau_m s) + K_t K_e} \tag{10}$$

As constantes de Torque e velocidade foram levantadas experimentalmente com o uso de um tacômetro estroboscópico e de uma balança. Os experimentos consistiram na mensuração de K_t usando-se um braço conhecido para pressionar a balança com o eixo do motor travado, neste etapa levanta-se ainda a resistência de enrolamento dos motores. K_e, constante de velocidade, foi encontrada medindo-se a DDP nos terminais do motor e a velocidade do mesmo em regime permanente. Os resultados se apresentam na *tabela 1*. Ilustrações dos ensaios encontram-se no anexo três.

Tabela 2: Parâmetros eletrodinâmicos do modelo dos motores

K_t	$2{,}68\ kgcm/A$	Constante de torque
K_e	$0{,}377\ Vs/rad$	Constante de velocidade
R_a	$5\ \Omega$	Resistência do enrolamento

O motor foi ensaiado com um degrau de partida unitário, e observamos a voltagem no transitório de aceleração do mesmo com o auxílio de um osciloscópio. Na saída obtida podemos notar características interessantes.

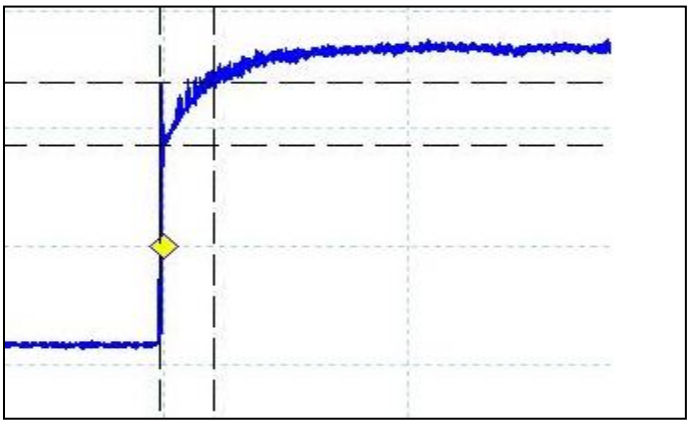

Figura 5: Resposta temporal transitória de voltagem nos terminais do motor, 200ms por divisão, T = 42ms.

A resposta tem notadamente um caráter de primeira ordem, o que leva a concluir que a dinâmica elétrica do motor pode ser desprezada fronte à sua dinâmica inercial; este característica é comum em motores elétricos DC de pequeno porte. Assim, podemos assumir o seguinte modelo (Ricci).

$$\frac{\Omega(s)}{V(s)} = \frac{K_m}{\tau s + 1}$$

Onde,

$$\frac{\Omega(s)}{V(s)} = \frac{K_t}{R_a B + K_t K_e}$$

Desprezando-se dissipações de Rayleigh no motor, devido às baixas velocidades nas quais a planta se encontrará oscilando em torno da posição ereta, podemos simplesmente adotar:

$$K_m = \frac{1}{K_e} = K_t$$

Assim, nos resta a equação:

$$\frac{\Omega(s)}{V(s)} = \frac{K_t}{\tau s + 1} \tag{10}$$

A subida íngreme representa o rotor sob o efeito do atrito estático de partida, podendo ser modelado como um pequeno atraso na planta, também desprezado. A inércia do rotor permanece desconhecida, assim usamos a constante de tempo obtida para modelá-la.

Procuramos uma função de transferência entre a voltagem nos terminais do motor e a respectiva saída de torque. As equações a seguir seguem do modelo do motor DC e da hipótese de indutância nula.

$$T(s) = K_t I(s) \tag{11}$$

$$\Omega(s) = K_v V(s) \tag{12}$$

$$V(s) = R_a I(s) \tag{13}$$

Procuramos uma relação entre $T(s)$ e $\Omega(s)$ para que possamos substituí-la na equação (10) e encontramos a função de transferência dos motores.

Das relações algébricas acima, obtemos:

$$\frac{T(s)}{\Omega(s)} = \frac{K_t}{K_v R_a}$$

Que, se substituído em (9) revela:

$$\frac{\Omega(s)}{V(s)}\frac{T(s)}{\Omega(s)} = \frac{K_t}{\tau s + 1} * \frac{K_t}{K_v R_a} = \frac{3{,}81}{0{,}042s + 1{,}01}$$

Após a obtenção das funções de transferência numéricas do sistema, concebemos o seguinte diagrama de blocos em malha aberta.

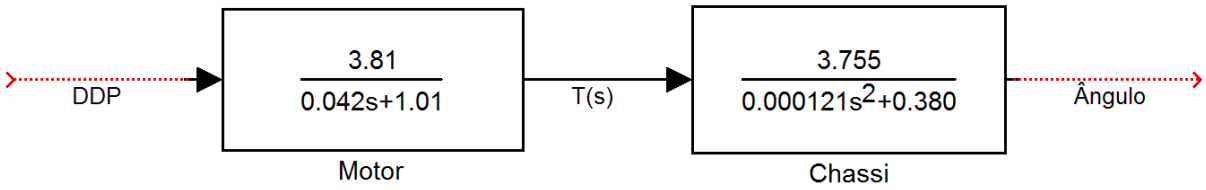

Figura 6: Diagrama de blocos em malha aberta do sistema Motor-Chassi

2.5. Técnicas de Controle

Em primeira análise, com intuito de validar o modelo teórico, realizou-se uma abordagem de controle clássico em ambiente virtual, a fim de analisar respostas e parâmetros do sistema previamente estabelecidos pela teoria. As simulações computacionais também possuem a finalidade de validar as hipóteses assumidas durante todo o processo de

modelagem da planta, como o comportamento dinâmico dos motores de corrente contínua e a preponderância das características inercias do pêndulo invertido sobre às dos motores.

Da técnica de controle clássica foram utilizados os métodos de design de controladores através das análises de diagramas do Lugar das Raízes e Diagramas de Bode.

O diagrama de Lugar das Raízes permite a visualização do lugar geométrico onde os polos dominantes de malha fechada da planta possam se localizar. Estes polos são os determinantes das características da resposta transiente do sistema controlado, e no caso deste projeto, caso a resposta transiente não seja conveniente, o pêndulo invertido jamais se estabilizará na posição de equilíbrio. Esta técnica é ainda essencial no projeto de controladores com requisitos temporais como tempo de resposta e acomodação do sistema. Para o projeto de controle do sistema de Pêndulo invertido sobre rodas foi determinado um tempo de acomodação de 150 ms.

Os Diagramas de Bode introduzem os diagramas de margem de ganho e fase da resposta em frequência do sistema; Estes diagramas são fundamentais para o entendimento de características frequências do sistema, permitindo um entendimento profundo sobre a dinâmica da planta e influência de controladores sobre a resposta dinâmica do sistema. No caso deste projeto, a análise destes diagramas permitirá a descrição da dinâmica da planta no espectro de frequências e a influência do ganho de controlador sobre a resposta da planta.

2.5.1. Análise do Sistema não compensado

A primeira etapa do projeto de controle consiste na análise da planta em malha aberta. Utilizando a ferramenta *sisotool* do software *Matlab* obtiveram-se os seguintes diagramas:

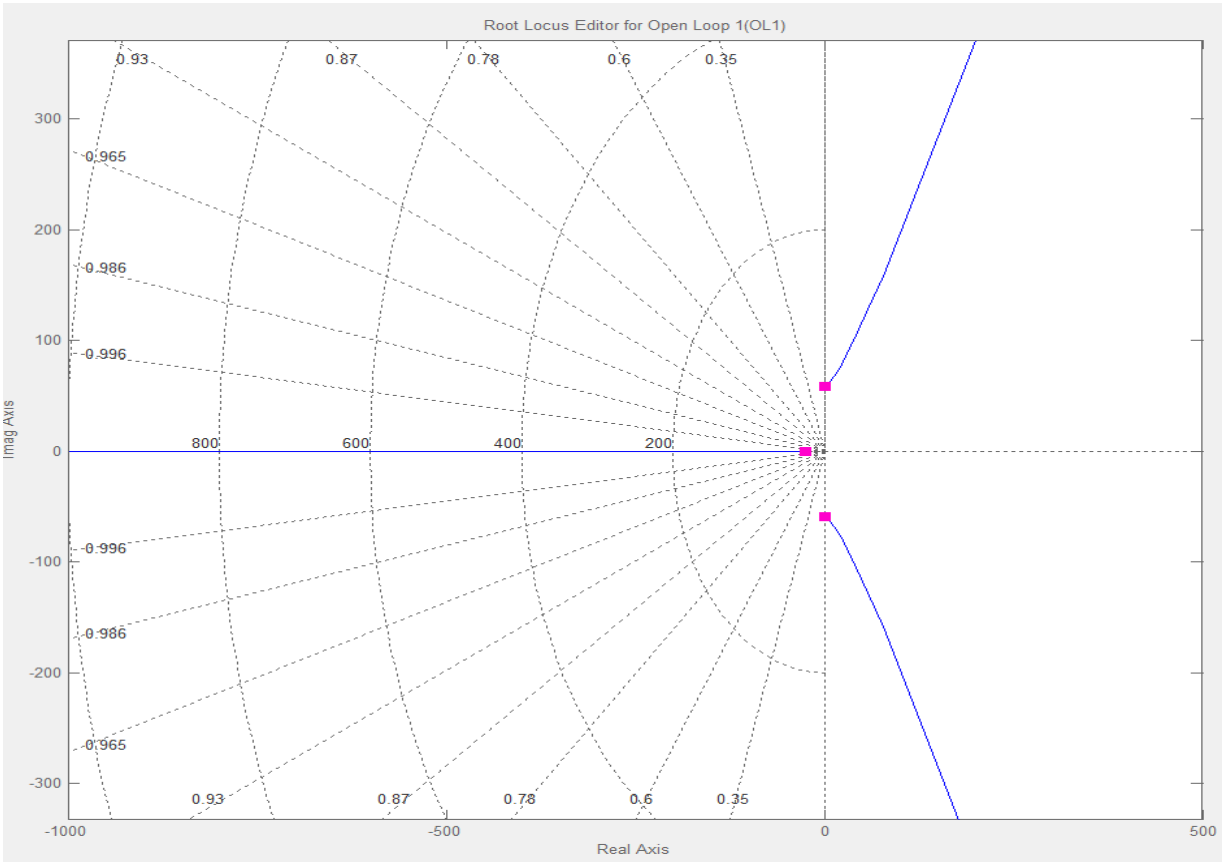

Figura 7: Lugar das Raízes do sistema não compensado

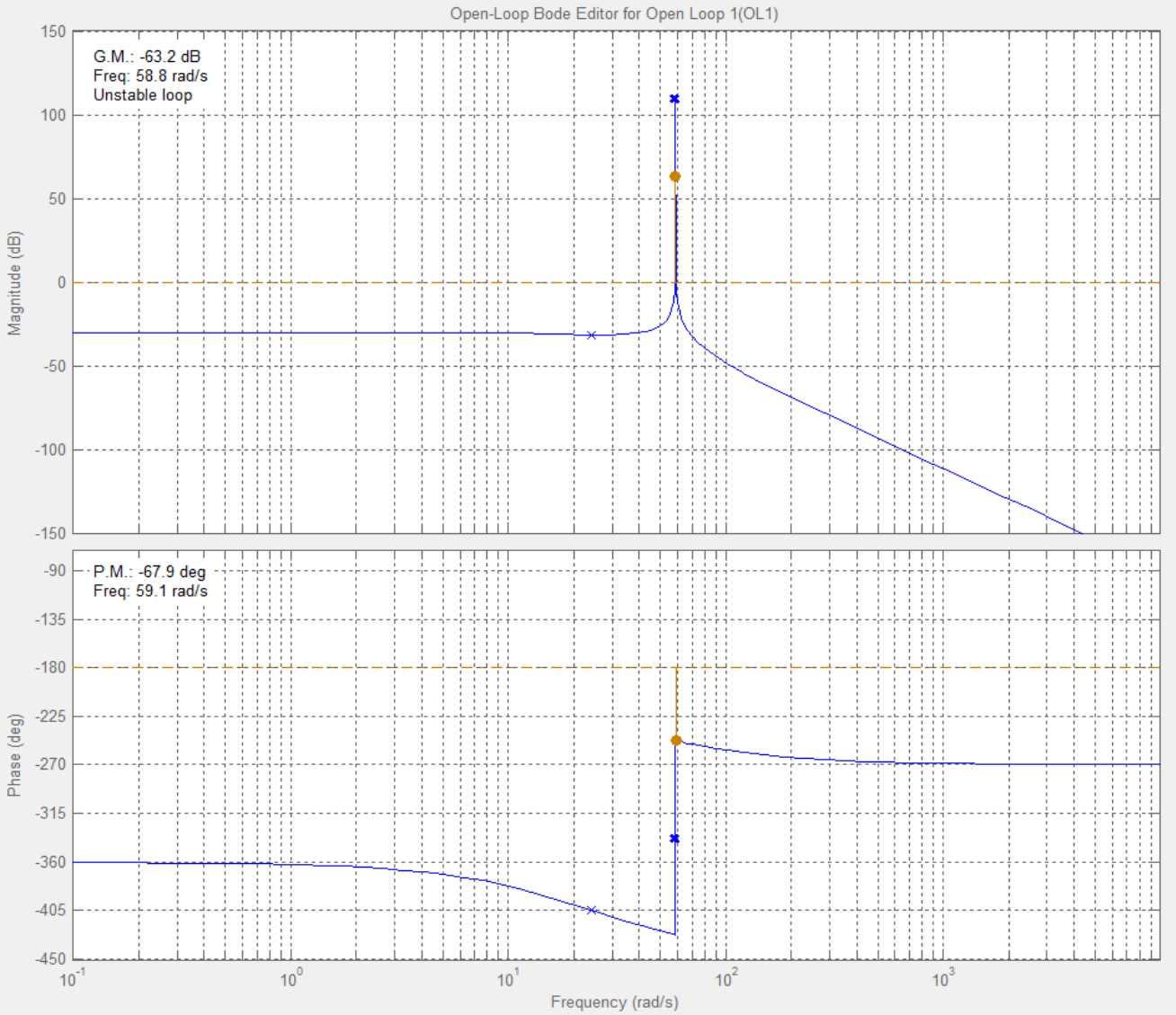

Figura 8: Diagrama de Bode do sistema não compensado

O Lugar das Raízes -LR- do sistema não compensado indica uma planta inerentemente instável, independentemente do ganho do sistema ele sempre possuirá polos dominantes em malha fechada sobre o eixo imaginário ou semiplano direito do plano S.

Os diagramas frequências corroboram com o LR sinalizando uma margem de fase sempre negativa e de no máximo -67,9 graus para a planta do projeto. Ainda, na região próximo à frequência 59 rad/s o sistema se torna altamente instável, devido as mudanças bruscas e degradantes da margem de ganho e fase.

2.5.2. Estratégia de Controle

Baseando-se nas informações obtidas anteriormente e nas teorias de controle clássico, adotou-se a abordagem de alteração do lugar das raízes com adição de compensadores, afim de aumentar a estabilidade do sistema e garantir margens de ganho e fase positivas, implicando em um sistema controlável.

Os compensadores de avanço de fase têm por característica principal adicionar fase à resposta do sistema com um zero e um polo, sendo o zero sempre mais próximo do eixo imaginário do plano s. Com isso, estes compensadores aumenta a margem de fase do sistema de controle em até 90 graus e deslocam o Lugar das Raízes da planta para à esquerda (Sistema mais estável implicando em uma melhora na resposta temporal do sistema.
O bloco típico de um compensador de avanço é (Castrucci, et al., 2011):

$$C(s) = \frac{s + z_c}{s + p_c}, z_c < p_c$$

2.5.3. Projeto do Compensador

A análise do diagrama de Bode fornece que a margem de fase máxima da planta não compensada é de aproximadamente -67,7 graus para a frequência de 58,8 rad/s. Portanto, através do método de tentativa e erro e alocação de polos e zeros respaldados pelas informações dos diagramas de LR e Bode, foi modelado um compensador de avanço que adiciona o máximo de margem de fase possível, 90 graus, ao sistema e desloca o LR para esquerda no plano S.
O compensador obtido foi:

$$C(s) = \frac{s + 6.46}{s + 400}, K_c = 0.2$$

E os diagramas de Lugar das Raízes e Bode resultante do Sistema Compensador + Planta foram:

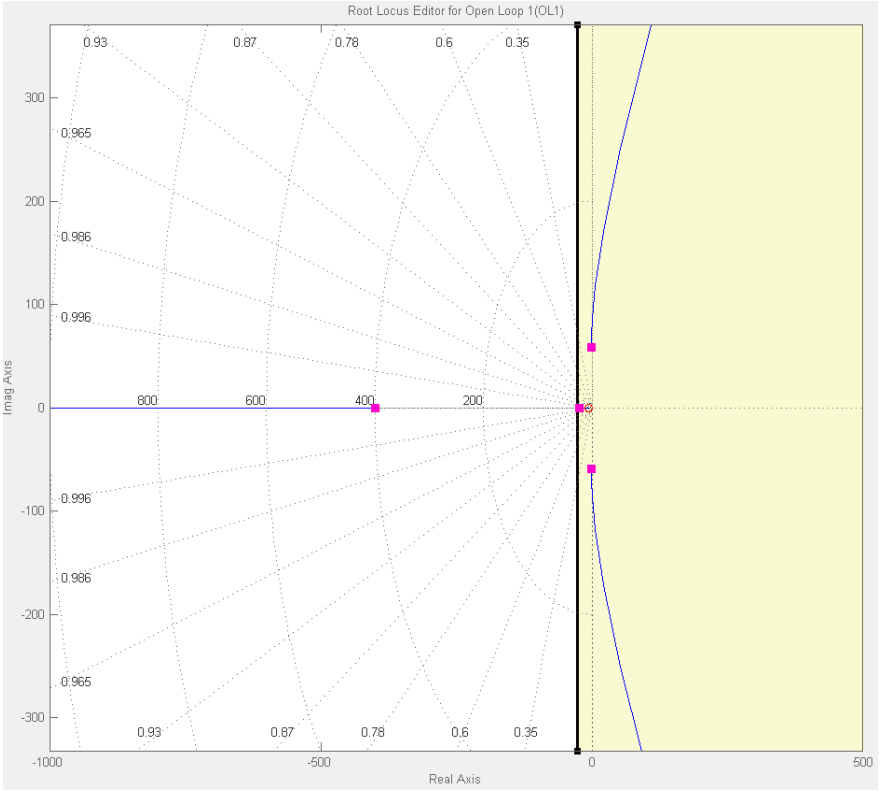

Figura 9: Lugar das Raízes - Sistema C(s)*G(s)

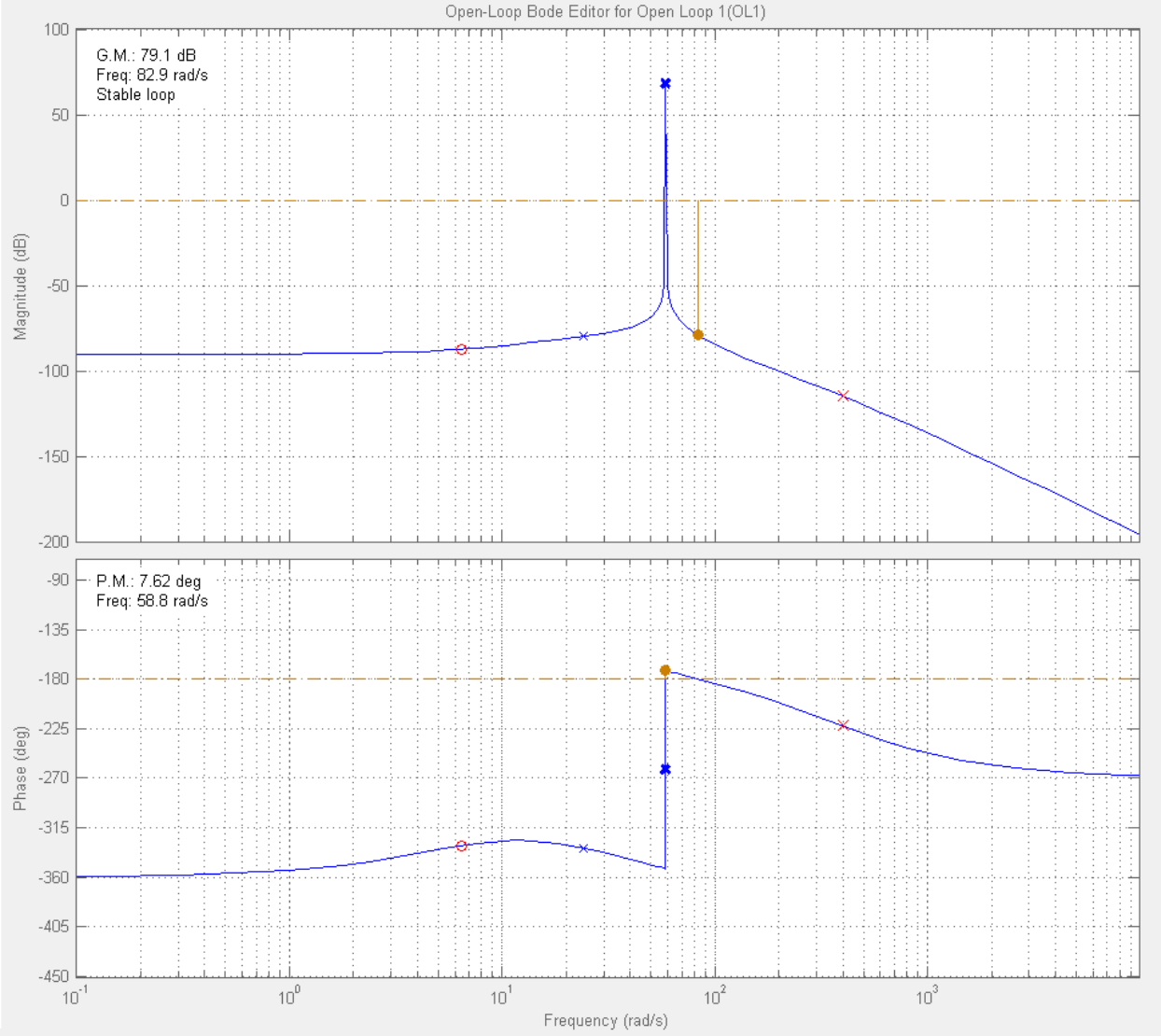

Figura 10: Diagrama de Bode - Sistema C(s)*G(s)

Com a adição do Compensador de Avanço na malha de controle nota-se uma mudança favorável no lugar das raízes, os ramos que se localizavam totalmente no semiplano direito do plano S se aproximaram do eixo imaginário, chegando mesmo à cruzá-lo e possuírem um intervalo de valores estáveis sobre o semiplano esquerdo de S. As margens de ganho e fase se tornaram positivas – MG = 79,1 dB e MF = 7,62 deg, o que indica que o sistema se tornou estável.

Para verificar as alterações nas características de resposta do sistema, simulou-se a aplicação de um distúrbio de 0.1 radianos (~5.7 graus) ao pêndulo invertido sobre rodas em sua posição de equilíbrio e obteve-se a seguinte malha de controle e gráficos de resposta e esforço de controle:

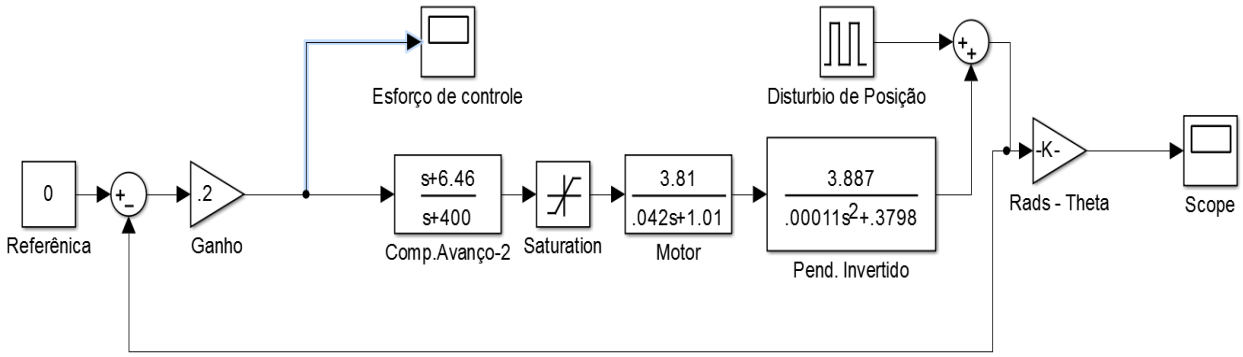

Figura 11: Malha de Controle em Simulink para Simulação

O modelo em *Simulink* simula um Compensador de Avanço com ganho de 0.1 em sequência com o a planta do motor e do pêndulo invertido. O bloco de saturação não permite que o motor opere acima das suas condições reais, fornecendo um torque impossível para o mesmo, o bloco referência em zero indica que o pêndulo se encontra na sua posição de equilíbrio, formando um ângulo de zero grau com a o eixo da direção vertical à base do sistema. O sistema gerador de pulsos gera um distúrbio de posição de 5.7 graus.
As leituras das saídas forneceram os seguintes gráficos temporais de resposta do sistema e esforço de controle:

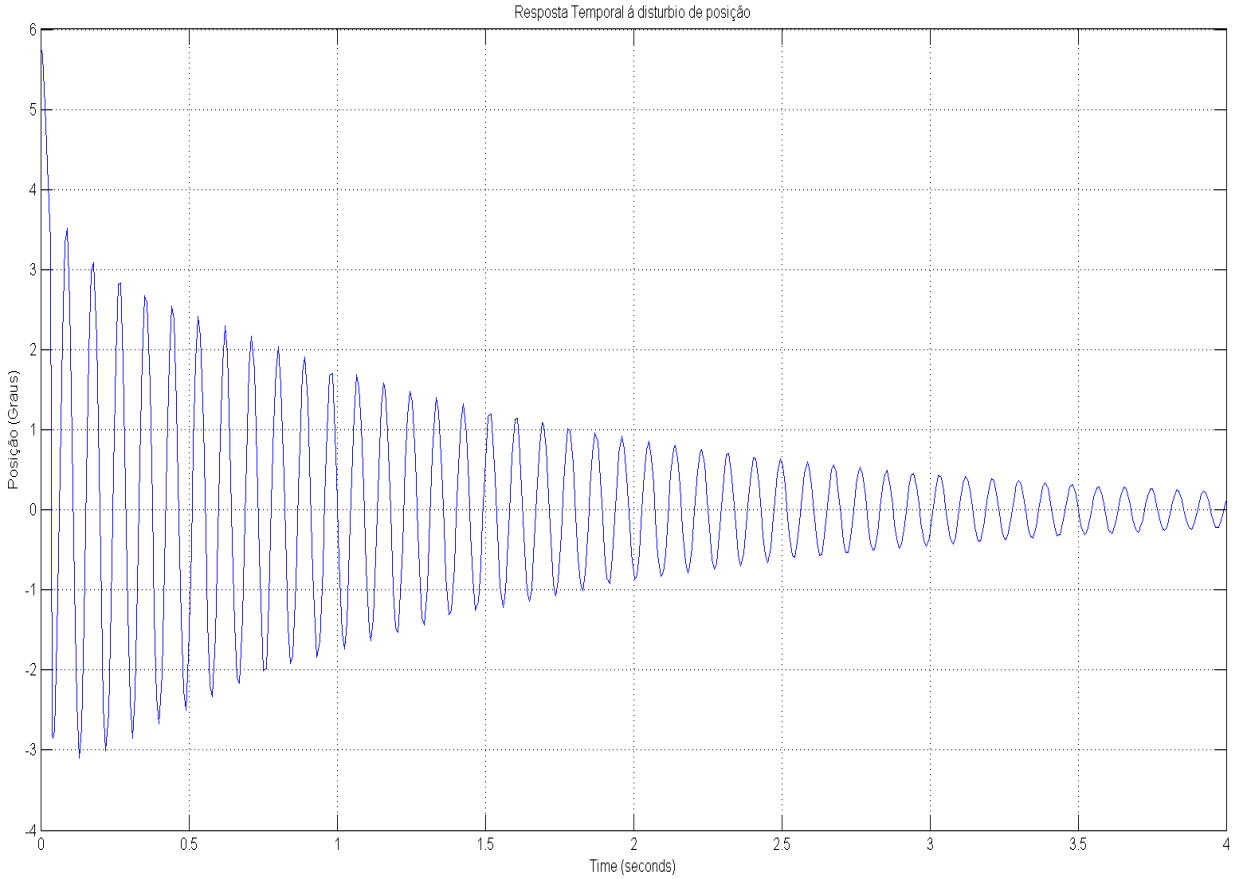

Figura 12: Resposta temporal do sistema C(s)G(s) à distúrbio de posição

O gráfico de resposta mostra um distúrbio de posição de aproximadamente 5.7 graus aplicado no tempo zero, causado por um impulso aplicado ao pêndulo invertido sobre rodas. Nota-se uma tendência à convergência da posição em torno do equilíbrio (Posição em zero graus), porém, de forma lenta, longe dos 150 ms definidos em projeto, e muito oscilatória. Isso implica, na prática, a impossibilidade de controle da planta, pois, no caso de novos distúrbios aplicados com alta frequência, o sistema não consegue se recuperar e a resposta do mesmo tenderá ao desequilíbrio.

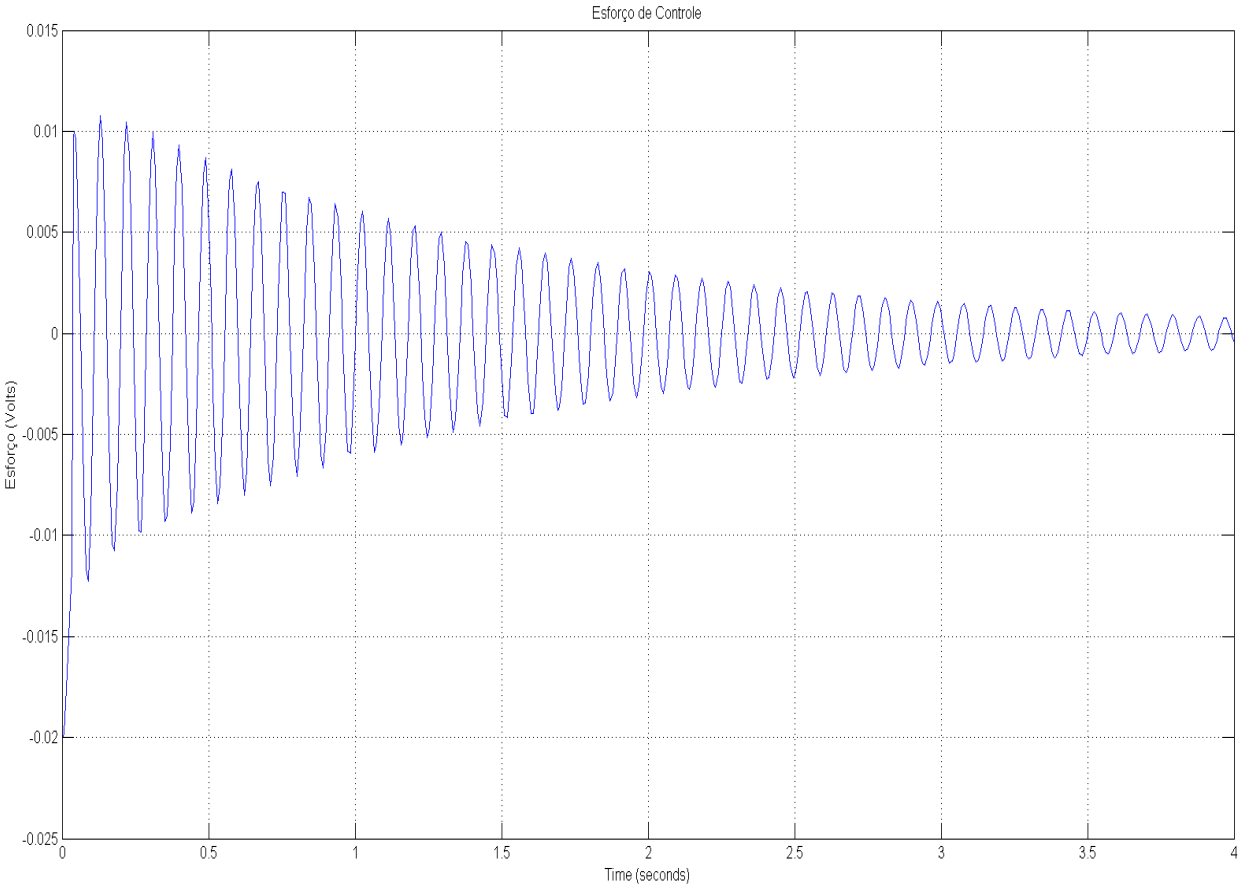

Figura 13: Esforço de Controle do Sistema C(s)*G(s)

O sinal de esforço de controle possui fase oposta ao de resposta, afinal a ação de controle da planta em questão sempre impõe uma rotação contrária à do sistema afim de que o sistema entre em equilíbrio e não tenha sua resposta degenerada, ou seja, vá para uma posição angular tal que o controlador não possua mais a capacidade de equilibrar o pêndulo invertido sobre rodas. A amplitude máxima do sinal de esforço de controle não passa de 20 milivolts, portanto, não é capaz de causar a saturação dos motores CC que suportam tensões de até seis Volts.

O fator mais crítico na ação de controle será a velocidade de resposta dos motores, já que o sinal de controle possui alta frequência de oscilação. Apesar do sistema de pêndulo invertido sobre rodas mais compensador de avanço ser teoricamente estável, a resposta é muito oscilatória e lenta e a margem de fase do sistema de 7,62 graus obtida através do diagrama do Lugar das Raízes é baixa. Portanto, visando garantir o tempo de assentamento da resposta de 150 ms definido para o projeto e a melhoria da estabilidade e robustez do sistema, adotou-se a estratégia de adicionar mais um compensador de avanço em série na malha de controle.

A abordagem seguida foi a adição de outro compensador em série idêntico ao já existente, ambos com equação

$$C(s) = \frac{s + 6.46}{s + 400}$$

A fim de adicionar 90 graus à margem de fase do sistema e deslocar o lugar das raízes ainda mais para a região de estabilidade, semiplano esquerdo do plano S.

Obtiveram-se os seguintes diagramas de Lugar das Raízes e Bode:

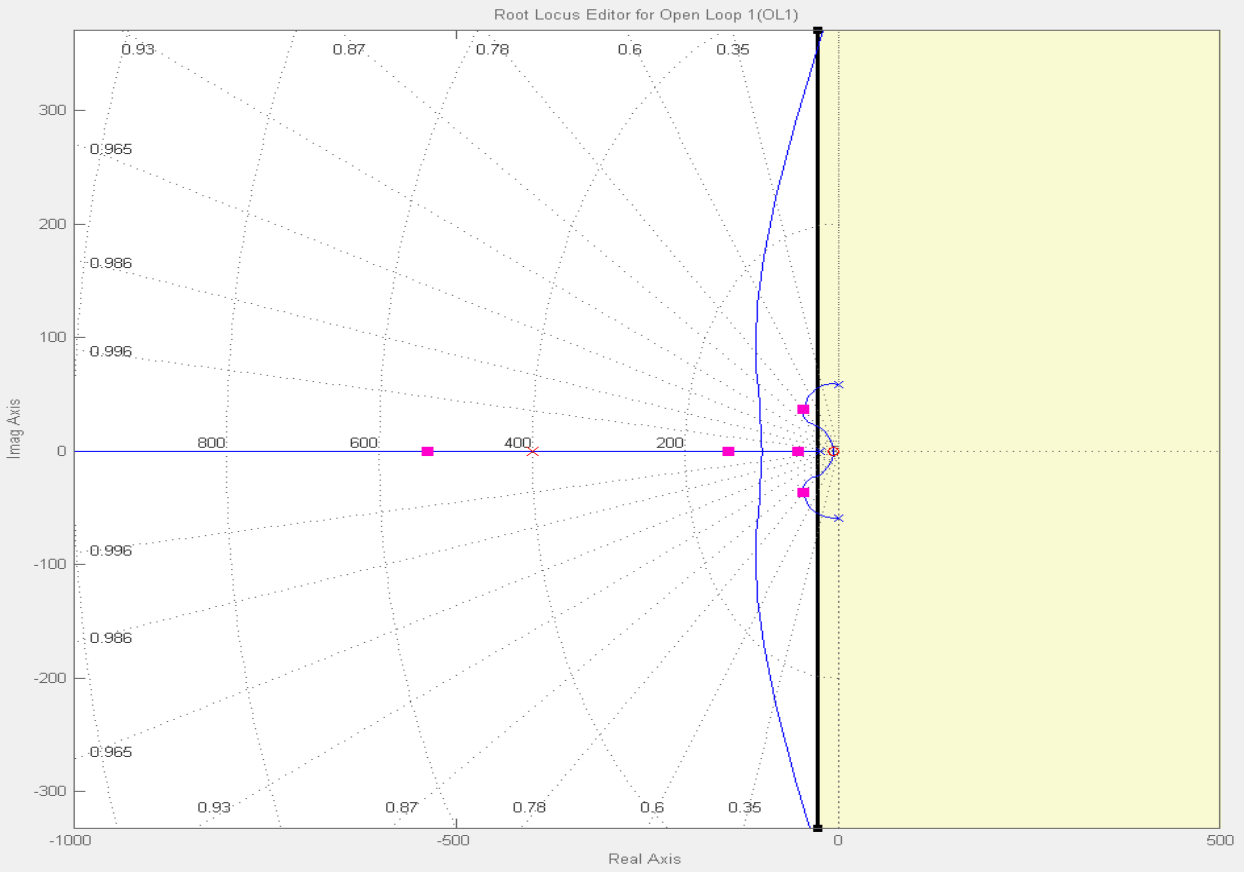

Figura 14: Lugar das Raízes do Sistema C(s)*C2(s)*G(s)

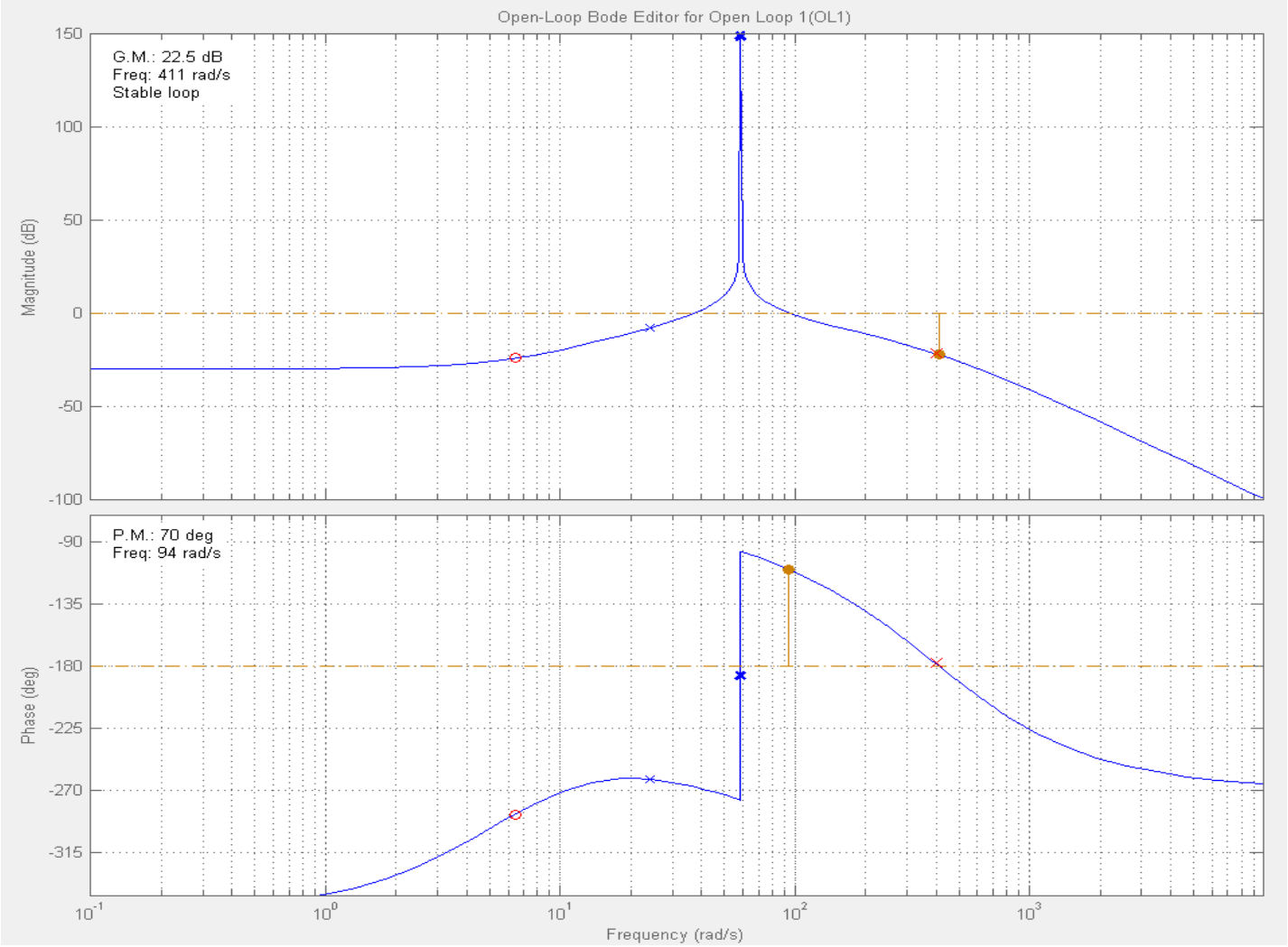

Figura 15: Diagrama de Bode do Sistema C(s)*C2(s)*G(s)

A análise do novo diagrama de Lugar das Raízes mostra que o novo sistema possui uma região de estabilidade ainda maior do sistema anterior com apenas um único compensador. Realizando-se um estudo sobre a influência do ganho dos dois compensadores de avanço nos polos em malha fechada do novo sistema, obteve-se um ganho de compensador $K_c = 2.2$, desta forma os polos dominantes em malha fechada do sistema estão localizados na região branca do LR, que é a região dos polos que garantem tempo de assentamento do sistema em 150 ms.

O diagrama de Bode mostra que as novas margem de fase e ganho são positivas e apresentam valores positivos e que garantem a estabilidade do pêndulo invertido sobre rodas. A nova malha de controle e o novo modelo em *Simulink* obtidos foram:

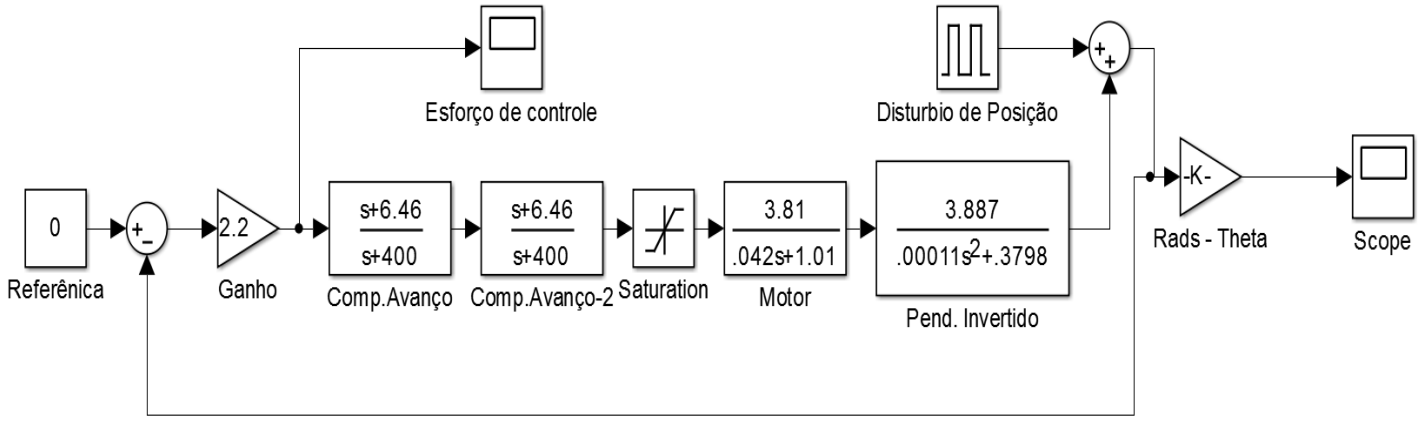

Figura 16: Malha de controle em *Simulink* para simulação do sistema C(s)*C2(s)*G(s)

Este novo modelo somente difere do anterior na adição de mais um bloco de compensador de avanço de fase e na alteração do ganho K_c

Os resultados obtidos na simulação à distúrbios de posição na forma de pulso de amplitude de 5.7 graus foram:

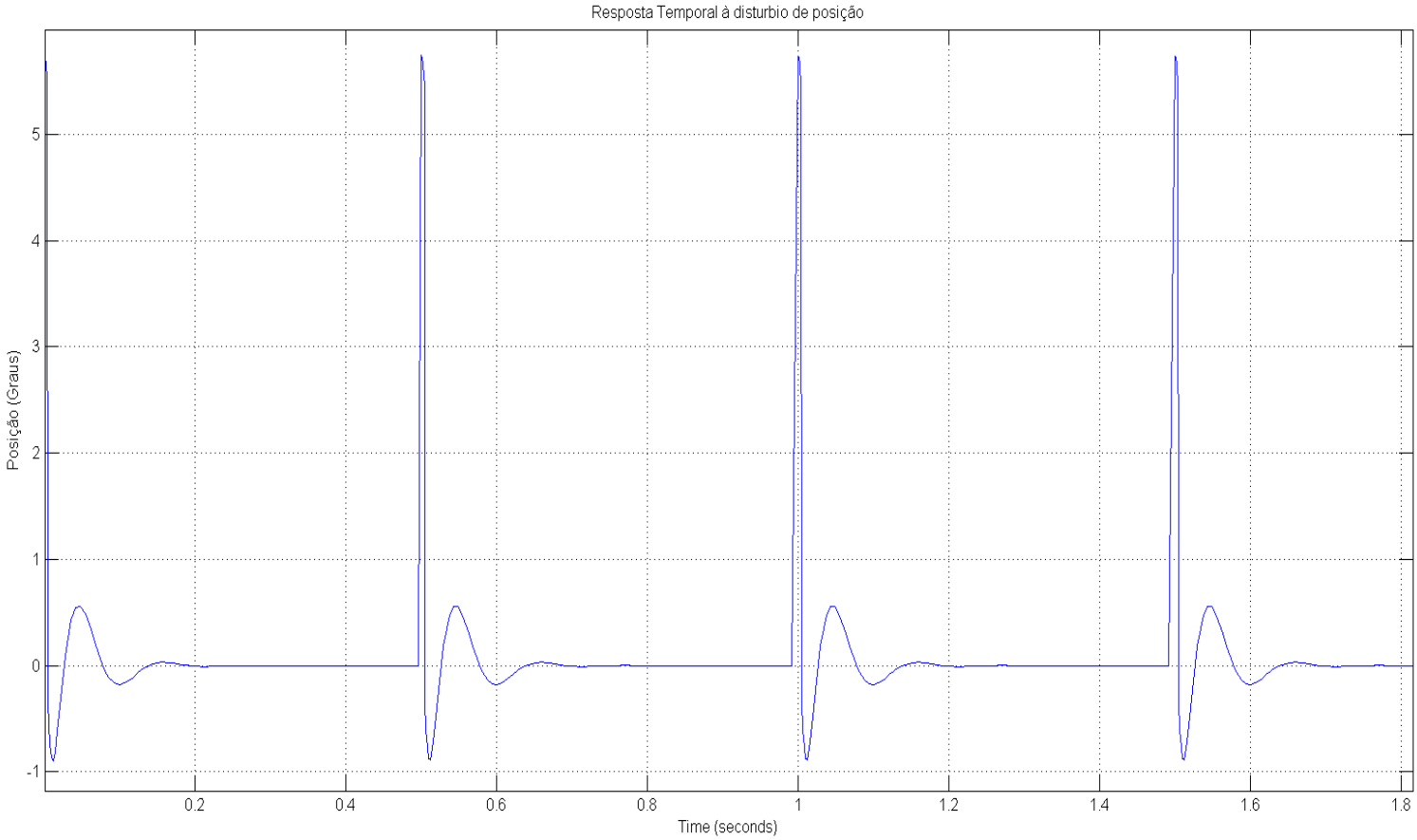

Figura 17: Resposta temporal à distúrbio de posição do sistema C(s)*C2(s)*G(s)

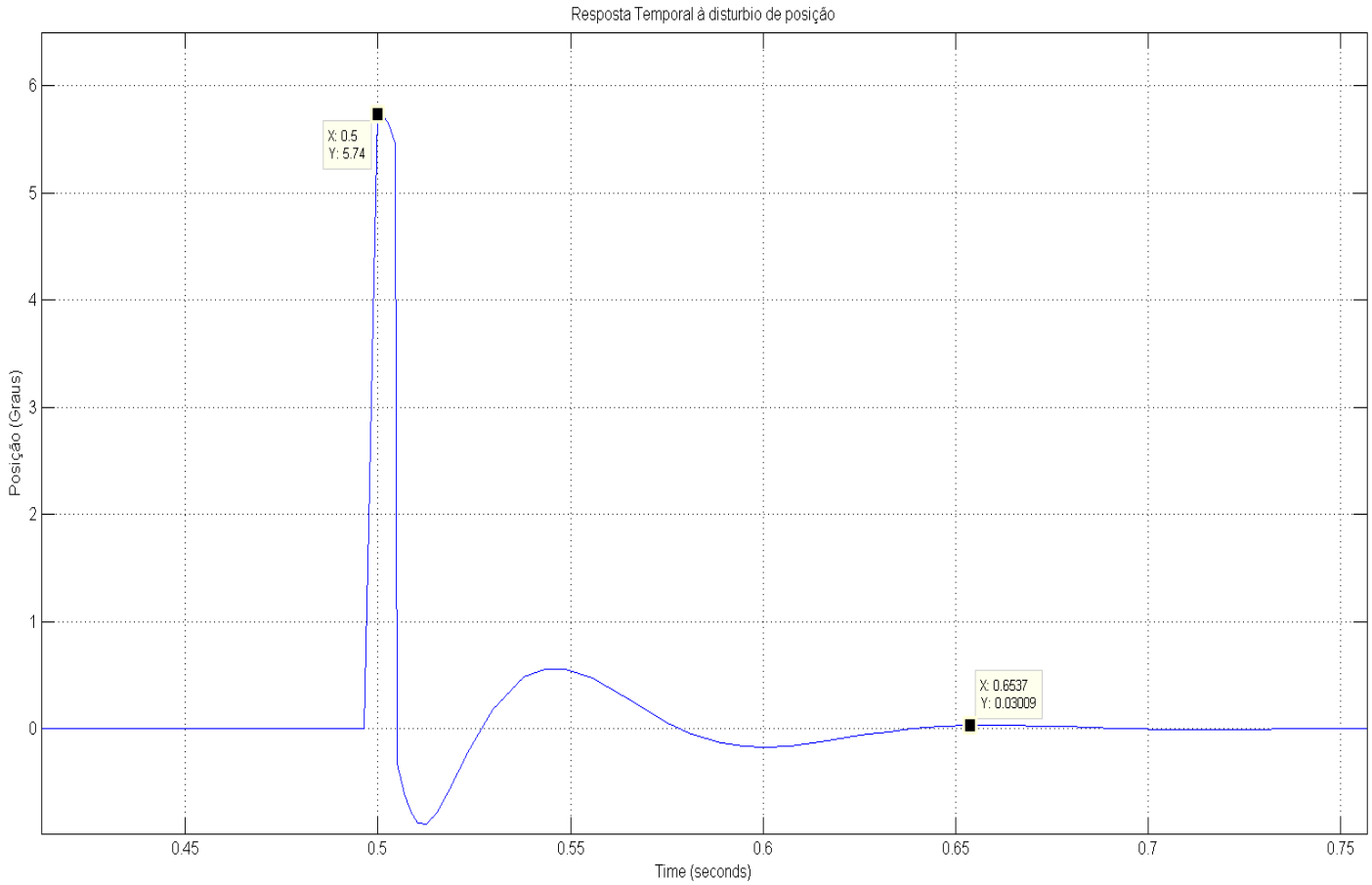

Figura 18: Ampliação do gráfico anterior.

Os gráficos da resposta temporal do sistema com dois compensadores de avanço de fase mostram que a resposta do sistema à aplicação de um distúrbio de posição na forma de um pulso de amplitude de 5.7 graus converge de forma mais rápida que o sistema proposto anteriormente. Ainda, da análise do gráfico de resposta ampliada nota-se que o sistema converge em 150 ms com um erro de 0.03 graus na resposta, um erro de resposta satisfatório. Diferentemente do sistema anterior, esta nova configuração de controlador garante um sistema robusto e com rápida recuperação, permitindo que o pêndulo invertido sobre rodas se mantenha em equilíbrio, mesmo exposto a ação de distúrbios com alta frequência, como nota-se da figura 17, onde foram aplicados distúrbios a cada 0.5 s e o sistema manteve-se equilibrado.

O sinal de esforço de controle obtido é dado pelo seguinte gráfico:

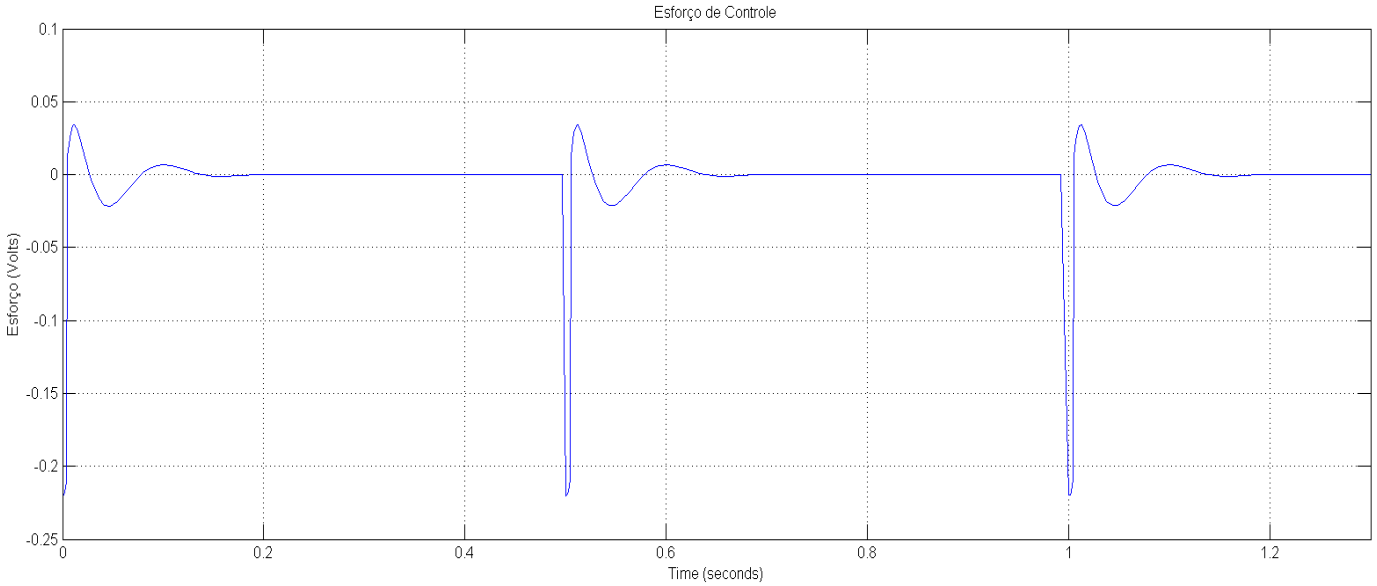

Figura 19: Esforço de controle para o sistema C(s)*C2(s)*G(s)

O sinal de esforço de controle não supera valores maiores que 250 milivolts e, portanto não é capaz de causar a saturação dos motores de corrente contínua utilizados como atuadores. Esta configuração final de controlador exige maior potência elétrica em comparação com a passada, porém garante as especificações de resposta definidas em projeto, além de tornar a planta menos suscetível a distúrbios externos.

2.5.4. Digitalização do Controlador

Definido o controlador C(s) no plano complexo s, espaço de tempo contínuo, há a necessidade de executar a discretização do mesmo e por fim obter uma equação de diferenças, obtendo-se assim uma equação que represente a dinâmica do controlador em função de tempo discreto. Na prática, esta equação de diferenças será implementada no microcontrolador. Estas operações são necessárias para implementação do controlador através de um dispositivo digital, no caso deste trabalho um microcontrolador ATMEGA-328P-AU do tipo AVR que funciona inerentemente em tempo discreto.

A aproximação digital de um controlador em tempo contínuo pode ser feita obtendo-se a solução da equação diferencial do controlador utilizando-se métodos numéricos de integração de EDO's. Porém, uma estratégia muito aplicada e bastante eficaz na área de engenharia de controle é a aproximação da função de transferência do controlador C(s) para uma função em tempo discreto C(z) e depois a obtenção da equação de diferenças do mesma.

Para isso existem inúmeros métodos de aproximar o tempo contínuo em discreto, neste projeto será utilizado o Método de aproximação de *Tustin* ou Transformação Bilinear que já é bastante consolidado e comprovadamente eficaz e propõem a seguinte relação:

$$H(z) = H(s)\big|_{s=\frac{2(Z-1)}{T_a Z+1}},$$

Onde T_a é o valor do tempo de amostragem.

A definição do período de amostragem é de fundamental importância para o controlador, se este tempo for muito grande a resposta do sistema se torna lenta e ineficaz, pois o controlador perderá muita informação sobre o estado da planta entre os duas amostragens consecutivas.

Para definir um Ta eficaz e coerente foi feita a análise dos gráficos de resposta temporal do sistema obtidos anteriormente e levado em conta o fato de que o controlador foi projetado em tempo contínuo e negligenciou atrasos inerentes aos processos de discretização e quantização dos sinais envolvidos e processamento do dados.

Os gráficos de resposta temporal do sistema mostram um tempo de estabilização de aproximadamente 150 ms, então foi definida um tempo de amostragem quinze vezes menor Ta=10 ms, para início de testes e experimentos.

Baseado nos conceitos e dados acima e sendo a FT do controlador obtido previamente

$$G(s) = \frac{s^2 + 12.92s + 41.73}{s^2 + 800s + 160000}$$

obtém-se então:

$$G(z) = \frac{0.1184z^2 - 0.222z + 0.104}{z^2 + 0.6667z + 0.1111}$$

Possuindo G(z) obteve-se então a seguinte equação de diferenças para implementação digital:

$$u[k] = -0.6667\,u[k-1] - 0.1111\,u[k-2] + 0.1184\,e[k] - 0.222\,e[k-1] + 0.104\,e[k-2]$$

onde o vetor u[] representa a saída da planta de controle e o vetor e[] representa o sinal de erro que é o valor do sinal referência menos o da saída da planta.

Com o auxílio do software MATLAB, o sistema de controle digital foi simulado obtendo o seguinte gráfico de resposta temporal:

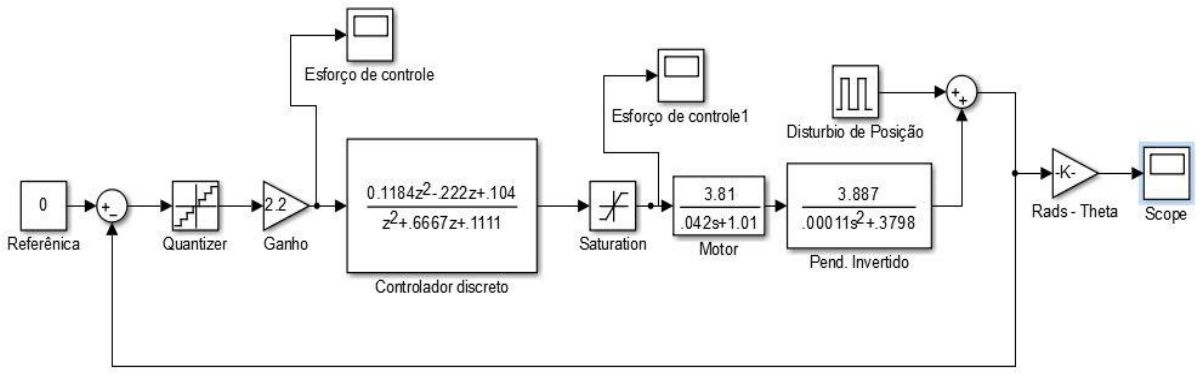

Figura 20: Malha de controle para Controlador discreto C(z)

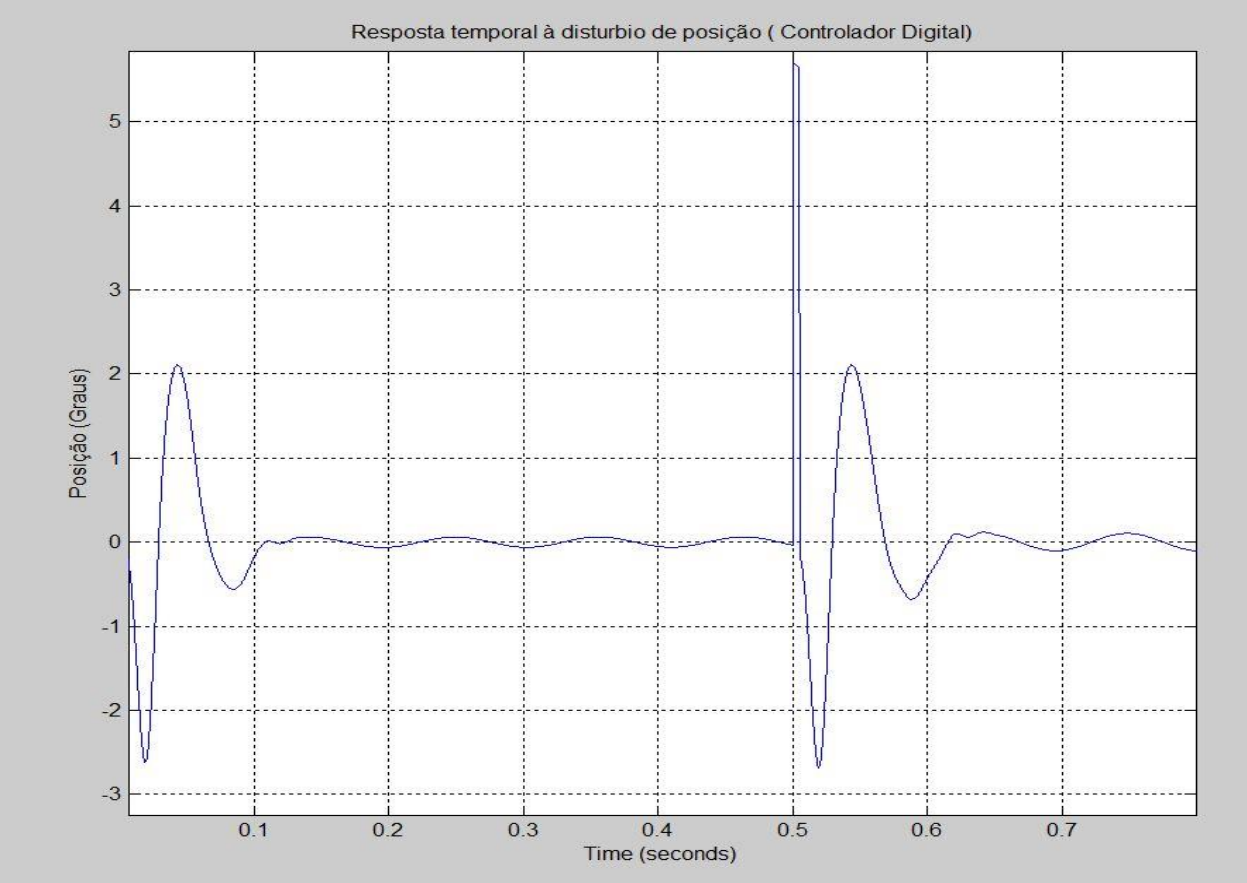

Figura 21 - Resposta temporal do sistema com Controlador digital

O gráfico acima mostra que o controlador digital projetado com o período de amostragem T_a de 10 ms é capaz de estabilizar o sistema do pêndulo invertido sobre rodas em aproximadamente 100 ms, validando a equação de diferenças obtida e as hipóteses adotadas durante todo o processo de projeto do controlador.

Com o projeto de controle finalizado e validado por testes computacionais, passa-se à etapa de implementação de software, hardware e posteriores testes com o modelo real.

2.6. Aquisição de dados dos sensores

2.6.1. Cálculo de ângulo utilizando Acelerômetros e Giroscópios

Os sensores de aceleração ou acelerômetros são transdutores capazes de medir acelerações em eixos coordenados, este sensor “enxerga” sempre as direções das forças aplicadas nele. Assim, utilizando a própria aceleração gravitacional da terra decomposta nos eixos coordenados de um acelerômetro, podemos obter através destas direções valores de posição angular do nosso sistema, fazendo uso de alguma trigonometria.

Como pode ser notado na figura seguinte, a força resultante sentida pelo acelerômetro é decomposta em três planos XY, XZ e YZ, sendo a direção Z sempre alinhada com a gravidade.

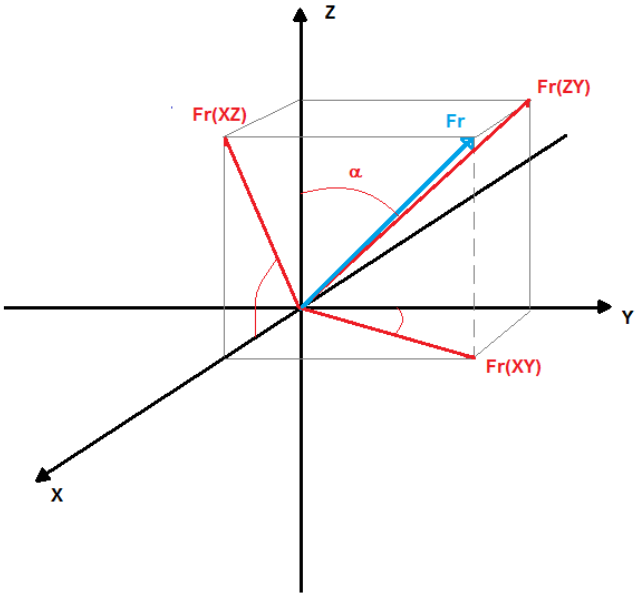

Figura 22 - Força arbitrária aplicada sobre um acelerômetro de três eixos e suas componentes nos planos coordenados

Fazendo-se uso desta técnica de decomposição trigonométrica, pode-se inferir a orientação de um acelerômetro de três eixos no espaço. No entanto, para o escopo deste trabalho, devemos calcular somente o ângulo de orientação da planta com a vertical, facilitando em muito o tratamento matemático.
O acelerômetro nos fornece as acelerações nas direções Z e Y, logo, o ângulo α que representa o ângulo de desalinhamento entre a posição de equilíbrio do pêndulo invertido e o próprio pêndulo é obtido por:

$$\tan^{-1}\left(\frac{A_Y}{A_Z}\right) .$$

De outra maneira, a posição angular do protótipo pode ser obtida através da integração leitura de velocidade angular oriunda de um giroscópio. Novamente, neste trabalho só teremos rotação em torno da direção X (ortogonal ao plano ZY), que quando integrada fornece o ângulo α imediatamente.

2.6.2. Problemática de Sensoriamento

Já mencionado neste trabalho, um dos principais problemas deste presente experimento tange a aquisição de dados precisos e coerentes do estado do pêndulo invertido sobre rodas, ou seja, de sua orientação no espaço.

Devido à sua construção física, na prática, um giroscópio em repouso, exatamente alinhado com o horizonte do planeta não registra valor de saída nulo, ou seja, ele possui um offset que é conhecido como erro de BIAS. Enquanto o BIAS pode ser sanado pela introdução de um fator corretivo em *software,* existe outro problema associado ao giroscópio que é conhecido como *drift* e é relacionado com a integração do sinal de saída do sensor, este é gerado quando se integra temporalmente grandezas sujeitas a ruídos aleatórios e erros de truncamento. O *drift* se caracteriza pela soma destes pequenos erros e ruídos de forma sucessiva, até que o valor gerado pela integração numérica perde completamente o sentido, fronte ao decaimento rápido

da proporção sinal-ruído. Como desejamos mensurar ângulo com a vertical em tempo real e não velocidade angular, o *drift* se torna inerente à nossa planta, e deve ser sanado de alguma forma.

Fenômenos inerentes à operação de acelerômetros modernos envolvem vibrações e sinais de alta frequência que adicionam ruído indesejado. A própria translação do acelerômetro conforme este oscila juntamente ao chassi do protótipo também prejudica a decomposição trigonométrica utilizada para calcular o ângulo instantâneo deste com a vertical.

No caso do projeto do pêndulo invertido sobre rodas, haverá momentos onde o sistema será submetido a movimentos de alta frequência (quando oscila em torno do equilíbrio, por exemplo) e somente o acelerômetro não será capaz de garantir dados precisos sobre a orientação do pêndulo no espaço.

Analisando os problemas de cada tipo de sensor acima, nota-se claramente que em certos momentos, se qualquer um dos dois for utilizado sem o auxílio do outro no projeto, não será possível a manutenção do sensoriamento robusto de ângulo conforme o veículo opera, portanto, o sistema de controle não será capaz de manter o equilíbrio do pêndulo invertido sobre rodas.

Uma estratégia muito eficaz e largamente aplicada na engenharia para evitar estes problemas é a técnica de *sensor fusion* que utilizará o sinal de ambos os sensores, giroscópio e acelerômetro, para combinar as medições de ambos e estimar o estado do protótipo de maneira precisa e eficaz.

Existem diversas técnicas e filtros de sinais para tal tarefa como, por exemplo, o filtro de Kalman, complementar e de Mahony (OLLIW, 2015).

Dadas as diferentes opções de filtragem, optou-se pela implementação em software de um filtro complementar, levando em conta seu algoritmo relativamente simples e de baixo custo computacional (OLLIW, 2015).

O filtro complementar nada mais é do que a fusão de um filtro passa alta e de um filtro passa baixa, atuando nas medidas de aceleração e de velocidade angular, respectivamente. A função de transferência de um filtro complementar típico segue:

$$\theta(s) = \frac{1}{1+Ts}a(s) + \frac{Ts}{1+Ts}\frac{1}{s}\omega(s)\ ,$$

O primeiro termo representa a função transferência do filtro passa baixa filtrando o sinal obtido do acelerômetro, enquanto o segundo termo representa a função transferência de um filtro passa alta filtrando o sinal de posição angular obtido do giroscópio.
A constante de tempo dos filtros se relaciona com a frequência de corte de ambos os filtros. Esta constante está intrinsicamente relacionada à operação do filtro em tempo real, e modifica seu comportamento na implementação digital do mesmo. Nesta, utiliza-se valores numéricos no lugar das parcelas filtrantes, e alguma manipulação matemática nos revela a seguinte relação:

$$\frac{1}{1+Ts} = p \quad , \quad \frac{Ts}{1+Ts} = (1-p)$$

Ou seja, o fator p delimita diretamente a zona de corte do filtro complementar, um valor mais baixo de p favorece o cálculo de ângulo a partir da leitura do giroscópio integrado no tempo, enquanto um valor mais alto favorece o cálculo trigonométrico empregado sobre as leituras do acelerômetro.

A resposta em frequência do Filtro Complementar é mostrada abaixo:

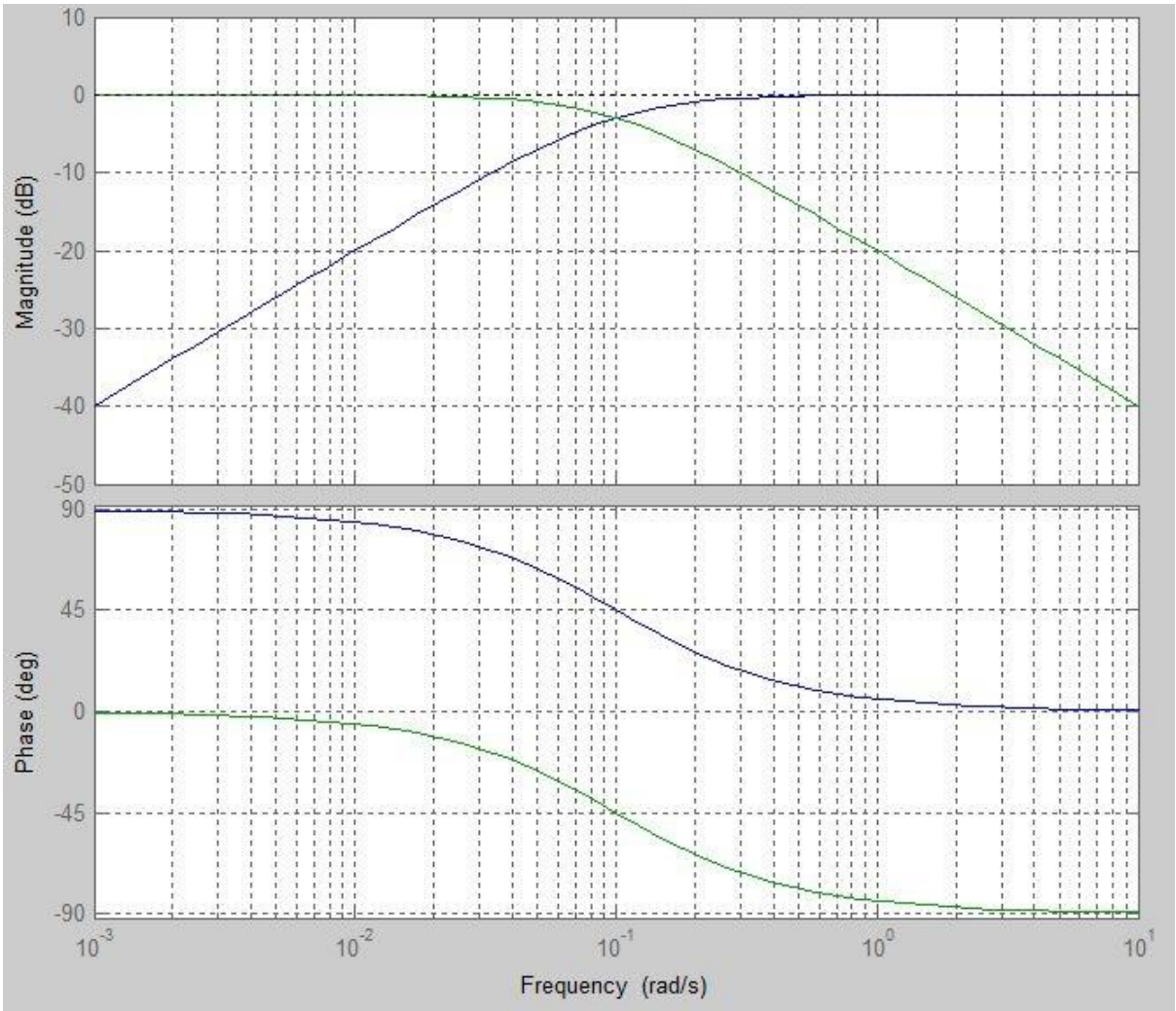

Figura 23 - Resposta em frequência de um filtro complementar típico

Realizando o processo de discretização dessa planta, utilizando o método de aproximação de derivação para trás, tem-se:

$$\theta(z) = \theta(s)|_{s=\frac{Z-1}{Z\,T_a}}$$

$$\theta(z) = \frac{T_a}{T_a + T(1 - z^{-1})}a(z) + \frac{T\,T_a}{T_a + T(1 - z^{-1})}\omega(z)$$

Por fim, a equação de diferenças resultante a ser implementada no microcontrolador é:

$\theta[k] = p\theta[k-1] + p\omega[k]T_a + (1-p)a[k]$, onde $p = \frac{T}{T+T_a}$.

A constante p então envolve o período de amostragem e a frequência de corte dos filtros passa alta e passa baixa. A constante p pode ser escolhida conforme a frequência natural da planta sob a qual o filtro atua e dimensionada de forma a fazer com que este gere

leituras de ângulo com rejeição de ruído otimizada, no entanto, neste trabalho o valor exato desta constante não é calculado previamente, pois é posteriormente calibrado experimentalmente na própria planta em função do comportamento do filtro.

2.7. Hardware e Sistemas Embarcados

A fim de realizar fisicamente o sistema de controle projetado e atuar na planta é realizado um projeto de hardware eletrônico. A plataforma superior presente no chassi do veículo foi idealizada de forma a abrigar uma PCI[1] capaz de suportar os subsistemas necessários, sendo críticos ao controle o microcontrolador central, drivers de acionamento dos motores, e o sensor inercial baseado em MEMs.

No hardware projetado para o protótipo encontram-se não só os componentes necessários para realizar o controle da planta, bem como um display de caracteres de cristal líquido, LEDs de indicação para possível *debugging* de software, reguladores de tensão para alimentação dos circuitos lógicos e de potência dos motores, e finalmente, um circuito de detecção que indica ao operador quando a bateria atinge níveis de tensão inferiores à sua faixa de tensão operacional nominal, a fim de evitar dano à mesma.

A interface com o usuário se manifesta de forma simples como um *encoder* óptico, e um botão presente no próprio *dial* deste componente, conjunto que possibilita a navegação em um sistema de menus. Este sistema é discutido na seção relacionada ao projeto de software. Tais funcionalidades, claramente não relacionadas à operação do sistema de controle em si, foram implementadas para abrir a possibilidade de desenvolvimento de uma interface com o usuário que torne possível a modificação de parâmetros de controle no próprio protótipo. A operação desta maneira confere à planta possível caráter didático, tornando-a objeto de demonstração da teoria de controle por posicionamento de polos no domínio da frequência em um sistema real.

A seguir divide-se o texto em subseções que descrevem o hardware eletrônico em detalhes. Esquemas elétricos de conexão e layout e fotos ilustrativas encontram-se no primeiro anexo.

2.7.1. Microcontrolador central e Sensor Inercial

Todo o processamento de dados oriundo dos sensores e cálculo de saída para os motores, e interface com o usuário serão geridos por um microcontrolador ATMEGA-328P-AU do tipo AVR. O microcontrolador está instalado numa placa de *breakout* compatível com a arquitetura ARDUINO.

O sensor inercial escolhido foi o popular MPU6050, que se manifesta em um pacote SMD diminuto e já incorpora acelerômetros e giroscópios. O pacote também se encontra em uma placa de *breakout* que facilita acesso aos pinos de I/O do mesmo. A interface entre o sensor e o microcontrolador se dá por meio de uma interface de dois fios (TWI – *two wire interface)* na qual circulam dados em conformidade com o protocolo I2C de comunicação serial. Um terceiro fio ainda conecta o sensor ao microcontrolador, pois aquele é capaz de gerar interrupções externas para este, possibilitando algumas funcionalidades de aquisição de dados que de outra forma não estariam disponíveis. Mais a respeito na seção relacionada ao software.

[1] A placa de circuito impresso foi fabricada em ambiente doméstico.

A placa ainda possui um *bus* I2C secundário que torna possível a integração com um magnetômetro, tornando possível a implementação de um sistema AHRS completo (Invensense). Apesar de ser capaz de medir acelerações e velocidades angulares em todas as direções, no nosso sistema utilizamos apenas leituras no eixo de oscilação vertical da planta; a escolha do MPU6050 como transdutor inercial se justifica por seu elevado grau de integrabilidade com o microcontrolador por vias de bibliotecas de software já existentes; tal fato evidencia a busca por uma maior facilidade no posterior desenvolvimento do mesmo.

2.7.2. Interface com *encoder* óptico e LCD de caracteres

O hardware conta ainda com elementos de navegação e interface com o usuário, tornando possível a execução de uma pequena máquina de estados a ser codificada em software. A navegação é realizada por um *encoder* óptico de dois bits em código Gray. Os bits são lidos pelo microcontrolador numa simples interface paralela.

O elemento central desta simples, porém eficaz interface, é um LCD de 16x2 caracteres, presente na parte central da placa. O LCD torna possível a criação de navegação em menus baseado em máquina de estados, tornando possível o ajuste em software de parâmetros de controle e de sensoriamento, e sua interface com o Microcontrolador é empregada por via de seis vias, sendo quatro delas uma interface de dados paralela de 4 bits, e as outras duas sinais de controle de fluxo de dados.

2.7.3. Drivers de acionamento dos motores

Os drivers consistem duas pequenas PCI’s ocupadas pelo versátil MC33926 e seus periféricos: um pacote diminuto PQFN-32 o manifesta. Este é capaz de realizar acionamentos usando diversas configurações de I/O, porém neste trabalho objetivou-se reduzir o número de pinos tomados pela interface. Pode-se fazer uso de apenas um bit comum a ambos os drivers visto que a planta descreverá movimento aproximadamente retilíneo, assim restando apenas duas saídas moduladas em ciclo de carga – *PWM,* segundo à fabricante dos componentes eletrônico *Sparkfun.*
O PWM é a chave para modular a potência elétrica entregue aos enrolamentos do motor, de forma a controlar efetivamente a planta.

2.7.4. Circuito de alimentação e monitoramento de bateria e LEDs de aviso

Todo o hardware eletrônico é alimentado via um borne de duas vias que recebe as conexões da bateria que repousa na plataforma inferior do veículo. Uma chave geral chaveia a entrada de energia, ligando ou desligando o sistema como um todo – O LED de 5mm verde indica alimentação.

O sistema é capaz de operar com baterias operantes acima de 8.5V devido à presença de reguladores de tensão no circuito. Estes são três no total, todos lineares, sendo dois LM7806 para os motores (cada driver é alimentado por um regulador independente, a fim de se maximizar a corrente disponível) e um LM7805 para gerar a tensão lógica de nível TTL utilizada na maioria do circuito. Um quarto regulador, presente na placa de *breakout* do microcontrolador, gera tensão de 3.3V para alimentação exclusiva do sensor inercial, já que este não opera em níveis TTL.

Vale ressaltar que técnicas de *level shifting* foram consideradas para adequar os níveis de tensão no bus I2C, porém tal hardware foi descartado por ter sido observada perfeita operação daquele na ausência do deste.

Limites superiores de alimentação são controlados pela tensão máxima de alimentação dos reguladores mencionados e por capacidade de dissipação térmica dos mesmos na configuração em que se encontram. O hardware foi ensaiado com baterias de chumbo-ácido e Lítio-Polímero entre a faixa de 10.5V a 13.8V e constatou-se perfeita operação. O protótipo final será ensaiado com ambas as baterias a fim de se observar mudanças inerciais na planta e desempenho do controle quando submetido às mesmas.

Para que seja evitado dano à bateria foi usado um comparador com histerese (LM311) que tem em sua saída um LED amarelo de aviso; suas entradas estão configuradas de forma a acender o LED quando o nível de tensão de alimentação cai abaixo de 11.1V [2].

Os três LEDs restantes estão conectados a pinos de I/O do ATMEGA. O primeiro, vermelho, indica o bit de direção que inverte o sentido de giro dos motores. Os dois restantes são somente para uso geral e serão possivelmente associados a rotin as relevantes de software, para tornar mais explícita a operação do mesmo conforme o protótipo se equilibra; um azul e outro verde, de 3mm.

2.8. Software

Todo o software foi desenvolvido usando-se linguagem C clássica. Foi feito uso extensivo de bibliotecas operacionais para os diversos componentes de hardware afim de expeditar as tarefas de programação.

Existem duas etapas nas quais o software opera, sendo estes o *setup* e o *loop,* que diferem somente no fato de a primeira rodar uma única vez quando o sistema é energizado, e a segunda rodar em loop indefinidamente. Na rotina de *setup* inicializa-se os elementos de hardware no microcontrolador, como por exemplo, o LCD e a unidade de medição inercial. Na rotina de *loop* temos duas rotinas que rodam independentemente da configuração do sistema, sendo estas a rotina de atualização do *encoder* e a de atualização do LCD, e ainda, uma máquina de estados responsável pela execução de todas as tarefas do controlador. Esta é descrita a seguir.

2.8.1. Máquina de estados e dinâmica de interação com usuário

Como descrito na seção relacionada ao hardware, foi projetado um pequeno sistema de interface com o usuário. Aqui descrevemos brevemente a operação da máquina de estados em software que tornará possível o ajuste de parâmetros de controle na própria planta, sem que seja necessária a reprogramação do microcontrolador.

A interface foi programada na língua inglesa, pois nesta as palavras são em geral mais curtas e o espaço disponível no display de caracteres pode ser mais bem aproveitado.

A máquina começa no estado #0, e percorre a árvore conforme o usuário aciona o *encoder* no sentido horário (E+), anti-horário (E-) ou pressiona do botão (B). Nos próprios estados apresenta-se um croqui de como será o texto no LCD; Um diagrama detalhado das transições entre os menus se encontra no anexo um.

[2] Baterias de Lítio Polímero de três células operam na faixa de 4.2V à 3.6V por célula – são três em série - sendo que abaixo desta faixa é aumentada consideravelmente a possibilidade de dano às mesmas.

No laço 0-1-2 encontra-se o menu inicial, sendo que os laços 3-4-5-6 e 10-11-12 evidenciam os submenus de ajuste de parâmetros e de seleção de modo de aquisição sensorial.

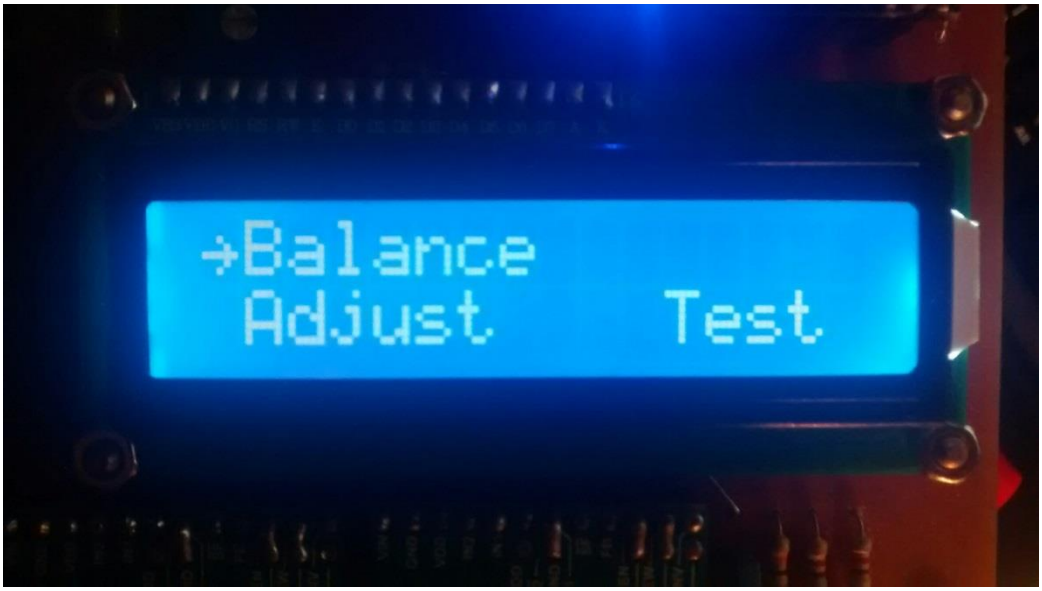

Figura 24 - Retrato do LCD de caracteres para máquina de estados no menu principal (estado #2). A flecha circula entre as opções conforme o usuário aciona o encoder óptico de navegação.

O menu principal possibilita ao usuário escolher entre três opções, ajuste de parâmetros, seleção de modos de aquisição sensorial e modo de teste. No último, pode-se observar o acionamento dos motores proporcionalmente ao valor de um contador interno que é incrementado usando-se o *encoder* de navegação. No submenu de ajuste, pode-se selecionar entre os ganhos a serem regulados e ajustar seus valores, enquanto que no submenu de seleção de modo de aquisição sensorial pode-se escolher se o veículo irá se equilibrar usando dados de ângulo calculados diretamente em hardware dentro da MPU6050, ou se o fará usando dados *raw* do acelerômetro e do giroscópio.

Vale notar do esquemático da máquina de estados, que os estados 7,8,9,13,14 e 15 são estados de ponta, onde o giro do encoder incrementa ou decrementa variáveis internas ao software, ao invés de navegar entre as diferentes opções de um menu.

2.8.2. Geração de ângulo em softare – O filtro complementar

O filtro complementar descrito na problemática de sensoriamento foi implementado de forma mais simples possível. A rotina responsável pelo cálculo dos ângulos para cada set de leituras *raw* dos sensores é listada em código como RUN_FILTER, e pode ser encontrada no Anexo 2. Programada para ser chamada somente quando a máquina de estados se encontra no estado #13, ela é executada numa frequencia de 100Hz, taxa muito superior à frequencia natural da planta, para satisfazer os requisitos de amostragem de sinais analógicos e decorrente processamento digital estabelecidos pelo Teorema de Nyquist. Nela ocorrem os seguintes eventos:

- Aquisição de leituras dos eixos Y e Z do acelerômetro e do eixo X do giroscópio.
- Atualização do parâmetro de corte p escolhido pelo usuário
- Tratamento dos valores *raw* para grandezas físicas reais

- Cálculo de ângulo via integração das leituras do giroscópio e correção do valor para radianos
- Cálculo trigonométrico do ângulo instantâneo
- Ponderação do Ângulo via filtragem complementar
- Atualização de um buffer FiFo de três posições que guarda a leitura atual, a passada e a retrasada

O parâmetro de corte p pode ser atualizado pelo usuário pela simples navegação até o estado#8, com a modificação do parâmetro G2. Em etapadas de *debugging* o filtro foi ensaiado usando-se uma interface serial com o microcontrolador central com valores de p variando na faixa de 0.5 até 0.99. Constata-se que para valores de p inferiores à 0.7 a parcela de cálculo de ângulo relacionada às medidas do acelerômetro introduz ruídos significativos no processo de filtragem, em decorrência de fatores já mencionados. O valor final de $p = 0.97$ foi escolhido.

2.8.3. Leitura direta de ângulo via DMP (Digital Motion Processing)

O sensor MPU6050 é notável pois possui integrado ao seu diminuto pacote hardware capaz de realizar em tempo real os cálculos necessários para gerar medidas angulares. Esta funcionalidade foi desenvolvida com o intuito de facilitar o trabalho das unidades de processamento central em sistemas que necessitam de informação inercial, como por exemplo celulares ou tablets, onde o processamento necessário na geração gráfica é intenso, e qualquer diminuição de *overhead* gerada por sensores ou hardware periférico é muito bem-vinda.

Para se obter leitura direta de ângulo da MPU6050 deve-se usar um bit de interrupção externa do ATMEGA328, visto que a biblioteca usada no projeto (Rowberg, 2015) para interfacear com o sensor foi realizada desta forma. Esta interrupção é gerada pela MPU a cada iteração de cálculo de ângulo, e o microcontrolador central interrompe suas atividades onde quer que elas estejam para fazer a leitura do ângulo. Imediatamente, nota-se um possível problema: o *overflow.* Este é causado quando o sensor atualiza leituras de ângulo mais rapidamente do que o controlador central consegue aquisitá-las, fazendo com que este entre em um estado conhecido como *lockup* ou travamento. O travamento é caracterizado em um sistema quando unidades de processamento são interrompidas externamente com uma frequencia maior do que a qual tornaria possível a execução não só das rotinas de comunicação com sensores mas também de todo o restante do código, fazendo com que o microcontrolador ou microprocessador não consiga realizar suas tarefas em tempo cabível.

A MPU6050 conta com um *buffer* FiFo de cinquenta posições, que armazena valores de ângulo conforme estes são calculados, assim, em um eventual travamento, não se perdem leituras de ângulo.

Entretanto, vale ressaltar que este *buffer*, quando totalmente preenchido, faz com que a MPU6050 gere um sinal de dados para o microcontrolador central comunicando o ocorrido, e durante este aviso o controle propriamente dito fica comprometido, em vista da indisponibilidade momentânea de dados angulares. Assim sendo, o travamento e o *overflow* deste buffer devem ser evitados a todo custo. Experimentos preliminares com o protótipo indicaram que nossa MPU estava gerando leituras de ângulo com frequencia superior ao desejado, e de fato obtivemos *overflow* no buffer interno da mesma. O problema foi sanado ao modificar o valor presente no registrador #26 interno à mesma (Invensense, 2013) para escolher taxas de atualização menores, diminuindo assim a frequencia da interrupção externa gerada pelo sensor e aliviando o ATMEGA328 para que este pudesse focar nas atividades de controle em tempo real.

A mudança foi implementada diretamente no arquivo de *header* da biblioteca utilizada. Onde o valor do registrador foi alterado para 0x05, diminuindo assim a taxa de atualização da MPU para 50Hz, a mesma frequencia utilizada para a taxa de atualização do controlador, descrito na seção seguinte. No software foi criado um pequenho buffer FiFo de três posições para armazenamento de leituras, de forma análoga à implementada usando-se o filtro complementar. A rotina de software responsável pela aquisição direta de ângulo usando o *Digital Motion Processing* está listada no anexo 2 como RUN_DMP.

2.8.4. Implementação do controlador

Usando a equação de diferenças calculada anteriormente, foi implementado compensador duplo por avanço de fase em código seguindo a equação previamente encontrada:

$$u[k] = -0.6667\,u[k-1] - 0.1111\,u[k-2] + 0.1184\,e[k] - 0.222\,e[k-1] + 0.104\,e[k-2]$$

Onde o valor $u[k]$ é o esforço atual de controle, seus equivalentes $u[k-1]$ e $u[k-2]$ os esforços passado e retrasado respectivamente, e os valores $e[k]$, $e[k-1]$ e $e[k-2]$ as leituras atual passada e retrasada de ângulo com a vertical. Vale ressaltar que temos um sistema regulador, pois a referência do mesmo é constante e sempre igual a zero (ângulo com a vertical deve ser nulo para configurar equilíbrio).

Para efeito de calibração, o software possibilita, via *encoder*, somar um valor de *offset* às leituras de ângulo. Tal valor tem o intuito de compensar o *setpoint* de ângulo zero na planta. Esta funcionalidade é interessante pois na fabricação do protótipo não foram estritamente observados ângulos presentes na montagem do sensor na placa de circuito impresso nem mesmo na montagem desta com o chassis. Assim, pode-se compensar estes ângulos para que o sistema possa referenciar a vertical virtualmente em qualquer posição, de acordo com o usuário. Isto ainda possibilita a compensação do sistema a posteriores mudanças na configuração da planta mecânica, caso a posição do centro de massa da mesma seja modificado para efeitos de teste.

No início da operação de controle, verifica-se que ambos DMP como o algoritmo do filtro complementar levam algum tempo para convergir para valores representativos, e é durante esta convergência que se ajusta o *offset* na leitura de ângulo manualmente para que o robô encontre ângulo zero com a vertical na posição em que seu CG está exatamente acima do eixo de giro dos motores. Todo este procedimento garante que o controle atuará de fato tentando regular uma posição de equilíbrio do sistema.

Após ensaios preliminares, observou-se que o modelo da planta utilizado no cálculo da equação de diferenças do controle possui diversas simplificações que acabaram por ignorar fenômenos físicos presentes no nosso protótipo. O primeiro deles, e talvez o mais preponderante, foi a histerese encontrada no sistema de acionamento dos motores. Teoricamente, estes deveriam girar com velocidade linearmente proporcional ao ciclo de carga do sinal PWM alimentado aos drivers, porém isto não foi observado. Tal histerese é nitidamente presente em regiões de acionamento com ciclo de carga pequeno, ou seja, em comandos para que os motores rotacionem lentamente. Notou-se uma região de velocidade nula no eixo do motor para ciclos de carga de 0% à até aproximadamente 32%, para os quais o motor simplesmente não é acionado.

Assim, foi introduzido na equação do controlador um fator corretivo, que o fez operar acima de 32% de ciclo de carga a todo tempo. Esta modificação, no entanto, fez com que a nossa equação de controle torna-se inválida, e após extensiva fase de experimentação com a mesma, foi constatado que a mesma deveria ser abandonada.

Entretanto, o projeto teórico do controlador duplo por avanço de fase nos revelou que, dadas as características dinâmicas da planta, esta pudesse ser estabilizada usando-se um controlador PD clássico em sua forma discreta (ATMEL, 2014) e (Aström, et al., 1988).

$$u[k] = K_p(e[k] + K_d(e[k] - e[k-1]))$$

Assim, a tarefa de controle revelou seu caráter mais experimental. A histerese foi combatida somando-se uma constante de atuação ao esforço de controle calculado, e criando-se uma zona de atuação nula entre ângulos de -0.008 a 0.008 Rad, para que o giro dos motores fosse interrompido quando a planta se encontrasse nesta região de inclinação (este valor foi dimensionado experimentalmente). Desta forma foi observada uma maior estabilidade da planta quando esta permanecia sob sua posição de equilíbrio vertical.

Da forma como este controlador PD foi implementado, pode-se regular o ganho proporcional do mesmo pela alteração da constante G1, que corresponde ao mesmo em código. Ainda, regula-se o ganho derivativo no valor de G3. O controlador foi programado com o fator K_p em evidência para que alterações experimentais neste ganho não modificassem a proporção existente entre a parcela de controle proporcional e a parcela derivativa, assim pode-se ajustar os valores independentemente.

A operação final do controlador segue o seguinte fluxograma:

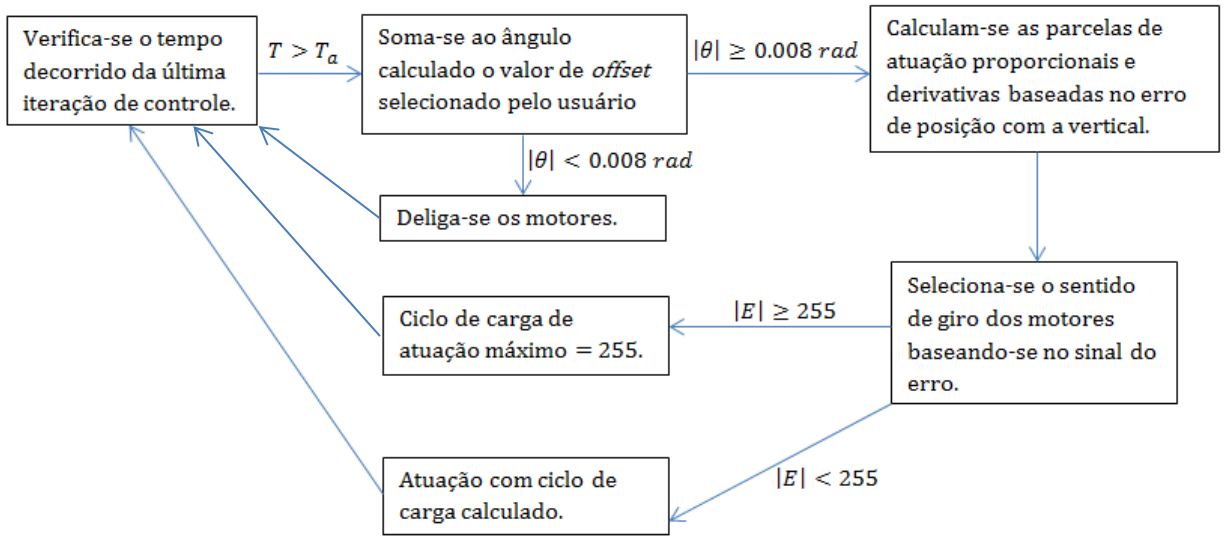

Figura 25 - Fluxograma de cálculo de esforço de controle E baseado no ângulo instantâneo com a vertical θ

Vale observar que o esforço de controle é quantizado num intervalo de 256 valores discretos, devido à operação em oito bits do microcontrolador. Na configuração final do controlador, foi escolhida a seguinte equação de controle:

$$u[k] = 2.46(e[k] + 3.94(e[k] - e[k-1])$$

Na regulagem do sistema via IHM do protótipo modificam-se valores normalizados do ganho que levam em consideração a conversão de Volts para Ciclo de carga de PWM atuante nos drivers.

Esta implementação foge completamente à metodologia de dimensionamento de controladores clássica, que opera sobre sistemas perfeitamente lineares e imunes à histerese. Vale notar fenômenos não modelados comumente se manifestam na realidade, e muitas vezes

estes comprometem a representatividade do projeto teórico. A rotina responsável por controlar a planta encontra-se novamente no Anexo 2 e é chamada CONTROL.

2.8.5. Desempenho Final da Planta sob Testes

Os valores finais dos ganhos proporcional e derivativo do controlador foram escolhidos de forma completamente experimental. Um desempenho satisfatório foi atingida quando a planta foi capaz de equilibrar-se na vertical por períodos maiores que 10 segundos. Durante a fase de testes foi observado que o protótipo se comporta de maneira distinta quando opera sobre diversas superfícies, particularmente superfícies lisas fronte a superfícies acolchoadas ou aveludadas, como tapetes e carpetes. Nestas, notou-se um visível amortecimento oscilatório na planta, possivelmente em decorrência de um maior assentamento dos pneus sobre as vilosidades do chão aveludado. A magnitude diferente do atrito na interface roda – superfície de contato também aumenta entre estas diferentes superfícies, o que favorece a ação de controle do protótipo.

Em superfícies lisas o protótipo consegue manter o equilíbrio em torno de 10s quando ocorrem escorregamentos que ocasionam a queda do modelo, enquanto em superfícies como os tapetes de maior atrito e amortecimento, o modelo se mantém estável indefinidamente.

Outro ponto importante analisado se deu na comparação do resultado obtido entre o uso do filtro complementar processado pela plataforma Arduino com o sinal de ângulo gerado pela DMP da MPU6050. Este último se mostrou mais estável e com resposta mais rápida, isso se deve ao fato de o processamento de dados ser feito diretamente pelo chip do sensor, o que permite que o microprocessador processe o sinal de controle mais rapidamente.

Deve-se levar em conta que o decorrente aumento de poder computacional requerido por processos de filtragem mais estáveis e preciso não é possível, dadas as limitações de processamento existentes e o alto requisito dinâmico de controle da planta, como a arquitetura em oito bits do controlador e sua frequência de *clock* relativamente baixa.

3. Conclusões

A planta mecânica foi modelada de forma clássica, assumindo operação de todos os sistemas de forma perfeitamente linear e desprezando fatores como folga mecânica presente nos motores, deformações dos pneus sob torque e histerese de atuadores. Esta abordagem possibilitou a concepção de um modelo matemático simplificado da planta, sobre o qual foi projetado um compensador por avanço de fase. Entretanto, fenômenos não modelados comprometeram a representatividade do modelo em relação ao protótipo real, o que forçou uma abordagem experimental de projeto de controle.

A possibilidade de ajuste de parâmetros de controle *on-the-fly* realizada por elementos de interface com o usuário no hardware facilitou em muito a tarefa de ajuste do controlador PD posteriormente escolhido para equilibrar a planta, e provavelmente sem esta a tarefa de dimensionar um controlador de forma puramente experimental seria muito dificultada dada a necessidade de reprogramação do software para cada nova configuração de ganhos a ser ensaiada.

Os testes finais com o protótipo nos levaram à escolha de um controlador satisfatório, e revelaram fenômenos físicos que, se desprezados, podem comprometer o projeto e realização de um sistema de controle real. O presente trabalho deixa em aberto a possibilidade de uma análise futura do controlador implementado e a simulação do mesmo em ambiente computacional para validar resultados experimentais. Ainda, vale frisar que fenômenos tribológicos relacionados ao atrito dos pneus com diversos tipos de substrato são capazes, neste caso, de modificar drasticamente a dinâmica da planta, e estudos mais extensos podem

ser realizados afim de se relacionar desempenho com grandezas como coeficiente de atrito do piso.

O trabalho envolveu todas as áreas de conhecimento fundamentais à competência de um Engenheiro Mecatrônico, incluindo programação de software, modelagem e projeto de hardware, constituindo uma prova irrefutável da importância destas tarefas na execução de projetos de engenharia reais.

4. Bibliografia

A. H. Nayfeh, E. Rahman. 2007. *Two-dimensional control for ship mounted cranes: a feasibility study.* Virginia : s.n., 2007. pp. 657-685.

Aström, Karl e Hagglund, Tore. 1988. *PID Controllers: Theory, Design and Tuning.* s.l. : Instrument Society of America, 1988.

ATMEL. 2014. Application Notes on discrete 8-bit PID controllers. [Online] 2014. http://www.atmel.com/images/doc2558.pdf.

Castrucci, Plinio de Lauro, Bittar, Anselmo e Sales, Roberto Moura. 2011. *Controle Automático.* s.l. : LTC, 2011.

Do, K. D. e Seet, G. 2010. *Motion Control of a Two-Wheeled Mobile Vehicle with an Inverted Pendulum.* School of Mechanical and Aerospace Engineering, Nanyang Technological University. Singapore : Springer Science/Business Media, 2010.

Huang Shiuh-Jer, Huang Chien-Lo. 2000. Control of and inverted pendulum using a grey prediction model. *IEEE transactions on industry applications.* 2000, Vol. 36.

Hurst, J. W. 2005. *The role of compliance in Legged Locomotion.* Carnegie Mellon. Pittsburgh : s.n., 2005. Thesis Proposal.

Invensense. 2013. [Online] 2013. http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/.

—. Motion Tracking devices. [Online] http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/.

IPT, Instituto de pesquisas tecnológicas. Informações sobre madeiras. [Online] http://www.ipt.br/informacoes_madeiras/3.htm.

Juang, H. S. e Y., Lum K. 2013. *Design and control of a two-wheel self-balancing Robot using the arduino microcontroller board.* s.l. : ICCA, 2013.

Kailath, T. 1980. *Linear Systems.* Eaglewood Cliffs : Prentice Hall, 1980.

Khalil, H. K. 2002. *Nonlinear Systems.* New Jersey : Prentice Hall, 2002.

Ogata, Katushito. 2010. *Engenharia de Controle Moderno.* São Paulo : Pearson, 2010. p. 61.

OLLIW. 2015. *OLLIW.* [Online] 16 de Janeiro de 2015. http://www.olliw.eu/2013/imu-data-fusing.

Ricci, Mário César. Motores elétricos e acionamentos em mecânica fina. São Paulo : INPE. Aulas 57-60.

Roberge, J. K. 1960. *The Mechanical Seal.* Massachusets Institute of Technology. Cambridge : s.n., 1960. Bachelor's Thesis.

Rowberg, Jeff. 2015. *GITHUB.* [Online] 2015. https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050.

Sadin, P.E. 2003. *Robotic Mechanisms and Mechanical Devices.* 1st. s.l. : McGraw-Hill/TAB Electronics, 2003.

Shaefer, J. F. e Cannon, R. H. 1966. On the control of unstable mechanical systems. 1966, Vol. 3, pp. 12-24.

V. Lopes Jr., et al. 2010. *Controle de um pêndulo invertido utilizando o modelo fuzzy takagi-sugeno.* FEMEC, Universidade Federal de Uberlândia. Uberlândia : s.n., 2010.

5. ANEXOS

5.1. Anexo 1: Desenhos de fabricação e esquemáticos de hardware

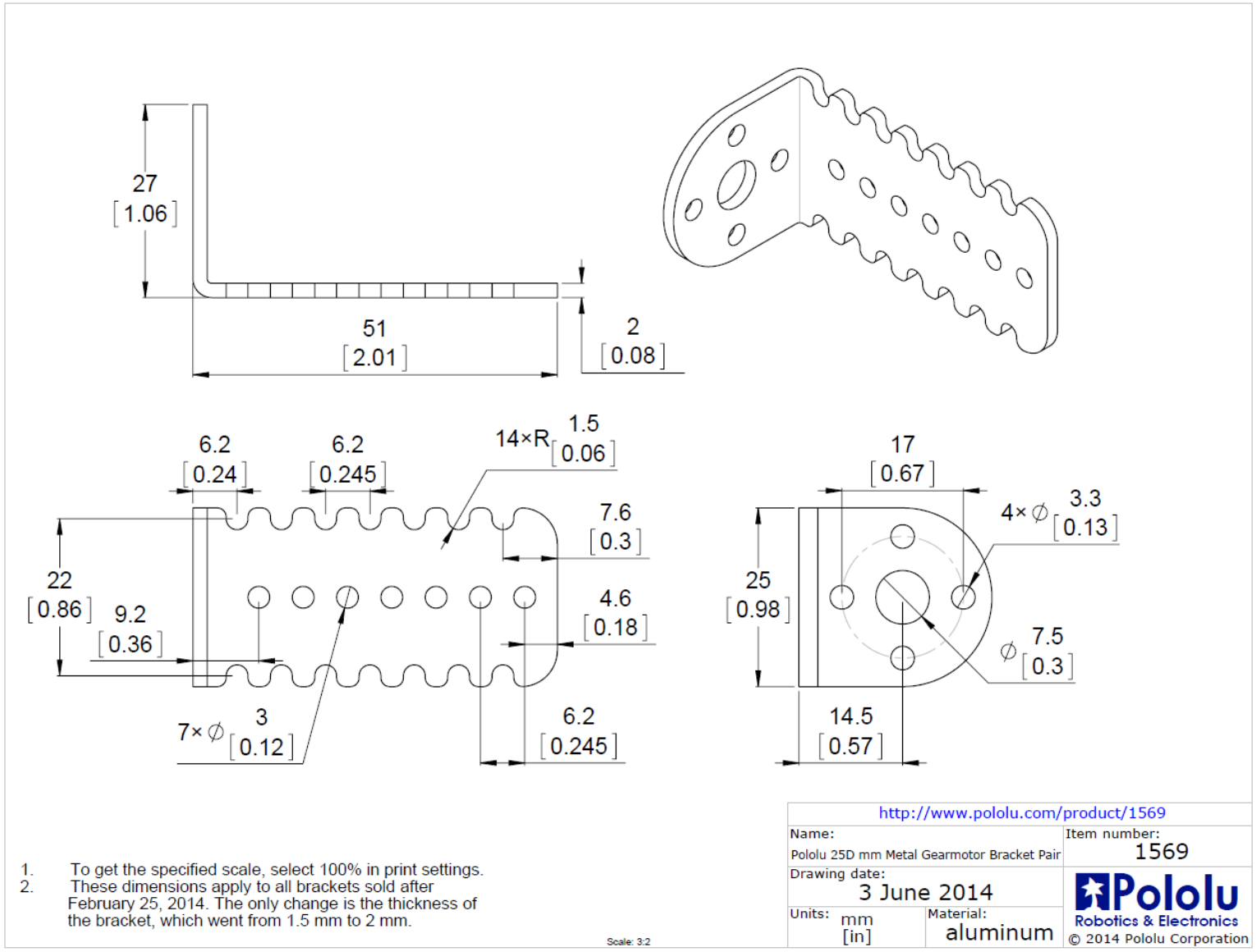

Figura 26: Bracket utilizado na fixação do motor ao chassi

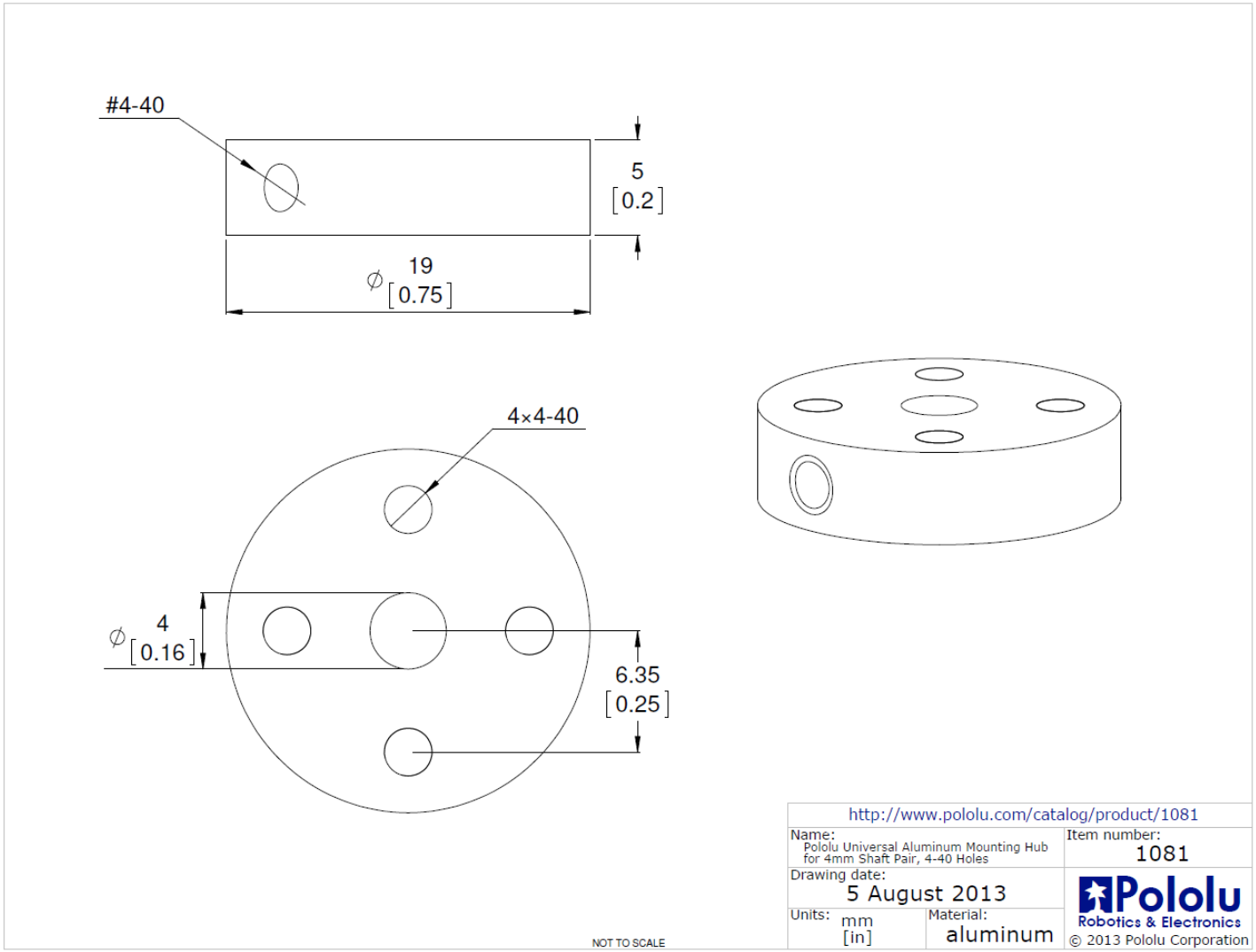

Figura 27: Flange cilíndrico usado na fixação das rodas no Chassis

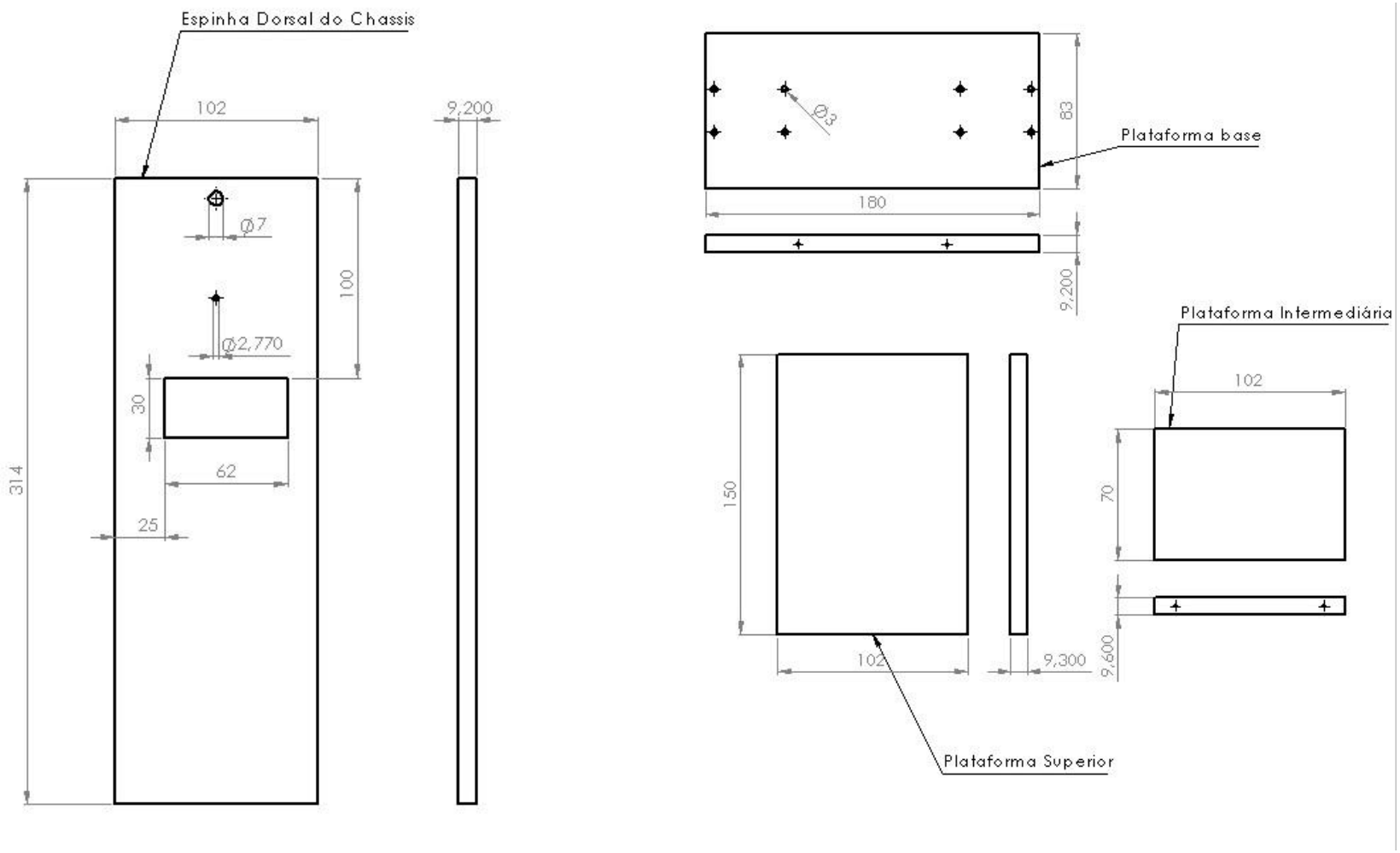

Figura 28: Estrutura em Tauari do chassi e dimensões

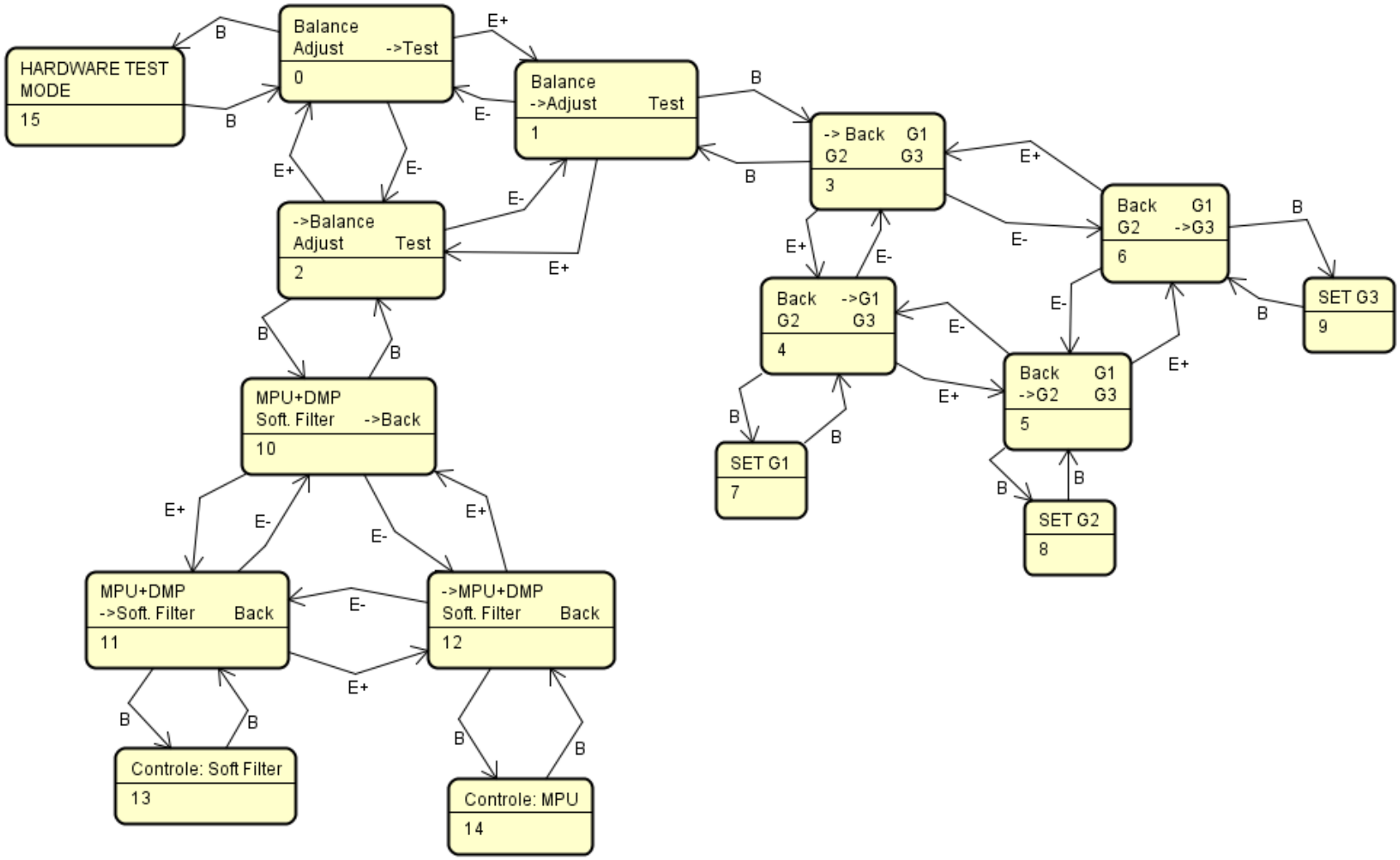

Figura 29 - Máquina de estados e sistema de menus.

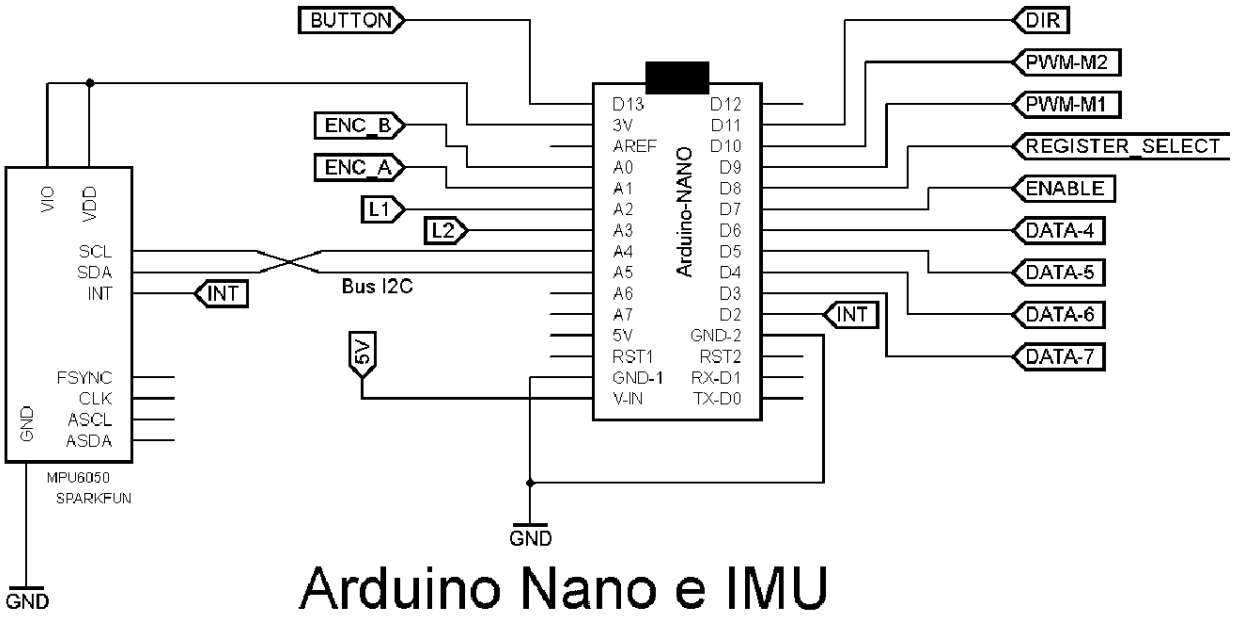

Figura 30 - Esquemático do circuito do microprocessador, com sinais listados.

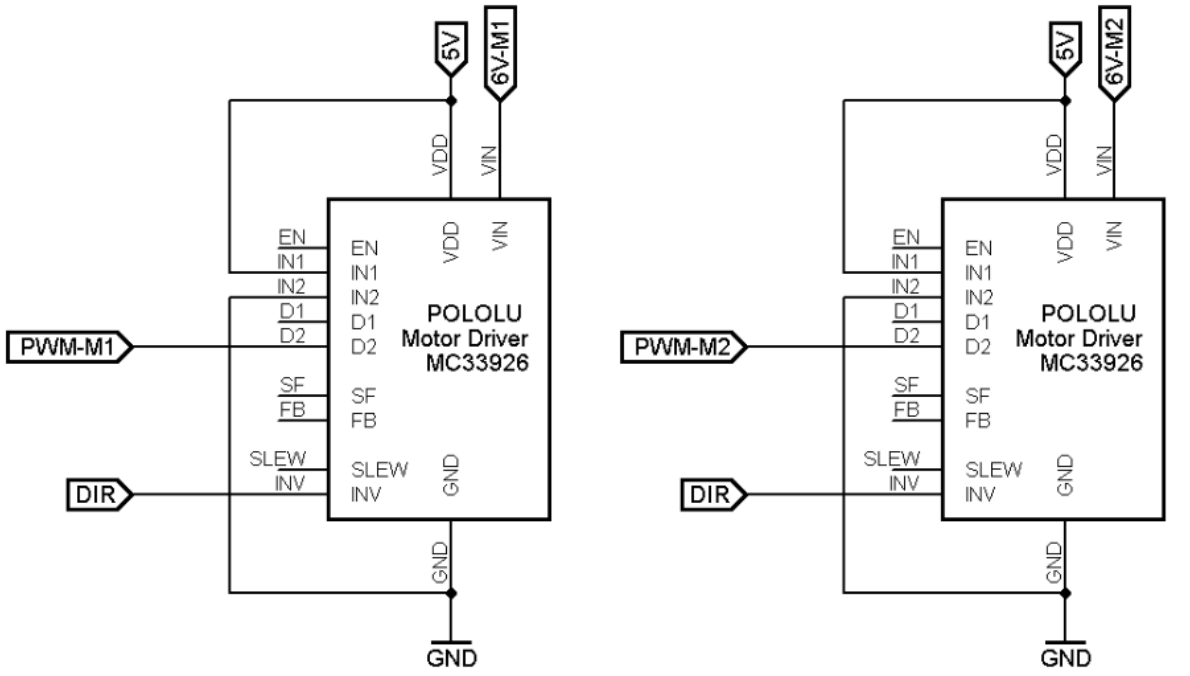

Figura 31 - Esquemático do circuito de drivers dos motores

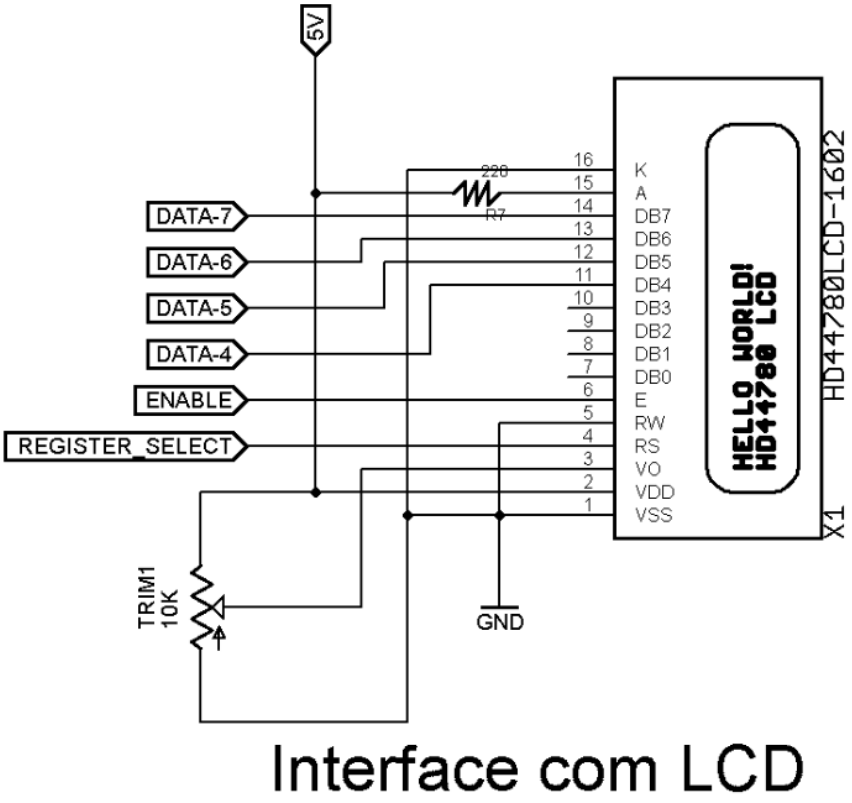

Figura 32 - Esquemático da interface de hardware entre LCD e Microcontrolador

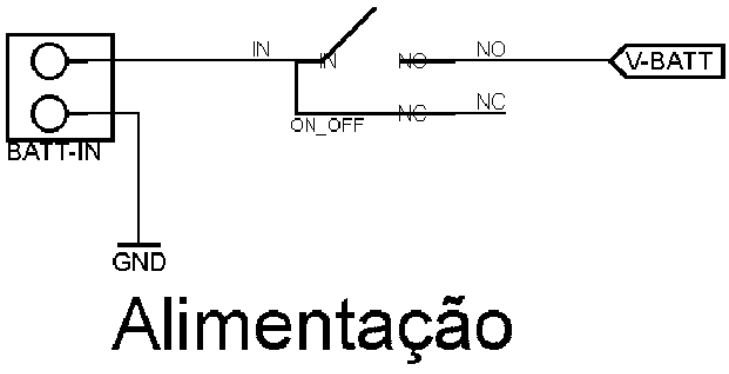

Figura 33 - Entrada de alimentação do Hardware eletrônico

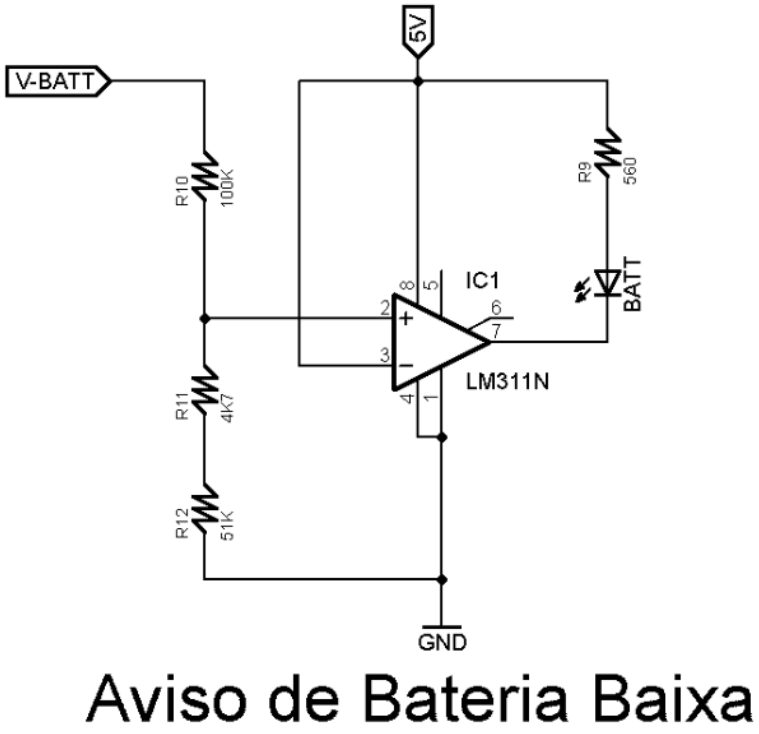

Figura 34 - Circuito comparador para detecção de Bateria Baixa

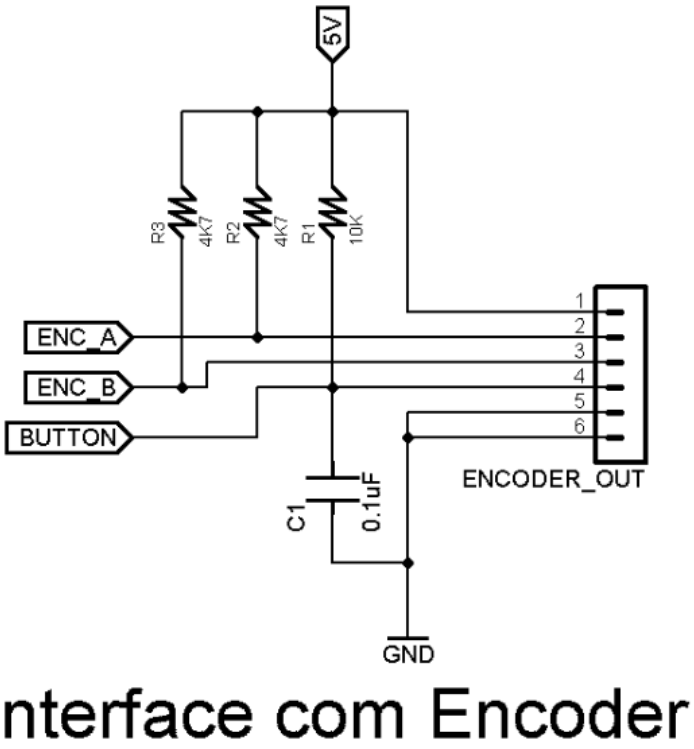

Figura 35 - Circuito de interface com Encoder óptico

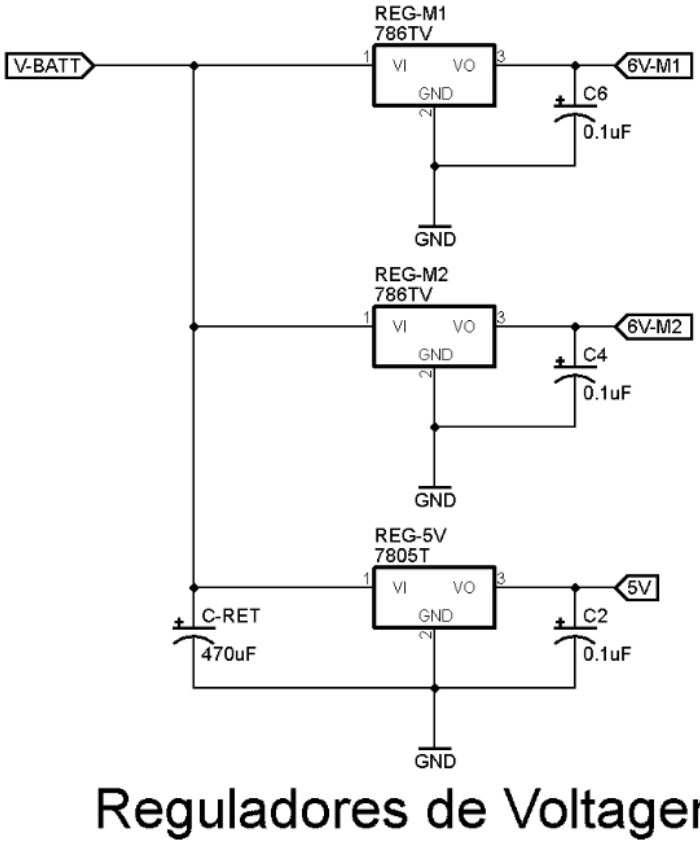

Figura 36 - Circuito de reguladores de Voltagem presentes na Placa.

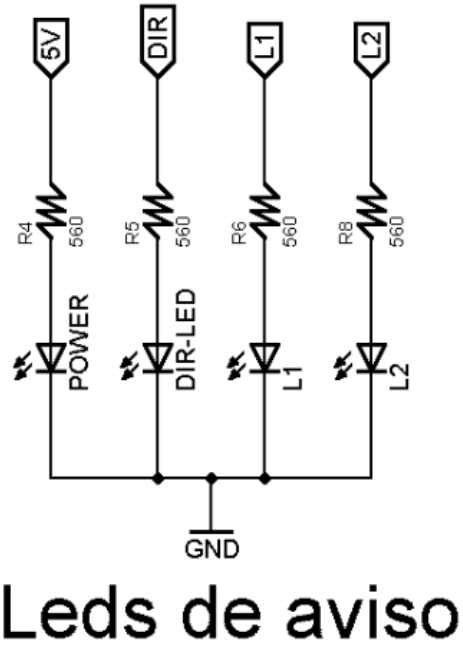

Figura 37 - Leds de aviso e respectivos sinais conectados

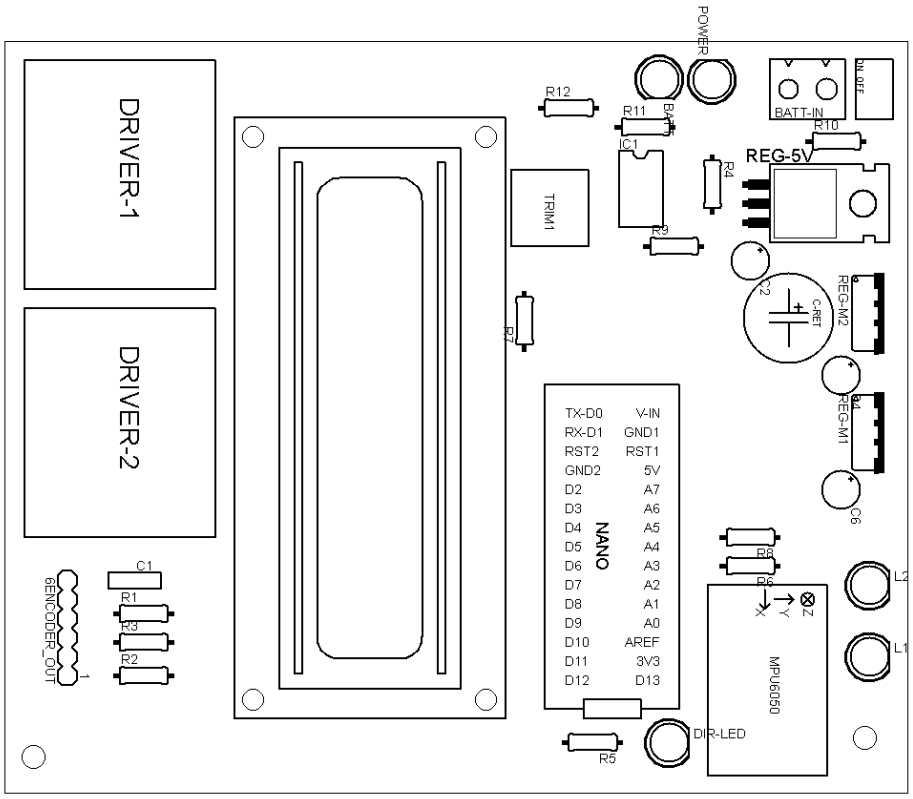

Figura 38 - Disposição de componentes na placa de circuito impresso

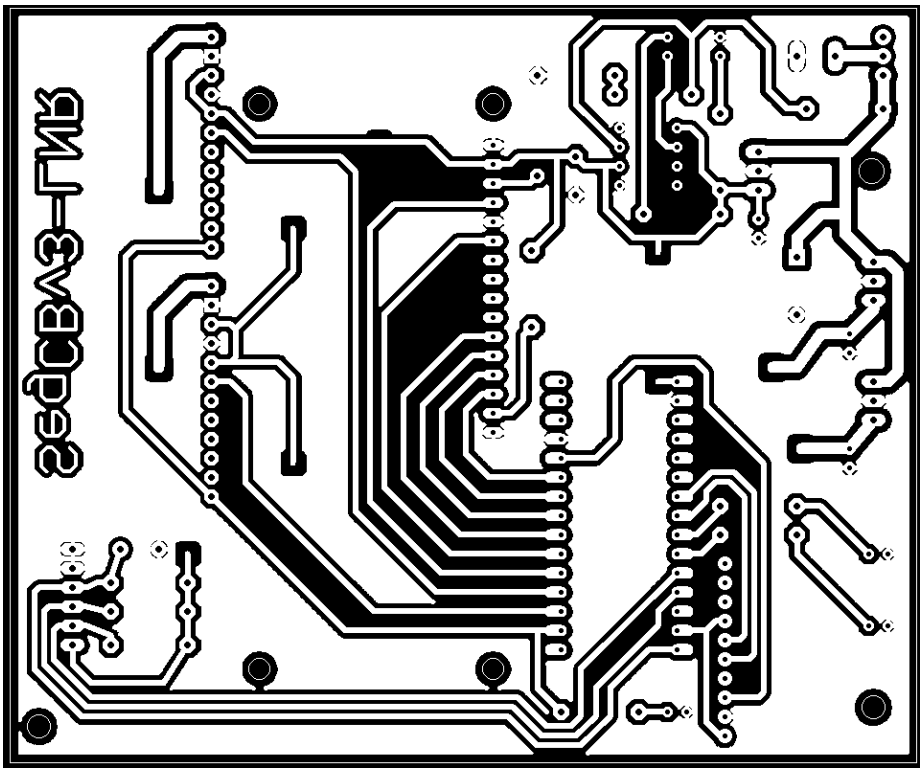

Figura 39 - Roteamento de trilhas na placa de circuito impresso

5.2. Anexo 2: Listagem de Código.

5.2.1. Rotinas principais – Setup e Loop

```
//**********************************//
//          SEGWAY                  //
//    Software final - TCC 2015 //
//********************************* //


// ======  BIBLIOTECAS
#include <LiquidCrystal.h>                              // Biblioteca para LCD
#include "I2Cdev.h"                                     // Suporte para headers das biblioteca
MotionApps
#include "MPU6050_6Axis_MotionApps20.h"         // Biblioteca MotionApps, para interfacear
MPU6050 e Arduino via I2C

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

// ====== DEFINIÇÕES de código
#define dir 11            // Direção de giro dos motores.
#define M1 9              // PWM - Motor #1 no pino D9 do Arduino
#define M2 10             // PWM - Motor #2 no pino D10 do Arduino
#define LED_GREEN A2      // Led verde na saída A2
#define LED_BLUE A3       // Led Azul na saída A3

#define D2R 0.01745       // Constante de conversao degree to radian
#define R2D 57.2958       // Constante de conversao radian to degree


// ====== INSTÂNCIAS de código
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL
LiquidCrystal lcd(8, 7, 6, 5, 4, 3);


// ====== VARIÁVEIS GLOBAIS
byte state;           //Variável que representa o estado atual no sistema de menus do LCD. São
16 estados no total, o inicial é naturalmente o estado ZERO.

bool dmpReady = false; // É setada para TRUE se a função INIT da mpu rodar com sucesso.
uint8_t mpuIntStatus;     // byte de interrupt status da MPU.
uint8_t devStatus;        // retorna o status do dispositivo após cada operação 0 = OK , !0 =
Ruim.
uint16_t packetSize;      // Byte de frame de comunicação com a MPU.
uint16_t fifoCount;       // Variável que conta o número de leituras presentes no FIFO.
uint8_t fifoBuffer[64];   // Buffer de armazenamento do FIFO.
Quaternion q;             // [w, x, y, z]        Container do quaternion, populado no modo DMP.
```

```
VectorFloat gravity;      // [x, y, z]           Vetor gravidade
float ypr[3];             // [yaw, pitch, roll]   Container dos angulos Yaw, Pitch e Roll.
float roll_buffer[3];     // FIFO de três elementos usados nos algoritmos de controle.
float angle_compensated;  // Variável que guarda o ângulo compensado. este ângulo é a soma daquele calculado pelo filtro complementar ou pela DMP com o offset setado pelo usuário
int abs_intensity, intensity_der, intensity_prop; // Variáveis que guardam as intensidades de controle respectivas à ativação do PWM, do ganho derivativo, e do ganho proporcional.
int counter=0;            // Contador usado na leitura do encoder óptico para navegação em menus.
byte old_actual, get_it = 0;  // Variáveis auxiliares usadas na manutenção da máquina de estados.

// Valor inicial dos ganhos G1, G2, G3. Na última configuração estes correspondem respectivamente a Kp, alpha e Kd.
float gain_one=0.43, gain_two= 0.98, gain_three=0.08, angle_offset=0.11;

// Variáveis que mantem registro de tempo, usadas no algoritmo de cotrole e também no filtro complementar.
static unsigned long lastTime, lastTime_filter;
int timeChange, timeFilter;

// Fator de proporção para setar frequencias de corte do filtro complementar. deve estar entre 0 e 1.
float alpha;

//================================
//|||| Variaveis Complementar  ||||
//================================
int16_t ax, ay, az;
int16_t gx, gy, gz;
uint8_t FS_SEL;
uint8_t AFS_SEL;
float accel_y=0.0;
float accel_z=0.0;
float gyro_x=0.0;
float accel_angle_x=0.0;
float gyro_angle_x=0.0;
float gyro_Scale_Factor;
float accel_Scale_Factor;
static unsigned long last_t;
float Filtered_Angle_x = 0.0;
float Filtered_Angle_x_Past_One = 0.0;
float Filtered_Angle_x_Past_Two = 0.0;
//================================

// ====== MATRIZ DE TRANSIÇÃO DE ESTADOS.
/* NOTA #1: Os elementos da matriz são os estados para os quais se transita, a partir do estado representado pelo número da linha do qual pertence o elemento considerado.
NA PRIMEIRA COLUNA: Encoder girado no sentido horário
NA SEGUNDA COLUNA:   Enconder girado no sentido anti-horário
NA TERCEIRA COLUNA: Botão pressionado.
```

```
 * NOTA #2: Os elementos de 50 À 91 sserão usados para decodificar mudanças em variáveis globais, como offset de ângulo, e ganhos internos do controlador.
*/
byte transition_matrix [16][3]=
{   /* E+      E-    Button */
/*S0*/ {1,      2,      15},
/*S1*/ {2,      0,      3},
/*S2*/ {0,      1,      10},
/*S3*/ {4,      6,      1},
/*S4*/ {5,      3,      7},
/*S5*/ {6,      4,      8},
/*S6*/ {3,      5,      9},
/*S7*/ {51,     50,     4},
/*S8*/ {61,     60,     5},
/*S9*/ {71,     70,     6},
/*S10*/{11,     12,     2},
/*S11*/{12,     10,     13},
/*S12*/{10,     11,     14},
/*S13*/{81,     80,     11},
/*S14*/{81,     80,     12},
/*S15*/{91,     90,     0},
};


// CRIAÇÃO DE CARACTER ESPECIAL PARA O LCD, UMA FLECHA À DIREITA.
byte direita[8] = {
  0b00000,
  0b00000,
  0b00100,
  0b00010,
  0b11111,
  0b00010,
  0b00100,
  0b00000
};

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
// FUNÇÃO DE INTERRUPÇÃO - Quando ela roda, o flag mpuInterrupt é setado.
void dmpDataReady() {
    mpuInterrupt = true;
}

// SETUP
void setup() {
    pinMode(LED_GREEN,OUTPUT);
    pinMode(LED_BLUE,OUTPUT);
    pinMode(M1,OUTPUT);
    pinMode(M2,OUTPUT);
    pinMode(11,OUTPUT);
    pinMode(A1, INPUT);
    pinMode(A0, INPUT);
```

```
    pinMode(13, INPUT);
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        TWBR = 12; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    start_lcd();      // INICIALIZA O LCD

    mpu.initialize(); // INICIALIZA A MPU6050

    lcd.print(mpu.testConnection() ? F("Conn OK ") : F(" Conn NOT "));

    // A variável devStatus vale 0 se a MPU for corretamente inicializada, vale 1 se ocorrer erro
de leitura inicial de memória, e 2 se ocorrer erro na atualização de dados da DMP.
    devStatus = mpu.dmpInitialize();

    // SETAR OFFSETS
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

    if (devStatus == 0) { // Se devStatus retornar 0, é porque a MPU foi inicializada com sucesso.
        mpu.setDMPEnabled(true);
        attachInterrupt(0, dmpDataReady, RISING); // Roda a função dmpDataReady quando a
interrupção externa ocorre.
        mpuIntStatus = mpu.getIntStatus();
        dmpReady = true;
        packetSize = mpu.dmpGetFIFOPacketSize();
        flash(LED_BLUE,10,200,0);               // Pisca o Led Azul 3 vezes para anunciar que a
MPU foi inicializada com sucesso.
    }else{    // Se devStatus retornar 1 ou 2, é porque ocorreu erro na inicialização.
        flash(LED_GREEN,3,500,1);
    }
    FS_SEL = 3 ;                                // Default = 3, aqui setamos zero para aumentar a
sensibilidade do gyro
    AFS_SEL = 1;                                // Controla a sensibilidade do acelerômetro +-4g
    mpu.setFullScaleGyroRange(FS_SEL);                  // Seta sensibilidade do gyro na MPU.
    mpu.setFullScaleAccelRange(AFS_SEL);                // Seta a sensibilidade do acelerômetro
na MPU.
    gyro_Scale_Factor = 131.0 / 8;                  // Seta fator de correção para a sensibilidade
do gyro escolhida
    accel_Scale_Factor = 16384.0 /2.0;                  // Seta fator de correção para a
sensibilidade do acelerômetro escolhida.
    lcd.createChar (0, direita);                    // Carrega a flecha para a direita na EEPROM do
LCD
    lcd.clear();
    Serial.begin(115200);                           // Inicia porta serial (somente para debugging)
    state = 0;
```

```
    delay(200);
} // END SETUP

// LOOP
void loop() {
  encoder_refresh();
  refresh_screen(state);
  switch (state){
    case 13:
       RUN_FILTER();
       CONTROL(Filtered_Angle_x, Filtered_Angle_x_Past_One , Filtered_Angle_x_Past_Two);
    break;
    case 14:
       RUN_DMP();
       CONTROL(roll_buffer[0],roll_buffer[1],roll_buffer[2]);  // Checado 22:37 out/26 DURATION:
90uS. //Checado 16:07 Nov/02 Duration: 7,5ms
    break;
    default:
      analogWrite(M1, 0);
      analogWrite(M2, 0);
    break;
  }

} // END LOOP
```

5.2.2. Rotina de aquisição de Ângulos via *Digital Motion Processing*

```
void RUN_DMP(){
    // Se o setup da DMP não funcionou, não faça nada.
    if (!dmpReady){
      lcd.print("-DMP FAIL-");
      return;
    }

    // resetar flag de interrupção e aquisitar INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // Aquisitar número de valores presente no FiFo interno à MPU6050
    fifoCount = mpu.getFIFOCount();

    // Checar o filtro para detector Overflow. Isto não deve acontecer.
    if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
       // reset so we can continue cleanly
       mpu.resetFIFO();
       Serial.println(F("FIFO overflow!"));

    // Checar pela interrupt DMPDATAREADY
    } else if (mpuIntStatus & 0x02) {
```

```
      // wait for correct available data length, should be a VERY short wait
      while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

      // read a packet from FIFO
      mpu.getFIFOBytes(fifoBuffer, packetSize);

      // track FIFO count here in case there is > 1 packet available
      // (this lets us immediately read more without waiting for an interrupt)
      fifoCount -= packetSize;

      // ATUALIZA OS VALORES DE YAW PITCH E ROLL.
      #ifdef OUTPUT_READABLE_YAWPITCHROLL
         mpu.dmpGetQuaternion(&q, fifoBuffer);
         mpu.dmpGetGravity(&gravity, &q);
         mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
      #endif
   }

   // atualiza os valores do Buffer de ângulos de giro:
  roll_buffer[2] = roll_buffer[1];
  roll_buffer[1] = roll_buffer[0];
  roll_buffer[0] = ypr[2];          //primeiro valor do buffer é a leitura mais atualizada de dados.

}
```

5.2.3. Rotina de aquisição de ângulos via Filtro Complementar

```
void RUN_FILTER(){
     unsigned long now_filter = millis();
     timeFilter = (now_filter - lastTime_filter);
     if(timeFilter >= 10){
       mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);  // Atualiza os valores raw de todos os
eixos.
       alpha = gain_two;                          // Fator alfa do complementary
       gyro_x = (gx) / gyro_Scale_Factor;           // Leva gyro_x para a escala em radianos
       accel_y = (ay) / accel_Scale_Factor;
       accel_z = (az) / accel_Scale_Factor;
       gyro_angle_x = gyro_x * timeFilter/1000.0 + Filtered_Angle_x_Past_One*R2D ;     //
Integra o valor do gyro, com o angulo anterior filtrado em vez do angulo ruidoso
       gyro_angle_x *= D2R; //


       //accel_angle_x = atan(accel_y/sqrt(pow(accel_x,2) + pow(accel_z,2)));   // Analisar qual
dos dois e mais estavel
       accel_angle_x = atan2(accel_y, accel_z );                          // Comparar com atan,
Ambos em radians

       // Atualização do filtro complementar e do FIFO.
       Filtered_Angle_x_Past_Two = Filtered_Angle_x_Past_One; // Atualiza a leitura retrasada
do FiFo
```

```
      Filtered_Angle_x_Past_One = Filtered_Angle_x;          // Atualiza a leitura passada do FiFo
      Filtered_Angle_x = alpha*gyro_angle_x + (1.0 - alpha)*accel_angle_x; // Atualiza algoritmo
de filtro complementar

      //DEBUGGING:
      Serial.print(timeFilter);             // imprime o tempo atual em milissegundos
      Serial.print(" ");
      Serial.print(gyro_angle_x);       // imprime a intensidade de atuação baseada no erro
proporcional
      Serial.print(" ");
      Serial.print(accel_angle_x);      // imprime a intensidade de atuação baseada no erro
proporcional
      Serial.print(" ");
      Serial.print(accel_z);               // imprime a intensidade de atuação baseada no erro
derivativo
      Serial.print(" ");
      Serial.println(Filtered_Angle_x + angle_offset); // imprime a intensidade do PWM.
      //FIM DO DEBUGGING.

      lastTime_filter = now_filter;
    }
}
```

5.2.4. Rotina de tratamento do *Encoder* de navegação

```
void encoder_refresh(){
 boolean A, B;    //Aqui a variável ser do tipo static significa que não muda de valor conforme a
função na qual ela se encontra declarada é chamada novamente.
 static boolean A_OLD, B_OLD;
 A = digitalRead(A0);
 B = digitalRead(A1);
 if(digitalRead(13)==0){
   state_refresh(state, 2); // ATUALIZA a variável "state" usando a terceira coluna da matriz de
transição.
   while(digitalRead(13)==0){
     continue;            // Não abandona o loop até que o botão seja solto, Este método realiza
debouncing de maneira muito efetiva.
   }
 }
 if(A!=A_OLD || B!=B_OLD){
  if(A ^ B_OLD){
    state_refresh(state, 1); // ATUALIZA a variável "state" usando a segunda coluna da matriz
de transição.
  }else{
    state_refresh(state, 0); // ATUALIZA a variável "state" usando a primeira coluna da matriz de
transição.
  }
// VOA LIXÃO!!!!
  A_OLD = A;
  B_OLD = B;
 }
```

```
}

void state_refresh(byte current_state, byte input){
  /*
  input = 0 -> GIRO NO SENTIDO HORÁRIO
  input = 1 -> GIRO NO SENTIDO ANTI HORÁRIO
  input = 2 -> BOTÃO.
  */
  get_it = transition_matrix[current_state][input];
  if(get_it < 50){
    state = transition_matrix[current_state][input];
  }else{
    switch (get_it){
      case 51:
        gain_one+=0.01;
        break;
      case 50:
        gain_one-=0.01;
        break;
      case 61:
        gain_two+=0.01;
        break;
      case 60:
        gain_two-=0.01;
        break;
      case 71:
        gain_three+=0.01;
        break;
      case 70:
        gain_three-=0.01;
        break;
      case 81:
        angle_offset+=0.01;
        break;
      case 80:
        angle_offset-=0.01;
        break;
      case 91:
        counter++;
        break;
      case 90:
        counter--;
        break;
      default:
      break;
    }
  }
}
```

5.2.5. Rotinas de atualização do LCD de caracteres

```
void start_lcd(){
   lcd.begin(16, 2);
   //lcd.print(0);
}

void refresh_screen(byte actual){
  if(actual!=old_actual){
    old_actual=actual;
    lcd.clear();
    switch (actual) {
      case 0:
        lcd.home();
        lcd.print(" Balance");
        lcd.setCursor(0,1);
        lcd.print(" Adjust ");
        lcd.setCursor(10,1);
        lcd.write(byte(0));
        lcd.print("Test");
        break;
      case 1:
        lcd.home();
        lcd.print(" Balance");
        lcd.setCursor(0,1);
        lcd.write(byte(0));
        lcd.print("Adjust ");
        lcd.setCursor(10,1);
        lcd.print(" Test");
        break;
      case 2:
        lcd.home();
        lcd.write(byte(0));
        lcd.print("Balance");
        lcd.setCursor(0,1);
        lcd.print(" Adjust ");
        lcd.setCursor(10,1);
        lcd.print(" Test");
        break;
      case 3:
        lcd.home();
        lcd.write(byte(0));
        lcd.print("Back");
        lcd.setCursor(9,0);
        lcd.print(" Set G1");
        lcd.setCursor(0,1);
        lcd.print(" Set G2");
        lcd.setCursor(9,1);
        lcd.print(" Set G3");
        break;
      case 4:
```

```
    lcd.home();
    lcd.print(" Back");
    lcd.setCursor(9,0);
    lcd.write(byte(0));
    lcd.print("Set G1");
    lcd.setCursor(0,1);
    lcd.print(" Set G2");
    lcd.setCursor(9,1);
    lcd.print(" Set G3");
    break;
  case 5:
    lcd.home();
    lcd.print(" Back");
    lcd.setCursor(9,0);
    lcd.print(" Set G1");
    lcd.setCursor(0,1);
    lcd.write(byte(0));
    lcd.print("Set G2");
    lcd.setCursor(9,1);
    lcd.print(" Set G3");
    break;
  case 6:
    lcd.home();
    lcd.print(" Back");
    lcd.setCursor(9,0);
    lcd.print(" Set G1");
    lcd.setCursor(0,1);
    lcd.print(" Set G2");
    lcd.setCursor(9,1);
    lcd.write(byte(0));
    lcd.print("Set G3");
    break;
  case 10:
    lcd.home();
    lcd.print(" MPU+DMP");
    lcd.setCursor(9,0);
    lcd.write(byte(0));
    lcd.print("Back");
    lcd.setCursor(0,1);
    lcd.print(" Soft. Filter");
    break;
  case 11:
    lcd.home();
    lcd.print(" MPU+DMP");
    lcd.setCursor(9,0);
    lcd.print(" Back");
    lcd.setCursor(0,1);
    lcd.write(byte(0));
    lcd.print("Soft. Filter");
    break;
  case 12:
```

```
        lcd.home();
        lcd.write(byte(0));
        lcd.print("MPU+DMP");
        lcd.setCursor(9,0);
        lcd.print(" Back");
        lcd.setCursor(0,1);
        lcd.print(" Soft. Filter");
        break;
      default:
//        lcd.clear();
//        lcd.home();
//        lcd.print("input invalid");
//        lcd.setCursor(0,1);
//        lcd.print("State:");
//        lcd.print(state);
//        //delay(3000);
      break;
    }
  }
switch (actual) {
      case 7:
        lcd.home();
        lcd.print("Setting G1:");
        lcd.setCursor(0,1);
        lcd.print(gain_one);
        break;
      case 8:
        lcd.home();
        lcd.print("Setting G2:");
        lcd.setCursor(0,1);
        lcd.print(gain_two);
        break;
      case 9:
        lcd.home();
        lcd.print("Setting G3:");
        lcd.setCursor(0,1);
        lcd.print(gain_three);
        break;
      case 13:
        lcd.home();
        lcd.print(Filtered_Angle_x+angle_offset);
        lcd.setCursor(6,0);
        lcd.print(alpha);
        lcd.setCursor(0,1);
        lcd.print(angle_offset);
        lcd.setCursor(6,1);
        lcd.print(abs_intensity);
        break;
      case 14:
        lcd.home();
        lcd.print(roll_buffer[0]+angle_offset);
```

```
        lcd.setCursor(0,1);
        lcd.print(angle_offset);
        lcd.setCursor(6,1);
        lcd.print(abs_intensity);
//        lcd.setCursor(10,1);
//        lcd.print(abs(intensity_der));
        break;
      case 15:
        lcd.home();
        lcd.print("Test Mode");
        lcd.setCursor(0,1);
        lcd.print(counter);
        break;
      default:
      break;
    }
}
```

5.2.6. Rotina de Controle dos Motores

```
void CONTROL(float angle_now, float angle_past_one, float angle_past_two){
      unsigned long now = millis();
      timeChange = (now - lastTime);
      if(timeChange >= 20 ){
      digitalWrite(LED_GREEN, HIGH);
      angle_compensated = angle_now + angle_offset ;// Gera o angulo que sera usado no
algoritmo de controle. nada mais é doq MPU somado ao offset calibrado na hora.
      if(angle_compensated < 0.06 && angle_compensated > -0.06){
          intensity_der = (int)1461*((gain_one*gain_three)*20)*(angle_now-angle_past_one);       //
Ganho 3 funciona como constante de tempo do Deriv.
          intensity_prop = (int)1461*gain_one*(angle_compensated);                              //
Parseando intensity_der como int, pois a variável é inteira.
          abs_intensity = abs(intensity_prop) + abs(intensity_der) + 87 ;                       // calcula a
intensidade do PWM. 0-255.
      }else{
          intensity_der = (int)1461*((gain_one*gain_three)*20)*(angle_now-angle_past_one);       //
Ganho 3 funciona como constante de tempo do Deriv.
          intensity_prop = (int)6574*gain_one*(angle_compensated);                              //
Parseando intensity_der como int, pois a variável é inteira.
          abs_intensity = abs(intensity_prop) + abs(intensity_der) + 87 ;
      }
        // Atualiza sentido de giro do motor baseado nos valores de intensidade.
        if(intensity_prop + intensity_der > 0){
          digitalWrite(dir, LOW);
        }else{
          digitalWrite(dir, HIGH);
        }


        if(abs_intensity<256){                //Se não saturarmos a saída:
          if(abs(angle_compensated)>=0.008){    //se o angulo não estiver entre +- 0.008 RAD.
```

```
          analogWrite(M1, abs_intensity);
          analogWrite(M2, abs_intensity);
        }else{                          // Se o ângulo for menor que 0.008 RAD em módulo, zera-se a
saída do motor.
          analogWrite(M1, 0);
          analogWrite(M2, 0);
        }
      }else{                            // Se saturarmos, joga o máximo.
        analogWrite(M1, 255);
        analogWrite(M2, 255);
      }
      lastTime = now;
      digitalWrite(LED_GREEN,LOW);

//      //DEBUGGING:
//      Serial.print(timeChange);      // imprime o tempo atual em milissegundos
//      Serial.print(" ");
//      Serial.print(angle_compensated);       // imprime a intensidade de atuação baseada no
erro proporcional
//      Serial.print(" ");
//      Serial.print(intensity_prop);  // imprime a intensidade de atuação baseada no erro
proporcional
//      Serial.print(" ");
//      Serial.print(intensity_der);   // imprime a intensidade de atuação baseada no erro
derivativo
//      Serial.print(" ");
//      Serial.println(abs_intensity); // imprime a intensidade do PWM.
//      //FIM DO DEBUGGING.
    }
}
```

5.2.7. Rotinas de apoio

```
void flash(int pin, int times, int interval, bool fim){
  for(int c=0;c<times;c++){
    digitalWrite(pin, HIGH);
    delay(interval/2);
    digitalWrite(pin,LOW);
    delay(interval/2);
  }
  if(fim)digitalWrite(pin,HIGH);
}
```

5.3. Anexo 3: Fotos do protótipo e ensaios

Figura 40: Chassi construído em Tauari (Maio de 2015)

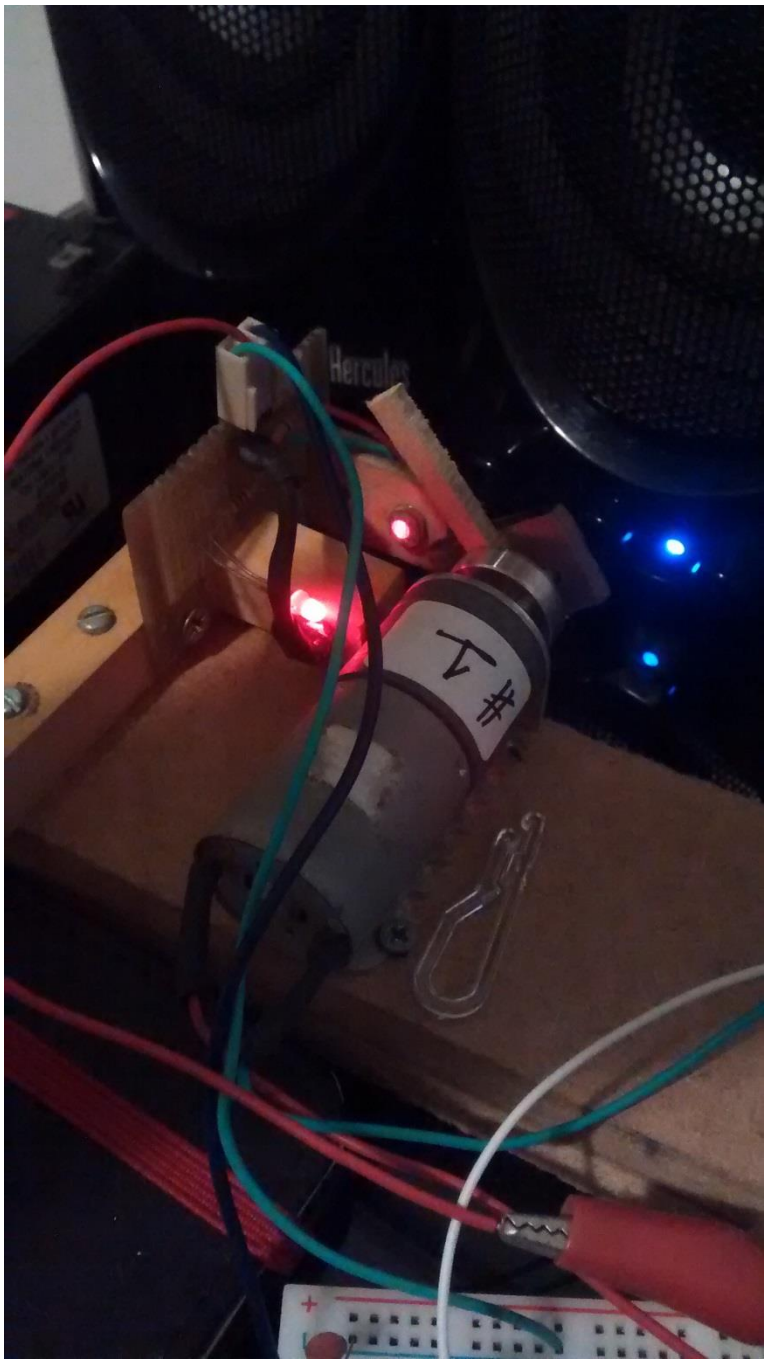

Figura 41: Tacômetro improvisado para medição da velocidade angular do motor em regime. Um diodo Laser e um divisor de tensão usando um LDR geram pulsos de tensão a cada vez que o Laser é interrompido (Maio de 2015)

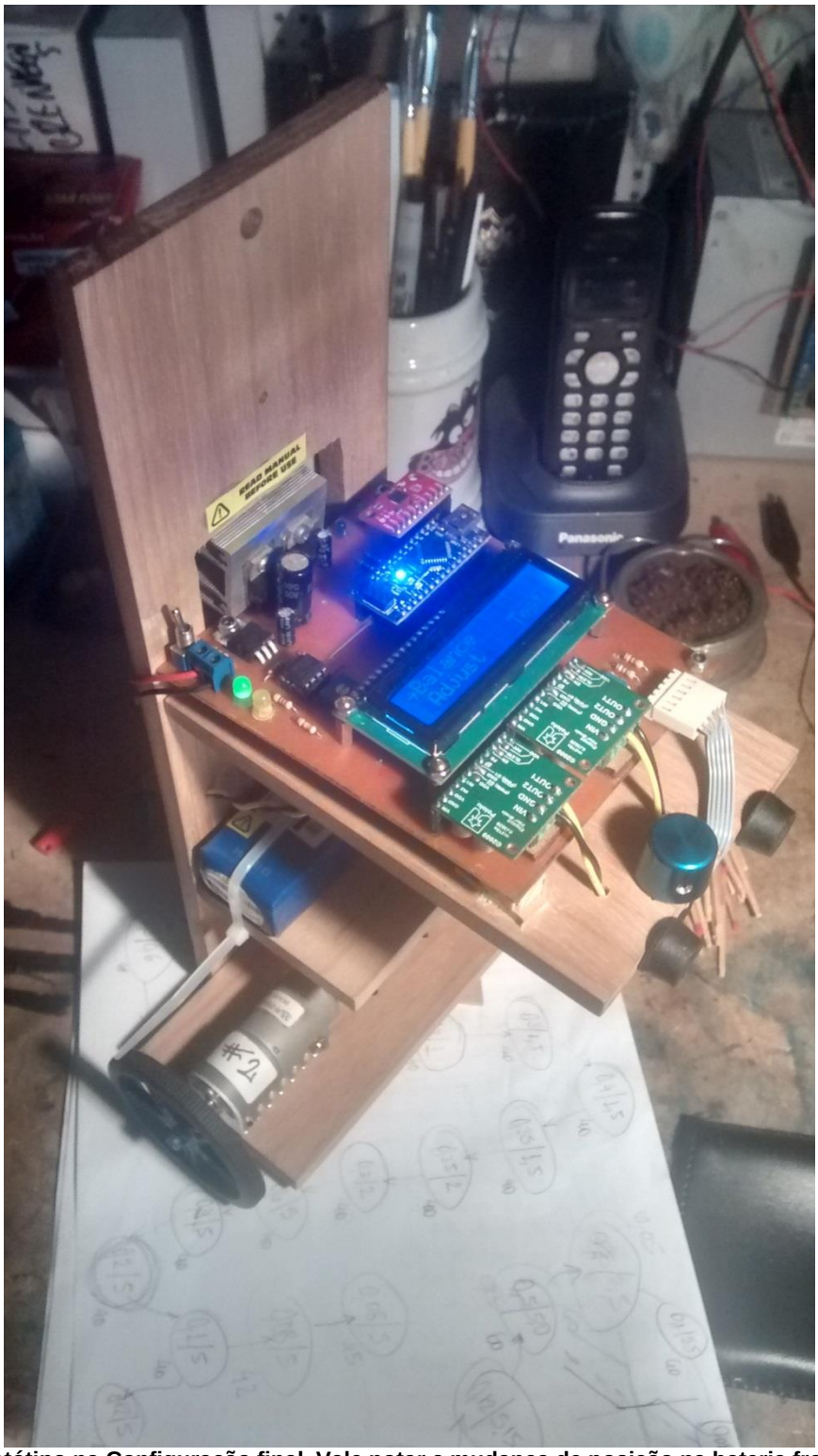

Figura 42 - Protótipo na Configuração final. Vale notar a mudança de posição na bateria fronte ao projeto inicial. Ao fundo, sequencias de testes de controladores PD e rascunhos de como sanar problemas de histerese (Novembro de 2015)