Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Eletrônica

Plataforma de teste para sistemas de controle baseada em pêndulo invertido sobre duas rodas

Autor: Pedro Eugênio Machado de Lima
Orientador: Prof. Dr. Renato Coral Sampaio

Brasília, DF
2019

Pedro Eugênio Machado de Lima

Plataforma de teste para sistemas de controle baseada em pêndulo invertido sobre duas rodas

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Renato Coral Sampaio

Brasília, DF

2019

Pedro Eugênio Machado de Lima

Plataforma de teste para sistemas de controle baseada em pêndulo invertido sobre duas rodas/ Pedro Eugênio Machado de Lima. – Brasília, DF, 2019-
89 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Renato Coral Sampaio

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2019.

1. sistemas de controle. 2. pêndulo invertidos sobre duas rodas. I. Prof. Dr. Renato Coral Sampaio. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Plataforma de teste para sistemas de controle baseada em pêndulo invertido sobre duas rodas

CDU 02:141:005.6

Pedro Eugênio Machado de Lima

Plataforma de teste para sistemas de controle baseada em pêndulo invertido sobre duas rodas

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 08 de julho de 2019 – Data da aprovação do trabalho:

Prof. Dr. Renato Coral Sampaio
Orientador

Prof. Dr. Roberto de Souza Baptista
Convidado 1

Prof. Dr. Daniel M. Muñoz Arboleda
Convidado 2

Brasília, DF
2019

Este trabalho é dedicado a minha família e meus amigos

Agradecimentos

À Universidade de Brasília pela oportunidade de estudo e crescimento.

Ao Prof. Me. Renato Coral Sampaio pela orientação e oportunidade de desenvolver este projeto.

Ao laboratório LEIA por disponibilizar recursos ao longo deste projeto.

Ao Fabián Barrera Prieto pela disponibilidade e ajuda no projeto.

À Camila Miranda Fettermann por ajudar a obter os artigos fundamentais neste trabalho.

Know all the theories, master all the techniques,
but as you touch a human soul be just another human soul.
(C. G. Jung)

Resumo

Sistemas de controle se tornaram comuns em várias áreas da engenharia e é esperado que os engenheiros tenham conhecimento na área de controle. Para auxiliar no ensino, ferramentas práticas de implementação do conhecimento se tornaram essenciais nos cursos de engenharia. É proposto neste trabalho a criação de uma plataforma de testes para sistemas de controle com base no modelo matemático de um pêndulo invertido sobre duas rodas. A plataforma foi desenvolvida usando sensores inerciais para fornecer as entradas do sistema de controle e dois motores DC para representar a saída do sistema. Foi usado um *driver* para fornecer a corrente necessária para os motores e para acompanhar o movimento deles foram usados dois *encoders*. Dessa forma, é possível implementar diferentes sistemas de controle para o modelo matemático do pêndulo invertido sobre duas rodas através de um *software* embarcado. O objetivo deste trabalho foi concluído com sucesso, um protótipo foi fabricado e todos os componentes funcionam de forma integrada.

Palavras-chave: sistema de controle. pêndulo invertido sobre duas rodas. sensor inercial. motores DC.

Abstract

Control systems are common in several areas of engineering, and it is expected that engineers are acquainted with them. Practical tools for knowledge implementation, used to aid the teaching process, are essential in engineering courses. This work proposes the creation of a platform for testing control systems based on the mathematical model of a two-wheels inverted pendulum. The platform was developed using inertial sensors to provide the control system input; and two DC motors to represent the system output. A driver was used to provide the current required by the motors and, to track their movement, two encoders were used. Thereby, it is possible to implement different control systems for the mathematical model of two-wheels inverted pendulum with the use of an embedded software. The objective of this work was successfully concluded, a prototype was produced and all the components work in an integrated manner.

Key-words: control system. two-wheeled inverted pendulum. inertial sensor. DC motors.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PID	*Proporcial-Integral-Derivativo*
TWID	*Two-wheeled Inverted Pendulum*
LQR	*Linear–quadratic Regulator*
MPC	*Model-based Predictive Control*
SMC	*Sliding Mode Control*
IMU	*Inercial Measurement Unit*
LKF	*LInear Kalman Filter*
DC	*Direct Current*
PWM	*Pulse Width Modulation*
I^2C	*Inter-Integrated Circuit*
SCL	*Serial Clock*
SDA	*Serial Data*
BBB	*BeagleBone Black*
BBBW	*BeagleBone Black Wireless*
GPIO	*General Purpose Input/Output*
PRU	*Programmable Real-Time Unit*
PRUSS	*Programmable Real-Time Unit Subsystem*
PRU-ICSS	*Programmable Real-Time Units And Industrial Controller Subsystem*
RPM	*Rotações por Minuto*
MEMS	*MIcro-Electro-Mechanical System*

Lista de símbolos

θ	Ângulo de Arfagem [rad]
ϕ	Ângulo de Rolagem [rad]
ψ	Ângulo de Guinada [rad]
x	Deslocamento [m]
F	Força [N]
φ_R	Ângulo de rotação da roda direita [rad]
φ_L	Ângulo de rotação da roda esquerda [rad]
T_R	Torque do motor direito [Nm]
T_L	Torque do motor esquerdo [Nm]
μ_R	Ação de controle do motor direito
μ_L	Ação de controle do motor esquerdo
H	Força de contato horizontal no eixo das rodas [N]
V	Força de contato vertical no eixo das rodas [N]
N	Força normal [N]
P	Peso [N]
g	Aceleraçõa gravitacional [$\frac{m}{s^2}$]
r	Raio das rodas [m]
m	Massa (do veículo) [kg]
M	Massa das duas rodas do veículo [kg]
l	Distância do centro de massa [m]
d	Distância entre as duas rodas [m]
V_{cc}	Tensão de alimentação [V]
b	Atrito dinâmico do motor [kg$\cdot m^2$]

C_a	Atrito dinânico no eixo das rodas [kg·m^2]
J_p	Momento de inércia do veículo [kg·m^2]
J_z	Momento de inércia do veículo no eixo z [kg·m^2]
J_w	Momento de inércia das rodas, engrenagens e eixo [kg·m^2]
K_m	Constante de torque do motor $[N \cdot m \cdot A^{-1}]$
K_e	Constante de força contra-eletromotriz $[V \cdot s]$
R	Resistência do motor [Ω]
dz	Zona morta do motor
F_c	Força Coriolis [N]
v	Velocidade $[\frac{m}{s}]$
a_c	Aceleração de Coriolis $[\frac{m}{s^2}]$
Ω	Velocidade angular $[\frac{rad}{s}]$
F_{el}	Força elástica [N]
k	constante da mola
a	Aceleração $[\frac{m}{s^2}]$
B_x	Campo magnético medido pelo magnetômetro no eixo x [T]
B_y	Campo magnético medido pelo magnetômetro no eixo y [T]
B_z	Campo magnético medido pelo magnetômetro no eixo z [T]
V_x	Efeito de *hard iron* no eixo x [T]
V_y	Efeito de *hard iron* no eixo y [T]
V_z	Efeito de *hard iron* no eixo z [T]
f_x	Aceleração medida pelo acelerômetro no eixo x $[\frac{m}{s^2}]$
f_y	Aceleração medida pelo acelerômetro no eixo y $[\frac{m}{s^2}]$
f_z	Aceleração medida pelo acelerômetro no eixo z $[\frac{m}{s^2}]$
$\dot{}$	Derivada
°	Graus

$_0$	Instante inicial
$_k$	Instante k
$_{k+1}$	Instante k posterior
$_{k-1}$	Instante k anterior

Sumário

1 Introdução

No século XVIII foi desenvolvido por James Watt um controlador de velocidade de máquinas a vapor considerado o primeiro sistema de controle significativo. Em 1922 foi desenvolvido por Minorsky um controlador para pilotagem de embarcações utilizando equações diferenciais e análise de estabilidade. Nos anos seguinte foram apresentados mais trabalhos importantes na área de controle por Nyquist e Hazen. Posteriormente foram desenvolvidos diversos métodos de controle sendo o PID (Proporcional-Integrado-Derivado) o mais característico da teoria clássica de controle (OGATA; SEVERO, 1998).

Com a criação dos computadores digitais e o crescimento da complexidade dos sistemas de controle, desenvolveu-se a teoria de controle moderno, que utiliza variáveis de estado para o controle de sistemas com múltiplas entradas, satisfazendo requisitos de aplicações militares, industriais e espaciais.

Ainda segundo Ogata e Severo (1998), atualmente já desenvolveu-se a teoria de controle robusto e os sistemas de controle são implementados em diversos áreas da ciência e da engenharia, desde sistemas robóticos até equipamentos de automação industrial. Portanto, é esperado de um engenheiro o conhecimento de controle automático e a capacidade de implementá-lo em sistemas reais.

O ensino de engenharia, por sua vez, é tradicionalmente centrada nos aspectos conceituais e na transmissão de conhecimento do professor, possuidor do conhecimento, para o aluno, receptor do conhecimento. Esse modelo estabeleceu-se bem no sistema no ensino de massa, entretanto, as novas condições apresentadas pela globalização geraram novas demandas das instituições de ensino. A academia reagiu, procurando novas respostas para as questões de aprendizagem, inserindo a tecnologia no ensino e valorizando novos métodos de ensino (BELHOT; FIGUEIREDO; MALAVÉ, 2001).

Ainda segundo Belhot, Figueiredo e Malavé (2001), foi desenvolvido o modelo de aprendizagem POCE, que constitui um ciclo de ensino com 4 etapas: "Porque"(P), "O quê"(O), "Como"(C) e "E se"(E), cada uma dependente da anterior. Na primeira etapa é apresentada uma contextualização do problema justificando a utilidade do tema ensinado. Na segunda etapa é apresentado o conhecimento usado na solução do problema. Na terceira etapa são desenvolvidas soluções para problemas bem estruturados. Na quarta etapa são feitas novas restrições para o problema que necessitaram de novas soluções. Na terceira etapa ("Como") podem ser implementadas ferramentas computacionais para auxiliar no ensino, geralmente usam-se simulações ou experimentos.

1.1 Descrição do problema

Levando-se em consideração o que foi apresentado anteriormente, nota-se que o desenvolvimento de novos métodos de aplicação prática do conhecimento são fundamentais no aperfeiçoamento do ensino de engenharia.

Portanto este trabalho propõe a criação de uma plataforma que permite implementar vários sistemas de controle para um modelo real. Dessa forma, a plataforma deve fornecer os parâmetros necessários para o funcionamento do modelo e o desenvolvimento de algoritmos de controle, bem como apresentar o comportamento esperado para as respostas do sistema de controle.

Para o modelo do sistema optou-se por um pêndulo invertido sobre duas rodas, pois o pêndulo invertido é um problema clássico na área de controle e vários algoritmos de controle já se mostraram eficientes para este caso, sendo um bom modelo para estudos.

1.2 Objetivos

1.2.1 Objetivo geral

Este trabalho tem como objetivo implementar uma plataforma baseada em um pêndulo invertido sobre duas rodas que permita a implementação de diferentes algoritmos de controle para auxiliar em futuros estudos na área de engenharia de controle.

1.2.2 Objetivos específicos

- Implementar o protocolo de comunicação I2C para leitura dos sensores inerciais.
- Realizar a calibração dos sensores inerciais de forma eficiente para obter dados mais confiáveis.
- Implementar um algoritmo de fusão sensorial para obter uma descrição da orientação espacial do veículo.
- Desenvolver um algoritmo de controle dos atuados por meio dos *PRU*'s da plataforma BeagleBone Black (BBB).
- Realizar a conexão da plataforma de controle (BBB) com os *encoders* dos motores.
- Desenvolver um programa embarcado na plataforma BeagleBone Black que forneça interface para a implementação de diversos sistemas de controle.

1.3 Metodologia

Neste trabalho optou-se por utilizar uma abordagem *top-down*, fragmentando o projeto em etapas, partindo da arquitetura geral até o detalhamento de cada componente. Na etapa de leitura dos sensores inerciais será usado o protocolo I2C e será realizado um processo de calibração dos sensores. O controle dos motores será realizado utilizando as unidade de tempo real da placa BeagleBone Black, que gerarão uma modulação por largura de pulso para acionar os motores e para acompanhar a resposta e corrigir o movimento deles, serão usados *encoders*. A plataforma embarcada será desenvolvida em linguagem C, exceto pelo algoritmo dos motores que será implementado em linguagem *assembly*, mas que deverá ser carregado nas unidades de tempo real através de um código também em linguagem C.

Adotou-se, então, a seguinte metodologia para o desenvolvimento desse trabalho

- Revisão bibliográfica.
- Leitura e calibração dos sensores através de do protocolo I2C.
- Implementação, em linguagem C, do algoritmo de um filtro complementar para obter a orientação dos objeto a partir dos dados dos sensores.
- Testes de validação dos dados obtidos pela filtragem.
- Controle dos motores e leitura dos *encoders* utilizando as unidades de tempo real da BeagleBone Black.
- Testes em bancada para o controle dos atuadores.
- Desenvolvimento de um *software* que integre os sensores e os atuadores em um única plataforma.
- Desenvolvimento da estrutura física do pêndulo invertido sobre duas rodas.
- Testes em bancada para a validação do projeto implementando algoritmos de controle menos sofisticados.

.

1.4 Organização do trabalho

Este trabalho está organizado em 5 partes. No capítulo 1 é apresentado uma contextualição do problema, assim como o objeto e a metodologia. No capítulo 2 é apresentada uma revisão bibliográfica das teorias necessárias para o desenvolvimento deste trabalho.

No capítulo 3 são apresentados os materiais utilizados e suas especificações técnicas, além de uma descrição detalhada da metodologia usada. No capítulo 4 são apresentados e discutidos os resultados deste trabalho. Por fim, no capítulo 5 é apresentada a conclusão deste trabalho.

2 Fundamentação Teórica

2.1 Estado da arte

Sendo o pêndulo invertido um sistema comum na área de controle, já foram propostos vários modelos para um sistema de controle do caso particular do pêndulo invertido sobre duas rodas (TWIP, do inglês *Two-Wheeled inverted pendulum*).

Um das técnicas de controle mais simples é o PID, que consiste na realização de três operações, derivação, integração e proporção, com o sinal de erro e posteriormente o somatório dos sinais de saída dessas operações. Esse método foi implementado e testado por Fagundes (2015) que também fez um comparativo com o regulador linear quadrático (LQR, do inglês *Linear Quadratic Regulator*), utilizando tanto um filtro complementar quanto um filtro de Kalman para estimar os valores de entrada do sistema de controle. A Figura 1 apresenta o projeto desenvolvido para os testes de ambos os controladores.

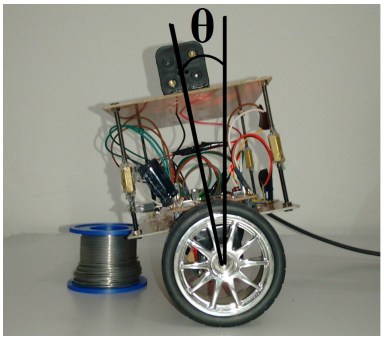


Figura 1 – Projeto de um pêndulos invertido sobre duas rodas para testes de controladores PID e LQR (FAGUNDES, 2015).

Um sistema de controle mais sofisticado é o controle preditivo baseado em modelo (MPC, do inglês *Model-based Predictive Control*). Um protótipo foi desenvolvido por Ohhira e Shimada (2017) para testar esse modelo de controle em um TWID. A Figura 2 apresenta o protótipo desenvolvido.

Outro sistema de controle que pode ser usado para o controle de um TWID é o controle de modo deslizante (SMC, do inglês *Sliding Mode Control*). Simulações foram desenvolvidas por Ha e Lee (2012) para validar a aplicação desse sistema de controle em um TWID.

Observa-se, então, que diferentes sistemas de controle podem ser implementados para um mesmo modelo dinâmico. Dessa forma, tendo uma plataforma que implemente o modelo dinâmico e forneça os estados desse sistema para um controlador, é possível testar diferentes controladores e estudar seus comportamentos.

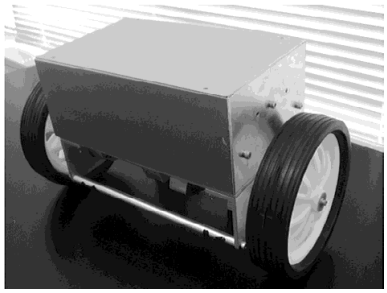


Figura 2 – Protótipo de um pêndulos invertido sobre duas rodas para testes de controladores MPC (OHHIRA; SHIMADA, 2017).

Foi desenvolvida na Universida da Califórnia, em San Diego, (USCD, no inglês *University of California San Diego*) o *Educational Mobile Inverted Pendulum*, ou EduMIP, uma versão comercial de um robô para estudo de sistemas de controle e robótica que segue o modelo do pêndulo invertido sobre duas rodas. Esse robô é utilizado em disciplinas para o ensino de controle embarcado e robótica, além de permitir o ensino de sensoriamento e atuação (BEWLEY, 2017). É compatível também com Python, MATLAB, Simulink e outros. A versão comercial do EduMIP é apresentado na Fig. 3

Figura 3 – EduMIP (BEWLEY, 2017)

2.2 Modelo matemático

A definição de um modelo do sistema dinâmico é a primeira etapa na construção de um sistema de controle. Geralmente o modelo verdadeiro é complexo e é simplificado em um modelo de projeto que preserva as características essenciais do sistema (TEIXEIRA; PIETROBOM; ASSUNÇÃO, 2000).

O modelo de projeto do veículo de duas rodas com base em um pêndulo invertido é apresentado na Fig 4 e foi descrito por Silva et al. (2017).

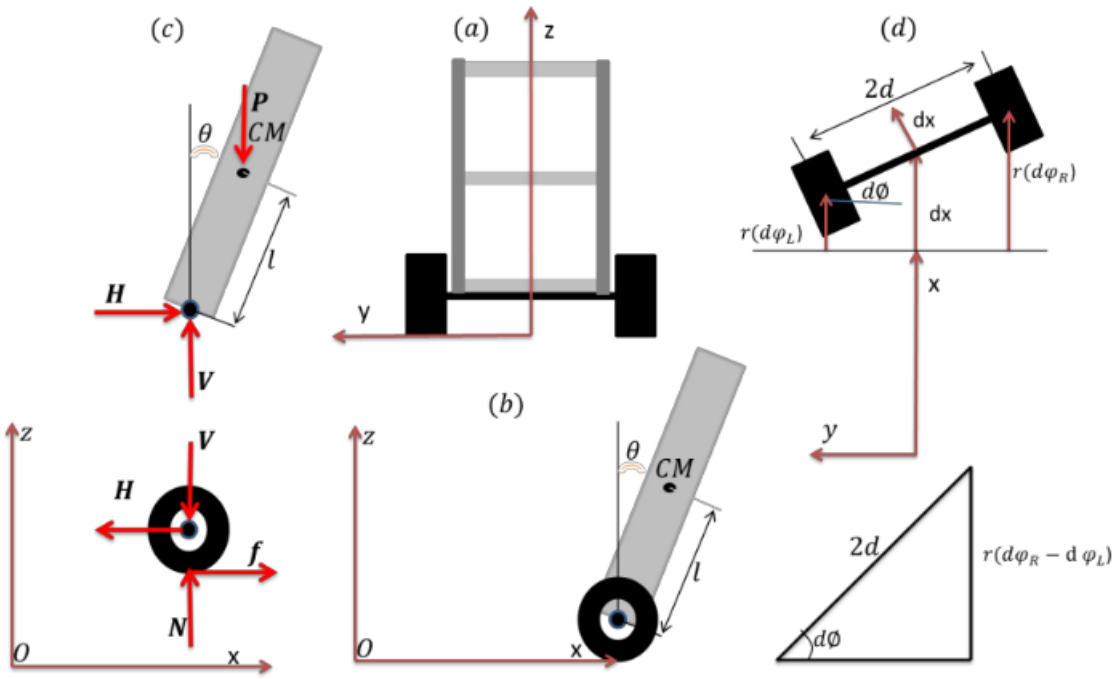


Figura 4 – Representação do veículo com base em um pêndulo invertido sobre duas rodas (SILVA et al., 2017).

Na Figura 4 são apresentados os eixos de referência do sistema, a posição do centro de massa e o diagrama de corpo livre. Para modelar o sistema são consideradas as variáveis e os parâmetros apresentados nas Tab. 1 e 2 como mostra Silva et al. (2017).

Tabela 1 – Variáveis do sistema

Simbolo	**Interpretação física**	**Unidade**
θ	Ângulo de inclinação (*Pitch angle*)	rad
x	Posição do veículo	m
ψ	Ângulo de rotação em torno do eixo z	rad
φ_R	Ângulo de rotação da roda direita	rad
φ_L	Ângulo de rotação da roda esquerda	rad
T_R	Torque do motor direito	Nm
T_L	Torque do motor esquerdo	Nm
μ_R	Ação de controle do motor direito	Adimensional
μ_L	Ação de controle do motor esquerdo	Adimensional
f	Força resultando do veículo	N
H	Força de contato horizontal no eixo das rodas	N
V	Força de contado vertical no eixo das rodas	N
N	Força normal	N
P	Peso do veículo	N

Da Figura 4 também é possível obter as Eq. 2.1 e 2.2 que descrevem o movimento

Tabela 2 – Parâmetros do sistema

Simbolo	**Interpretação física**	**Valor**(Unidade)
J_p	Momento de inércia do veículo	48×10^{-4} (kg·m^2)
J_Z	Momento de inércia do veículo no eixo Z	7.18×10^{-3} (kg·m^2)
J_W	Momento de inércia das rodas, engrenagem e eixo	3.1×10^{-5} (kg·m^2)
g	Aceleração da gravidade	9.79 (m·s^{-2})
r	Raio das rodas	0.0316 (m)
M	Massa das duas rodas	0.062 (Kg)
m	Massa do pêndulo	1.304 (Kg)
d	Metade da distância entre as duas rodas	0.115 (m)
l	Posição do centro de massa	0.0748 (m)
C_a	Atrito dinâmico no eixo das rodas	0.0005 (kg·m^2)
b	Atrido dinâmico do motor	0.002 (kg·m^2)
V_{cc}	Força eletromotriz máximo no motor	6 (V)
K_m	Constante do torque do motor	0.5373 (N·m·A^{-1}))
K_e	Constante da força contra-eletromotriz	0.34 (V·s)
R	Resistência do motor	0.9246 (Ω)
dz	Zona morta do motor	0.03 (adimensional)

do veículo no eixo x e o ângulo de rotação do eixo z (ψ) respectivamente.

$$x = \frac{r}{2}(\varphi_R + \varphi_L) \tag{2.1}$$

$$\psi = \frac{r}{2d}(\varphi_R - \varphi_L) \tag{2.2}$$

A Equação 2.3 descreve o comportamento das forças no eixo x

$$m\frac{d^2}{dt^2}(x + l \cdot \sin\theta) = H \tag{2.3}$$

A Equação 2.4 apresenta a relação da força no eixo vertical.

$$m\frac{d^2}{dt^2}(l \cdot \cos\theta) = v - m \cdot g \tag{2.4}$$

A Equação 2.5 descreve o movimento de rotação do veículo em relação ao eixo y, ou seja, sua inclinação.

$$J_P\frac{d^2\theta}{dt^2} = V \cdot l\sin\theta - H \cdot l\cos\theta - C_a \cdot (\dot{\theta} - \dot{\varphi}_R) - C_a \cdot (\dot{\theta} - \dot{\varphi}_L) \tag{2.5}$$

As Equações 2.3 a 2.5 descreve os efeitos das forças no centro de massa do veículo. O comportamento das forças na base do veículo, ou seja no eixo da roda, é apresentado

na Eq. 2.6.

$$M\frac{d^2x}{dt^2} = f - H \tag{2.6}$$

A força que os motores transmitem para o veículo é apresentada na Eq. 2.7

$$f = \frac{1}{r}(T_R - b\dot{\theta}_R - J_W\ddot{\theta_R}) + \frac{1}{r}(T_L - b\dot{\theta}_L - J_W\ddot{\theta_L}) \tag{2.7}$$

Na Equação 2.7 os efeitos do atrito dinâmico e do momento de inercia podem ser desprezados, assim a equação é reduzida para a Eq. 2.8

$$f^* = \frac{1}{r}(T_R + T_L) \tag{2.8}$$

Por fim, são derivadas as equações que modelam a posição, a inclinação (Eq. 2.9 e 2.10) e a rotação do veículo (Eq. 2.11).

$$(ml\cos\theta)\ddot{\theta} + \left(M + m + \frac{2J_W}{r^2}\right)\ddot{x} = f^* - \left(\frac{2b}{r^2}\right)\dot{x} + (ml\sin\theta)\dot{\theta} \tag{2.9}$$

$$(J_P + ml^2)\ddot{\theta} + (ml\cos\theta)\ddot{x} = mlg\sin\theta - 2C_a\dot{\theta} - \left(\frac{2C_a}{r}\right)\dot{x} \tag{2.10}$$

$$\left(J_z + 2\left(\frac{d}{r}\right)^2\right)\ddot{\psi} = \left(\frac{d}{r^2}\right)(T_R - T_L) - 2b\left(\frac{d}{r^2}\right)\dot{\psi} \tag{2.11}$$

Observa-se que as variáveis necessárias para descrever o modelo dinâmico do sistema são a posição, a inclinação e a rotação do veículo, além de suas derivadas. Portanto, para estimar a inclinação do veículo, serão usados sensores inerciais, a posição será calcular de acordo com leitura dos *encoders* e a rotação será definida pela combinação dos sensores e dos *encoders*.

2.3 Sensoriamento inercial

Para equilibrar um pêndulo invertido é necessário conhecer qual orientação o mantem em equilibrio e qual a sua orientação de fato. A relação entre a orientação do pêndulo e a orientação de referência é chamada de atitude do objeto e pode ser descrita de diversar formas, como ângulos de Euler, matrizes de rotação e quatérnios. Para conhecer a atitude de um objeto são usados sensores inercias (SILVA, 2016). Uma unidade de medida inercial, ou *IMU* (*Inertial Measurement Unit*) é composta de alguns sensores inerciais, geralmente giroscópio e acelerômetro, mas também pode incluir magnetômetro, e com o devido tratamento dos dados é capaz de fornecer a orientação de um objeto (GONÇALVES, 2017).

2.3.1 Giroscópio

O giroscópio é responsável por medir a velocidade angular do objeto e funciona com base no efeito Coriolis. Nos modelos comerciais, como apresentado na Fig 5, uma massa de prova mantém-se vibrando por um estímulo eletroestático ou eletromagnético criando um velocidade no sentido da vibração. Pelo efeito Coriolis, quando um objeto com velocidade v sofre uma rotação é gerada uma força perpendicular a velocidade, a força de Coriolis, como mostra a Eq. 2.12 onde v é a velocidade linear, m é a massa e Ω é a velocidade angular. A velocidade angular é calculada pela integração da aceleração de Coriolis, que é obtida pela 2ª Lei de Newton (Eq. 2.14) e é apresentada na Eq. 2.13 (FORHAN, 2010).

$$F_c = 2mv \times \Omega \tag{2.12}$$

$$a_c = 2v \times \Omega \tag{2.13}$$

Portanto, quando as estruturas vibrantes do giroscópio sofrem uma rotação, essas estruturas são deslocadas no eixo perpendicular alterando a capacitância entre os eletrodos.

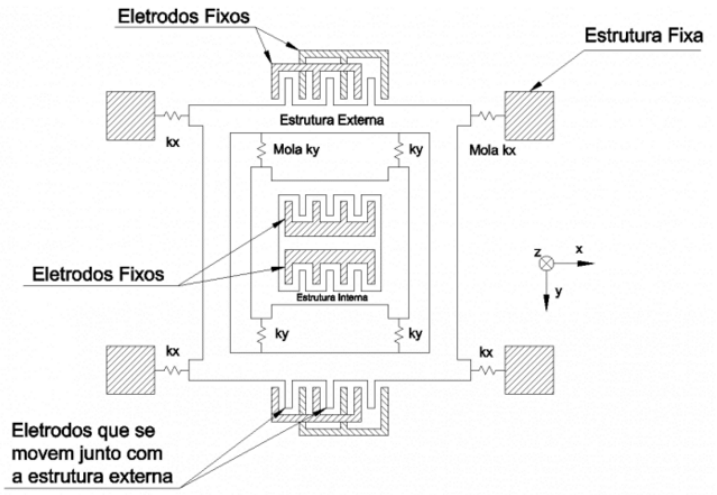


Figura 5 – Estrutura do giroscópio (TORRES, 2015)

O giroscópio apresenta uma deficiência, um efeito chamado *drift* (ou deriva) que ocorre devido ao processo de integração dos dados para obter a posição angular e é caracterizado por uma progressão dos valores mesmo com o sensor imóvel (VIDA, 2016).

2.3.2 Acelerômetro

O acelerômetro é um dispositivo capaz de medir a aceleração própria de um objeto com base na 2ª Lei de Newton (Eq. 2.14) e na Lei de Hooke (Eq. 2.15). O acelerômetro

é constituído de um sistema massa-mola que quando sofre uma aceleração provoca um variação na posição da mola em relação ao referencial fixo do sensor. A variação da posição é medida de forma capacitiva ou piezo-resistiva. A relação entre a aceleração e o deslocamento da massa é mostrado na Eq. 2.16 (GONÇALVES, 2017).

$$F = m \cdot a \tag{2.14}$$

$$F_{el} = k \cdot x \tag{2.15}$$

$$a = \frac{k \cdot x}{m} \tag{2.16}$$

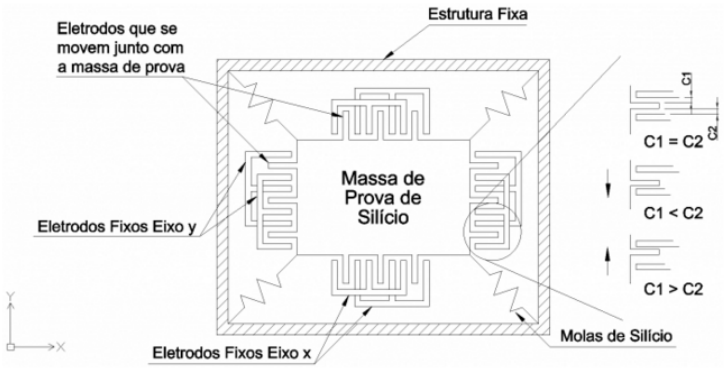


Figura 6 – Estrutura do acelerômetro (TORRES, 2015)

Devido a sua natureza, o acelerômetro também apresenta deficiências, sendo a mais característica a presença de ruídos causados por pequenas trepidações (VIDA, 2016).

2.3.3 Magnetômetro

Magnetômetros são dispositivos capazes de medir campos magnéticos com base no efeito Hall. O efeito Hall descreve a geração de uma tensão elétrica, a Tensão Hall, devido a deflexão das cargas elétricas em um condutor de corrente elétrica na presença de um campo magnético (GONÇALVES, 2017). O Efeito Hall é apresentado na Fig. 7.

O magnetômetro é afetado por dois tipos de interferência, o *hard iron* e o *soft iron*. O primeiro é causado por campo magnéticos aditivos e invariantes no tempo e e gera um erro constante, similar ao *offset*. O segundo é causado por materiais magnéticos capazes de alterar a magnitude ou a direção dos campos magnéticos e gera erro que dependem da orientação do material (GONÇALVES, 2017).

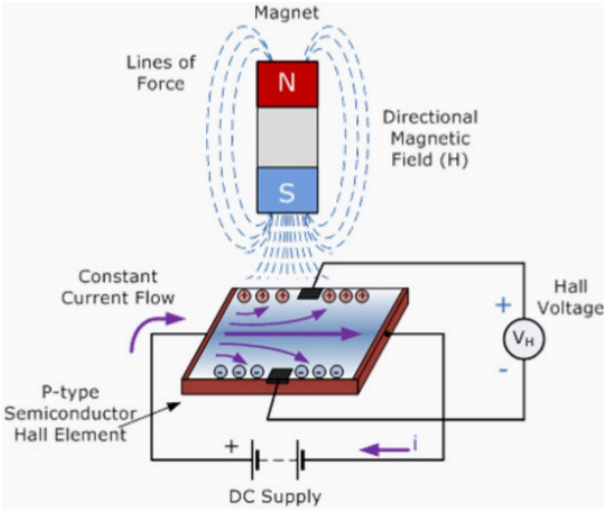


Figura 7 – Efeito Hall (GONÇALVES, 2017)

2.4 Representação de atitude

2.4.1 Ângulos de Euler

Uma das forma de representar a orientação de um objeto é através dos ângulos de Euler. Nessa representação são definidos dois sistemas ortogonais independentes, um fixo e um móvel, o primeiro associado ao ambiente e o segundo associado ao objeto, de modo que a orientação do objeto pode ser descrita como uma sequência de rotações do sistema móvel em relação ao sistema fixo (ALMEIDA, 2014).

A sequência dos ângulos de Euler são definidos na literatura como:

- *Roll*: ângulo de rotação em torno do eixo longitudinal, o eixo X, representado pela letra grega ϕ
- *Pitch*: ângulo de rotação em torno do eixo lateral, o eixo Y, representado pela letra grega θ
- *Yaw*: ângulo de rotação em torno do eixo vertical, o eixo Z, representado pela letra grega ψ

Os ãngulos de Euler são representados na Fig. 8 onde é possível observar o sentido de rotação em relação as eixos do objeto.

2.5 Fusão sensorial

Como visto na seção 2.3, cada um dos sensores apresenta um tipo de erro. O acelerômetro capta muitos ruídos de alta frequência, como trepidações, o giroscópio apresenta um efeito de *drift* e o magnetômetro apresenta os efeitos de *hard iron* e *soft iron*.

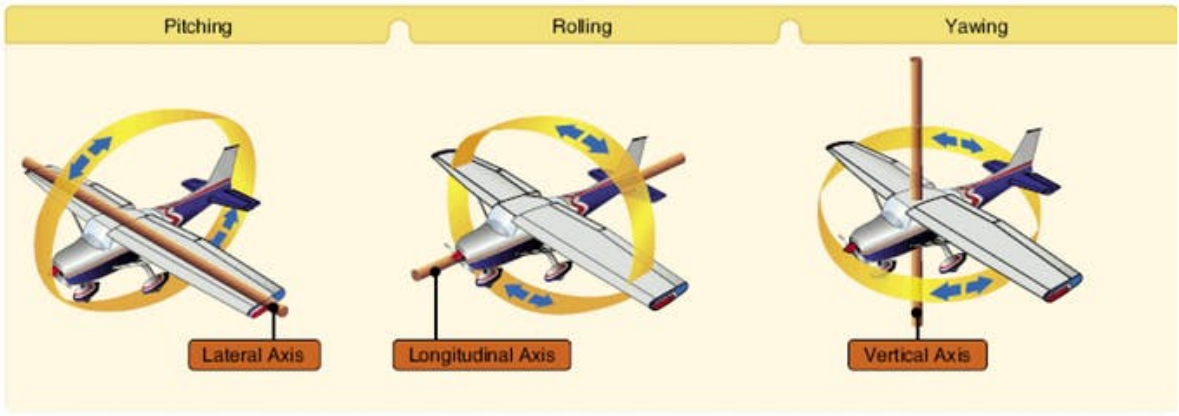


Figura 8 – Representação dos ângulos de Euler (PAUL; GANGULY, 2016)

Entretanto é possível combinar os sensores através da fusão sensorial e obter dados mais confiáveis. Nesta seção será apresentado o método de fusão sensorial usando um filtro complementar.

2.5.1 Filtro complementar

O fitro complementa consiste de um filtro passa-baixas aplicado ao acelerômetro e a um filtro passa-altas aplicado ao giroscópio como mostra a Fig. 9.

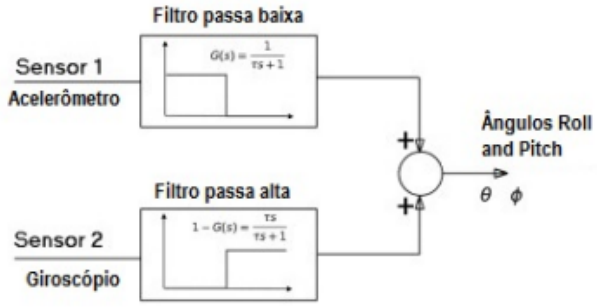


Figura 9 – Filtro complementar

Dessa forma, é possível evitar os ruídos de trepidações do objeto captados pela acelerômetro e evitar o efeito de *drift* que ocorre no giroscópio. Entretanto esse método conserva as características ruidosas dos dados brutos dos sensores, logo os dados permanecem não confiáveis, além de tornar o sistema lento (VIDA, 2016).

2.6 Atuadores

Atuadores são definidos como dispositivos que convertem energia em movimento, sendo os mais comuns os motores DC, os servos-motores e os motores de passos, e são usa-

dos para diversas aplicações desde impressoras 3D até automação residencial (MOLLOY, 2014).

2.6.1 Motor DC

Segundo Basilio e Moreira (2001) os motores de corrente contínua têm significativa importância na área de controle por serem usados no desenvolvimento de servomecanismos. Ele podem ser abstraídos para um modelo de primeira ordem que tem como entrada a tensão de armadura e como saída a velocidade angular, sendo o suficiente para projetos de controladores.

Nos motores DC, o torque é geralmente proporcional a corrente elétrica e como esses motores podem gerar muito torque, é necessário um circuito de alimentação apropriado. Segundo Molloy (2014), esse modelo é controlado por um sinal modulado por largura de pulse (PWM, do inglês *Pulse Width Modulation*) e são usados *encoders* para implementar um sistema de malha fechada.

2.6.2 Encoder

Encoders são dispositivos eletrônicos, geralmente ópticos, que permitem detectar a variação do movimento de um motor. Estruturalmente, são formados por um emissor de luz, um sensor de luminosidade e um disco com seções opacas e translúcidas alternada, dessa forma, a variação de luminosidade indicada pelo sensor fornece uma medida de movimento do disco. A Figura 10 apresenta a estrutura e os sinais apresentados por um *encoder* simples (RODRIGUES, 2017).

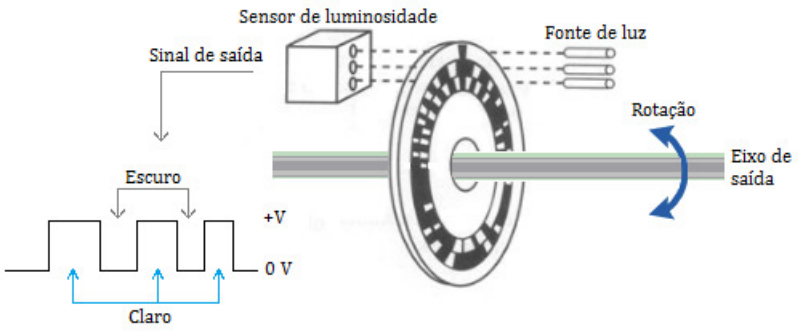


Figura 10 – Estrutura e funcionamento de *encoders* simples (RODRIGUES, 2017).

Encoders simples apresentam um canal e fornecem apenas uma medida da quantidade de movimento, para obter também o sentido de rotação do motor usa-se um *encoder* em quadratura que apresenta dois canais defasados em 90°. A Figura 11 apresenta os sinais de saída de *encoder* em quadratura durante uma mudança no sentido de rotação.

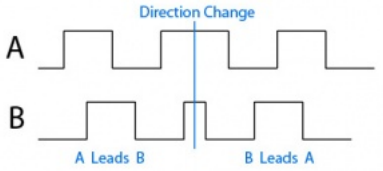


Figura 11 – Sinais de saída de *encoders* em quadratura (PHIDGETS, 2017)

Os modelos de *encoder* que fornecem apenas o sentido de rotação e a a variação de movimento são chamados de incrementais, mas há também o modelo absoluto que fornece a posição do eixo do motor. A Figura 12 apresenta um comparativo entre os modelos de *encoders* incremental em quadratura e absoluto com código de Grey (PHIDGETS, 2017).

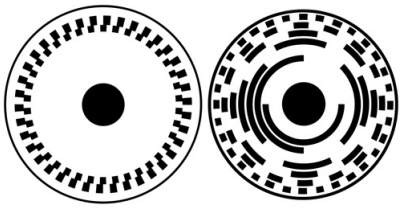


Figura 12 – Diagramas dos *encoder* incremental (a esquerda) e absoluto (a direita) (PHIDGETS, 2017).

Portanto, é possivel obter o deslocamento angular pelo número de pulso gerados e derivando esse valor obtêm-se a velocidade angular. Pela análise das fases identifica-se o sentido de rotação também (FAGUNDES, 2015).

2.7 Plataforma Computacional

Atualmente os sistemas de controle são implementados em computadores digitais, como microcontroladores. Esses sistemas de controle são chamados de modelos discretos e necessitam de uma amostragem dos valores de saída para realimentar o controlador. Essa amostragem é feita de duas formas, uma através de um sinal de interrupção de acordo com uma determinada taxa de amostragem f_{sample} ou a cada ciclo de execução de código. No primeiro caso, a taxa de amostragem é fixa e, no segundo caso, a taxa pode variar de acordo com os desvios lógicos do código. (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

Os microcontroladores são constituídos de um microprocessador, memória de dados, memória de programas, pinos de entrada e saída de dados e um conjunto de periféricos, como barramento de comunicação serial, controlador de interrupção, temporizadores, geradores de PWM, entre outros. Apesar do conjunto de recursos uteís na áres de controle, como os controladores de interrupção e os temporizadores, os microcontroladores possuem processadores com velocidade de processamento de poucos *megahertz* (MARTINS, 2005).

Como sugerido por (FAGUNDES, 2015), que implementou um controlador LQR com filtro de Kalman em um processador de 80MHz a uma taxa de amostragem de 500Hz, a implementação de algoritmos mais sofisticados demanda um esforço computacional maior e necessita de um *hardware* com capacidade de processamento maior.

Portanto, para o desenvolvimento de um sistema compatível com diversos tipos de controladores é necessário um processador que suporte algoritmos complexos e que ofereça os componentes de *hardware* necessários. No mercado, há algumas plataformas que cumprem os requisitos deste trabalho, como a placa BeagleBone Black.

2.8 Protocolo I^2C

O interface I^2C (*Inter-Integrated Circuit*) é a forma mais comum de comunicação serial, sendo muito usado em aplicações com microcontroladores, sensores e entre outros dispositivos eletrônicos (CAMARA, 2013). A comunicação com I^2C é feita através de 2 sinais, o SCL (*serial clock*), responsável pela transmissão do sinal de *clock* e, consequentemente, pela sincronização dos dispositivos, e o SDA (*serial data*), responsável pela transmissão dos dados. A comunicação é feita entre dois tipos de dispositivos, mestre (ou *master* e escravo (ou *slave*). O mestre é a unidade de controle que transmite o *clock*, solicita os dados e define com qual escravo será feita a comunicação. O escravo é o dispositivo que responde as solicitações do mestre (LONGARETTI; GIRARDI; ENGROFF, 2016). Essa forma de comunicação é vantajosa, pois permite a transmissão de dados com vários escravos em um mesmo barramento como é apresentado na Fig. 13.

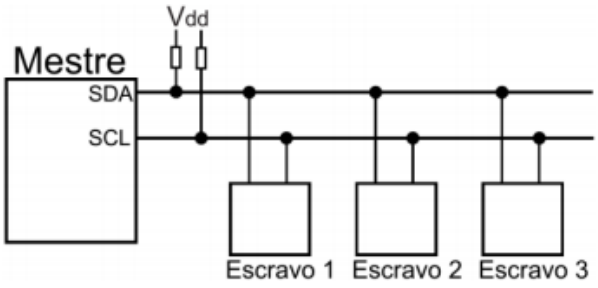


Figura 13 – Representação da comunicação I2C (LONGARETTI; GIRARDI; ENGROFF, 2016)

2.9 Modulação por largura de pulso

A modulação por largura de pulso (ou PWM, do inglês *Pulse Width Modulation*) é uma forma de controlar a tensão de um sinal através da variação, ou modulação, do ciclo ativo desse sinal, ou seja, tendo uma fonte de tensão constante é possível variar a

potência média, e a tensão consequentimente, variando os períodos em que se aplica ou não a tensão constante.(SILVA, 2008).

Definindo-se uma frequência para a aplicação de tensão na carga, a tensão média aplicada será proporcional a porcentagem do período em que há tensão na carga (SILVA, 2008). A figura 14 apresenta o funcionamento do PWM e o cálculo do ciclo ativo dele. Além disso, essa forma de controlar a potência média transferida para uma carga é muito eficiente para o acionamento de motores DC.

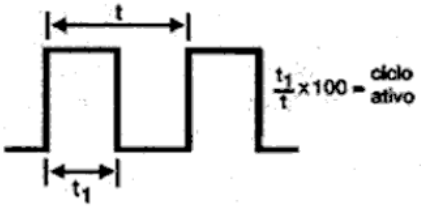


Figura 14 – Definição de ciclo ativo do PWM (SILVA, 2008)

2.10 Controlador PID

Segundo Ogata e Severo (1998), o controle PID apresenta uma aplicabilidade a maioria dos sistemas de controle, estando presente em mais da metado dos controladores industriais em uso atualmente. O PID também se mostra útil em sistemas com o modelo matemático da planta desconhecido. Apesar dos controladores PID apresentarem um controle satisfatório, em muitas situações ele não oferece um controle ótimo do sistema. Quanto ao ajuste da sintonia do PID, na maioria dos casos ele é ajustado em campo e existem variadas regras de sintonia, entretanto, já existem métodos de ajustes automático.

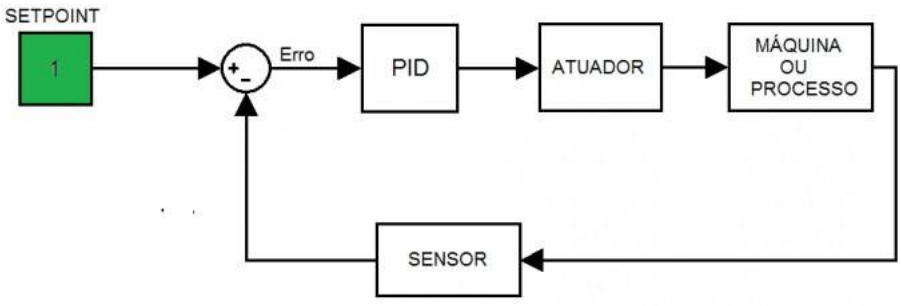


Figura 15 – Diagrama de blocos de um sistemas de controle com PID (FREITAS, 2014)

O controlador PID pode ser descrito pela Eq. 2.17 (MUNIZ, 2017). Cada parcela da equação do PID é responsável por efeitos característicos no sinal de saída do controlador. A primeira parcela é proporcional ao erro entre o sinal de entrada e o sinal de saída e é ajustado pelo ganho K_p. Quanto maior o ganho proporcional mais rápido é o tempo de

resposta e menor o erro do sinal, porém o erro nunca é corrigido e há um aumento do sobressinal. A segunda parcela é proporcional a integral do erro e é ajustado pelo ganho K_i. O aumento desse ganho corrige o erro em regime estacionário, porém também causa um aumento do sobressinal em regime transitório. A terceira parcela é proporcional a derivado do erro e é ajustado pelo ganho K_d. O aumento desse ganho reduz o sobressinal e diminui um pouco o tempo de resposta, porém não apresenta bons resultados em sistemas com sinal ruidoso.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{2.17}$$

3 Materiais e Métodos

Este capítulo apresenta as especificações técnicas de todos os materiais utilizados no projeto, bem como os detalhes das metodologias de desenvolvimento.

3.1 Materiais

3.1.1 Placa BeagleBone Black Wireless

A BeagleBone Black (BBB) é uma plataforma que permite desenvolver projetos tanto em alto nível (*software*) quanto em baixo nível (*hardware*). Tendo um sistema operacional Debian GNU/Linux 8 (ou Debian Jessie) embarcado ele permite o uso de diversos *softwares* de código aberto e mais acesso aos processos do sistema operacional. Com a BBB ainda é possível acessar os recursos de (*hardware*) facilmente, tendo a disposição diversos pinos de propósito geral (GPIO, do inglês *General Purpose Input/Output*) com diversas funções de comunicação e duas unidades de tempo real (PRUs, do inglês *Programmable Real-time Units*). A Tabela 3 apresenta algumas das especificações da BeagleBone Black (MOLLOY, 2014).

Tabela 3 – Especificações do BeagleBone Black

Preço	US$45–55
Processador	1 GHz AM335x
OS	Debain Jessie
Memória	512 MB DDR3
Armazenamento	4 GB eMMC
Vídeo	HDMI
Entradas/saídas	92 GPIOs

A BBBW também conta com os seguintes recursos:

- 2 unidades programáveis de tempo real
- 3 barramentos de comunicação I2C
- 4 barramentos de comunicação UART
- 2 barramentos de comunicação SPI
- Pinos de alimentação de 5V e 3.3V
- Interface para configuração do Wireless

- Suporte a linguagens de alto nível como Python
- Ambiente de desenvolvimento integrado via *web browser* Cloud9

Figura 16 – BeagleBone Black Wireless

3.1.1.1 *Real-Time* com a BeagleBone Black

Uma das grandes vantagens da BBB é a união de um sistema operacional Linux com um *hardware* cheio de recursos. Entretanto, um SO Linux é, por padrão, não preemptivo, ou seja, é possível criar apenas aplicações *soft real-time* (MOLLOY, 2014). Para ampliar as possibilidades de projetos para desenvolvedores, a BBB apresenta alguns recursos para desenvolver sistemas de tempo real. Hamilton (2015) apresenta um conjunto maior de soluções para sistemas de tempo real, como listado a seguir.

- RT_PREEMPT
- Xenomai
- PRU's
- PRU's + Xenomai

O *RT_PREEMPT* é um recurso que permite modificar o *Kernel* no Linux para configurar a preemptividade dele, assim, tarefas com menor prioridade podem superar as de maior prioridade para cumprir as demandas de tempo. *Xenomai* é uma ferramenta que permite criar um *kernel* secundário para as demandas de tempo real (HAMILTON, 2015). Os PRU's são os recursos usados neste trabalhos e serão descritos adiante.

Ainda segundo Hamilton (2015) cada um desses recursos apresenta um tempo de resposta próprio mesmos sendo todos de tempo real. A tabela 4 apresenta uma comparação entre os tempos de latência de cada recurso.

Tabela 4 – Especificações de latência

Sistema	Latência
RT_PREEMPT	200 ms
Xenomai	20 ms
PRU's	5 ns

3.1.1.2 Unidades Programáveis de Tempo Real

As Unidades Programáveis de Tempo Real e o Subsistema de Controlador Industrial (ou PRU-ICSS, do inglês *Programmable Real-Time Units And Industrial Controller Subsystem*), ou PRU's, são 2 microcontroladores de 32 *bits* que operam a 200 Mhz. É importante informar que os PRU's são um *hardware* separado da CPU principal, portanto não são afetados pela preempitividade do sistema operacional ou do *kernel*. Como vantagem, os PRU's apresentam mais eficiência no tratamento de eventos em tempo real e no controle dos pinos de entrada e saída de sinais, sendo classificado por Hamilton (2015) como *100 percent hard real-time*.

Entretanto, como os PRU's estão separados da CPU principal, é necessário realizar algumas modificações nas configurações da BBB para habilitá-los e para dar-lhes acesso ao GPIO (HAMILTON, 2015).

Para facilitar o uso dos PRU's foi desenvolvido o *PRU Speak* através do projeto *Google Summer of Code* de 2014. Ele é constituído de três partes: um módulo kernel para carregar os códigos nos PRU's e manter a comunicação com eles (*pru_speak.ko*), arquivos de *firmware* (*pru0_firmware* e *pru1_firmware*), e bibliotecas de *Python* para desenvolver códigos com *scripts* em *Python* (*bs_tcp_server.py* e *pru_speak.py*). O módulo *kernel* trabalho em associação com a *Device Tree Overlay* para conceder aos PRU's acesso aos pinos de I/O. (YODER; KRIDNER, 2015)

3.1.2 Motor DC 6V 210RPM com Encoder

A Figura 17 apresenta o modelo do motor que é usado neste trabalho e a seguir não apresentadas suas especificações técnicas (FILIPEFLOP,).

- Tensão: 6V DC
- Velocidade: 210 RPM
- Diâmetro do eixo: 4mm
- Comprimento do eixo: 12mm
- Máxima eficiência: 2.0kg.cm/170rpm/2.0W/0.60A

- Máximo torque: 5.2kg.cm/110rpm/3.1W/1.10A
- Cabo de conexão: 6 fios
- Comprimento do cabo: 20 cm
- Comprimento total motor e eixo: 73mm
- Encoder na placa
- Largura da roda: 27mm
- Diâmetro da roda: 65mm

Figura 17 – Modelo comercial do motor DC 6V 210RPM com Encoder

Também é apresentada a pinagem do motor na Fig. 18.

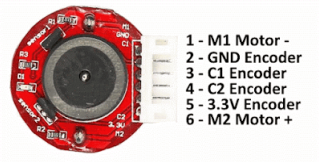


Figura 18 – Pinagem do motor DC 6V 210RPM com Encoder

3.1.3 Driver TB6612FNG

Como os motores DC operam com correntes superiores as permitidas pela BBB, será necessário uma alimentação separada e um circuito de controle dos motores chamado ponte H. Para o circuito será usado um circuito integrado (ou CI) TB6612FNG, que permite o controle de até 2 motores DC. A Tabela 5 apresenta algumas especificações técnicas do CI.

É apresentado na Fig. 19 o funcionamento do CI. Os sinais *IN1* e *IN2* são sinais lógicos usados para definir o sentido de rotação dos motores, podendo ser no sentido horário (ou CW, do inglês *clockwise*) ou anti horário (ou CCW, do inglês *counter-clockwise*).

Tabela 5 – Especificações do TB6612FNG (TOSHIBA, 2008)

Simbolo	Parâmetro	Valor	Unidade
V_M	Alimentação	até 15	V
V_{CC}	Alimentação lógica	até 6	V
V_{IN}	Voltage de entrada	-0.2 até 6	V
V_{OUT}	Tensão de saída	15	V
I_O	Corrente de operação DC	1.2 (por canal)	A
T_{op}	Temperatura de operação	-20 a 85	°C

Input				Output		
IN1	IN2	PWM	STBY	OUT1	OUT2	Mode
H	H	H/L	H	L	L	Short brake
L	H	H	H	L	H	CCW
		L	H	L	L	Short brake
H	L	H	H	H	L	CW
		L	H	L	L	Short brake
L	L	H	H	OFF (High impedance)		Stop
H/L	H/L	H/L	L	OFF (High impedance)		Standby

Figura 19 – Tabela verdade do CI TB6612FNG (TOSHIBA, 2008)

O sinal *PWM* controla a velocidade dos motores e os sinais *OUT1* e *OUT2* alimentam os motores.

A Figura 20 apresenta o módulo usado no projeto que contém o *driver* para os motores. Este circuito, cedido pelo LEIA, foi testado antes de ser usado no projeto e apresentou o comportamento esperado.

Figura 20 – Circuito de controle dos motores (CLUSTERBOT!,)

3.1.4 Unidade de medida inercial

O sensor usado neste trabalho é o módulo MPU9250 da empresa InvenSense (Fig. 21) constituído de dois *chips*, um contém o acelerômetro e o giroscópio e o outro contém o magnetômetro, todos sensores triaxiais, e utiliza um barramento de comunicação I^2C para

a transmissão dos dados. O MPU9250 é um sensor de baixo custo fabricado com tecnologia MEMS (*Micro-Electro-Mechanical System*) (SILVA, 2016). Algumas especificações dos sensores são apresentados nas Tab. 6 a 8 (INVENSENSE, 2016).

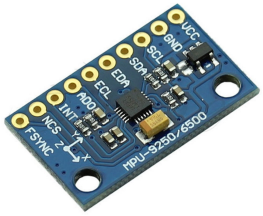


Figura 21 – Modelo do sensor MPU9250

Tabela 6 – Especificações do giroscópio

Caracterísitca	**Valor**
Escala	±250,±500,±1000,±2000 (°/s)
Sinal de saída	16 bits
Corrente de operação	3.2 mA

Tabela 7 – Especificações do acelerômetro

Caracterísitca	**Valor**
Escala	±2,±4, ±8,±16 (g)
Sinal de saída	16 bits
Corrente de operação	450 μA

Tabela 8 – Especificações do magnetômetro

Caracterísitca	**Valor**
Escala	±4800 (μT)
Sinal de saída	14 ou 16 bits
Corrente de operação	280 μA

3.2 Métodos

Este trabalho será realizado utilizando a plataforma computacional BeagleBone Black Wireless que oferece a capacidade de preocessamento necessária para algoritmos de controle sofisticados e os recursos de *hardware* necessários. A orientação do objeto será calculada utilizando um algoritmo desenvolvido em lingugem C para o filtro de Kalman. As medições usadas no filtro serão feitos com o sensor MPU9250, que deverá ser calibrado. A leitura do sensor é feita usando o protocolo de comunicação I^2C já configurado no GPIO

da BBB. Os motores serão controlados usando PWM gerados pelo GPIO. E o controle do motor seá feito usando os PRU's para a leitura dos *encoders*. A alimentação dos motores será feita usando uma ponte H.

3.2.1 Leitura dos sensores

A leitura do módulo MPU9250 foi realizada utilizando o barramento I^2C, sendo que a placa BeagleBone já possui pinos configurados para as funções de transmissão de dados (SDA) e sincronização (SCL) da comunicação I2C. Também foram usados os pinos do GPIO para alimentar o módulo com 3.3V. Os pinos do GPIO configurados como barramento I^2C são apresentados na Tab 9.

Tabela 9 – Pinagem para a IMU

Pino (BBB)	Modo de operação	Pino (MPU9250)
P9_19	I2C2_SCL	SCL
P9_20	I2C2_SDA	SDA

Como a BeagleBone Black possui um sistema operacional Linux embarcado, foram utilizadas algumas ferramentas desse ambiente para a comunicação I^2C, que incluem as bibliotecas *i2c.h* e *i2c-dev.h*. Como nos sistemas Linux a comunicação I^2C é feito através da manipulação de arquivos também foi usada a biblioteca *ioctl.h*.

No módulo MPU9250, o magnetômetro é encapsulo separamente, logo a comunicação é feita com dois dispositivos escravos diferentes e é necessário informar os endereços destes dispositivos antes da leitura ou da escrita na comunicação I^2C, entretanto a comunicação só pode ser feita com um dispositivo por vez, portanto o endereçamento deve ser feito sempre que se muda de dispositivo. Além disso, os dados de cada eixo de cada sensor são armazenados em dois registradores separados, logo os dados devem ser montados para serem usados. Para resolver isso foram implementadas funções de leitura para cada sensor que indica o dispositivo escravo e organiza os dados lidos internamente.

Inicialmente será feita uma calibração dos sensores e, posteriormente, os dados obtidos com a leitura dos sensores serão usados para estimar a orientação do objeto.

3.2.2 Calibração dos sensores

Na fabricação de sensores de baixo custo, como dispositivos MEMS, não é comum que a calibração seja feita pelo fabricante por ser um processo demorado e com custo alto. Portanto, em aplicações que demandam maior precisão nas medições, é necessário que seja feita a calibração do sensor (SILVA, 2016).

3.2.2.1 Calibração do giroscópio e do acelerômetro

Para calibrar esses sensores é necessário calcular o valor do *offset* (ou *bias*) de cada um dos eixos de ambos os sensores. Optou-se por usar a técnica apresentada por Gonçalves (2017) que consiste em calcular o valor médio de um número significativo de amostras como indica a Eq. 3.1.

$$offset_{x,y,z} = \frac{1}{100}\sum_{i=1}^{100} x_i \tag{3.1}$$

3.2.2.2 Calibração do magnetómetro

A calibração do magnetômetro é um processo mais complexo uma vez que esse sensor é mais afetado por ruídos aditivos causado por campos magnéticos gerados por materiais elétricos e magnéticos próximos (GONÇALVES, 2017). Para o magnetômetro é necessário calcular um ajuste de escala e um *offset*. Esta calibração é realizada em duas etapas.

1. O eixo z deve ser posicionado perpencicular a superfície e rotacionado 360° em torno do eixo z, assim são coletados os valores máximos e mínimos nos eixo x e y

2. O eixo y deve ser posicionado perpendicular a superfície e rotaciona-se o sensor em torno desse eixo, assim são coletados os valores máximos e mínimos no eixo z

Obtidos os valores máximos e mínimos, são calculados a escalas e o *offset* de cada eixo como mostram as Eq. 3.2 e 3.3, respectivamente.

$$escala_{x,y,z} = \frac{max_{x,y,z} - min_{x,y,z}}{2} \tag{3.2}$$

$$offset_{x,y,z} = \frac{max_{x,y,z} + min_{x,y,z}}{2} \tag{3.3}$$

Para obter o valor do ajuste de escala de cada eixo é necessário calcular um valor de escala média (Eq. 3.4) e então calcular o ajuste pela Eq. 3.5.

$$escala_{média} = \frac{escala_x + escala_y + escala_z}{3} \tag{3.4}$$

$$ajuste_{x,y,z} = \frac{escala_{media}}{escala_{x,y,z}} \tag{3.5}$$

3.2.3 Representação da orientação

Os dados obtidos pelo sensor não representam as variáveis usadas no modelo matemático, mas, sim, medições de fenômenos físicos. Para obter os valores das variáveis do modelo é necessário uma etapa de cálculo dessas variáveis. Nesta seção serão apresentados os métodos usados inicialmente, que calculam a atitude do objeto usando ângulos de Euler. Posteriormente esses valores de ângulo podem ser usados para obter quatérnios e matrizes de rotação, como apresentado na subseção **??**.

3.2.3.1 Representação com giroscópio

Como o giroscópio mede a velocidade angular, é necessário integrar o sinal lido para obter a posição angular (PAULA, 2015b). A Equação 3.6 apresenta a relação entre os ângulos de Euler e as velocidades angulares medidas pelo giroscópio, onde p, q e r são as medidas do sensor.

$$\begin{aligned} \phi_k &= \phi_{k-1} + \Delta t \cdot p \\ \theta_k &= \theta_{k-1} + \Delta t \cdot q \\ \psi_k &= \psi_{k-1} + \Delta t \cdot r \end{aligned} \tag{3.6}$$

Outra forma de calcular a posição angular usando o giroscópio é apresentado por Gonçalves (2017)

3.2.3.2 Representação com acelerômetro

A Equação 3.7 apresenta a relação das acelerações medidas pelo acelerômetro e os ângulos de Euler através da decomposição do vetor da aceleração gravitacional. Como discutido no capítulo 2, o acelerômetro é incapaz de fornecer o ãngulo *yaw* eficientemente, portanto as equações usadas neste trabalhas foram baseadas no texto de Paula (2015a) que utiliza apesar as duas primeiras linhas da Eq. 3.7. A terceira linha foi incluída para complementar o trabalho e incluir a possibilidade de uso dela como é apresentado por Lima (2016).

$$\begin{aligned} \phi &= \arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \\ \theta &= \arctan\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \\ \psi &= \arctan\left(\frac{a_z}{\sqrt{a_x^2 + a_y^2}}\right) \end{aligned} \tag{3.7}$$

3.2.4 Filtragem dos dados

Para obter dados mais confiáveis será necessário implementado um filtro. algumas opções foram apresentadas na seção 2.5. Para os testes iniciais foi implementado um filtro complementar. optou-se por esse filtro pela simplicidade de implementação e pela razoável eficiência (PAULA, 2015b). A abordagem usada para implementar o filtro foi apresentada por Paula (2015b) e a Eq. 3.8 apresenta o cálculo dos dados para o eixo Y.

$$\theta = HPF \cdot \theta_{gyro} + LPF \cdot \theta_{accel} \tag{3.8}$$

3.2.5 Controle dos motores

Como serão usados os motores DC, apresentados na Seção 3.1, será necessário o uso de uma fonte de alimentação externa, porque a BBB não é capaz de fornece corrente o bastante para o acionamente deles. Será usada uma ponte H como é indicado por Molloy (2014).

O controle da velocidade do motor será feito com o PWM e para acompanhar o comportamento de motor, que representa o sinal de saída do sistema de controle, será implementado um sistema de amostragem para ler os *encoders* utilizando os PRU's.

3.2.5.1 Sinais de controle com o GPIO

Para o acionamento dos motores será usado um PWM e a melhor solução é usar o GPIO, pois a BBB já apresenta vários pinos com essa função (modo 4 para P8 e modo 6 para P9). Além disso, tanto Molloy (2014) quanto Hamilton (2015) usam esse método para acionamento de motores DC.

A Tabela 10 apresenta a configuração do pinos necessários para o controle da ponte H.

Tabela 10 – Pinagem de controle da Ponte H

Pino (BBB)	Modo de operação	Pino (*driver*)
P9_16	EHRPWM1B	PWMA
P9_17	GPIO 5	AIN1
P9_18	GPIO 4	AIN2
P9_22	EHRPWM0A	PWMB
P9_24	GPIO 15	BIN1
P9_26	GPIO 14	BIN2
P9_23	GPIO 49	STBY

Todos os pinos apresentados na tabela não apresentam configuração alguma, portanto é necessário uma etapa de configuração da funções dos pinos. Todos são con-

figurados através dos arquivos disponíveis nos diretório */sys/devices/platform/ocp* (ou */sys/devices/ocp.**, dependendo da placa), que é responsável pela multiplexação dos pinos. Neste diretório há um diretório para cada pino disponibilizado através da *Device Tree Overlay* carregada, alguns usados nesse projeto são, por exemplo, *ocp:P9_16_pinmux* ou *ocp:P9_23_pinmux*.

No diretório de cada pino há um arquivo chamado *state* onde deve-se escrever a função desejada para o pino. No projeto foram usadas as funções "*pwm*", "*gpio*"(e "pruin", que será discutido adiante).

Para configurar e manipular os PWM's é necessários acessar os diretórios de cada subsistema de PWM, portanto é necessário identificar esse diretórios primeiro. Para isso, identifica-se qual *chip* está associado a qual PWM. A indicação do *chip* de cada PWM pode ser descoberta através de diretórios nomeados a partir dos endereços de memória dos PWM's. Esses endereços de memória estão indicados no manual de referência e apresentados na Tab. 11 (INSTRUMENTS, 2017).

Primeiro verifica-se o diretório */sys/devices/platform/ocp*, nele há os diretórios de cada subsistema de PWM nomeados como *48300000.epwmss*, *48302000.epwmss* e *48304000.epwmss*, que são os endereços dos subsistemas de PWM. Em cada um deles, há o diretório do PWM em si, ou seja, o diretório da função de PWM, nomeados por *48300200.pwm*, *48302200.pwm* e *48304200.pwm*, que são os endereços de memórias dos PWM's de cada subsistema. Em cada um desses diretórios, há um diretório chamado *pwm*, nele há o diretório, cujo nome indica o *chip*, por exemplo, *pwmchip0*.

No caso, do PWM0, que será usado no projeto o diretório referente a ele é:

/sys/devices/platform/ocp/48300000.epwmss/48300200.epwmss/pwm/pwmchip1

E no caso do PWM1 é:

/sys/devices/platform/ocp/48302000.epwmss/48302200.epwmss/pwm/pwmchip4

Entretanto, para fins de praticidade e segurança, o controle dos PWM's é feito através dos diretórios */sys/class/pwm/pwmchip1* e */sys/class/pwm/pwmchip4*.

Tabela 11 – Organização dos *chips* dos subsistemas de PWM na BBB

PWM	Endereço	*chip*
PWM0	0x48300200	*pwmchip1*
PWM1	0x48302200	*pwmchip4*
PWM2	0x48304200	*pwmchip7*

Cada subsistema de PWM e capaz de gerar até dois sinais de PWM, por exemplo *EHRPWM0B* e *EHRPWM0A*, e é necessário habilitá-los separadamente. Optou-se por não usar dois sinais de mesmo subsistema.

Os pinos usados como GPIO também podem ser configurados usando o arquivo "*export*"presente no direto */sys/class/gpio*. A configuração é feita escrevendo o número do GPIO no arquivo citado. O número do GPIO de um pino é definido da seguinte forma, o pino tem um nome na forma GPIOX_Y e seu número será dado pela fórmula $(X \times 32) + Y$. Um exemplo é o pino P9_23, que tem a função de GPIO1_17, possui o número 49 no diretórios do GPIO. O pino devidamente configurado gera um diretorio próprio com o número do pino no GPIO, por exemplo "/gpio49". A numeração do GPIO dos pinos usados está apresentado na Tab. 10

Para os sinais de controle *AIN1*, *AIN2*, *BIN1* e *BIN2* é necessário configurar os pinos *P9_17*, *P9_18*, *P9_24* e *P9_26* também com a função de saída (ou "*output*"). No diretório de cada GPIO há um arquivo chamado "*direction*", nele deve-se escrever o modo de operação "*out*"para saída e "*in*"para entrada.

Por fim. para gerar os valores lógicos nas saídas do GPIO basta escrever o valor "0"ou "1"no arquivo "*value*"presente no diretório de cada pino.

3.2.5.2 Lógica de controle

Para o controle de velocidade e direção dos motores será usada a seguinte lógica.

- O sinal PWMA controla a velocidade do motor A
- Os sinais AIN1 e AIN2 definem a direção do motor A
- O sinal PWMB controla a velocidade do motor B
- Os sinais BIN1 e BIN2 definem a direção do motor B

É importante notar que os circuitos da ponte H e dos motores são idênticos, mas como eles ficam espelhados na montagem, é necessário configurar os valores dos sinais corretamente. A Tabela 12 apresenta como os sinais do GPIO afetam as ações dos motores.

Tabela 12 – Tabela verdade da Ponte H para um motor

AIN1	AIN2	Direção
L	H	para frente
H	L	para trás
H/L	H/L	parado
BIN1	**BIN2**	**Direção**
L	H	para trás
H	L	para frente
H/L	H/L	parado

Conhecendo os sentidos de movimento dos motores é possível saber como controlar o robô para andar para frente e para trás, ou girar, tanto no sentido horário quanto anti-horário.

3.2.6 Unidades programáveis de tempo real

Como dito antes, os PRU's são um *hardware* separado e, portanto, é necessário habilitá-los e concedê-los acesso aos pinos de I/O. Além disso, a programação deles é feito de uma forma diferente do restante do projeto e há uma forma específica para a comunicação dos PRU's com o programa principal. Essa subseção abordará os processos necessários para o uso dos PRU's de acordo com o objetivo do projeto.

3.2.6.1 Habilitando os PRU's

Como a BBB possui muitas ferramentas, como comunicações I^2C e SPI, GPIO, PWM e outros, que usam os mesmos recursos de *hardware*, como os pinos de I/O, é necessário organizá-los para que não haja conflitos. A organização é feita através da *Device Tree* e, para modificá-la, é necessário usar uma *Device Tree Overlay* (MOLLOY, 2014). Na versão 9.5 do Debian, usada nesse projeto, há uma *overlay* para usar o *PRU Linux Application Loader*. Ela é habilitada usando o arquivo */boot/uEnv.txt* e descomentando a linha que indica os recurso do PRU no *userspace AM335X-PRU-UIO-00A0.dtbo*. Deve-se comentar também a linha que indica os recursos de *RemoteProc*, também do PRU, indicado por *AM335x-PRU-RPROC-4-14-TI-00A0.dtbo*.

Além disso, deve-se instalar um pacote com os recursos para desenvolder os códigos para os PRU's. O pacote chama-se *PRU Package* e inclui as bibliotecas para os PRU's e um compilador de linguagem *assembly*, chamado *pasm*. Uma explicação mais detalhada do processo de instalação desse pacote é apresentado por Molloy (2014).

3.2.6.2 Programando os PRU's

Para o uso dos PRU's serão feitos dois códigos separados, como explica Molloy (2014). O código que será executado no PRU deve ser escrito em *assembly* em um arquivo no formato *.p*. Ele deve ser compilado para gerar um arquivo no formato *.bin*.

O segundo código deve ser escrito em linguagem C usando as funções das bibliotecas do *PRU Package*. Entre elas há uma função específica que carrega o arquivo no formato *.bin* para o PRU desejado.

A comunicação entre o PRU e o programa no Linux é feita através de compartilhamento de memória e chamadas de interrupção. Tanto o mapeamento de interrupções quanto o mapeamento da memória é feito usando as funções do *PRU Package*. A Figura 22 apresenta de forma visual o método de programação dos PRU's.

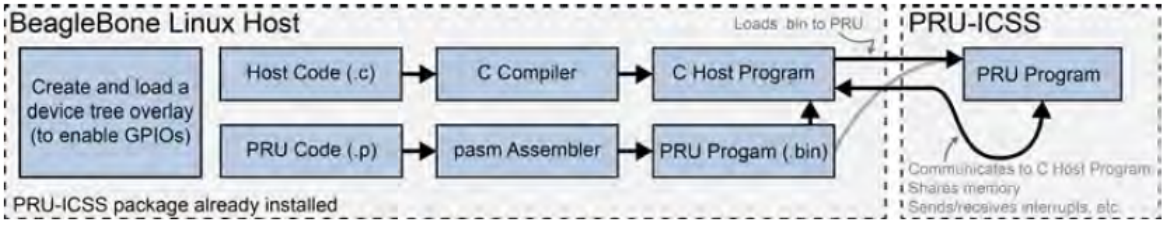


Figura 22 – Diagrama de funcionamento do PRU (MOLLOY, 2014)

3.2.7 Leitura dos *encoders*

Para a leitura dos encoder será necessário que o PRU tenha acesso a 4 pinos no modo de entrada. A pinagem é organizada como apresenta a Tab. 13 e a escolha dos pinos foi feita para que os sinais correspondessem aos *bits* de 0 a 3 do registrador de entrada do PRU0.

Tabela 13 – Pinagem para *encoders*

Pino (BBB)	Modo de operação	Pino (*encoders*)
P9_28	pr1_pru0_pru_r31_3	C2B
P9_29	pr1_pru0_pru_r31_1	C1B
P9_30	pr1_pru0_pru_r31_2	C2A
P9_31	pr1_pru0_pru_r31_0	C1A

Como os *encoders* estão em quadratura, o lógica de decodificação dos pulsos é apresentado na Tab. 14. Primeiro é identificado a mudança de estado de um dos canais, então testá-se o estado do outro canal para definir a direção em que o motor está atuando, e por fim é atualizado o número de pulsos. A implementação do algoritmo é feita em *assembly* e carregado no PRU0.

Tabela 14 – Lógica de interpretação dos *encoders*

A ↑	B=1	Incrementa
	B=0	Decrementa
A ↓	B=1	Decrementa
	B=0	Incrementa
B ↑	A=1	Decrementa
	A=0	Incrementa
B ↓	A=1	Incrementa
	A=0	Decrementa

O valor calculado pela leitura dos *encoders* é atualizado diretamenta na memória, dessa forma é possível para um programa no Linux acessá-lo usando um mapeamento de memória e manipulá-lo como um ponteiro. O pacote instalado para uso do PRUSS já possui funções prontas para esse mapeamento e acesso a memória de ambos os PRU's (MOLLOY, 2014).

3.2.8 Controle do sistema com PID

Para validação do projeto como ferramenta de testes de sistemas de controle optou-se por implementar um controlador PID. O controlador funciona de forma semelhante aos trabalhos feitos por Freitas (2014) e Oliveira (2017) e segue a teoria apresentada na seção 2.10.

3.2.8.1 Ajuste da sintonia do PID

Para o PID funcionar com eficiência é necessário ajustar os ganhos K_p, K_i e K_d corretamente. Existem diversas formas de realizar esse ajuste, uma delas é o método de Ziegler-Nichols, explicado por Ogata e Severo (1998) e Muniz (2017).

Esse método é um ajuste de campo e consistem em ajustar os ganhos K_d e K_i para zero e incrementar o ganho K_p até que o sistema apresente uma oscilação constante. O valor do K_p que gera essa oscilação é chamado de ganho crítico, ou K_u, e a oscilação fornece um período chamado de período crítico, ou T_u. Obtidos esse valores críticos, basta seguir as regras apresentada na Tab.15 para calcular os ganhos do PID.

Tabela 15 – Método Ziegler-Nichols

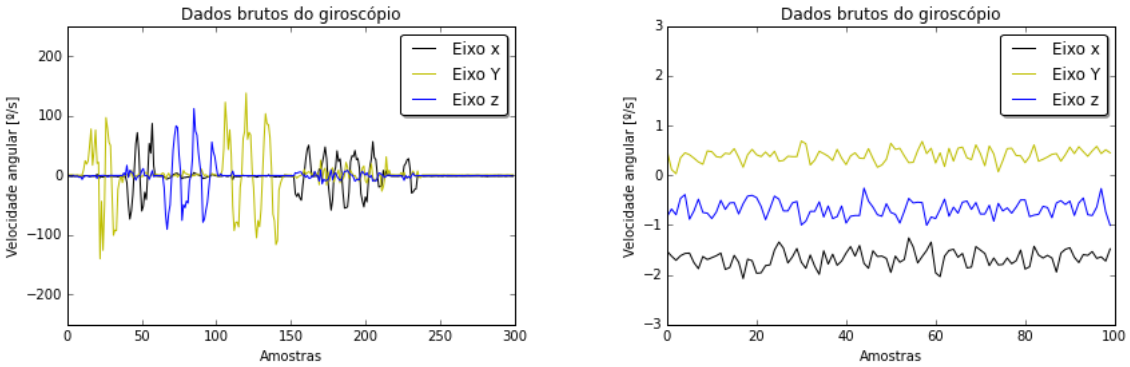
Ganho	K_p	K_i	K_d
P	$0.5\ K_u$		
PI	$0.45\ K_u$	$1.2\ \frac{K_p}{T_u}$	0
PID	$0.6\ K_u$	$2\ \frac{K_p}{T_u}$	$\frac{K_p T_u}{8}$

4 Resultados e discussões

Neste capítulo são apresentados os resultados finais deste trabalho. Foram concluídos, o sensoriamento inercial, a fusão sensorial, o acionamento dos motores, a leitura dos *encoders* e a integração de todos esses componentes em um protótipo da plataforma.

4.1 Leitura dos sensores

As Figuras 23 a 25 apresentam os resultados das leituras dos três sensores.



(a) Dados do giroscópio em movimento

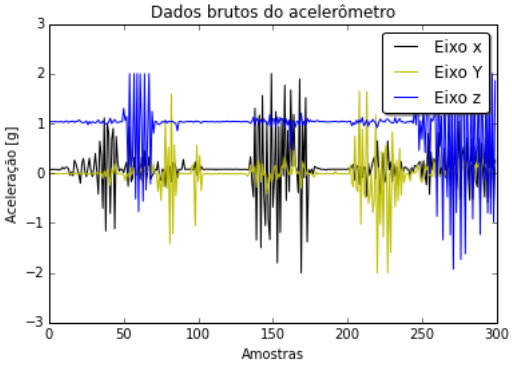
(b) Visão ampliada do offset do giroscópio

Figura 23 – Leitura dos dados brutos do giroscópio

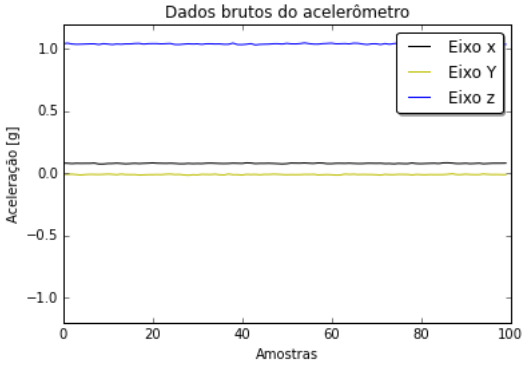
Na Figura 23a é possível observar o movimento de rotação do giroscópio em cada um dos eixos. Primeiro é apresentado o movimento de rotação no eixo y, em amarelo, em seguida o movimento no eixo x, em preto, e por fim o movimento no eixo z, em azul. Ao lado, na Fig. 23b, são apresentados os dados de forma ampliada e é possível ver a discrepância do valores lidos e dos valores esperados, ou seja, nulos e uma presença de muita oscilação nos valores, mas neste caso é irrelevante, pois a variação ocorre em uma escala muito pequena e não apresenta significância na escala original.

Na Figura 24a é possível observar os movimentos de deslocamento do acelerômetro através das acelerações lineares registradas. Neste gráfico observa-se que o eixo z, diferente dos eixos x e y, registra a aceleração da gravidade por estar perpendicular a superfície da Terra. Na Figura 24b observa-se que os valores brutos do acelerômetro também apresentam uma discrepância dos valores esperados, que devem ser $1g$ no eixo perpendicular a superfície da Terra (eixo z) e nulos nos eixos restantes (eixos x e y).

Na Figura 25 são apresentados os valores lidos do magnetômetro. é possível observar os movimentos de rotação em cada um dos eixos, entretanto, observa-se algumas discrepâncias, por exemplo, quando o eixo z é mantido parado perpendicularmente ao solo



(a) Dados do acelerômetro em movimento



(b) Visão do offset do acelerômetro

Figura 24 – Leitura dos dados brutos do acelerômetro

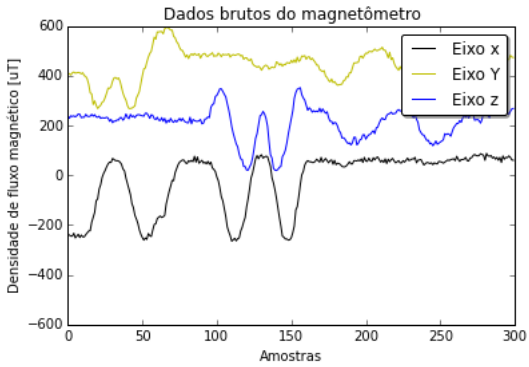


Figura 25 – Dados brutos do magnetômetro em movimento

e o sensor é rotacionado, em determinado momento da rotação o eixo y deve apresentar o mesmo valor do eixo x, entretanto, esse comportamtento não foi registrado no gráfico.

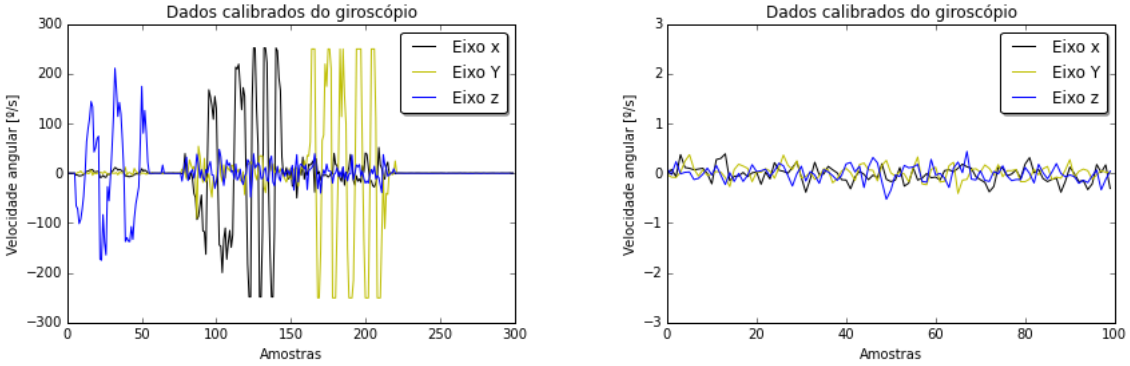
Desses resultados, conclui-se que é necessário corrigir os dados lidos para obter informações mais confiáveis dos sensores, logo a calibração dos sensores é fundamental.

4.2 Calibração dos sensores

Evidenciou-se na seção 4.1 a necessidade de calibração dos sensores para obter dados mais confiáveis, portanto foram implementadas funções de calibração dos sensores com um algoritmo que está de acordo com a descrição do método de calibração apresentado na seção 3.2.2. As Figuras 26 a 29 apresentam os resultados da calibração dos sensores.

Na Figura 26a são apresentados os dados do giroscópio em movimento de rotação. Inicialmente, não é perceptível o efeito da calibração, entretanto na Fig. 26b observa-se o ajuste do *offset* e o correto posicionamento dos valores próximos de zero.

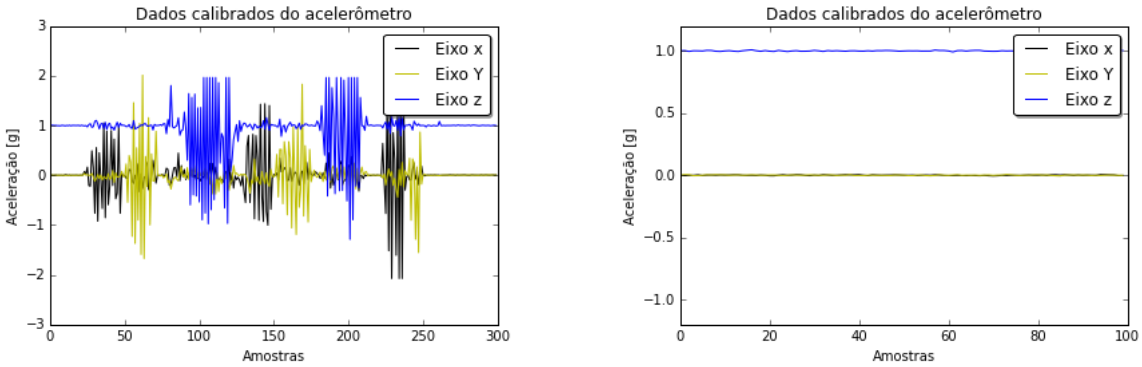
Na Figura 27a são apresentados os dados lidos do acelerôemtro em movimento. Novamente é pouco evidente o efeito da calibração, mas ele pode ser observado claramente na Fig. 27b, onde os valores dos eixos x e y estão próximos de zero e o valor do eixo z



(a) Dados calibrados do giroscópio em movimento

(b) Visão ampliada do offset calibrado do giroscópio

Figura 26 – Leitura dos dados calibrados do giroscópio



(a) Dados calibrados do acelerômetro em movimento

(b) Visão do offset calibrado do acelerômetro

Figura 27 – Leitura dos dados calibrados do acelerômetro

está próximo de $1g$.

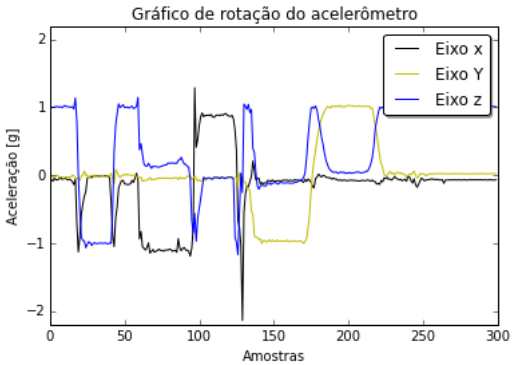


Figura 28 – Dados calibrados do acelerômetro em movimento de rotação

Na Figura 28 são apresentados os dados obtidos do acelerômetro realizando movimentos de rotação. Observa-se no gráfico o registro da aceleração da gravidade nos eixos conforme o sensor é rotacionado. Dessa forma, a rotação do objeto gera componentes da aceleração da gravidade nos eixos e isso permite usar o acelerômetro para estimar a

orientação de um objeto,.

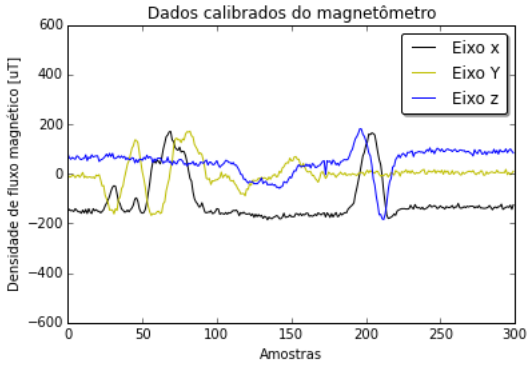


Figura 29 – Dados calibrados do magnetômetro em movimento

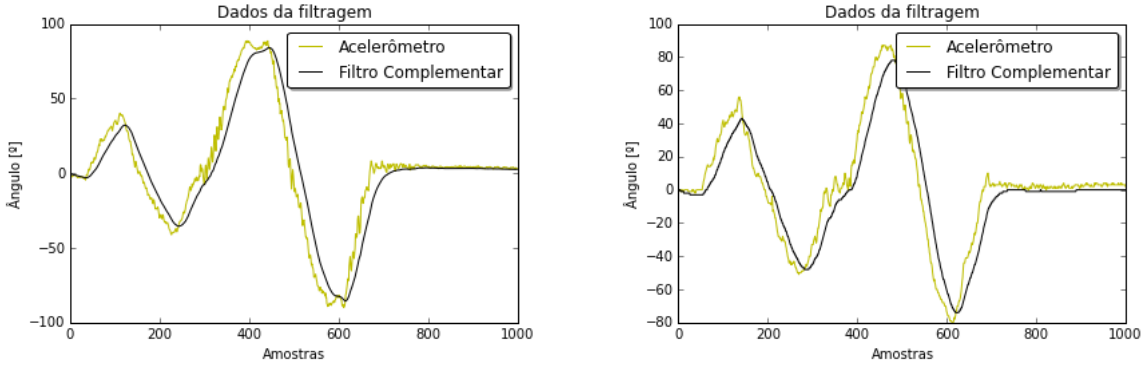
Na Figura 29 são apresentados os dados obtidos do magnetômetro calibrado. Para o magnetômetro, uma análise do *offset* é desnecessária, porque o sensor ainda registra o campo magnético da Terra. Para observar o efeito da calibração é necessário analisar a concordância dos dados. Como explicado anteriormente, mantendo-se um eixo fixo e rotacionando o sensor, os valores dos eixos restantes devem apresentar uma diferença de fase de 90°. De modo geral, esse comportamento pode ser observado no gráfico. Além disso é perceptível o efeito do ajuste das escalas de cada um dos eixos.

4.3 Representação da orientação e filtragem dos dados

Para obter a orientação dos pêndulo, foram usadas as Eq. 3.6 e 3.7. E para a filtragem dos dados usou o método do filtro complementar, implementado através da Eq. 3.8. Para análise dos resultados foram medidas mil amostras a uma taxa de 100 HZ, seguindo um padrão de movimento de uma oscilação curta seguida de uma oscilação ampla, mas dentro do limite de 90º do movimento do pêndulo.

A Figura 30 apresenta os resultados da representação da orientação e da filtragem dos dados, nela é possivel observar o atraso do filtro, como era esperado. O atraso do filtro foi explicado na seção 2.5.1 e os resultados estão de acordo com o resultado apresentado por Paula (2015b). Observa-se também que o sinal de saída do filtro não apresenta ruídos do acelerômetro e nem o efeito de *drift* do giroscópio.

A Figura 30a apresenta o resultado da filtragem com precisão decimal e a Fig. 30b apresenta os resultados com precisão truncada em números inteiros. Observa-se pouco diferença nos resultados quanto ao movimentos, de todo modo, a precisão decimal gera curvas mais suaves no gráfico e as partes constantes tem oscilações menores.



(a) Representação da orientação do pêndulo com precisão decimal

(b) Representação da orientação do pêndulo com precisão inteira

Figura 30 – Resultados do filtro complementar

4.4 Controle dos Motores

A seguir são apresentados os resultados para uso do PWM na BBB, da leitura dos *encoders* com o PRU e a interpretação dessas informações, que formam uma malha fechada de controle dos motores.

4.4.1 Sinais de PWM

No trabalho foram usados sinais de PWM para controlar a velocidade dos motores. Optou-se por usar os recursos do GPIO que já apresentam *hardware* configurado para essa função. Foram usados dois subsistemas separados para evitar interferências. As Figuras 31 e 32 apresentam os resultados para essa etapa do projetos.

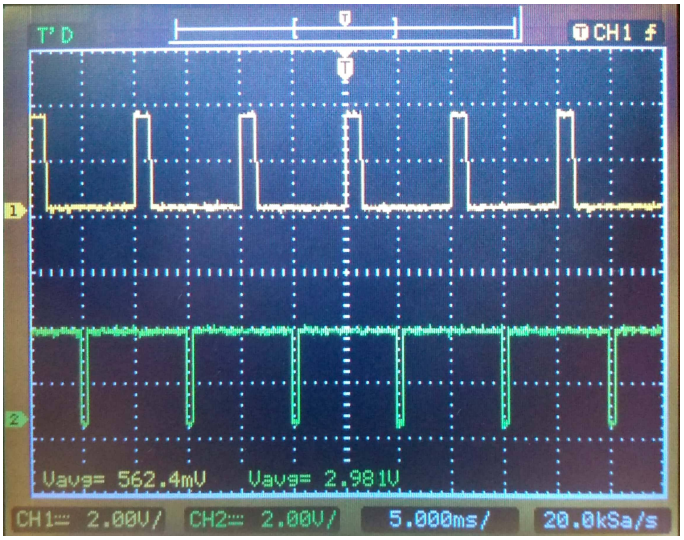


Figura 31 – Sinais de PWM a 100Hz

Na Figura 31 são apresentados os sinais dos dois subsistemas usados no projeto com uma frequência de 100Hz. O sinal mais acima apresenta um *duty cycle* de 15%, ou

seja, 1.5 milissegundo. O sinal mais abaixo apresenta um *duty cycle* de 95%, ou seja, 9.5 milissegundos. Na imagem também são apresentados os valores médios de tensão de cada sinal. Para o primeiro sinal temos uma tensão média de 562,4mV e para o segundo sinal temos 2.981V. A Tabela 16 faz um comparativo dos parâmetros do sinal gerado com os parâmetros esperados. Observa-se um erro na transferência da tensão média para o motor.

Tabela 16 – Resultados dos sinais de PWM

Sinal	***Duty cycle* (%)**	$\mathbf{V}_{Máx}$**(V)**	$\mathbf{V}_{média}$**(V)**	**(%) real**	**Erro (%)**
EHPWM0B	15	3.3	0.5624	17.04	13.6
EHPWM2A	95	3.3	3.32.981	90.33	4.92

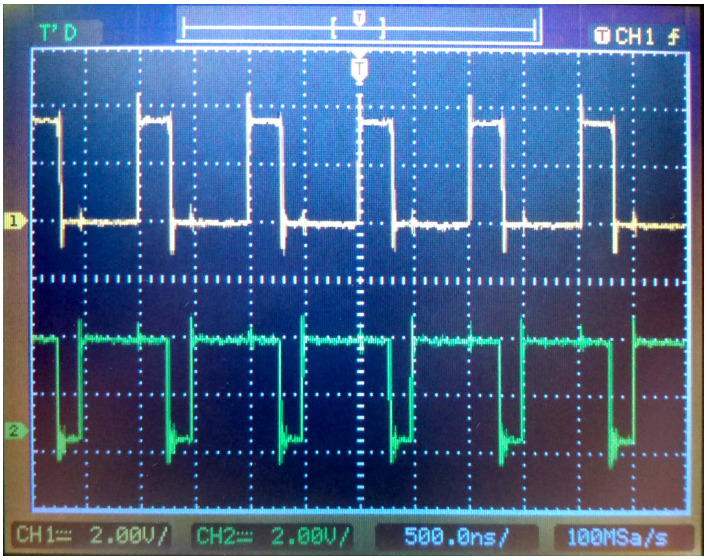


Figura 32 – Sinais de PWM a 1MHz

Na Figura 32 são apresentados os sinais de PWM com uma frequência de 1MHz. Os *duty cycles* são de 30% e 80% para os sinais EHPWM0B e EHPWM2A respectivamente. Observa-se nessa figura que o aumento da frequência gera ruídos no sinal.

Segundo (MOLLOY, 2014) usar frequências altas, acima 1kHz, gera oscilações na tensão do PWM, chamadas *ringing*. Ainda segundo ele, frequências baixas, como 100Hz, já são o suficiente para controle de motores e servos. Entretanto, não observou-se a vairação de tensão média em nenhuma frequência

4.4.2 Aplicação do *driver* para motor

Para o acionamento dos motores testou-se alguns modelos de *drives*, após alguns testes os componentes apresentaram mal funcionamento. Decidiu-se usar o modelo apresentado na seção 3.1.3, que apresentou um bom funcionamento. Esse *driver* possui uma lógica de controle como apresentado na seção 3.2.5.2.

Com o ajuste da lógica no código do projeto, o motor foi acionado com eficiência, entretanto, a lógica de controle desse *driver* é diferente de outros modelos comerciais, ou seja, outros modelos não serão compatíves com o projeto caso seja necessário trocar o componente.

4.4.3 Leitura dos *encoders*

Para um controle eficiente dos motores é necessário acompanhar a rotação dos motores e os modelos usados no projeto já apresentam um ciruito de *encoders* embutidos capazes de realizar essa função. A figura 33 apresenta o sinal de um dos *encoders* com o motor ligados em um sinal contínuo de 6V, ou seja, um *duty cycle* de 100%. É possível observar a defasagem entre os canais e a forma de onda com poucos ruídos. Dessa forma, para a leitura dos *encoders*, bastou seguir a lógica de decodificação apresentada na seção 3.2.7 para saber o sentido do movimento e contar os pulsos.

Usou-se um dos PRU's para a leitura dos pulsos em tempo real. A frequência de 200 MHz do PRU é muito alta em relação a frequência de 1.12 kHz do *encoder*, como mostra a Fig. 33, mas os resultados dos testes não apresentaram diferenças significativas com a redução da frequência de leitura dos *encoders*.

O uso do *PRU Package* para implementar a leitura dos *encoders* foi satisfatório, pois a afinidade com a linguagem *assembly* permitiu desenvolver o código com a lógica de decoficação com razoável tranquilidade. Além disso, a manipulação de memória tornou-se simples usando os recursos desse pacote.

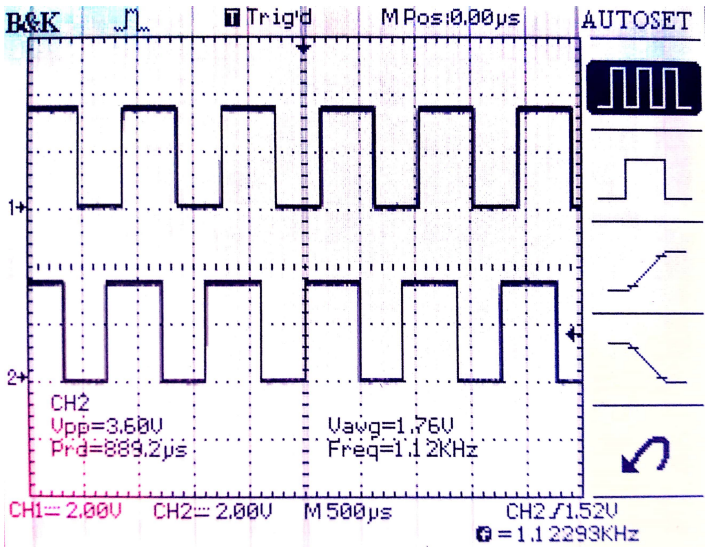


Figura 33 – Sinais do *encoder* em quadratura

Como os motores usados não tem referências técnicas o suficiente para saber a relação entre os pulsos do *encoder* e a rotação do eixo do motor, realizou-se alguns testes

para estimar um valor razoável. Foram realizadas 10 voltas do eixo do motor e mediu-se os valores com o PRU, esse processo foi retido algumas vezes e os resultados foram coerente.

Entretanto, controlar motores não é algo trivial. Entre o instante em que o *encoder* determina que o motor atingiu a posição desejada e o momento em que o sinal de controle para o motor é interrompido há um atraso que mantém o motor acionado e isso gera um erro no cálculo de posição. Além disso, o motor não para imediatamente quando o sinal de controle é interrompido, ele ainda demanda um tempo para desacelerar e isso também provoca um erro no controle de posição. É necessário, portanto, um controlador para corrigir esse comportamento do motor.

4.5 Sistema de controle

Para validar a plataforma como uma ferramenta de testes de sistemas de controle, optou-se por implementar um controlador PID simples, como apresentado na seção 2.10. Não obteve-se um bom resultado no ajuste dos ganhos do PID, dessa forma não foi possível realizar o equilíbrio do pêndulo como esperado.

Inicialmente, optou pelo método de Ziegler-Nichols para ajustar a sintonia do PID, como apresentado na seção 3.2.8, entretanto, esse método de ajuste em campo é particularmente problemático. Ele exige que o sistema seja colocado em oscilação constante e isso não foi possível devido ao estado frágil do protótipo construído. Portanto, a opção de ajuste manual de ganho representa a melhor opção para o protótipo.

4.6 Protótipo

A seguir são apresentados os resultados da estrutura desenvolvida para o protótipo e da integração com os demais componentes.

4.6.1 Estrutura

Para o projeto de um pêndulo invertido, optou-se por fazer uma estrutura mais verticalizada. O projeto possui apenas restrições de tamanho no plano dos motores, ou seja, é necessário que a largura seja suficiente para encaixar os dois motores. Além disso, é necessário um suporte horizontal para fixar a IMU e posicionr corretamente os eixos para garantir uma leitura eficiente dos sensores.

Para a sustentação do pêndulo foi fabricada uma base para o protótipo em alumínio. Foram fabricados usando impressão 3D, um suporte vertical para moldar a geometria do pêndulo, um suporte para os sensores e *driver* e um suporte para a BBB. Os diagrams dos componentes e o esquemático geral estão apresentados no Apêndice A. O suporte ver-

tical foi desenvolvido para que haja espaço suficiente para acoplar uma bateria comercial para o sistema.

4.6.2 Integração

Com todas as peças devidamente fabricadas foi realizada a integração do protótipo. A Figura 34 apresenta o resultado da integração. O protótipo apresentou bom funcionamento de suas partes. Os sensores, os motores e o *driver* funcionaram tão bem quanto nas etapas de desenvolvimento de cada componente. Fabricou-se também uma placa de circuito impresso para garantir robustez às conexões do protótipo.

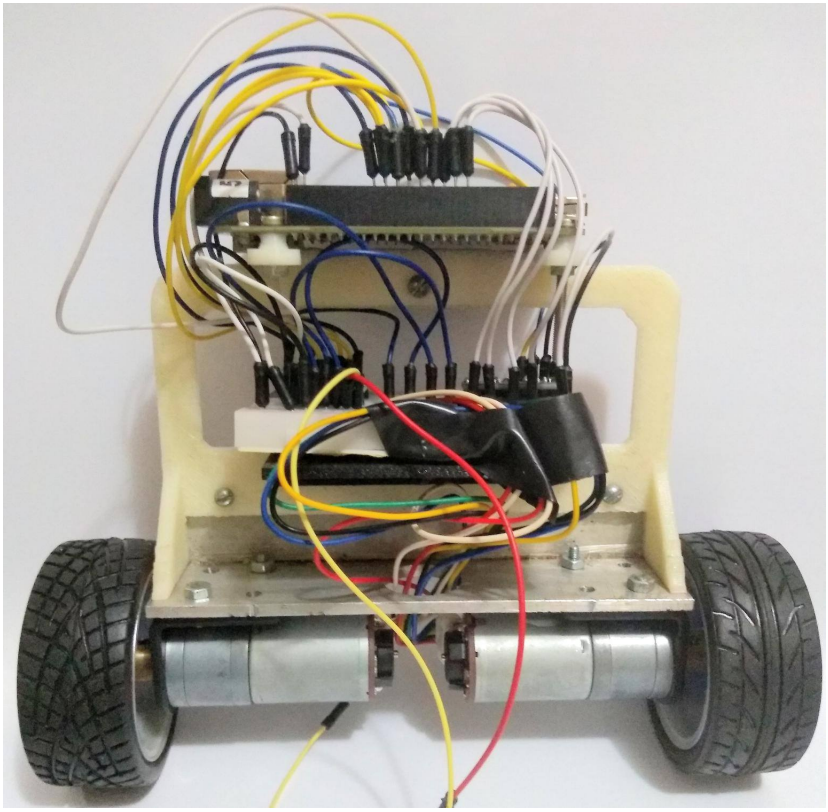


Figura 34 – Resultado da integração do protótipo

4.7 Arquitetura geral do projeto

Com todos os blocos funcionais finalizados e o integrados é possível compreender melhor como eles se relacionam no projeto. A Figura 35 apresenta o diagrama da arquitetura geral do projeto, que permite visualizar essas relações.

Na parte superior esquerda da Fig. 35, é apresentado o programa principal que é executado pelo Linux e foi desenvolvido em linguagem C. Nesse programa é implementado um controlador para os motores como discutido na seção 4.4 e um controlador para o modelo dinãmico do pêndulo imvertido. Os códigos desses controladores são distintos de

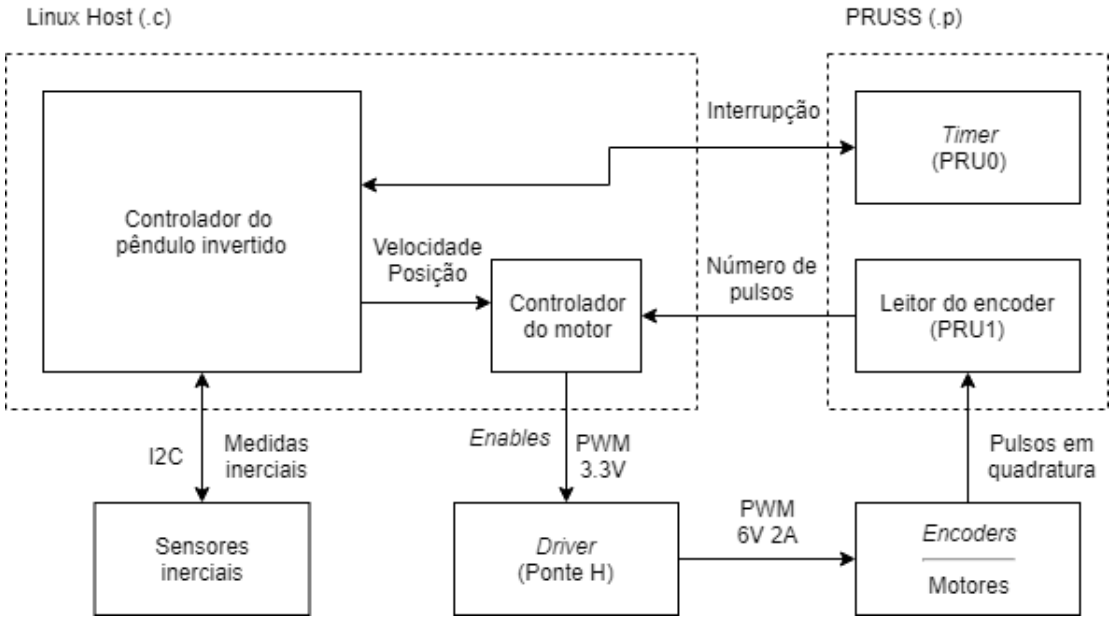


Figura 35 – Arquitetura geral do projeto

forma que o usuário da plataforma desenvolve o controlador do pêndulo e envia o sinal de controle para o controlador do motor. O controle do motor é feito de forma encapsula e o usuário não precisa ter conhecimento dos seus métodos.

Na parte superior direita da Fig. 35, é apresentado os componentes desenvolvidos para o PRUSS em linguagem *assembly*. Como o PRUSS possui dois microcontroladores, um foi usado para a leitura dos encoders em tempo real e o outro foi reservado para tarefas de temporização do sistema. A leitura dos *encoders* mostrou-se eficiente e é importante para o controle dos motores.

Observa-se também na Fig. 35 o sistema de malha fechada que envolve o controlador do motor, o *driver*, os motores e os *encoders*. O controlador do motor recebe um sinal, como posição ou velocidade e gera os sinais de controle do motor, o PWM e os *enables*. O *driver* recebe os sinais do controlador e alimenta o motor com a corrente necessária. O motor é acionado pelo *driver* e realiza o processo físico do sistemas. O *encoder* registra a ação do motor e retorna essa informação para o controlador, que pode corrigir o processo do motor gerando novos sinais.

Ainda na Fig, 35 é apresentado o bloco dos sensores inerciais. Ele é ligado diretamente ao controlador do pêndulo invertido, assim, o controlador obtém a orientação através dele e calcula um sinal de controle a partir disso.

4.8 Arquitetura do *firmware*

Como o objetivo desde trabalho é desenvolver uma plataforma de testes de sistemas de controle, o usuário precisa apenas desenvolver o algoritmo de controle. Os processos

por trás da manipulção de *hardware* são encapsulados e o usuário não tem conhecimentos dos métodos usados. Assim foi desenvolvido uma camada de *firmware* para servir como interface do usuário com o *hardware*. A Figura 36 apresenta a arquitetura do *firmware* onde é possível observar a dependência das bibliotecas bem como a sua aplicação no projeto.

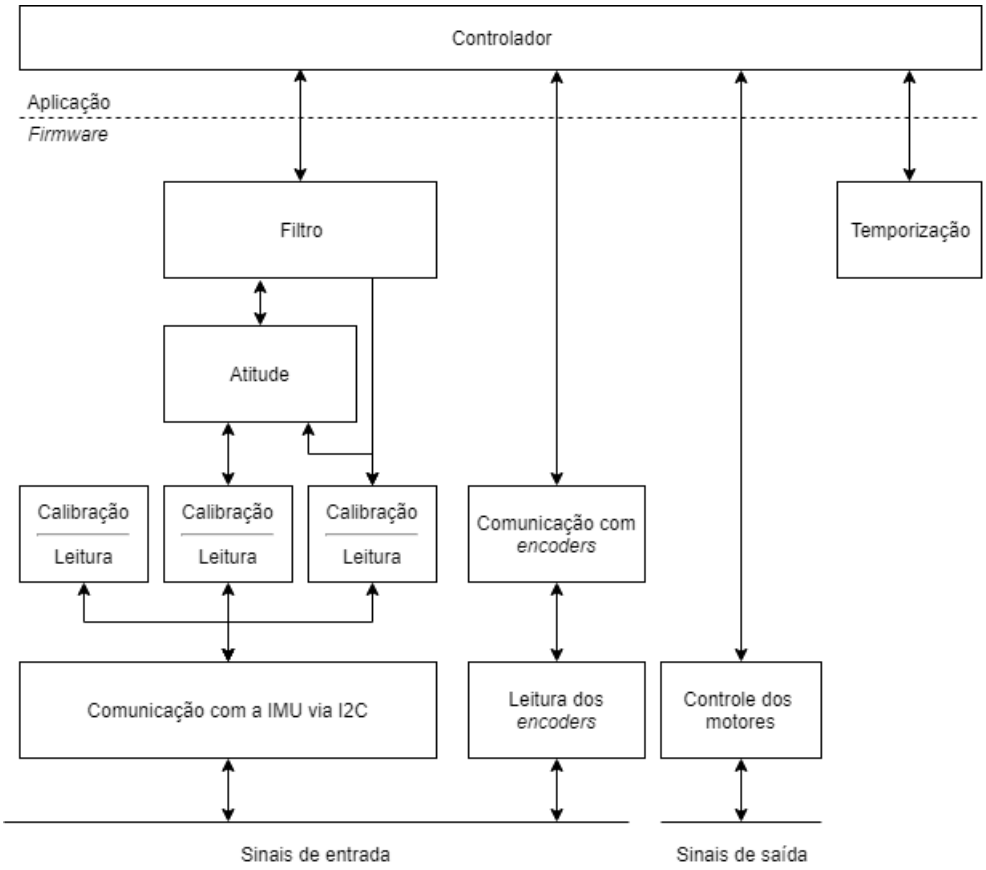


Figura 36 – Arquitetura do firmware

Na parte mais a esquerda do diagrama observa-se as camadas de *firmware* para o sensoriamente, orientação e filtragem dos dados. Para o usuário final o funcionamento do protocolo I^2C fica omitido na camada de comunicação com IMU através de funções de leitura e escrita implementadas na biblioteca *MPU9250.h*. Acima dessa camada, há as funções próprias de cada sensor e estão encapsuladas na bibliotecas *gyroscope.h*, *accelerometer.h* e *magnetometer.h*. As bibliotecas dos sensores depende da biblioteca de comuncação por I^2C.

Como discutido na seção 3.2.3, os dados lidos dos sensores não representam a orientação do pêndulo. Para saber a orientação há uma camada de *firmware* que implementa em código as equações de representação de atitude como apresentado na seção 3.2.3. As funções para cálculo da atitude estão presentes na biblioteca *atitude.h*.

Há mais uma camada de *firmware* que implementa uma fusão sensorial para melhorar a representação de atitude. Essa camada usa funções tanto de atitude do acelerômetro

quando de leitura dos dados brutos do giroscópio e está implementada na biblioteca *filtro.h*.

Para a leitura do encoder foram criadas duas camadas de código. a primeira, mais baixa, é o código em *assembly* executado pela PRU. e a segunda camada é código em linguagem C que carrega o código no PRU e cria uma interface de comunicação usando as funções do *PRU Package*. A divisão em duas camadas é feita devido ao funcionamento padrão do PRUSS, como foi apresentado na seção 3.2.6. Essa camada de *firmware* está presente na biblioteca *encoder.h*.

Para o controle dos pinos do GPIO para gerar os sinais de *enable* e para o controle dos subsistemas de PWM, foi implementado uma camada de *firmware* que omite os processos de manipulação desses recursos. O usuário apenas envia o sinal de controle e as funções dessa camada realizam a ação desejada. As funções de controle do motor estão presentas na biblioteca *motor.h*.

Todas essa camadas de *firmware* apresentaram um bom funcionamento, isso significa que todos os processos de *hardware* estão satisfatoriamente encapsulados. Como não foi implementado um sistema de temporização do sistema em tempo real, a camada de temporização não será discutida.

5 Conclusão

Neste trabalho foi desenvolvida uma plataforma de testes de sistemas de controle com base no modelo do pêndulo invertido sobre duas rodas. A plataforma tem o objetivo de ensinar aos alunos de gradução o funcionamento de diferentes sistemas de controle. O projeto tem como base o modelo matemático de pêndulo invertido sobre duas rodas, que garante algumas características interessantes para testes de sistemas de controle, como a instabilidade intrínseca ao modelo.

Para o sensoriamento do projeto foi usado uma unidade inercial, que contém um giroscópio, um acelerômetro e um magnetômetro. Devido as características dos sensores foi implementado um método de calibração e os resultados foram satisfatórios. Além disso foi implementada uma fusão sensorial usando um filtro complementar para eliminar os problemas intrínsecos do giroscópio e do acelerômetro. Os resultados da leitura e calibração dos sensores, da representação de atitude e da fusão sensorial forma satisfatórios e coerentes com o referencial teórico.

Para a atuação da plataforma foram usados dois motores DC de 6V. Para controlar esses motores através da BBB, foi necessário usar um *driver* para não causar danos a placa. Como os motores apresentam comportamentos físicos característicos, é necessário um controle de malha fechada. O sensoriamento dos motores foi feito usando *encoders* e implementou-se a leitura desses *encoders* usando uma unidade programável de tempo real da BBB. O controle dos motores não foi concluído, porém o acionamento dos motores e a leitura dos *encoders* apresentaram bons resultados.

O objetivo desde trabalho é entregar um produto, a plataforma de testes, e este objetivo foi alcançado. Foi fabricado um protótipo, com todos os componentes integrados e funcionando. Além disso, os subprodutos resultantes dos objetivos parciais foram produzidos com sucesso. O sensoriamento inercial, a fusão sensorial, o controle dos motores e a leitura dos *encoders* já produziram muito conhecimento e são suficientes para demonstrar o sucesso desse trabalho. O projeto desde o início se mostrava mais complexo do que é possível realizar em um único trabalho de graduação, entretanto, ele foi desenvolvido o bastante e de forma que pode ser continuado e criar possibilidades de conhecimento para mais pessoas.

Um dos maiores legados deste trabalho é o estudo aprofundado da BBB como recurso educacional. Apesar dos diversos problemas com o plataforma ao longo do projeto, é nítido o potencial da placa para as mais diversas aplicações, incluindo este projeto. Ainda há muito para ser explorado nessa placa e muito para ser aperfeiçoado. Uma pequena parte desse conhecimento foi produzido para a comunidade acadêmica brasileira e registrado em

lingua portuguesa, assim espera-se que mais pessoas se interessem pela BBB e que mais conhecimento seja produzido para as futuras gerações.

5.1 Proposta de melhoria

A seguir são apresentadas algumas propostas de melhoria que podem ser desenvolvidas em futuros trabalhos.

- Temporização do sistema. Como discutido nesse trabalho, a BBB possui recursos que permitem uma temporização eficiente do sistema, mas é necessário algum tempo de estudo sobre esses recurso, porque nem a literatura nem a comunidade de usuário possui imformação o bastante sobre isso.
- Ajuste da preemptividade do sistema operacional. A BBB possui um sistema operacional Linux que é originalmente não preemptivo. É apresentado nesse trabalho formas de modificar essa característica do sistema e melhorar os resultados do projeto.
- Implementação de um filtro de Kalman. O filtro de Kalman pode representar uma solução melhor para a fusão sensorial por não causar um atraso no sinal.
- Implementação de um controlador para os motores. Os motores usados nesse trabalhos não possuem referência técnica alguma. Um estudo mais detalhado dos motores e das metodologias de controle, é um possibilidade de aprendizado a mais para futuros trabalhos.
- Aprofundamento dos recursos da BBB. Nesse projeto foram explorados alguns recursos da BBB, entretanto, isso representa pouco perto de todo o seu potencial.
- Alimentação do sistema. Para um movimento mais livre do pêndulo um forma de alimentação embutida na estrutura do pêndulo seria o ideal.

Referências

ALMEIDA, T. E. P. Sistema de sensoriamento de orientação para um veículo aquático de superfície utilizando sensores de baixo custo. Escola de Engenharia de São Carlos, 2014. Citado na página 40.

BASILIO, J. C.; MOREIRA, M. V. Experimentos para estimação dos parâmetros de motores de corrente contínua. *VII Encontro de Educação em Engenharia*, 2001. Citado na página 42.

BELHOT, R. V.; FIGUEIREDO, R. S.; MALAVÉ, C. O. O uso da simulação no ensino de engenharia. In: *Congresso Brasileiro de Ensino de Engenharia, XXIX COBENGE*. [S.l.: s.n.], 2001. p. 445–451. Citado na página 29.

BEWLEY, T. edumip: the world's most extensible low-cost open platform for learning embedded control robotics. 2017. Disponível em: <https://www.ucsdrobotics.org/edumip>. Citado 2 vezes nas páginas 15 e 34.

CAMARA, R. C. P. Protocolo i2c. 2013. Disponível em: <http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C.pdf>. Citado na página 44.

CLUSTERBOT! Disponível em: <https://www.instructables.com/id/Clusterbot/>. Citado 2 vezes nas páginas 15 e 51.

FAGUNDES, R. S. Robô pêndulo invertido. 2015. Citado 4 vezes nas páginas 15, 33, 43 e 44.

FILIPEFLOP. *Motor DC 6V 210RPM com Encoder e Adaptador*. Disponível em: <https://www.filipeflop.com/produto/motor-dc-6v-210rpm-com-encoder-e-adaptador/>. Citado na página 49.

FORHAN, N. A. E. Giroscópio mems. São José dos Campos, 2010. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m19@80/2010/01.25.18.42>. Citado na página 38.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. *Sistemas de controle para engenharia*. [S.l.]: Bookman Editora, 2013. Citado na página 43.

FREITAS, C. M. Controle pid em sistemas embarcados. 2014. Disponível em: <https://www.embarcados.com.br/controle-pid-em-sistemas-embarcados/>. Citado 3 vezes nas páginas 15, 45 e 61.

GONÇALVES, H. M. Instrumentação eletrônica de uma bengala para auxiliar no monitramento de marcha de usuários de exoesqueleto inferior. Brasília, DF, 2017. Citado 6 vezes nas páginas 15, 37, 39, 40, 54 e 55.

HA, J.-S.; LEE, J.-J. Position control of mobile two wheeled inverted pendulum robot by sliding mode control. In: IEEE. *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*. [S.l.], 2012. p. 715–719. Citado na página 33.

HAMILTON, C. A. *BeagleBone Black Cookbook*. [S.l.]: Packt Publishing Ltd, 2015. Citado 3 vezes nas páginas 48, 49 e 56.

INSTRUMENTS, I. T. Technical reference manual - am335x and amic110 sitara^{TM} processors. 2017. Disponível em: <http://www.ti.com/lit/ug/spruh73p/spruh73p.pdf>. Citado na página 57.

INVENSENSE. *MPU-9250 Product Specification Revision 1.1*. [S.l.], 2016. Disponível em: <https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>. Citado na página 52.

LIMA, T. Aprenda a medir inclinação com a genuino 101. 2016. Disponível em: <https://www.embarcados.com.br/inclinacao-na-genuino-101/>. Citado na página 55.

LONGARETTI, D.; GIRARDI, A. G.; ENGROFF, A. M. Implementação de uma interface i2c em fpga. *Anais do Salão Internacional de Ensino, Pesquisa e Extensão*, v. 7, n. 2, 2016. Citado 2 vezes nas páginas 15 e 44.

MARTINS, N. A. Sistemas microcontrolados. *Uma abordagem com o Microcontrolador PIC 16F84. Editora Novatec Ltda, 1ª edição*, 2005. Citado na página 43.

MOLLOY, D. *Exploring BeagleBone: tools and techniques for building with embedded Linux*. [S.l.]: John Wiley & Sons, 2014. Citado 8 vezes nas páginas 15, 42, 47, 48, 56, 59, 60 e 68.

MUNIZ, S. R. Resumo sobre controladores pid. 2017. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/4132451/mod_resource/content/0/Resumo_controladores_PID.pdf>. Citado 2 vezes nas páginas 45 e 61.

OGATA, K.; SEVERO, B. *Engenharia de controle moderno*. [S.l.]: Prentice Hall do Brasil, 1998. Citado 3 vezes nas páginas 29, 45 e 61.

OHHIRA, T.; SHIMADA, A. Model predictive control for an inverted-pendulum robot with time-varying constraints. Japão, 2017. Citado 3 vezes nas páginas 15, 33 e 34.

OLIVEIRA, E. L. de. minicurso de arduino - robô equilibrista. 2017. Disponível em: <https://kpacitor.teachable.com/p/robo-equilibrista>. Citado na página 61.

PAUL, S.; GANGULY, S. Using the mpu9250 to get real-time motion data. 2016. Disponível em: <https://www.hackster.io/30503/using-the-mpu9250-to-get-real-time-motion-data-08f011>. Citado 2 vezes nas páginas 15 e 41.

PAULA, F. O. de. Sensores imu - uma abordagem completa - parte 1. 2015. Disponível em: <http://www.decom.ufop.br/imobilis/sensores-imu-uma-abordagem-completa-parte-1/>. Citado na página 55.

PAULA, F. O. de. Sensores imu - uma abordagem completa - parte 2. 2015. Disponível em: <http://www.decom.ufop.br/imobilis/sensores-imu-uma-abordagem-completa-parte-2/>. Citado 3 vezes nas páginas 55, 56 e 66.

PHIDGETS. Encoder primer. 2017. Disponível em: <https://www.phidgets.com/docs/Encoder_Primer#Quadrature_Encoding>. Citado 2 vezes nas páginas 15 e 43.

RODRIGUES, A. P. C. Desenvolvimento de uma interface para acionamento de atuadores e leitura de encoders para um exoesqueleto de membro inferior com a plataforma soc-fpga zybo. Brasília, 2017. Citado 2 vezes nas páginas 15 e 42.

SILVA, C. P. Aplicação de algoritmos evolutivos na calibração de sensores inerciais mems. Londrina, 2016. Citado 3 vezes nas páginas 37, 52 e 53.

SILVA, F. C. da. Planta didática – controle pid digital para motor dc. Curitiba, 2008. Disponível em: <https://www.up.edu.br/blogs/engenharia-da-computacao/wp-content/uploads/sites/6/2015/06/2008.14.pdf>. Citado 2 vezes nas páginas 15 e 45.

SILVA, G. D. et al. Nonlinear modeling, simulation and control of a two-wheeled inverted pendulum. Universidade Federal de Uberlândia, 2017. Citado 3 vezes nas páginas 15, 34 e 35.

TEIXEIRA, M. C. M.; PIETROBOM, H. C.; ASSUNÇÃO, E. Novos resultados sobre a estabilidade e controle de sistemas não-lineares utilizando modelos fuzzy e lmi. *Controle and Automacao*, p. 37–48, 2000. Citado na página 34.

TORRES, H. Sensores inerciais - parte 2. 2015. Disponível em: <https://www.embarcados.com.br/sensores-inerciais-parte-2/>. Citado 3 vezes nas páginas 15, 38 e 39.

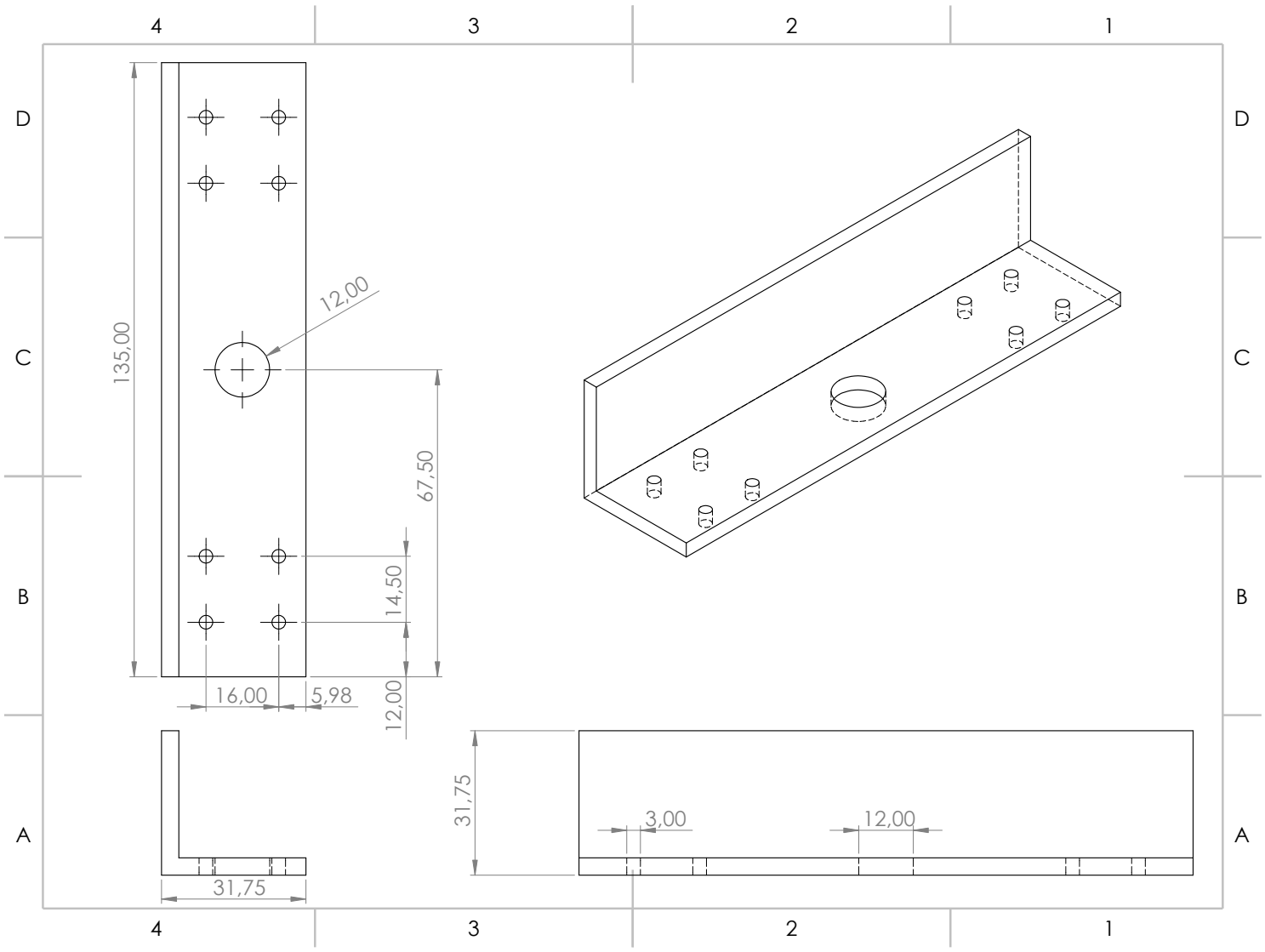
TOSHIBA, S. *TB6612FNG driver IC for dual DC motor*. 2008. Disponível em: <https://www.pololu.com/file/0J86/TB6612FNG.pdf>. Citado 3 vezes nas páginas 15, 17 e 51.

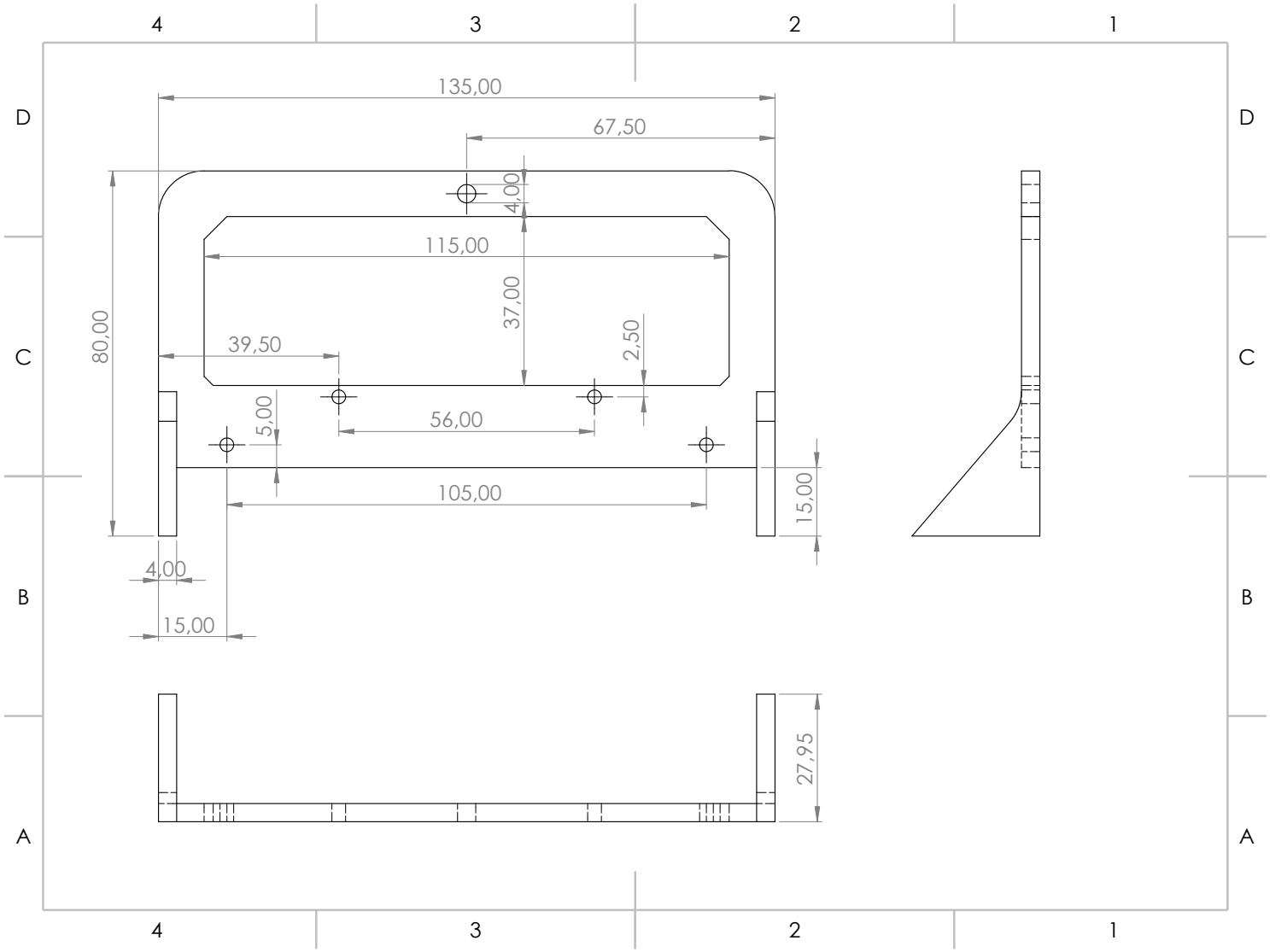
VIDA, J. B. da. Implementação em sistema embarcado de método para estimação de orientação utilizando filtro de kalman, sensores inerciais e magnetômetro. São Carlos, 2016. Citado 3 vezes nas páginas 38, 39 e 41.

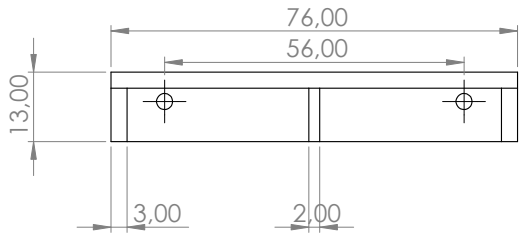
YODER, M. A.; KRIDNER, J. *BeagleBone Cookbook: Software and Hardware Problems and Solutions*. [S.l.]: "O'Reilly Media, Inc.", 2015. Citado na página 49.

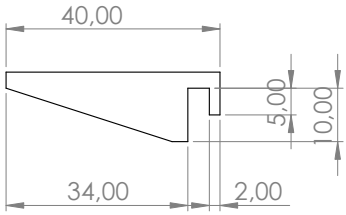
Apêndices

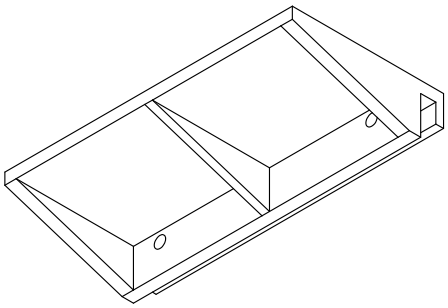
APÊNDICE A – Desenhos técnicos do protótipo

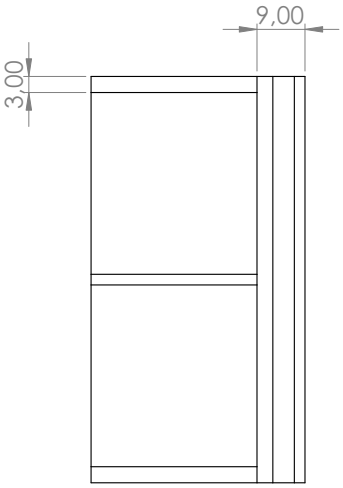

4
3
2
1
D
C
B
A
135,00
12,00
67,50
14,50
16,00
5,98
12,00
31,75
31,75
3,00
12,00

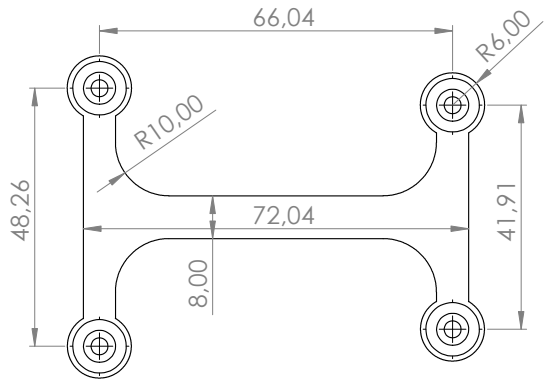

4
3
2
1
D
C
B
A
135,00
67,50
4,00
115,00
37,00
80,00
39,50
2,50
5,00
56,00
105,00
15,00
4,00
15,00
27,95

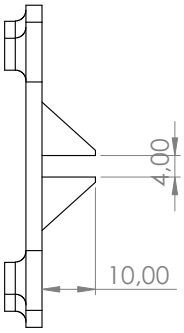

76,00
56,00
13,00
3,00
2,00

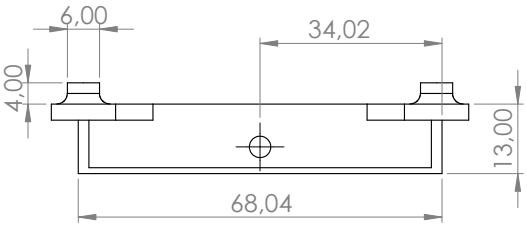

40,00
5,00
10,00
34,00
2,00

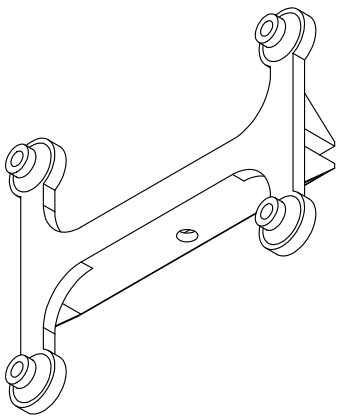


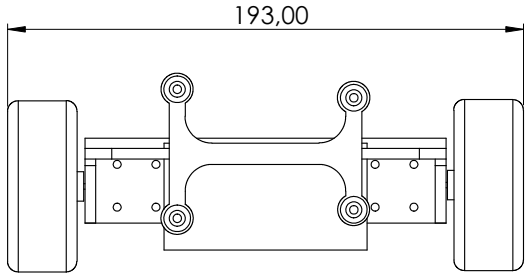

9,00
3,00

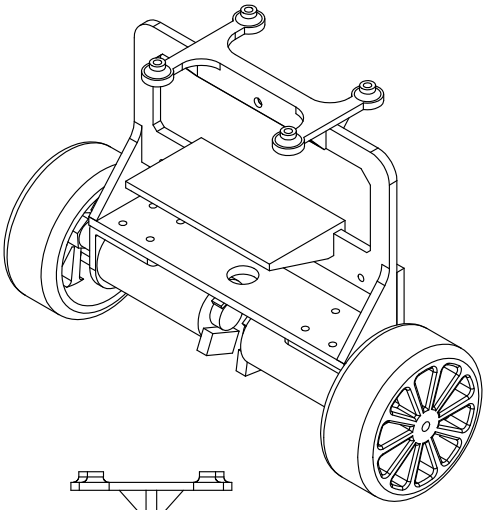

66,04
R6,00
R10,00
48,26
72,04
41,91
8,00

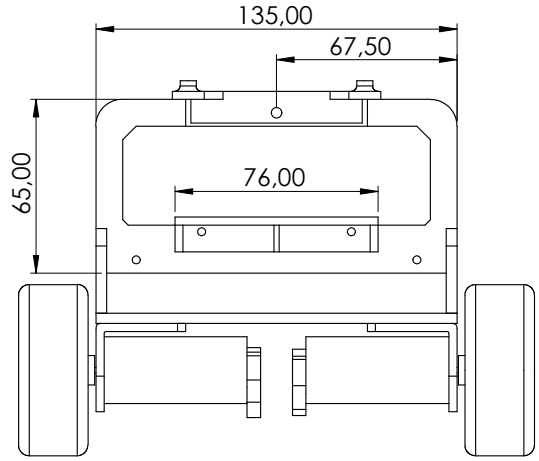

4,00
10,00

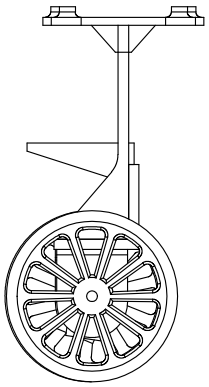

6,00
34,02
4,00
13,00
68,04




193,00




135,00
67,50
65,00
76,00



APÊNDICE B – Repositório do projeto

https://gitlab.com/PedroEugenioML/pendulo_invertido