JOSÉ BATISTA DE OLIVEIRA JÚNIOR

**CONTROLE DE UMA PLATAFORMA INERCIAL ESTABILIZADA COM
TRÊS GRAUS DE LIBERDADE**

São Paulo

2016

JOSÉ BATISTA DE OLIVEIRA JÚNIOR

**CONTROLE DE UMA PLATAFORMA INERCIAL ESTABILIZADA COM TRÊS
GRAUS DE LIBERDADE**

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do título de Mestre em
Ciências

**São Paulo
2016**

JOSÉ BATISTA DE OLIVEIRA JÚNIOR

**CONTROLE DE UMA PLATAFORMA INERCIAL ESTABILIZADA COM TRÊS
GRAUS DE LIBERDADE**

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do título de Mestre em
Ciências

Área de Concentração:
Engenharia de Sistemas

Orientador: Prof. Dr. José Jaime da Cruz

**São Paulo
2016**

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, dede 2016

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Oliveira Jr, José Batista

Controle de uma Plataforma Inercial Estabilizada com Três Graus de Liberdade / J. B. Oliveira Jr -- versão corr. -- São Paulo, 2016.

109 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.

1.ENGENHARIA ELÉTRICA 2.ENGENHARIA ELÉTRÔNICA

3.ENGENHARIA DE CONTROLE E AUTOMAÇÃO I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II.t.

À minha família, que sempre esteve ao meu lado em todos os momentos.

AGRADECIMENTOS

Aos professores José Jaime da Cruz e André Alves Ferreira pela orientação e todo o apoio durante o desenvolvimento desse trabalho.

Aos colegas de trabalho e estudo que contribuíram direta ou indiretamente para a conclusão desse trabalho.

Agradecimento especial à minha esposa Samanta pelo apoio e compreensão, ao pequeno Victor, que renovava minhas energias para seguir em frente a cada sorriso e a minha mãe Maria que sempre me incentivou a buscar o conhecimento e o desenvolvimento humano e profissional.

RESUMO

Esta dissertação apresenta o desenvolvimento de uma plataforma inercial autônoma com três graus de liberdade para aplicação em estabilização de sensores – por exemplo, gravimétricos estacionários e embarcados – podendo ser utilizada também para estabilização de câmeras. O sistema é formado pela Unidade de Medida Inercial, IMU, desenvolvida utilizando um sensor micro eletromecânico, MEMS – que possui acelerômetro, giroscópio e magnetômetros nos três eixos de orientação – e um microcontrolador para aquisição, processamento e envio dos dados ao sistema de controle e aquisição de dados. Para controle dos ângulos de inclinação e orientação da plataforma, foi implementado um controlador PID digital utilizando microcontrolador. Este recebe os dados da IMU e fornece os sinais de controle utilizando as saídas PWM que acionam os motores, os quais controlam a posição da plataforma. Para monitoramento da plataforma foi desenvolvido um programa para aquisição de dados em tempo real em ambiente Matlab, por meio do qual se pode visualizar e gravar os sinais da IMU, os ângulos de inclinação e a velocidade angular. Testou-se um sistema de transmissão de dados por rádio frequência entre a IMU e o sistema de aquisição de dados e controle para avaliar a possibilidade da não utilização de *slip rings* ou fios entre o eixo de rotação e os quadros da plataforma. Entretanto, verificou-se a inviabilidade da transmissão em razão da baixa velocidade de transmissão e dos ruídos captados pelo receptor de rádio frequência durante os movimentos da plataforma. Sendo assim, dois pares de fios trançados foram utilizados para conectar o sensor inercial ao sistema de aquisição e processamento.

Palavras chave: Plataforma inercial. Sensor micro eletromecânico. Estabilização de sensores. Unidade de medida inercial. Controle PID.

ABSTRACT

This work presents the development of a three-degree of freedom autonomous inertial platform for the use in sensors stabilization – for example, stationary and embedded gravimeters. It can also be used to stabilize cameras. The system is composed by the Inertial Measurement Unit, IMU, developed using a micro electromechanical sensor, MEMS – which has an accelerometer, a gyroscope and a magnetometer in the three axes of orientation – and a microcontroller for data acquisition, data processing and data sending to the control and data acquisition system. To control the platform angles and its orientation, a digital PID controller was implemented using a microcontroller. It receives data from the IMU and provides the control signals using the PWM outputs that drive the motors to control the platform position. In order to supervise the platform operation, a real time data acquisition software was developed in Matlab, where IMU signals, inclination angles and angular velocities can be displayed and recorded. Data transmission via radio frequency between the IMU and the data acquisition and control system was tested in order to evaluate the possibility of not using slip rings or wires between the rotation axis and platform frames. This approach was unsuccessful due to the low speed of data transmission and to the noise that affected the radio frequency receiver during the platform's movements. In view of that wire was used to directly connect the inertial sensor to the acquisition and processing system.

Keywords: Inertial platform. Micro electromechanical sensor. Stabilization sensors. Inertial measurement unit. PID control.

LISTA DE FIGURAS

Figura 1: Plataforma Inercial Estabilizada com três graus de liberdade

Figura 2: Coordenadas centralizadas da Terra e Coordenadas geodésicas em relação a elipsoide da Terra

Figura 3: Sistema de coordenadas NED

Figura 4: Coordenadas nos quadros s e t

Figura 5: Velocidades angulares em relação ao quadro-n

Figura 6: Perspectiva do protótipo

Figura 7: Montagem do protótipo

Figura 8: Quadro externo e intermediário

Figura 9: Quadro externo, intermediário e interno

Figura 10: Integração do centro da plataforma

Figura 11a e 11b: Montagem Final da Plataforma

Figura 12: Diagrama de blocos do sistema eletrônico

Figura 13: MPU-9250

Figura 14a: Acelerômetros e giroscópios

Figura 14b: Magnetômetro

Figura 15a e 15b: PWM para controle dos motores

Figura 16: Resposta do sistema, Pitch: $K_p=1$, $K_i=0$, $K_d=0$; Roll: $K_p=1$, $K_i=0$, $K_d=0$

Figura 17: Resposta do sistema, Pitch: $K_p=10.3$, $K_i=0$, $K_d=0$; Roll: $K_p=3.3$, $K_i=0$, $K_d=0$

Figura 18: Sinais de erro – Ganho proporcional

Figura 19: Resposta do sistema, Pitch: $K_p=10.3$, $K_i=0.55$, $K_d=0$; Roll: $K_p=3.3$, $K_i=0.3$, $K_d=0$

Figura 20: Sinais de erro – Ganho Proporcional e Integral

Figura 21: Resposta do sistema, Pitch: $K_p=10.3$, $K_i=0.55$, $K_d=10$; Roll: $K_p=3.3$, $K_i=0.3$, $K_d=5$

Figura 22: Sinais de erro – Ganho PID

Figura 23: Programa de aquisição de dados

Figura 24: Interferência eletromagnética no ângulo yaw (α)

Figura 25: Ângulos de pitch (η) e roll (χ) em regime estacionário

Figura 26: Ângulos de pitch (η) e roll (χ) determinação do ruído bruto e filtrado

Figura 27: Deriva e nível de ruído dos giroscópios

Figura 28: Sinais dos acelerômetros

Figura 29: Inclinação teste de navegação

Figura 30: Resposta do 1º teste de navegação

Figura 31: Variável de controle – 1º teste de navegação

Figura 32: Resposta do 1º teste de navegação – 1º ciclo

Figura 33: Resposta do 1º teste de navegação – 2º ciclo

Figura 34: Resposta do 2º teste de navegação

Figura 35: Variável de controle - 2º teste de navegação

Figura 36: Resposta do 2º teste de navegação – 1º ciclo

Figura 37: Resposta do 2º teste de navegação – 2º ciclo

LISTA DE ABREVIATURAS E SIGLAS

IMU	Unidade de Medida Inercial (Inertial Measure Unit)
MEMS	Micro Sistema Eletro-Mecânico (Micro-Electro-Mechanical Systems)
NED	Sistema de Coordenadas Geográficas (North East Down)
USB	Protocolo de Comunicação (Universal Serial Bus)
RS-232	Protocolo de Comunicação Serial
Giro	Giroscópios
Acc	Acelerômetro
Mag	Magnetômetros
I ² C	Interface de Comunicação (Inter-Integrated Circuit)
DMP	Processador Digital de Movimento (Digital Motion Processor)
FIFO	Estrutura de Entrada e Saída de Dados (First In, First Out)
ADC	Conversor Analógico Digital (Analog Digital Converter)
quadro-n	Quadro de navegação
quadro-e	Quadro fixado a Terra
quadro-b	Quadro da base/corpo
quadro-i	Quadro inercial
quadro-s	Quadro concêntrico com orientação relativa arbitrária
quadro-t	Quadro concêntrico com orientação relativa arbitrária
RF	Rádio frequência

LISTA DE SIMBOLOS

η	Ângulo de inclinação da plataforma (pitch)
χ	Ângulo de inclinação da plataforma (roll)
α	Ângulo de inclinação da plataforma (yaw)
x, y, z	Sentido de Orientação nos Eixos (Norte, leste e Vertical)
F	Força aplicada
m_i	Massa inercial de uma partícula
$\dot{x}$	Velocidade de uma partícula
$\ddot{x}$	Aceleração de uma partícula
m_g	Massa gravitacional
g	Aceleração da gravidade
F_g	Força gravitacional resultante
M	Massa de uma partícula
K	Constante gravitacional de Newton
a	Aceleração de uma partícula
ϕ	Latitude geodésica
λ	Longitude geodésica
h	Altura da elipsoide
x^s	Coordenadas de um ponto no quadro-s
x^t	Coordenadas de um ponto no quadro-t
C_t^s	Matriz de transformação do quadro-t para o quadro-s
C_i^e	Matriz de transformação do quadro-i para o quadro-e

C_n^e	Matriz de transformação do quadro-n para o quadro-e
θ	Ângulo de inclinação
Ψ	Matriz simétrica de inclinação (skew symmetric matrix)
$\alpha \ \eta \ \chi$	Ângulos de Euler
ω_e	Velocidade angular de rotação da Terra.
ω_{ie}^e	Velocidade angular do quadro-e em relação ao quadro-i
ω_{en}^n	Velocidades angulares no quadro-n referentes ao quadro-e
ω_{nb}^b	Velocidades angulares no quadro-b referentes ao quadro-n dadas pelos giroscópios
v^n	Vetor de velocidade
m^n	Vetor de campo magnético
a^n	Vetor de aceleração
$\bar{g}^n$	Vetor de gravidade
Kp	Ganho proporcional do controlador PID
Ki	Ganho integral do controlador PID
Kd	Ganho derivativo do controlador PID
Ti	Tempo de integração do controlador PID
Td	Tempo derivativo do controlador PID
FK	Filtro de Kalman

SUMARIO

1. INTRODUÇÃO	16
1.1 Objetivo	18
1.2 Justificativas	19
1.3 Revisão Bibliográfica	20
1.4 Estrutura do trabalho	21
2. FUNDAMENTOS TEÓRICOS	22
2.1 Quadro de Navegação Inercial	22
2.2 Transformação de Coordenadas	23
2.3 Matrizes de Rotação	24
2.4 Ângulos de Euler	25
3. MATERIAIS E MÉTODOS	29
3.1. Sistema mecânico do Protótipo.....	29
3.1.1. Montagem da plataforma	29
3.2. Sistema Eletrônico	32
3.2.1. Unidade de Medida Inercial	33
3.2.2. Filtro de Kalman	35
3.2.3. Sistema de Controle.....	37
3.3. Programa de aquisição de dados.....	44
3.4. Materiais e equipamentos utilizados.....	45
3.4.1. Sistema mecânico	45
3.4.2. Sistema eletrônico	45
3.4.3. Equipamentos utilizados	46
4. RESULTADOS E DISCUSSÃO.....	47
4.1.. Parâmetros do controlador	47
4.2. Análise e discussão dos resultados	48
4.2.1. Plataforma em regime estacionária.....	48
4.2.2. Plataforma em regime de Navegação	51
5. CONCLUSÕES	58

REFERENCIAS BIBLIOGRÁFICAS.....	60
APENDICE A – PROGRAMA DE AQUISIÇÃO DE DADOS EM TEMPO REAL	62
APENDICE B – PROGRAMA DO MICROCONTROLADOR PRINCIPAL PARA AQUISIÇÃO DE DADOS E CONTROLE	90
APENDICE C – PROGRAMA DO MICROCONTROLADOR AUXILIAR PARA AQUISIÇÃO DE DADOS.....	98
APENDICE D – PROGRAMA DO MICROCONTROLADOR PARA APLICAÇÃO DO FILTRO DE KALMAN	102
APENDICE E – PROGRAMA DO MICROCONTROLADOR PARA CONFIGURAÇÃO DO MPU-9250	103
ANEXO 1 – DIAGRAMA ELÉTRICO IMU	108

1. INTRODUÇÃO

Sistemas de estabilização inercial são objetos de estudos em diferentes áreas, como na aviação comercial e militar, marinha, aeroespacial e sistemas estacionários para aplicações científicas, comerciais e militares, como por exemplo, estabilização de sensores aplicados em estudos e pesquisas geofísicas, estabilização de câmeras em aeronaves, helicópteros e automóveis, orientação de armamento militar, lançamento de satélites e orientação de telescópios.

Uma plataforma inercial é composta por:

1. Um conjunto de três quadros montados um sobre o outro, com eixos de articulação ortogonais para garantir que um objeto montado sobre o quadro interno se mantenha orientado independente da movimentação e ou perturbação na sua base de fixação.
2. Uma unidade de medida inercial (IMU), composta por um conjunto de sensores:
 - Acelerômetros, medem as acelerações;
 - Giroscópios, medem as velocidades angulares;
 - Magnetômetros, medem as componentes do campo magnético.

A IMU é montada em uma base comum para manter a mesma orientação e garantir a manutenção da taxa de estimativa da posição, velocidade e atitude em relação ao local ou veículo onde está instalada, como ilustrado na figura (1).

Figura 1: Plataforma Inercial Estabilizada com três graus de liberdade [21]

A estabilização e a orientação da plataforma são realizadas por três motores: um deles localizado no centro da plataforma, responsável por controlar a orientação no eixo de yaw (α), o segundo e o terceiro motores são responsáveis pela inclinação da plataforma, respectivamente nos eixos pitch (η) e roll (χ), conforme figura (1).

Utilizando os sinais dos acelerômetros e giroscópios para a estabilização da plataforma, ocorrem dois problemas que afetam consideravelmente o cálculo dos ângulos de inclinação e orientação e conseqüentemente exigem maior esforço de controle, são eles: o ruído e a deriva dos sinais respectivamente provenientes dos acelerômetros e giroscópios [1] e [2].

Conforme descrito em [20] foi implementado o Filtro de Kalman discreto, a ser apresentado na seção 3.2.2. Utilizando os ângulos calculados com base nos sinais dos acelerômetros e as velocidades angulares obtidas dos giroscópios como sendo os estados do sistema, o ruído dos acelerômetros e a deriva dos giroscópios foram reduzidos ou eliminados, garantindo maior precisão nos ângulos da plataforma.

Para o controle da plataforma é utilizado um controlador PID digital implementado no microcontrolador, este fornece os sinais de comando para a ponte H, responsável pelo controle do torque e orientação de rotação dos motores, que será apresentado na seção 3.2.3.

Além da IMU, do Filtro de Kalman e do controlador PID, foi implementado no mesmo microcontrolador o sistema de transmissão de dados a um computador via comunicação USB para monitoramento e gravação dos dados pelo programa de aquisição a ser apresentado na seção 3.3.

1.1. Objetivo

O objetivo deste trabalho é o desenvolvimento de uma plataforma inercial estabilizada com três graus de liberdade, pitch (η), roll (χ) e yaw (α) e um sistema de aquisição de dados em tempo real para monitoramento e gravação dos dados da plataforma durante seu funcionamento.

A plataforma poderá ser aplicada a sistemas estacionários ou embarcados utilizados para estabilização e orientação de sensores gravimétricos, câmeras fotográficas e de vídeo e equipamentos e sensores que precisem manter sua orientação independente de perturbações externas.

1.2. Justificativas

Sistemas inerciais complexos geralmente têm custos de desenvolvimento e produção elevados, inviabilizando seu uso para pequenas e médias aplicações.

Com o desenvolvimento da microeletrônica e a produção de MEMS, como acelerômetros, giroscópios e magnetômetros integrados em um único componente, os custos foram reduzidos, facilitando o acesso aos componentes para aplicação em pequenos e médios sistemas inerciais, smartphones, tablets e notebooks.

A utilização dos MEMS proporciona melhor custo benefício para pequenos sistemas inerciais em relação aos sistemas que utilizam acelerômetros e giroscópios eletromecânicos de alta precisão. Suas faixas de operação podem ser ajustadas de acordo com a aplicação, para movimentos rápidos ou lentos, e os sinais são transmitidos digitalmente através da interface de comunicação I_2C , evitando perda de dados e ruídos na transmissão.

A integração dos MEMS com um sistema eletrônico de aquisição e processamento e um sistema mecânico composto de uma base giratória, quadros, rolamentos e motores possibilita o desenvolvimento de uma plataforma inercial de baixo custo com precisão, sensibilidade de orientação e posicionamento para atender as especificações em diversas aplicações acadêmicas, industriais e de pesquisa.

1.3. Revisão bibliográfica

Nesta seção está apresentada e brevemente descrita, em seus aspectos relevantes, a bibliografia utilizada como base no desenvolvimento da dissertação.

O trabalho (Bredenkamp, A.F.L., 2007) apresenta o desenvolvimento e controle de uma plataforma inercial aplicada à estabilização de câmera de vídeo aerotransportada, baseado na teoria de funcionamento de um giroscópio com três graus de liberdade, com a utilização de sensores analógicos para determinação dos ângulos da plataforma, sinais dos acelerômetros e giroscópios que foram aplicados para determinação dos parâmetros dos controladores implementados.

Em (Khodadadi; Motlagh; Gorji, 2011) está a modelagem linear e implementação de um controle robusto de um sistema inercial com dois graus de liberdade, azimute e elevação, aplicado para estabilização de um sistema óptico. Na modelagem do sistema linearizado foram consideradas todas as não linearidades de torque, como a distúrbios devidos à inércia dos quadros, fricção, ruídos e distúrbios externos. Dessa forma o uso de um controlador linear PI foi aplicado ao sistema atendendo às especificações do projeto.

(McCallum; Lahiji; Meregany, 2011) apresenta o desenvolvimento de uma micro Unidade de Medida Inercial aplicada a um sistema estacionário, composta por um microcontrolador, memória, acelerômetro e giroscópio com saída de dados digital, aplicada no controle de uma micro plataforma com três graus de liberdade.

(Miranda, J.L.C; 2009) apresenta a aplicação do Filtro de Kalman em conjunto com controlador PID para controle de um pendulo invertido. O sistema foi desenvolvido integrando acelerômetro e giroscópio para controle do posicionamento e velocidade de dois motores responsáveis pelo controle do pêndulo invertido. Utilizou-se sistema microcontrolado para implementação do FK e do controlador PID. Parte da teoria aplicada é utilizada como base para a implementação do FK neste trabalho.

1.4. Estrutura do Trabalho

A presente dissertação apresenta o desenvolvimento de uma plataforma inercial com três graus de liberdade e um sistema de aquisição e processamento de dados para monitoramento e gravação dos dados em um computador pessoal.

O capítulo 1 apresenta a introdução aos sistemas inerciais, os objetivos do trabalho, justificativas, a metodologia aplicada e revisão bibliográfica.

O capítulo 2 apresenta os fundamentos teóricos utilizados no desenvolvimento do sistema.

O capítulo 3 apresenta os materiais e métodos utilizados, desenvolvimento do plataforma, hardware e software de aquisição de dados e controle, técnica de controle aplicada ao sistema, materiais e equipamentos utilizados.

O capítulo 4 apresenta os resultados obtidos e discussão.

O capítulo 5 apresenta as conclusões, considerações sobre alguns aspectos que tiveram que ser mudados durante o trabalho e sugestões de melhorias para trabalhos futuros.

2. FUNDAMENTOS TEÓRICOS

2.1. Quadro de Navegação Inercial

Considerando o sistema referencial geodésico como um sistema de coordenadas curvilíneas (ϕ, λ) que define a direção da NORMAL para um elipsoide adotado de revolução. A latitude geodésica (ϕ) de um ponto é o ângulo no plano meridional da NORMAL através do ponto, com relação à Linha do Equador, a latitude pode ser positiva ao norte e negativa ao sul, a longitude (λ) é o ângulo no plano equatorial, do Meridiano de Greenwich ao plano meridional do ponto. A altura da elipsoide, (h) , é a distância a partir da elipsoide ao longo da normal até o ponto.[1]

As coordenadas (ϕ, λ, h) constituem o conjunto de coordenadas ortogonais que podem ser utilizadas para o plano cartesiano de coordenadas para descrever as posições do quadro conforme figura 2.

Figura 2: Coordenadas centralizadas a Terra e Coordenadas Geodésicas em relação a elipsoide da Terra [1]

O sistema de coordenadas local pode ser definido como um conjunto de eixos de coordenadas cartesianas em que o primeiro eixo aponta para o norte, o segundo eixo aponta para o leste e o terceiro eixo está alinhado com a normal elipsoidal como ilustrado na figura 3.

Figura 3: Sistema de coordenadas NED [1]

2.2. Transformação de Coordenadas

A transformação de coordenadas se faz necessária para obtenção das coordenadas de um quadro para outro priorizando sua orientação relativa e como as coordenadas dos quadros são ortogonais, três ângulos são suficientes para descrever sua orientação relativa, figura 4. A transformação é linear e pode ser obtida pela direção dos cossenos e ângulos de rotação (Euler). [1]

Figura 4: Coordenadas nos quadros s e t [1]

Considerando dois quadros concêntricos, quadro-s e quadro-t as coordenadas de um ponto podem ser representadas por um vetor x^s no quadro-s e x^t no quadro-t.

2.3. Matrizes de Rotação

As coordenadas de um ponto em qualquer quadro são ortogonais as projeções dos vetores nos respectivos eixos do quadro. [1]

$$x^s = \begin{pmatrix} x_1^s \\ x_2^s \\ x_3^s \end{pmatrix}; x^t = \begin{pmatrix} x_1^t \\ x_2^t \\ x_3^t \end{pmatrix} \quad (2.1)$$

$$x^s = x_1^s e_1^s + x_2^s e_2^s + x_3^s e_3^s \quad (2.2)$$

$$x^t = x_1^t e_1^t + x_2^t e_2^t + x_3^t e_3^t \quad (2.3)$$

Onde $x_j^s = e_j^s \cdot x^s, j = 1, 2, 3 \dots$ são as coordenadas no quadro-s, e $e_j^s, = 1, 2, 3 \dots$ corresponde aos vetores unitários ao longo dos eixos do quadro, procedimento similar para as coordenadas e vetores unitários para o quadro-t.

Analogamente, as coordenadas do vetor unitário e_k^t do quadro-t, no quadro-s são:

$$c_{j,k} = e_j^s \cdot e_k^t \quad (2.4)$$

Logo

$$e_k^t = c_{1,k} e_1^s + c_{2,k} e_2^s + c_{3,k} e_3^s \quad (2.5)$$

Substituindo (2.45) para cada k e comparando o resultado com (2.2), temos a transformação de coordenadas.

$$x^s = C_t^s x^t$$

Assim, a matriz de transformação é:

$$C_t^s = \begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix}$$

Da equação (2.4) e da propriedade do produto escalar de vetores, os coeficientes $c_{j,k}$ são os cossenos dos ângulos entre os j^{th} eixos do quadro-s e os k^{th} eixos do quadro-t. Podemos então chamar a matriz C_t^s de matriz de rotação ou matriz de direção dos cossenos, onde as colunas k^{th} de C_t^s representam o vetor unitário e_k^t no quadro-s e as linhas j^{th} representam o vetor unitário e_j^s no quadro-t, assim, as linhas e colunas são mutualmente ortogonais, logo, a matriz C_t^s é ortogonal, isto implica que todos os elementos da matriz são dependentes o que significa que a orientação é regida por três graus de liberdade.

$$C_t^s (C_t^s)^T = I \Rightarrow C_t^s \equiv (C_t^s)^{-1} = (C_t^s)^T \quad (2.6)$$

A transformação de uma matriz $A_{3 \times 3}$ considerando a matriz de transformação ortogonal C_t^s , pode ser descrita da seguinte forma:

$$y^t = A^t x^t \Rightarrow C_s^t y^s = A^t C_s^t x^s \Rightarrow y^s = C_t^s A^t C_s^t x^s \quad (2.7)$$

Logo:

$$A^s = C_t^s A^t C_s^t \quad (2.8)$$

2.4. Ângulos de Euler

A orientação relativa dos quadros-s/t, pode ser descrita por meio de uma sequência de rotações. Uma alternativa para a transformação da matriz cossenos diretores é a aplicação sucessiva das matrizes de rotação em torno de um eixo específico, adotando a convenção de rotação ZYX dos ângulos de Eules, conforme descrito em [1] e [5].

Rotação em torno do eixo Z, movimento de *yaw* (α) :

Rotação em torno do eixo Y, movimento de *pitch* (η):

Rotação em torno do eixo X, movimento de *roll* (χ) :

As matrizes de rotação para cada eixo são:

$$R_1(\chi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\chi & \sen\chi \\ 0 & -\sen\chi & \cos\chi \end{pmatrix}; R_2(\eta) = \begin{pmatrix} \cos\eta & 0 & -\sen\eta \\ 0 & 1 & 0 \\ \sen\eta & 0 & \cos\eta \end{pmatrix};$$

$$R_3(\alpha) = \begin{pmatrix} \cos\alpha & \sen\alpha & 0 \\ -\sen\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2.9)$$

R_j representa a rotação em torno do eixo j^{th} pelo ângulo de referência, positivo de acordo com a regra da mão direita. Cada R_j é uma matriz de cossenos particular, sendo ela ortogonal e sua inversa é a matriz de rotação inversa $R_j^{-1} = R_j$. [1]

Considerando que o quadro-s é o resultado da rotação no quadro-t, primeiro no eixo-1 (χ), depois no eixo-2 (η), e por último no eixo-3 (α), a matriz de transformação total do quadro-t para o quadro-s é $R_3(\alpha)R_2(\eta)R_1(\chi)$ [1], logo:

$$x^s = R_3(\alpha)R_2(\eta)R_1(\chi)x^t \quad (2.10)$$

Quando usamos os cossenos diretores ou ângulos de Euler, a transformação deve ser a mesma.

$$x_t^s = R_3(\alpha)R_2(\eta)R_1(\chi) \quad (2.11)$$

Sendo $R_3(\alpha)R_2(\eta)R_1(\chi)$ igual a:[1]

$$\begin{pmatrix} \cos(\alpha)\cos(\eta) & \cos(\alpha)\sen(\eta)\sen(\chi) + \sen(\alpha)\cos(\chi) & -\cos(\alpha)\sen(\eta)\cos(\chi) + \sen(\alpha)\sen(\chi) \\ -\sen(\alpha)\cos(\eta) & -\sen(\alpha)\sen(\eta)\sen(\chi) + \cos(\alpha)\sen(\alpha)\sen(\eta)\cos(\chi) & \sen(\alpha)\sen(\eta)\cos(\chi) + \cos(\alpha)\sen(\chi) \\ \sen(\eta) & -\cos(\eta)\sen(\chi) & \cos(\eta)\cos(\chi) \end{pmatrix} \quad (2.12)$$

A relação entre os ângulos de Euler e os cossenos diretores depende da escolha dos ângulos e da sua ordem de rotação, usando a transformação da equação (2.10), os cossenos diretores $c_{j,k}$ são os elementos da matriz em (2.12). Sejam:

$$\chi = \tan^{-1}\left(\frac{-c_{3,2}}{c_{3,3}}\right); \eta = \sen^{-1}(c_{3,1}); \alpha = \tan^{-1}\left(\frac{-c_{2,1}}{c_{1,1}}\right) \quad (2.13)$$

Para pequenos ângulos, χ, η, γ , podemos aproximar $\cos(\chi) \approx 1, \sin(\chi) \approx \chi$, e assim para α, η , teremos:

$$R_3(\alpha)R_2(\eta)R_1(\chi) \approx \begin{pmatrix} 1 & \alpha - \eta \\ -\alpha & 1 & \chi \\ \eta & -\chi & 1 \end{pmatrix} = I_{3 \times 3} - \begin{pmatrix} 0 & -\alpha & \eta \\ \alpha & 0 & -\chi \\ -\eta & \chi & 0 \end{pmatrix} = I - \Psi \quad (2.14)$$

Sendo Ψ uma matriz anti-simétrica para pequenos ângulos de rotação (α, β, γ) .

$$C_t^s \approx I - \Psi = \begin{pmatrix} 1 & \alpha - \eta \\ -\alpha & 1 & \chi \\ \eta & -\chi & 1 \end{pmatrix} \quad (2.15)$$

Considerando pequenos ângulos de rotação entre os respectivos eixos do quadro-s e quadro-t, a ordem de rotação nos eixos é irrelevante, como mostra a transformação reversa dada por:

$$C_s^t \approx \begin{pmatrix} 1 & \alpha - \eta \\ -\alpha & 1 & \chi \\ \eta & -\chi & 1 \end{pmatrix}^T = I - \begin{pmatrix} 0 & \alpha & -\eta \\ -\alpha & 0 & \chi \\ \eta & -\chi & 0 \end{pmatrix} = I - \Psi^T \quad (2.16)$$

Utilizando os ângulos de Euler para definirmos uma transformação de coordenadas adicional, do quadro-b para o quadro-n, os ângulos nos eixos de roll (η), pitch (χ) e yaw (α) da plataforma com relação aos eixos NED, conforme orientações mostradas na figura 5, são definidos como:

Figura 5: Velocidades angulares em relação ao quadro-n [17]

$$C_b^n = R_3(-\alpha)R_2(-\chi)R_1(-\eta) \quad (2.17)$$

A relação entre as taxas de velocidade angulares ω_{nb}^b e as taxas de atitudes $\dot{\eta}, \dot{\chi}$ e $\dot{\alpha}$.

$$C_b^n = R_3(-\alpha)R_2(-\chi)R_1(-\eta) = C_{b_2}^n C_{b_1}^{b_2} C_b^{b_1} \quad (2.18)$$

Onde os quadros b_1 e b_2 são quadros intermediários estabelecidos por η e χ . Os quadros b , b_1 e b_2 , rotacionam com o tempo em relação a eles e o quadro-n, logo:

$$\omega_{nb}^b = \omega_{nb_2}^b + \omega_{b_2b_1}^b + \omega_{b_1b}^b = \omega_{b_1}^b C_{b_2}^{b_1} \omega_{nb_2}^{b_2} + C_{b_1}^b \omega_{b_2b_1}^{b_1} + \omega_{b_1b}^b \quad (2.19)$$

$$\omega_{b_1b}^b = \begin{pmatrix} \dot{\eta} \\ 0 \\ 0 \end{pmatrix}; \omega_{b_2b_1}^{b_1} = \begin{pmatrix} 0 \\ \dot{\chi} \\ 0 \end{pmatrix}; \omega_{nb_2}^{b_2} = \begin{pmatrix} 0 \\ 0 \\ \dot{\alpha} \end{pmatrix} \quad (2.20)$$

Usando as equações (2.18) e (2.19), temos:

$$\omega_{nb}^b = R_1(\eta)R_2(\chi) \begin{pmatrix} 0 \\ 0 \\ \dot{\alpha} \end{pmatrix} + R_1(\eta) \begin{pmatrix} 0 \\ \dot{\chi} \\ 0 \end{pmatrix} + \begin{pmatrix} \dot{\eta} \\ 0 \\ 0 \end{pmatrix}$$

$$\omega_{nb}^b = \begin{pmatrix} 1 & 0 & -\sin\chi \\ 0 & \cos\eta & \cos\chi\sin\eta \\ 0 & -\sin\eta & \cos\chi\cos\eta \end{pmatrix} \begin{pmatrix} \dot{\eta} \\ \dot{\chi} \\ \dot{\alpha} \end{pmatrix} \quad (2.21)$$

Resultando em:

$$\begin{pmatrix} \dot{\eta} \\ \dot{\chi} \\ \dot{\alpha} \end{pmatrix} = \begin{pmatrix} 1 & \sin(\eta)\tan(\chi) & \cos(\eta)\tan(\chi) \\ 0 & \cos(\eta) & -\sin(\eta) \\ 0 & \sin(\eta)\sec(\chi) & \cos(\eta)\sec(\chi) \end{pmatrix} \omega_{nb}^b \quad (2.22)$$

Ao integrar as velocidades angulares obtemos os ângulos pitch (η), roll (χ) e yaw (α) da plataforma em função do tempo dadas as velocidades angulares fornecidas pelos giroscópios.

3. MATERIAIS E MÉTODOS

3.1. Sistema mecânico do Protótipo

Foi desenvolvida uma plataforma em metal formada por três quadros montados um dentro do outro com eixos de articulação ortogonais e uma base circular definida como o centro da plataforma.

Nos quadros externo e intermediário foram fixados respectivamente os motores dos eixos de pitch (η) e roll (χ) responsáveis por manter a orientação vertical do quadro interno, onde está fixado o centro da plataforma. Em sua parte inferior, foi instalado o motor responsável pelo eixo yaw (α), conforme figura 6. Os quadros e a mesa foram montados para permitir que sua inclinação nos eixos de pitch (η) e roll (χ) pudessem ser de até $\pm 45^\circ$ e o eixo yaw (α) pudesse girar em torno de si mesmo de 0 a 360° . As especificações do protótipo serão apresentadas na seção 3.4.

Figura 6: Perspectiva do protótipo

3.1.1. Montagem da plataforma

O quadro interno, eixo de pitch (η), é formado por uma barra de metal, na qual foi fixado o eixo yaw (α) e na parte inferior o motor responsável pela sua orientação, conforme figura 7.

Figura 7: Montagem do protótipo

O quadro externo foi montado com chapa de alumínio em formato de U e acoplado a uma base circular, nesse quadro estão fixados o motor responsável pelos movimentos no eixo de roll (χ) e o quadro intermediário conforme figura 8.

Figura 8: Quadro externo e intermediário

Ao quadro intermediário, eixo de roll (χ), formado por uma base circular em metal, um contrapeso para balancear o centro de massa da plataforma e o motor responsável pelo eixo pitch (η), foi acoplado o eixo interno de pitch (η), figura 9.

Figura 9: Quadro externo, intermediário e interno

O centro da plataforma foi fixado na parte superior do quadro interno com o uso de um rolamento e um dispositivo de fixação com quatro pontos para facilitar o alinhamento com o quadro intermediário, conforme figura 10.

Figura 10: Integração do centro da plataforma

Após a integração do sistema mecânico, a IMU foi instalada na plataforma da seguinte maneira: sensor inercial MPU-9250 foi instalado na mesa central da plataforma com um par de fios para alimentação e outro par de fios para comunicação, ambos passando pelo centro do eixo de rotação e pelo eixo oposto ao motor de roll (χ) até ser ligado ao sistema eletrônico que está fixado na base da plataforma conforme figuras 11a e 11b.

Figura 11a: Montagem Final da Plataforma

Figura 11b: Montagem Final da Plataforma

3.2. Sistema Eletrônico

O sistema eletrônico consiste em duas IMUs, responsáveis pela aquisição, processamento e envio dos dados recebidos dos MEMS ao programa de aquisição de dados, drivers para acionamento dos motores e programa de aquisição de dados conforme figura 12. A IMU principal é responsável também pelo sistema de controle da plataforma, a IMU auxiliar fornece a inclinação da base da plataforma ou do veículo em que estiver instalada, conforme descrito no Apêndice C.

Figura 12: Diagrama de blocos do sistema eletrônico

3.2.1. Unidade de Medida Inercial

A IMU baseia-se na integração do MEMS Invensense MPU-9250 [8] e [11], mostrado na figura 13 e esquema elétrico no anexo 1. Este possui termômetro, acelerômetros, giroscópios e magnetômetros em três eixos conforme orientação nas figuras 14a e 14b com o microcontrolador Microchip PIC18F4550 [9] que é responsável pela inicialização do MEMS, aquisição, processamento e envio dos sinais. Também foram implementados o Filtro de Kalman e o controlador PID, que serão apresentados nas seções 3.2.2 e 3.2.3.

Figura 13: MPU-9250 [8]

Figura 14a: Acelerômetros e giroscópios [8]

figura 14b: Magnetômetros [8]

O microcontrolador utiliza um cristal oscilador de 20MHz em conjunto com seu PLL interno gera um clock de 48MHz. Para se comunicar com o MEMS o PIC18F4550 utiliza o protocolo de comunicação I^2C , conforme descrito em [8]. Para a comunicação com o software de aquisição de dados, foi utilizada a transmissão de dados via USB. Para envio dos sinais de controle aos drivers dos motores foram utilizadas as saídas PWM do microcontrolador, conforme apêndice B

Ao energizar a IMU, o microcontrolador envia à MPU-9250 as configurações necessárias para sua inicialização e autocalibração conforme descrito no Apêndice E. Em seguida inicia-se o processo de leitura dos dados dos acelerômetros, giroscópios, magnetômetros. A cada ciclo de leitura são calculados os ângulos da plataforma pitch (η) e roll (χ), conforme equações (3.1) e (3.2).

$$\eta = \tan^{-1} \left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}} \right) \quad (3.1)$$

$$\chi = \tan^{-1} \left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}} \right) \quad (3.2)$$

Para o cálculo do ângulo de yaw (α), primeiramente foi realizado o procedimento para compensação dos efeitos magnéticos próximos ao MEMS, conforme descrito em [19] obtendo os coeficientes a serem aplicados nos dados fornecidos pelos magnetômetros, m_y e m_x , conforme apêndice A. Substituindo (3.3) e (3.4) em (3.5) temos o ângulo yaw (α).

$$M_y = m_y \cos(\eta) + m_z \sin(\eta) \quad (3.3)$$

$$M_x = m_x \cos(\chi) + m_y \sin(\chi) - m_z \cos(\chi) \sin(\eta) \quad (3.4)$$

$$\alpha = \tan^{-1} \left(\frac{M_y}{M_x} \right) \quad (3.5)$$

Os ângulos da plataforma apresentam ruídos e derivas ao longo do tempo, provenientes respectivamente dos acelerômetros e dos giroscópios. Para minimizar os efeitos dos ruídos e das derivas, foi aplicado o Filtro de Kalman, que será apresentado na seção 3.2.2. O algoritmo para implementação do Filtro de Kalman está descrito no apêndice D. Após a filtragem dos sinais os dados são transmitidos ao computador via USB, conforme descrito no apêndice B.

3.2.2. Filtro de Kalman

O Filtro de Kalman é um filtro recursivo que fornece a estimativa do estado de um processo com base em dados medidos, minimizando o erro quadrático.

Conforme descrito em [20] e [18], para a implementação do Filtro de Kalman, o sistema deve ser modelado como um sistema linear, sendo descrito pela representação em espaço de estados conforme equações (3.6) e (3.7).

$$x_k = Fx_{k-1} + Bu_k + w_k \quad (3.6)$$

$$z_k = Hx_k + v_k \quad (3.7)$$

Em que, no instante k , x_k é o vetor de estados, u_k é o vetor de entrada (controle), z_k é o vetor de variáveis medidas, enquanto w_k e v_k são os vetores de ruído no estado (no proceso) e na medida, respectivamente. F , B e H são as matrizes do modelo de estados, cujas dimensões são compatíveis com as operações indicadas.

Os ruídos no estado w_k e na medida v_k , supostos não correlacionados entre si, são gaussianos, de média nula e covariâncias Q_k e R_k , respectivamente.

Na prática, o uso do Filtro de Kalman tem por objetivo diminuir o efeito dos ruídos provenientes dos sinais dos acelerômetros – utilizando os sinais dos giroscópios – e eliminar o efeito da deriva dos giroscópios – utilizando os sinais dos acelerômetros –, garantindo maior precisão na determinação dos ângulos da plataforma.

Um modelo simplificado utilizando os ângulos da plataforma calculados através dos sinais dos acelerômetros e as velocidades angulares provenientes dos giroscópios é descrito pela equação (3.6) com:

$$\begin{aligned}
x_k &= \begin{bmatrix} \theta_k \\ \dot{\theta}_k \end{bmatrix} \\
F &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \\
B &= \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \\
H &= [1 \quad 0]
\end{aligned} \tag{3.8}$$

Em que Δt é o passo de discretização do tempo, θ é o ângulo que, de maneira genérica, pode representar *roll* χ , *pitch* η ou *yaw* α , calculado a partir dos sinais dos acelerômetros conforme (3.1) e (3.2) e $\dot{\theta}$, a partir dos sinais dos giroscópios. A matriz de observação H considera que a variável observada seja somente o ângulo proveniente dos acelerômetros.

A matriz de covariância Q_k foi adotada como:

$$Q_k = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}} \end{bmatrix}, \tag{3.9}$$

Em que $Q_\theta > 0$ e $Q_{\dot{\theta}} > 0$ são supostos conhecidos.

O Filtro de Kalman tem duas fases: a primeira, de predição, e a segunda, de atualização.

Assim, inicialmente o FK efetua uma predição do estado $\hat{x}_{k|k-1}$ com base na estimativa do estado anterior $\hat{x}_{k-1|k-1}$ e no sinal enviado pelos giroscópios, $\dot{\theta}_k$, conforme equação (3.10).

$$\hat{x}_{k|k-1} = F \hat{x}_{k-1|k-1} + B \dot{\theta}_k \tag{3.10}$$

Em seguida, é realizada a predição da matriz de covariância $P_{k|k-1}$ do erro com base na estimativa anterior $P_{k-1|k-1}$:

$$P_{k|k-1} = F P_{k-1|k-1} F^T + Q_k \tag{3.11}$$

A atualização tem início calculando primeiramente o que se denomina *inovação*, que é a diferença entre a medida atual e a obtida com base no modelo de observação e na predição do estado (3.12). Posteriormente calcula-se a *inovação* para a covariância conforme equação (3.13):

$$\tilde{y}_k = z_k - H \hat{x}_{k|k-1} \tag{3.12}$$

$$S_k = H P_{k|k-1} H^T + R_k \tag{3.13}$$

O cálculo da matriz de ganho do FK, $K_{2 \times 1}$, é realizado conforme a equação (3.14).

$$K_k = P_{k|k-1}H^T + S_k^{-1} \quad (3.14)$$

Por fim, são calculadas as atualizações das estimativas do estado e da covariância do erro na estimativa do estado:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \quad (3.15)$$

$$P_{k|k} = (I - K_k H)P_{k|k-1} \quad (3.16)$$

Conforme descrito em [18] e [20], foram realizadas adaptações das equações do FK para a linguagem C do microcontrolador baseado nas aproximações descritas. Após alguns testes para melhor desempenho do FK, foram adotados os valores $Q_\theta=0.001$; $Q_{\dot{\theta}}=0.003$ e $R_k=0.03$ [18]. Para maiores detalhes, veja o Apêndice D.

3.2.3. Sistema de Controle

Para controle dos ângulos da plataforma foi implementado o algoritmo de um controlador PID no microcontrolador (veja detalhes no apêndice B).

Após a leitura e o processamento dos dados fornecidos pelo sensor MPU-9250, os ângulos de orientação da plataforma pitch (η), roll (χ) e yaw (α) filtrados são subtraídos dos ângulos definidos como *setpoint*, iguais a zero, gerando os sinais de erro, $e(t)$, aplicados aos controladores. A estrutura adotada para o PID foi a forma paralela, isto é,

$$C(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d de(t)/dt \quad (3.17)$$

Com base no descrito em [14] e [15], para a implementação do controlador PID digital foram feitas aproximações dos termos integral e derivativo por meio de

$$\int_0^t e(\tau) d\tau \sim T_s \sum_{i=1}^n e(iT_s) \quad (3.18)$$

$$de(t)/dt \sim [e(t) - e(t - T_s)]/T_s, \quad (3.19)$$

Em que T_s é o período de discretização e $t = nT_s$.

Substituindo (3.18) e (3.19) em (3.17), temos

$$C(n) = K_p e(nT_s) + K_i T_s \sum_{i=1}^n e(iT_s) + K_d [e(nT_s) - e((n-1)T_s)]/T_s \quad (3.20)$$

O controlador fornece três sinais de controle, um para cada ângulo de inclinação da plataforma. Esses sinais são utilizados como parâmetros para gerar as saídas PWM do PIC18F4550, as quais são enviadas aos *drivers* dos motores. O sentido de rotação dos motores é definido pelo sinal algébrico do erro: quando positivo, gira em um sentido e, quando negativo, gira em sentido contrário.

Para os *drivers* dos motores são utilizados dois circuitos integrados L293D [12]. Estes possuem 4 pontes H, possibilitando a ligação de dois motores em cada componente que podem ser acionados em ambos os sentidos de rotação. Os circuitos L293D recebem os sinais dos sentidos de rotação dos motores e os sinais PWM gerados pelo microcontrolador, os quais são mostrados nas figuras 15a e 15b, respectivamente para a plataforma estabilizada e para a plataforma inclinada. Esses circuitos também acionam os motores com as polaridades corretas para fornecerem às plataformas os torques e velocidades necessários ao controle do sistema.

Figura 15a: PWM para controle dos motores

Figura 15b: PWM para controle dos motores

A sintonia do controlador, conforme o procedimento descrito em [13], apresenta alguma semelhança com o 2º método de Ziegler-Nichols. Assim, os termos integral e derivativo foram inicialmente adotados como nulos. Em seguida, foi atribuído o valor unitário ao ganho proporcional a fim de verificar a resposta do sistema para os ângulos de pitch (η), roll (χ). Os resultados obtidos estão mostrados na figura 16.

Figura 16: Resposta do sistema, Pitch: $K_p=1$, $K_i=0$, $K_d=0$; Roll: $K_p=1$, $K_i=0$, $K_d=0$

Sendo a resposta do sistema insatisfatória com ganho proporcional unitário, este foi aumentado até que a plataforma começasse a oscilar levemente, mas com o cuidado de preservar a estabilidade. Com isso, foram obtidos para o ângulo de pitch (η), $K_p=10.3$ e, para o ângulo de roll (χ), $K_p=3.3$. A figura 17 mostra a resposta oscilando em torno do set point nulo.

Figura 17: Resposta do sistema, Pitch: $K_p=10.3$, $K_i=0$, $K_d=0$; Roll: $K_p=3.3$, $K_i=0$, $K_d=0$

Os sinais de erro são mostrados na figura 18 para os ângulos de pitch e de roll.

Figura 18: Sinais de erro – Ganho proporcional

Definido o ganho proporcional, o ganho do termo integral foi aumentado de forma a reduzir o erro estacionário, porém com a ocorrência de picos na saída, conforme mostra a figura 19. Para o ângulo de pitch (η) temos os seguintes parâmetros $K_p=10.3$, $K_i=0.55$, $K_d=0$ e para roll (χ) $K_p=3.3$, $K_i=0.3$, $K_d=0$. Os sinais de erro não foram reduzidos, como mostrado na figura 20, provocando picos de sinal da saída.

Figura 19: Resposta do sistema, Pitch: $K_p=10.3$, $K_i=0.55$, $K_d=0$; Roll: $K_p=3.3$, $K_i=0.3$, $K_d=0$

Figura 20: Sinais de erro – Ganho Proporcional e Integral

Para diminuir os valores de pico, o termo derivativo foi gradativamente aumentado até que se obtivesse uma resposta satisfatória (figura 21).

Para o controlador do ângulo de pitch (η) resultaram os seguintes parâmetros: $K_p=10.3$, $K_i=0.55$, $K_d=10$; para o controlador de roll (χ): $K_p=3.3$, $K_i=0.3$, $K_d=5$.

Figura 21: Resposta do sistema, Pitch: $K_p=10.3$, $K_i=0.55$, $K_d=10$; Roll: $K_p=3.3$, $K_i=0.3$, $K_d=5$

Na figura 22 podemos perceber a diminuição dos sinais de erro após a estabilização da plataforma.

Figura 22: Sinais de erro – Ganho PID

3.3. Programa de aquisição de dados

Este programa, desenvolvido em Matlab através da ferramenta *guide*, é responsável por fornecer em tempo real todos os dados da plataforma, ângulos, acelerações, velocidades angulares, temperatura. Seu código fonte encontra-se no apêndice A.

O programa tem por funções apenas o monitoramento e a gravação dos dados, sem nenhuma interferência no sistema de controle da plataforma, que é totalmente autônomo.

O programa de aquisição de dados recebe os dados da IMU através de comunicação por meio de interface USB. Não há nenhuma manipulação dos dados, sendo eles apenas separados e atribuídos às suas variáveis internas para serem mostrados na tela.

A interface com o usuário é bastante simples, possuindo dois botões de controle, a saber, Start e Stop, para leitura, plotagem e gravação dos dados em tempo real; ela tem ainda um botão denominado Plot, para leitura e plotagem posterior dos dados gravados durante o teste. Os menus para escolha dos dados a serem visualizados permitem a possibilidade de exibir até oito sinais simultaneamente, conforme mostra a figura 23.

Figura 23: Tela do programa de aquisição de dados

3.4. Materiais e equipamentos utilizados

3.4.1. Sistema Mecânico

Foram utilizados os seguintes materiais para a montagem mecânica do protótipo, conforme descrito na seção 3.1:

- Base circular de alumínio, com raio de 28 cm.
- Quadro externo de alumínio em formato de U, com distância de 17,5 cm.
- Quadro intermediário metálico em formato circular com raio de 15,8 cm.
- Eixo do quadro interno, constituído por um tubo de metal de comprimento 13,7 cm.
- Mesa circular interna em alumínio com raio de 12,9cm.
- Contrapeso circular com raio de 3,9 cm e diâmetro de 1,5 cm.
- Rolamento central da mesa.

Todos os materiais utilizados na montagem mecânica foram peças de reaproveitamento.

3.4.2. Sistema Eletrônico

Para a montagem do sistema eletrônico de aquisição de dados e controle foram utilizados os seguintes componentes e materiais:

- Placa de fenolite com dimensões de 10x15cm.
- Sensor MPU9250.
- Microcontrolador PIC18F4550.

- Ponte H L293.
- Cristal oscilador de 20 MHz.
- Regulador de tensão 7805.
- Regulador de tensão LPZ 2950 3.3V.
- Motor Modelo B070(12V) com redutor para 69 rpm (sem carga) – 63 rpm (máxima carga) - torque de 4 kgf.cm, com as seguintes dimensões: H= 67mm e L= 23mm.
- Resistores e capacitores.

3.4.3. Equipamentos Utilizados

Foram utilizados os seguintes equipamentos e ferramentas para o desenvolvimento do projeto:

- Placa de aquisição de dados – Desenvolvida para a IMU, Sistema de aquisição de dados, Sistema de Controle e Interface USB para envio dos dados ao programa de aquisição de dados, conforme anexo 1.
- Ferramentas manuais e elétricas.
- Fontes de alimentação de 12V e de 5V.
- Gravador de PIC MultiprogC.
- Notebook.
- Osciloscópio.
- Software Mplab para programação dos microcontroladores PIC.
- Software Matlab para desenvolvimento do programa de aquisição de dados.
- Software Eagle para elaboração do esquema elétrico.

4. RESULTADOS E DISCUSSÃO

4.1. Parâmetros do controlador

Por meio da sintonia do controlador pelo método descrito na seção 3.2.3, foram obtidos os valores para os ganhos proporcional, integral e derivativo para o controle dos ângulos de pitch (η) e roll (χ), respectivamente:

$$C_{pitch}(t) = 10.3e(t) + 0.55 \int_0^t e(\tau) d\tau + 10de(t)/dt \quad (4.1)$$

$$C_{roll}(t) = 3.3e(t) + 0.3 \int_0^t e(\tau) d\tau + 5de(t)/dt \quad (4.2)$$

Os valores dos tempos de integração e derivativo (forma ISA) obtidos para os ângulos de pitch (η) e de roll (χ) foram:

$$T_{i\ pitch} = \frac{K_{p\ pitch}}{K_{i\ pitch}} = \frac{10.3}{0.55} = 18.73\ ms \quad (4.3)$$

$$T_{d\ pitch} = \frac{K_{d\ pitch}}{K_{p\ pitch}} = \frac{10}{10.3} = 0.97\ ms \quad (4.4)$$

$$T_{i\ roll} = \frac{K_{p\ roll}}{K_{i\ roll}} = \frac{3.3}{0.3} = 11\ ms \quad (4.5)$$

$$T_{d\ roll} = \frac{K_{d\ roll}}{K_{p\ roll}} = \frac{5}{3.3} = 1.51\ ms \quad (4.6)$$

Para o ângulo de yaw (α) não foi possível a determinação dos parâmetros do controlador devido às interferências eletromagnéticas que afetaram diretamente a leitura dos campos magnéticos durante o funcionamento da plataforma. Com os motores desligados, a leitura do yaw (α) é estável devido ao procedimento para eliminar os campos magnéticos da plataforma, conforme descrito no Apêndice B. No entanto, ao energizar os motores de pitch (η) e roll (χ), ele se torna instável devido ao campo eletromagnético gerado pelos motores, conforme mostra a figura 24.

Figura 24: Interferência eletromagnética no ângulo yaw (α)

4.2. Análise e discussão dos resultados

Os resultados experimentais estão divididos em duas partes: a primeira parte se refere aos testes realizados com a plataforma em regime estacionário para avaliar sua estabilidade ao longo do tempo podendo ou não sofrer perturbações externas; a segunda parte foi realizada com a plataforma em regime de navegação para avaliar a resposta do sistema às possíveis condições encontradas durante testes com a plataforma embarcada.

4.2.1. Plataforma em regime estacionário

Os resultados da plataforma em regime estacionário mostram sua estabilidade por meio dos ângulos de pitch (η) e roll (χ) em testes de longa duração conforme ilustrado na figura 25. São apresentados os gráficos com os dados brutos e filtrados.

Figura 25: Ângulos de pitch (η) e roll (χ) em regime estacionário

A eficiência do Filtro de Kalman é mostrada na figura 26, na qual é determinado a amplitude do ruído. Para os ângulos brutos, a amplitude dos ruídos são de 0.04° e 0.05° respectivamente para os ângulos de pitch (η) e roll (χ), a amplitude dos ruídos nos ângulos filtrados são de 0.01° para pitch (η) e roll (χ).

Figura 26: Ângulos de pitch (η) e roll (χ) determinação do ruído bruto e filtrado

As amplitudes dos ruídos e das derivas nos sinais dos giroscópios durante o teste são mostradas na figura 27. Observa-se uma deriva de 0.05°/s em duas horas de navegação estacionária. Isso indica que a deriva é de 90°/h com a amplitude máxima do ruído de 0.2.

Figura 27: Deriva e nível de ruído dos giroscópios

Os valores dos sinais dos acelerômetros são mostrados na figura 28. Pode-se perceber que os valores variam sempre dentro dos mesmos limites, indicando a ocorrência do regime estacionário.

Figura 28: Sinais dos acelerômetros

4.2.2. Plataforma em regime de navegação

Para a obtenção dos resultados em regime de navegação foram produzidas inclinações circulares com o objetivo de reproduzir as condições extremas a que o sistema estaria exposto em uma navegação veicular ou aérea.

Inicialmente a plataforma foi inclinada em $+52^\circ$ no ângulo de pitch (η) para iniciar o ciclo de um giro de 360° em seu próprio eixo, de forma que o sistema de controle atue também no eixo do ângulo de roll (χ), como mostrado na figura 29.

Figura 29: Inclinação teste de navegação

Os ângulos de inclinação da base da plataforma (definidos como *pitch base* e *roll base*) são fornecidos pela IMU instalada na base da plataforma e mostrados no 2° e 4° gráficos da figura 30.

Os ângulos de pitch (η) e roll (χ) da plataforma – brutos e filtrados – são mostrados no 1° e 3° gráficos da figura 30 e a variável de controle, na figura 31. Pode-se notar a ação de controle durante a navegação, com pequeno sobressinal no início de cada movimento imposto de inclinação. Isso é decorrência do fato de os parâmetros do controlador PID terem sido ajustados para uma resposta relativamente lenta de forma a evitar oscilações.

Figura 30: Resposta do 1° teste de navegação

Figura 31: Variável de controle – 1º teste de navegação

O primeiro ciclo do teste foi realizado com ângulos de aproximadamente $+52^\circ$ e -42° no eixo de pitch (η) com período de 35s. As inclinações no eixo de roll (χ) foram de -49° e $+51^\circ$ com um período de aproximadamente 30s, conforme mostra a figura 32.

Figura 32: Resposta do 1º teste de navegação – 1º ciclo

Após o término do teste, aguardou-se a estabilização da plataforma e iniciou-se o segundo ciclo com ângulos de aproximadamente $+56^\circ$ e -34° no eixo de pitch (η) com período de 28s; as inclinações no eixo de roll (χ) foram de $+51^\circ$ e -28° com período de 34s, conforme mostrado na figura 33.

Figura 33: Resposta do 1º teste de navegação – 2º ciclo

O segundo teste de navegação foi realizado para verificar a resposta da plataforma às inclinações menores, assim como o comportamento da variável de controle nesse modo de navegação. Primeiramente foi considerado um período maior e, posteriormente, um menor. Veja as figuras 34 e 35.

Figura 34: Resposta do 2º teste de navegação

Figura 35: Variável de controle - 2º teste de navegação

O primeiro ciclo do teste foi realizado com ângulos de aproximadamente -33° e $+30^\circ$ no eixo de pitch (η) com período de 52s. As inclinações no eixo de roll (χ) foram de $+42^\circ$ e -36° com um período de aproximadamente 50s, conforme ilustrado pela figura 36.

Figura 36: Resposta do 2º teste de navegação – 1º ciclo

Após o término do primeiro ciclo do teste, iniciou-se o segundo ciclo com ângulos de aproximadamente -38° e $+32^\circ$ no eixo de pitch (η) e período de aproximadamente 25s; as inclinações no eixo de roll (χ) foram de $+41^\circ$ e -28° com período de aproximadamente 15s. Veja os resultados na figura 37.

Figura 37: Resposta do 2º teste de navegação – 2º ciclo

5. CONCLUSÕES

O propósito deste trabalho foi a integração de um sistema mecânico de eixos, quadros e bases com um sistema eletrônico e computacional de controle e aquisição de dados para o desenvolvimento de uma plataforma inercial com três graus de liberdade.

Para transmissão dos sinais do sensor inercial para o sistema de controle e aquisição de dados, inicialmente foi testado um sistema de radiofrequência para evitar o uso de *slip rings* ou a passagem de fios pelo eixo de rotação da plataforma. A transmissão sem fio não apresentou resultados satisfatórios devido aos altos níveis de ruído presentes durante o funcionamento da plataforma e à baixa velocidade na transmissão dos dados.

O prosseguimento do trabalho se deu com a utilização de dois pares de fios trançados para a ligação entre o sensor inercial e o sistema eletrônico. Esses fios foram passados pelos eixos dos quadros e do eixo de rotação de maneira a não prejudicar nem o movimento da plataforma e nem os fios de transmissão dos sinais.

O sistema de aquisição de dados, tanto no que se refere ao software como ao hardware, realizou perfeitamente as funções de monitorar, gravar e plotar os sinais adquiridos durante o teste, sem provocar nenhuma interferência no sistema de controle.

Conforme mostrado na seção 4.1, não foi possível a implementação do controle para o ângulo de yaw (α) devido às interferências eletromagnéticas surgidas durante o funcionamento da plataforma, as quais prejudicam a medida dos campos magnéticos fornecidas pelos magnetômetros da MPU-9250. Em razão dessas interferências, a plataforma opera de fato com apenas dois graus de liberdade.

Os testes mostrados no capítulo anterior foram satisfatórios tanto para a plataforma funcionando em regime estacionário quanto em regime de navegação. Em regime estacionário a plataforma se manteve estável durante um teste de aproximadamente 120 minutos de navegação com diferença entre os ângulos iniciais e finais em torno de 0.003° nos valores filtrados e deriva nos sinais dos giroscópios de $90^\circ/\text{h}$.

Com a implementação do Filtro de Kalman foi possível diminuir consideravelmente – em torno de 20 vezes – a amplitude dos ruídos e derivas provenientes dos acelerômetros e giroscópios, possibilitando uma melhor precisão e uma menor sensibilidade nos ângulos da plataforma.

O controlador PID digital implementado atuou satisfatoriamente para manter o posicionamento da plataforma, conforme teste de navegação apresentado na seção 4.2.2. Com a precisão obtida em ambos os testes, pode-se concluir que o sistema possui condições de ser utilizado tanto para a estabilização de sensores de precisão estacionários ou embarcados como para a estabilização de câmeras de vídeo ou fotográficas.

Para o prosseguimento do desenvolvimento da plataforma com três graus de liberdade poderão ser realizados os seguintes itens:

- Estudo e implementação de um sistema de transmissão de sinais sem fio para comunicação entre o MEMS e o sistema eletrônico,
- Integração de um sistema de GPS para determinação da orientação e localização da plataforma, possibilitando o cálculo e controle do ângulo de yaw (α),
- Realização de testes com a plataforma embarcada e
- Instalação de sensores gravimétricos para avaliar seu uso em pesquisas geofísicas.

REFERENCIAS BIBLIOGRÁFICAS

- [1] Jekeli, C., Inertial navigation systems with geodetic applications. Walter de Gruyter, New York 2001.
- [2] Wrigley, W., Hollister, W.M. and Denhard, W.G., Gyroscopic Theory, Design and Instrumentation. The M.I.T. Press, Cambridge, Massachusetts, 1969.
- [3] Titterton, D.H., Weston, J.L., Strapdown inertial navigation technology. Institution of Electrical Engineers, 2nd edition, 2004.
- [4] Kenneth, R. Britting, Inertial navigation systems analysis. John Wiley & Sons Canada, Limited, The University of Michigan, 1971.
- [5] Bredenkamp, A.F.L., Development and Control a 3-axis Stabilised Platform. Department of Electrical and Electronic Engineering, University of Stellenbosch – South Africa, 2007.
- [6] Analog Devices, AN-1057, Using an Accelerometer for Inclination Sensing, 2010.
- [7] Analog Devices, ADXL335, Datasheet, 2010.
- [8] InvenSense, MPU-9250 Product Specification Revision 1.0, 2014.
- [9] Microchip, PIC18F2455/2550/4555/4550, Datasheet, 2009.
- [10] Freescale Semiconductor, AN-3157, Measuring Tilt With Low-g Accelerometers, Rev 0, 05/2005.
- [10] Freescale Semiconductor, AN-3462, Tilt Sensing Using a Tree-Axis Accelerometer, Rev 6, 03/2013.
- [11] InvenSense, MPU-9250 Register Map and Descriptions Revision 1.4 , 2013.
- [12] Texas Instruments, L293X Quadruple Half-H Drivers, 2016.
- [13] National Instruments, PID Theory Explained, Mar 29, 2011, disponível em: <http://www.ni.com/white-paper/3782/en/> . Ultimo acesso em Dezembro de 2015.
- [14] Microchip, AN-937, Implementing a PID Controller Using a PIC18 MCU, 2004.
- [15] Microchip, AN-964, Software PID Control of an Inverted Pendulum Using the PIC16F684, 2004.
- [16] Welch, G., Bishop G., An Introduction to the Kalman Filter, Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175, July 24, 2006
- [17] Oliveira, J.B., Cruz, J.J, Ferreira. Ferreira, Desenvolvimento de uma Unidade de Medida Inercial com Sistema Digital de Aquisição de Dados em Tempo Real, Laboratório de Automação e Controle, Departamento de Engenharia de Telecomunicações e Controle, Universidade de São Paulo, VIII Simpósio Brasileiro de Engenharia Inercial (VIII SBEIN), Junho, 2015.

- [18] Lauszus K., A practical approach to Kalman filter and how to implement it. Guides, TKJ Electronics, September, 2012, Disponível em: <http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/> . Ultimo acesso em dezembro de 2015.
- [19] Caruso M.J., G., Applications of Magnetoresistive Sensors in Navigation Systemsr, Honeywell Inc.
- [20] Miranda, J.L.C, Application of Kalman Filtering and PID Control for Direct Inverted Pendulum Control. Department of Electrical and Computer Engineering, California State University – USA, 2009.
- [21] Basic Principles of Inertia Navigation, Seminar on Inertial Navigation System, Tampere University of Technology. Disponível em: <http://aerostudents.com/files/avionics/InertialNavigationSystems.pdf>. Ultimo acesso em dezembro, 2015.

APENDICE A – PROGRAMA DE AQUISIÇÃO DE DADOS EM TEMPO REAL

Funções de inicialização

```
function varargout = Teste_V2(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',      mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @Teste_V2_OpeningFcn, ...
                  'gui_OutputFcn',  @Teste_V2_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Teste_V2_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);

function varargout = Teste_V2_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
```

Função para leitura dos dados em tempo real

```
function pushbutton1_Callback(hObject, eventdata, handles)
```

Configurações para comunicação via interface USB

```
s = serial('COM2');
s1 = serial('COM4');
set(s, 'BaudRate', 19200);
set(s1, 'BaudRate', 19200);
disp('Running...');
```

Determinação das variáveis

```
id=(0);
outx = ones(0);
outy = ones(0);
outz = ones(0);
outAx = ones(0);
outAy = ones(0);
outAz = ones(0);
```

```

outGx = ones(0);
outGy = ones(0);
outGz = ones(0);
outKx = ones(0);
outKy = ones(0);
outT = ones(0);
outmx = ones(0);
outmy = ones(0);
data = ones(0);
time = ones(0);
outYaw = ones(0);
outKxb = ones(0);
outKyb = ones(0);
quitit = 0;
t=1;

```

Inicialização da comunicação via USB

```

fopen(s);
fopen(s1);
clc
disp('Running...');

```

Início da aquisição de dados

```

while( quitit == 0)
    s.ReadAsyncMode = 'manual';
    s1.ReadAsyncMode = 'manual';
    f=str2num(fscanf(s));
    f1=str2num(fscanf(s1));
    sz=size(f);
    sz1=size(f1);
    if ((sz(:,2)==13) && (sz1(:,2)==2))
        id=(f(:,1));
        if (id==113)
            outAx(t,:)=(f(:,2));
            outAy(t,:)=(f(:,3));
            outAz(t,:)=(f(:,4));
            outGx(t,:)=(f(:,5));
            outGy(t,:)=(f(:,6));
            outGz(t,:)=(f(:,7));
            outT(t,:) = (f(:,8));
            outx(t,:)=(f(:,9));
            outKx(t,:)=(f(:,10));
            outy(t,:)=(f(:,11));
            outKy(t,:)=(f(:,12));
            outYaw(t,:)=f(:,13);
            outKxb(t,:)=f1(:,1);
            outKyb(t,:)=f1(:,2);
            data(t,:)=[f(:,2) f(:,3) f(:,4) f(:,5) f(:,6) f(:,7) f(:,8) f(:,9) f(:,10) f(:,11)
                f(:,12) f(:,13) f1(:,1) f1(:,2)];

```

```
save('TESTE.mat','data')
```

```
label = {'Plot'};
```

Rotina para plotagem dos sinais

```
popup_sel_index = get(handles.popupmenu1, 'Value');  
axes(handles.axes1)
```

```
switch popup_sel_index
```

```
case 1
```

```
plot(outAx);
```

```
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
```

```
ylabel('mg');
```

```
grid on;
```

```
title('Acelerometro X');
```

```
h = zoom;
```

```
set(h,'ActionPostCallback',@mypostcallback);
```

```
set(h,'Motion','both','Enable','off');
```

```
hp = pan;
```

```
set(hp,'ActionPostCallback',@mypostcallback);
```

```
set(hp,'Enable','off');
```

```
case 2
```

```
plot(outAy);
```

```
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
```

```
ylabel('mg');
```

```
grid on;
```

```
title('Acelerometro Y');
```

```
h = zoom;
```

```
set(h,'ActionPostCallback',@mypostcallback);
```

```
set(h,'Motion','both','Enable','off');
```

```
hp = pan;
```

```
set(hp,'ActionPostCallback',@mypostcallback);
```

```
set(hp,'Enable','off');
```

```
case 3
```

```
plot(outAz);
```

```
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
```

```
ylabel('mg');
```

```
grid on;
```

```
title('Acelerometro Z');
```

```
h = zoom;
```

```
set(h,'ActionPostCallback',@mypostcallback);
```

```
set(h,'Motion','both','Enable','off');
```

```
hp = pan;
```

```
set(hp,'ActionPostCallback',@mypostcallback);
```

```
set(hp,'Enable','off');
```



```

case 4
plot(outGx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro X');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 5
plot(outGy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Y');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 6
plot(outGz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Z');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 7
plot(outT);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('°C');
grid on;
title('Temperatura');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');

```

```

hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 8
plot(outx,'b');;hold on; plot(outKx,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 9
plot(outKx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
%xlabel('Tempo');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 10

plot(outy,'b');;hold on; plot(outKy,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 11
plot(outKy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))

```

```

ylabel('Graus');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 12
plot(outYaw);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Yaw');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 13

plot(outKxb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 14
plot(outKyb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);

```

```

        set(hp,'Enable','off');
    end

    popup_sel_index = get(handles.popupmenu2, 'Value');
    axes(handles.axes2)
    switch popup_sel_index

        case 1
            plot(outAx);
            set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
            ylabel('mg');
            grid on;
            title('Acelerometro X');
            h = zoom;
            set(h,'ActionPostCallback', @mypostcallback);
            set(h,'Motion','both','Enable','off');
            hp = pan;
            set(hp,'ActionPostCallback', @mypostcallback);
            set(hp,'Enable','off');

        case 2
            plot(outAy);
            set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
            ylabel('mg');
            grid on;
            title('Acelerometro Y');
            h = zoom;
            set(h,'ActionPostCallback', @mypostcallback);
            set(h,'Motion','both','Enable','off');
            hp = pan;
            set(hp,'ActionPostCallback', @mypostcallback);
            set(hp,'Enable','off');

        case 3
            plot(outAz);
            set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
            ylabel('mg');
            grid on;
            title('Acelerometro Z');
            h = zoom;
            set(h,'ActionPostCallback', @mypostcallback);
            set(h,'Motion','both','Enable','off');
            hp = pan;
            set(hp,'ActionPostCallback', @mypostcallback);
            set(hp,'Enable','off');

        case 4
            plot(outGx);

```

```

set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro X');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 5
plot(outGy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Y');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 6
plot(outGz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Z');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 7
plot(outT);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('°C');
grid on;
title('Temperatura');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);

```

```

set(hp,'Enable','off');

case 8
plot(outx,'b');;hold on; plot(outKx,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 9
plot(outKx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
%xlabel('Tempo');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 10

plot(outy,'b');;hold on; plot(outKy,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 11
plot(outKy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;

```

```

title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 12
plot(outYaw);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Yaw');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 13

plot(outKxb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 14
plot(outKyb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');
end

```

```

popup_sel_index = get(handles.popupmenu3, 'Value');
axes(handles.axes3)
switch popup_sel_index

    case 1
        plot(outAx);
        set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
        ylabel('mg');
        grid on;
        title('Acelerometro X');
        h = zoom;
        set(h,'ActionPostCallback',@mypostcallback);
        set(h,'Motion','both','Enable','off');
        hp = pan;
        set(hp,'ActionPostCallback',@mypostcallback);
        set(hp,'Enable','off');

    case 2
        plot(outAy);
        set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
        ylabel('mg');
        grid on;
        title('Acelerometro Y');
        h = zoom;
        set(h,'ActionPostCallback',@mypostcallback);
        set(h,'Motion','both','Enable','off');
        hp = pan;
        set(hp,'ActionPostCallback',@mypostcallback);
        set(hp,'Enable','off');

    case 3
        plot(outAz);
        set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
        ylabel('mg');
        grid on;
        title('Acelerometro Z');
        h = zoom;
        set(h,'ActionPostCallback',@mypostcallback);
        set(h,'Motion','both','Enable','off');
        hp = pan;
        set(hp,'ActionPostCallback',@mypostcallback);
        set(hp,'Enable','off');

    case 4
        plot(outGx);
        set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
        ylabel('graus/s');
        grid on;

```



```

title('Giro X');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 5
plot(outGy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Y');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 6
plot(outGz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Z');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 7
plot(outT);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('°C');
grid on;
title('Temperatura');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 8

```

```

plot(outx,'b');;hold on; plot(outKx,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 9
plot(outKx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
%xlabel('Tempo');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

case 10

```

plot(outy,'b');;hold on; plot(outKy,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 11
plot(outKy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);

```

```

set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 12
plot(outYaw);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Yaw');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 13
plot(outKxb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 14
plot(outKyb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');
end

```

```

popup_sel_index = get(handles.popupmenu4, 'Value');
axes(handles.axes4)
switch popup_sel_index

```

```

case 1
plot(outAx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro X');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

```

case 2
plot(outAy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro Y');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

```

case 3
plot(outAz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro Z');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

```

case 4
plot(outGx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro X');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);

```

```

set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 5
plot(outGy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Y');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 6
plot(outGz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Z');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 7
plot(outT);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('°C');
grid on;
title('Temperatura');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 8
plot(outx,'b');;hold on; plot(outKx,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');

```

```

grid on;
title('Pitch');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

```

case 9
plot(outKx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
%xlabel('Tempo');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

case 10

```

plot(outy,'b');;hold on; plot(outKy,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

```

case 11
plot(outKy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);

```

```

set(hp,'Enable','off');

case 12
plot(outYaw);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Yaw');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

case 13
plot(outKxb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Base');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

case 14
plot(outKyb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll Base');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');
end

popup_sel_index = get(handles.popupmenu5, 'Value');
axes(handles.axes5)
switch popup_sel_index

case 1
plot(outAx);

```

```

set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro X');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 2
plot(outAy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro Y');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 3
plot(outAz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro Z');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 4
plot(outGx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro X');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);

```



```

set(hp,'Enable','off');

case 5
plot(outGy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Y');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

case 6
plot(outGz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Z');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

case 7
plot(outT);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('°C');
grid on;
title('Temperatura');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

case 8
plot(outx,'b');;hold on; plot(outKx,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch');
h = zoom;

```

```

set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 9
plot(outKx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
%xlabel('Tempo');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 10
plot(outy,'b');;hold on; plot(outKy,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 11
plot(outKy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 12

```

```

plot(outYaw);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Yaw');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

case 13

```

plot(outKxb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

case 14

```

plot(outKyb);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll Base');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');
end

```

```

popup_sel_index = get(handles.popupmenu6, 'Value');
axes(handles.axes6)
switch popup_sel_index

```

case 1

```

plot(outAx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');

```

```

grid on;
title('Acelerometro X');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 2
plot(outAy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro Y');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 3
plot(outAz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('mg');
grid on;
title('Acelerometro Z');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 4
plot(outGx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro X');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

```

```

case 5
plot(outGy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Y');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 6
plot(outGz);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('graus/s');
grid on;
title('Giro Z');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 7
plot(outT);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('°C');
grid on;
title('Temperatura');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback', @mypostcallback);
set(hp,'Enable','off');

```

```

case 8
plot(outx,'b');;hold on; plot(outKx,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch');
h = zoom;
set(h,'ActionPostCallback', @mypostcallback);
set(h,'Motion','both','Enable','off');

```

```

hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 9
plot(outKx);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
%xlabel('Tempo');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 10

plot(outy,'b');;hold on; plot(outKy,'r');hold off;
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Roll');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 11
plot(outKy);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
ylabel('Graus');
grid on;
title('Pitch Filtrado');
h = zoom;
set(h,'ActionPostCallback',@mypostcallback);
set(h,'Motion','both','Enable','off');
hp = pan;
set(hp,'ActionPostCallback',@mypostcallback);
set(hp,'Enable','off');

case 12
plot(outYaw);
set(gca, 'YTickLabel', num2str(get(gca,'YTick')))

```

```

        ylabel('Graus');
        grid on;
        title('Yaw');
        h = zoom;
        set(h,'ActionPostCallback',@mypostcallback);
        set(h,'Motion','both','Enable','off');
        hp = pan;
        set(hp,'ActionPostCallback',@mypostcallback);
        set(hp,'Enable','off');

        case 13

            plot(outKxb);
            set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
            ylabel('Graus');
            grid on;
            title('Pitch Base');
            h = zoom;
            set(h,'ActionPostCallback',@mypostcallback);
            set(h,'Motion','both','Enable','off');
            hp = pan;
            set(hp,'ActionPostCallback',@mypostcallback);
            set(hp,'Enable','off');

            case 14
                plot(outKyb);
                set(gca, 'YTickLabel', num2str(get(gca,'YTick')))
                ylabel('Graus');
                grid on;
                title('Roll Base');
                h = zoom;
                set(h,'ActionPostCallback',@mypostcallback);
                set(h,'Motion','both','Enable','off');
                hp = pan;
                set(hp,'ActionPostCallback',@mypostcallback);
                set(hp,'Enable','off');
            end
        end
        t = t + 1;
    end
end

```

Função para encerrar a aquisição de dados

```

function pushbutton2_Callback(hObject, eventdata, handles)
    quitit = 1;
    fclose (s);
    fclose (s1);
    delete (s);

```

```
delete (s1);
fclose(instrfind);
```

Funções de configuração dos menus para escolha dos sinais

```
function popupmenu1_Callback(hObject, eventdata, handles)
```

```
function popupmenu1_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
set(hObject, 'String', {'aX','aY','aZ','gX','gY','gZ','temp','Pitch','Pitch Filt','Roll','Roll
Filt','Yaw','Pitch Base','Roll Base'});
```

```
function popupmenu2_Callback(hObject, eventdata, handles)
```

```
function popupmenu2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
set(hObject, 'String', {'aX','aY','aZ','gX','gY','gZ','temp','Pitch','Pitch Filt','Roll','Roll
Filt','Yaw','Pitch Base','Roll Base'});
```

```
function popupmenu3_Callback(hObject, eventdata, handles)
```

```
function popupmenu3_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
set(hObject, 'String', {'aX','aY','aZ','gX','gY','gZ','temp','Pitch','Pitch Filt','Roll','Roll
Filt','Yaw','Pitch Base','Roll Base'});
```

```
function popupmenu4_Callback(hObject, eventdata, handles)
```

```
function popupmenu4_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
set(hObject, 'String', {'aX','aY','aZ','gX','gY','gZ','temp','Pitch','Pitch Filt','Roll','Roll
Filt','Yaw','Pitch Base','Roll Base'});
```

```
function popupmenu6_Callback(hObject, eventdata, handles)
```



```
function popupmenu6_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    set(hObject, 'String', {'aX','aY','aZ','gX','gY','gZ','temp','Pitch','Pitch Filt','Roll','Roll
    Filt','Yaw','Pitch Base','Roll Base'});
```

```
function popupmenu7_Callback(hObject, eventdata, handles)
```

```
function popupmenu7_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    set(hObject, 'String', {'aX','aY','aZ','gX','gY','gZ','temp','Pitch','Pitch Filt','Roll','Roll
    Filt','Yaw','Pitch Base','Roll Base'});
```

Função para leitura e plotagem dos sinais gravados

```
function pushbutton3_Callback(hObject, eventdata, handles)
    clc
    load TESTE.mat

    goto('Plot')
    return

    set(h,'ActionPostCallback', @mypostcallback);
    set(h,'Motion','both','Enable','off');
    hp = pan;
    set(hp,'ActionPostCallback', @mypostcallback);
    set(hp,'Enable','off');

    end
```

APENDICE B – PROGRAMA DO MICROCONTROLADOR PRINCIPAL PARA AQUISIÇÃO DE DADOS E CONTROLE

Configurações do microcontrolador

```
#include <18F4550.h>
#fuses SPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5,
      CPUDIV1, VREGEN
#use delay(clock=96MHz, USB_FULL)
#include <usb_cdc.h>
#use rs232(uart1, baud=19200)
#define MPU_SDA PIN_B0
#define MPU_SCL PIN_B1
#define LED    PIN_D2
#define M1_D   PIN_A0
#define M1_E   PIN_A1
#define M2_D   PIN_A2
#define M2_E   PIN_A3
#define MPU    PIN_E1
#define M3_D   PIN_D0
#define M3_E   PIN_D1
#define OFF    output_low
#define ON     output_high
#use I2C(master, fast, sda=MPU_SDA, scl=MPU_SCL)
#use TIMER(timer=1, tick=1us, bits=8, NOISR)
```

Inclusão de arquivos auxiliaries

```
#include "MPU9250_V6.c"
#include "math.h"
#include "Kalman_p.h"
#include "Kalman_r.h"
#include "Kalman_y.h"
```

Definição de variáveis

```
#define RAD_TO_DEG 180/PI
#define outmin 0
#define outmax 300
#define sampletime 100
#define fim 500
int x,m,j;
double k_term[3], i_term[3], d_term[3],kp1[3],ki1[3],kd1[3],dt, ax_rate, ay_rate,
      az_rate, gx_rate, gy_rate, gz_rate, t, mx_rate, my_rate, mz_rate, mx_rate2,
      my_rate2, mz_rate2,output[3],setpoint,serro[3],erro[3],lastinput[3],erroY;
      ax_offset, ay_offset, az_offset, gx_offset, gy_offset, gz_offset, mx_offset,
      my_offset, mz_offset,
      pitch,roll,yaw,yawf,yawf2,k_pitch,k_roll,yawf22,k_pitch2,k_roll2, MOS[3],
      mx2,my2,ti_counter, td_counter, i_time, d_time,acc_res, gyro_res, mag_res;
int8 periodo=0, ciclo1;
```

```
unsigned int16 timer;
```

```
void main()  
{
```

Determinação da resolução dos sinais

```
acc_res=2.0/32768.0;  
gyro_res=250.0/32768.0;  
mag_res=10.0*4912.0/32760.0;
```

Inicialização das variáveis

```
yawf22=0.0;  
k_pitch2=0.0;  
k_roll2=0.0;  
setpoint=0.0;  
lastinput[0]=0.0;  
lastinput[1]=0.0;  
lastinput[2]=0.0;  
serro[0]=0.0;  
serro[1]=0.0;  
serro[2]=0.0;  
double offset0=620.0;  
double offset1=625.0;  
i_time=5.0;//Tempo de integração  
d_time=2.0;//Tempo derivativo  
ti_counter=i_time;  
td_counter=d_time;  
OFF(M2_D);  
OFF(M2_E);  
OFF(M1_D);  
OFF(M1_E);  
ON(MPU);
```

Inicialização modulo PWM

```
setup_timer_2(T2_DIV_BY_16,248,1);  
#use pwm(CCP1,TIMER=2,FREQUENCY=2000,DUTY=100)  
#use pwm(CCP2,TIMER=2,FREQUENCY=2000,DUTY=100)  
pwm_off ();
```

Inicialização da MPU-9250

```
AK8963_asa ();  
delay_ms (50) ;  
Mpu9250_Init ();  
delay_ms (50) ;
```

Leitura dos dados de identificação

```
m = AK8963_Read(WHO_AM_I_AK8963);  
x = Mpu9250_Read(WHO_AM_I);
```

```

timer = get_ticks ();
set_ticks(0);
pwm_off ();
OFF(PIN_A5);

```

Início da leitura dos dados de offset da MPU-9250

```

if ((x == 0x71) && (j != 0x18)){
ax_offset=(signed int16)(Mpu9250_GetData (XA_OFFSET_H))/16384.0;
ay_offset=(signed int16)(Mpu9250_GetData (YA_OFFSET_H))/16384.0;
az_offset=(signed int16)(Mpu9250_GetData (ZA_OFFSET_H))/16384.0;
gx_offset=(signed int16)(Mpu9250_GetData (XG_OFFSET_H))/131.0;
gy_offset=(signed int16)(Mpu9250_GetData (YG_OFFSET_H))/131.0;
gz_offset=(signed int16)(Mpu9250_GetData (ZG_OFFSET_H))/131.0;

MOS[0]=(double)((((signed int16)(AK8963_Read (AK8963_ASAX)))-
128.0)/256.0)+1.0;
MOS[1]=(double)((((signed int16)(AK8963_Read (AK8963_ASAY)))-
128.0)/256.0)+1.0;
MOS[2]= (double)((((signed int16)(AK8963_Read (AK8963_ASAZ)))-
128.0)/256.0)+1.0;
}

```

Início do loop principal

```

while(1){
pwm_on ();

```

Leitura em tempo real dos dados da MPU9250

```

j = AK8963_Read(AK8963_ST2)
if ((x == 0x71) && (j != 0x18)){
ax_rate=(((signed int16)(Mpu9250_GetData (ACCEL_XOUT_H)))-
ax_offset)*acc_res;
ay_rate=(((signed int16)(Mpu9250_GetData (ACCEL_YOUT_H)))-
ay_offset)*acc_res;
az_rate=(((signed int16)(Mpu9250_GetData (ACCEL_ZOUT_H)))-
az_offset)*acc_res;
gx_rate=(((signed int16)(Mpu9250_GetData (GYRO_XOUT_H)))-
gx_offset)*gyro_res;
gy_rate=(((signed int16)(Mpu9250_GetData (GYRO_YOUT_H)))-
gy_offset)*gyro_res;
gz_rate=(((signed int16)(Mpu9250_GetData (GYRO_ZOUT_H)))-
gz_offset)*gyro_res;
mx_rate=((signed
int16)(AK8963_GetData(AK8963_XOUT_L))*MOS[0])*0.15*mag_res;
my_rate=((signed
int16)(AK8963_GetData(AK8963_YOUT_L))*MOS[1])*0.15*mag_res;
mz_rate=((signed
int16)(AK8963_GetData(AK8963_ZOUT_L))*MOS[2])*0.15*mag_res;

```

```
t=((signed int16)(Mpu9250_GetData (TEMP_OUT_H))/333.87)+21.0;
```

Aproximação para cálculo da derivada do tempo

```
dt =(double)(get_ticks () - timer)/1333;//68239;  
timer = get_ticks ();  
set_ticks(0);
```

Cálculo dos ângulos brutos e Aplicação do Filtro de Kalman

```
roll =(atan2 (ax_rate,sqrt(ay_rate * ay_rate + az_rate * az_rate))) *  
RAD_TO_DEG;  
k_roll = kalmanCalculate (roll, gy_rate, dt);//Filtro de Kalman  
  
pitch =(atan2 (ay_rate,sqrt(ax_rate * ax_rate + az_rate * az_rate))) *  
RAD_TO_DEG;  
k_pitch = kalmanCalculate( pitch, gx_rate, dt); //Filtro de Kalman  
  
mx_rate2=((mx_rate-32218.0)/(-12.05)*2.0)-1.0;  
my_rate2=((my_rate-(-55804.0))/(45.3550)*2.0)-1.0;  
mz_rate2=((mz_rate-(-4.1145))/(85.4650)*2.0)-1.0;  
  
yaw=atan2(my2,mx2)*RAD_TO_DEG;  
yaw+=(-0.3659)*RAD_TO_DEG;  
yawf =kalmanCalculate (yaw, gz_rate, dt); //Filtro de Kalman
```

Definição dos parâmetros do Controlador PID

```
kp1[0]=10.3;  
ki1[0]=0.55;  
Kd1[0]=10.0;  
kp1[1]=3.3;  
ki1[1]=0.3;  
kd1[1]=5.0;  
kp1[2]=0.0;  
ki1[2]=0.0;  
kd1[2]=0.0;
```

Definição do erro

```
erro[0]=setpoint-k_pitch;  
erro[1]=setpoint-k_roll;  
erro[2]=setpoint-yawf;
```

Somatória do erro ao longo do tempo para o controlador integral

```
serro[0]+=erro[0];
```

```
serro[1]+=erro[1];  
serro[2]+=erro[2];
```

Rotina definir o sentido de rotação dos motores com base no sinal do erro

```
if (erro[0] < 0){  
    kp1[0]*= -1;  
    ki1[0]*= -1;  
    kd1[0]*= -1;  
    OFF(M1_D);  
    ON(M1_E);  
}  
if (erro[0] > 0){  
    ON(M1_D);  
    OFF(M1_E);  
}  
if (erro[1] < 0){  
    kp1[1]*= -1;  
    ki1[1]*= -1;  
    kd1[1]*= -1;  
    OFF(M2_D);  
    ON(M2_E);  
}  
if (erro[1] > 0){  
    ON(M2_D);  
    OFF(M2_E);  
}  
if (erro[2] < 0){  
    kp1[2]*= -1;  
    ki1[2]*= -1;  
    kd1[2]*= -1;  
}  
if (yawf2 < 0){  
    OFF(M3_D);  
    ON(M3_E);  
}  
if (yawf2 > 0){  
    ON(M3_D);  
    OFF(M3_E);  
}
```

Cálculo do controlador Proporcional

```
k_term[0]=kp1[0]*erro[0];  
k_term[1]=kp1[1]*erro[1];  
k_term[2]=kp1[2]*erro[2];
```

Cálculo do controlador Integral

```

if (ti_counter == 0.0){
    serro[0]/=i_time;
    serro[1]/=i_time;
    serro[2]/=i_time;
    i_term[0]=Ki1[0]*serro[0];
    if(i_term[0] > outmax){
        i_term[0]= outmax;
    }
    i_term[1]=Ki1[1]*serro[1];
    if(i_term[1] > outmax){
        i_term[1]= outmax;
    }
    i_term[2]=Ki1[2]*serro[2];
    if(i_term[2] > outmax){
        i_term[2]= outmax;
    }
    ti_counter=i_time;
}
ti_counter-=1.0;

```

Cálculo do controlador Derivativo

```

if (td_counter == 0.0){
    d_term[0]=Kd1[0]*(k_pitch - lastinput[0])*2.0;
    d_term[1]=Kd1[1]*(k_roll - lastinput[1])*2.0;
    d_term[2]=Kd1[2]*(yawf - lastinput[2])*2.0;
    td_counter=d_time;
    if(d_term[0] > outmax){
        d_term[0]= outmax;
    }
    d_term[1]=Ki1[1]*serro[1];
    if(d_term[1] > outmax){
        d_term[1]= outmax;
    }
    d_term[2]=Ki1[2]*serro[2];
    if(d_term[2] > outmax){
        d_term[2]= outmax;
    }
    td_counter=d_time;
}
td_counter-=1.0;

```

Cálculo da saída do controlador PID

```

output[0]=k_term[0]+i_term[0]+d_term[0];
output[1]=k_term[1]+i_term[1]+d_term[1];
output[2]=k_term[2]+i_term[2]+d_term[2];

```

Rotina para limitação da saída do controlador

```
if(output[0] > outmax){
output[0]= outmax;
}
if(output[0] < outmin){
output[0]= outmin;
}
if(output[1] > outmax){
output[1]= outmax;
}
if(output[1] < outmin){
output[1]= outmin;
}
if(output[2] > outmax){
output[2]= outmax;
}
if(output[2] < outmin){
output[2]= outmin;
}
lastinput[0]=k_pitch;
lastinput[1]=k_roll;
lastinput[2]=yawf2;
```

Configuração da saída PWM

```
ciclo1 = output[2];
set_pwm1_duty(abs((int16)(offset0+output[0])));
set_pwm2_duty(abs((int16)(offset1+output[1])));
```

Rotina para transmissão de dados via interface USB

```
usb_cdc_init();
usb_task();
if (usb_enumerated()) {
usb_cdc_putc_fast <<x;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<ax_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<ay_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<az_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<gx_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<gy_rate;
printf(usb_cdc_putc, " ");
```



```

usb_cdc_putc_fast <<gz_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<t;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<pitch;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<k_pitch;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<roll;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<k_roll;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<yawf;
printf(usb_cdc_putc, "\n\r");
output_toggle(LED);
}
}
}
}

```

APENDICE C – PROGRAMA DO MICROCONTROLADOR AUXILIAR PARA AQUISIÇÃO DE DADOS

Configurações do microcontrolador

```
#include <18F4550.h>
#fuses SPL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5,
      CPUDIV1, VREGEN
#use delay(clock=96MHz, USB_FULL)
#include <usb_cdc.h>
#use rs232(uart1, baud=19200)
#define MPU_SDA PIN_B0
#define MPU_SCL PIN_B1
#define LED PIN_D2
#define OFF output_low
#define ON output_high
#use I2C(master, fast, sda=MPU_SDA, scl=MPU_SCL)
#use TIMER(timer=1, tick=1us, bits=8, NOISR)
```

Inclusão de arquivos auxiliares

```
#include "MPU9250_V6.c"
#include "math.h"
#include "Kalman_p.h"
#include "Kalman_r.h"
#include "Kalman_y.h"
```

Definição de variáveis

```
#define RAD_TO_DEG 180/PI
#define outmin 0
#define outmax 300
#define sampletime 100
#define fim 500
int x,m,j;
double dt, ax_rate, ay_rate, az_rate, gx_rate, gy_rate, gz_rate, t, mx_rate, my_rate,
      mz_rate, mx_rate2, my_rate2, mz_rate2, ax_offset, ay_offset, az_offset,
      gx_offset, gy_offset, gz_offset, mx_offset, my_offset, mz_offset,
      pitch,roll,yaw,yawf,yawf2,k_pitch,k_roll,yawf22,k_pitch2,k_roll2, MOS[3],
      mx2,my2, acc_res, gyro_res, mag_res;
unsigned int16 timer;
```

```
void main()
{
```

Determinação da resolução dos sinais conforme []

```
    acc_res=2.0/32768.0;
    gyro_res=250.0/32768.0;
    mag_res=10.0*4912.0/32760.0;
```

Inicialização da MPU-9250

```
AK8963_asa ();  
delay_ms (50) ;  
Mpu9250_Init ();  
delay_ms (50) ;
```

Leitura dos dados de identificação

```
m = AK8963_Read(WHO_AM_I_AK8963);  
x = Mpu9250_Read(WHO_AM_I);  
timer = get_ticks ();  
set_ticks(0);  
pwm_off ();
```

Início da leitura dos dados de offset da MPU-9250

```
if ((x == 0x71) && (j != 0x18)){  
ax_offset=(signed int16)(Mpu9250_GetData (XA_OFFSET_H))/16384.0;  
ay_offset=(signed int16)(Mpu9250_GetData (YA_OFFSET_H))/16384.0;  
az_offset=(signed int16)(Mpu9250_GetData (ZA_OFFSET_H))/16384.0;  
gx_offset=(signed int16)(Mpu9250_GetData (XG_OFFSET_H))/131.0;  
gy_offset=(signed int16)(Mpu9250_GetData (YG_OFFSET_H))/131.0;  
gz_offset=(signed int16)(Mpu9250_GetData (ZG_OFFSET_H))/131.0;  
  
MOS[0]=(double)((((signed int16)(AK8963_Read (AK8963_ASAX)))-  
128.0)/256.0)+1.0;  
MOS[1]=(double)((((signed int16)(AK8963_Read (AK8963_ASAY)))-  
128.0)/256.0)+1.0;  
MOS[2]= (double)((((signed int16)(AK8963_Read (AK8963_ASAZ)))-  
128.0)/256.0)+1.0;  
}
```

Início do loop principal

```
while(1){  
pwm_on ();
```

Leitura em tempo real dos dados da MPU9250

```
j = AK8963_Read(AK8963_ST2)  
if ((x == 0x71) && (j != 0x18)){  
ax_rate=((signed int16)(Mpu9250_GetData (ACCEL_XOUT_H)))-  
ax_offset)*acc_res;  
ay_rate=((signed int16)(Mpu9250_GetData (ACCEL_YOUT_H)))-  
ay_offset)*acc_res;  
az_rate=((signed int16)(Mpu9250_GetData (ACCEL_ZOUT_H)))-  
az_offset)*acc_res;  
gx_rate=((signed int16)(Mpu9250_GetData (GYRO_XOUT_H)))-  
gx_offset)*gyro_res;  
gy_rate=((signed int16)(Mpu9250_GetData (GYRO_YOUT_H)))-  
gy_offset)*gyro_res;
```

```

gz_rate=((signed int16)(Mpu9250_GetData (GYRO_ZOUT_H)))-
gz_offset)*gyro_res;
mx_rate=((signed
int16)(AK8963_GetData(AK8963_XOUT_L))*MOS[0])*0.15*mag_res;
my_rate=((signed
int16)(AK8963_GetData(AK8963_YOUT_L))*MOS[1])*0.15*mag_res;
mz_rate=((signed
int16)(AK8963_GetData(AK8963_ZOUT_L))*MOS[2])*0.15*mag_res;
t=((signed int16)(Mpu9250_GetData (TEMP_OUT_H))/333.87)+21.0;

```

Aproximação para cálculo da derivada do tempo

```

dt =(double)(get_ticks () - timer)/1333;//68239;
timer = get_ticks ();
set_ticks(0);

```

Cálculo dos ângulos brutos e Aplicação do Filtro de Kalman

```

roll =(atan2 (ax_rate,sqrt(ay_rate * ay_rate + az_rate * az_rate))) *
RAD_TO_DEG;
k_roll = kalmanCalculate (roll, gy_rate, dt);//Filtro de Kalman

pitch =(atan2 (ay_rate,sqrt(ax_rate * ax_rate + az_rate * az_rate))) *
RAD_TO_DEG;
k_pitch = kalmanCalculate( pitch, gx_rate, dt); //Filtro de Kalman

mx_rate2=((mx_rate-32218.0)/(-12.05)*2.0)-1.0;
my_rate2=((my_rate-(-55804.0))/(45.3550)*2.0)-1.0;
mz_rate2=((mz_rate-(-4.1145))/(85.4650)*2.0)-1.0;

yaw=atan2(my2,mx2)*RAD_TO_DEG;
yaw+=(-0.3659)*RAD_TO_DEG;
yawf =kalmanCalculate (yaw, gz_rate, dt); //Filtro de Kalman

```

Rotina para transmissão de dados via interface USB

```

usb_cdc_init();
usb_task();
if (usb_enumerated()) {
usb_cdc_putc_fast <<x;
printf(usb_cdc_putc," ");
usb_cdc_putc_fast <<ax_rate;
printf(usb_cdc_putc," ");
usb_cdc_putc_fast <<ay_rate;
printf(usb_cdc_putc," ");
usb_cdc_putc_fast <<az_rate;
printf(usb_cdc_putc," ");
}

```

```

usb_cdc_putc_fast <<gx_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<gy_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<gz_rate;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<t;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<pitch;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<k_pitch;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<roll;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<k_roll;
printf(usb_cdc_putc, " ");
usb_cdc_putc_fast <<yawf;
printf(usb_cdc_putc, "\n\r");
output_toggle(LED);
}
}
}
}

```

APENDICE D – PROGRAMA DO MICROCONTROLADOR PARA APLICAÇÃO DO FILTRO DE KALMAN

```
#ifndef _Kalman_h
#define _Kalman_h

float Q_angle = 0.001
float Q_gyrro = 0.003
float R_angle = 0.03

float y_bias = 0.0;
float y_angle = 0.0;
float P_00 = 0.0, P_01 = 0.0, P_10 = 0.0, P_11 = 0.0;
float y, S;
float K_0_r, K_1_r;

float kalmanCalculate(float newAngle, float newRate, unsigned int16 looptime)
{
    float dt = (float)(looptime_r)/1000;
    y_angle += dt * (newRate - y_bias);

    P_00 += dt * (dt * P_11 - P_01 - P_10 + Q_angle);
    P_01 -= dt * P_11;
    P_10 -= dt * P_11;
    P_11 += Q_gyrro * dt;

    S = P_00 + R_angle;

    K_0 = P_00 / S;
    K_1 = P_10 / S;

    y = newAngle - y_angle;

    y_angle += K_0 * y;
    y_bias += K_1 * y;

    P_00 -= K_0 * P_00;
    P_01 -= K_0 * P_01;
    P_10 -= K_1 * P_00;
    P_11 -= K_1 * P_01;

    return y_angle;
}
#endif
```

APENDICE E – PROGRAMA DO MICROCONTROLADOR PARA CONFIGURAÇÃO DO MPU-9250

```
#define MPU9250_I2C_Wr(value) i2c_write(value)
#define MPU9250_I2C_Rd(value) i2c_read(value)
#define MPU9250_I2C_Stop() i2c_stop()
#define MPU9250_I2C_Start() i2c_start()

#define AK8963_I2C_Wr(value) i2c_write(value)
#define AK8963_I2C_Rd(value) i2c_read(value)
#define AK8963_I2C_Stop() i2c_stop()
#define AK8963_I2C_Start() i2c_start()

#define MPU9250_ADDRESS          0xD0
#define MPU9250_ADDRESS_AD0_LOW 0x68
#define MPU9250_ADDRESS_AD0_HIGH 0x69
#define MPU9250_DEFAULT_ADDRESS MPU9250_ADDRESS_AD0_LOW
//Magnetometer Registers
#define AK8963_ADDRESS 0x0C<<1
#define WHO_AM_I_AK8963 0x00
#define INFO 0x01
#define AK8963_ST1 0x02
#define AK8963_XOUT_L 0x03
#define AK8963_XOUT_H 0x04
#define AK8963_YOUT_L 0x05
#define AK8963_YOUT_H 0x06
#define AK8963_ZOUT_L 0x07
#define AK8963_ZOUT_H 0x08
#define AK8963_ST2 0x09
#define AK8963_CNTL 0x0A
#define AK8963_ASTC 0x0C
#define AK8963_I2CDIS 0x0F
#define AK8963_ASAX 0x10
#define AK8963_ASAY 0x11
#define AK8963_ASAZ 0x12
#define SELF_TEST_X_GYRO 0x00
#define SELF_TEST_Y_GYRO 0x01
#define SELF_TEST_Z_GYRO 0x02
#define SELF_TEST_X_ACCEL 0x0D
#define SELF_TEST_Y_ACCEL 0x0E
#define SELF_TEST_Z_ACCEL 0x0F
#define SELF_TEST_A 0x10
#define XG_OFFSET_H 0x13
#define XG_OFFSET_L 0x14
#define YG_OFFSET_H 0x15
#define YG_OFFSET_L 0x16
#define ZG_OFFSET_H 0x17
#define ZG_OFFSET_L 0x18
```

```

#define SMPLRT_DIV      0x19
#define CONFIG          0x1A
#define GYRO_CONFIG     0x1B
#define ACCEL_CONFIG    0x1C
#define ACCEL_CONFIG2   0x1D
#define LP_ACCEL_ODR    0x1E
#define WOM_THR         0x1F
#define MOT_DUR         0x20
#define ZMOT_THR        0x21
#define ZRMOT_DUR       0x22
#define FIFO_EN         0x23
#define I2C_MST_CTRL    0x24
#define I2C_SLV0_ADDR   0x25
#define I2C_SLV0_REG    0x26
#define I2C_SLV0_CTRL   0x27
#define I2C_SLV1_ADDR   0x28
#define I2C_SLV1_REG    0x29
#define I2C_SLV1_CTRL   0x2A
#define I2C_SLV2_ADDR   0x2B
#define I2C_SLV2_REG    0x2C
#define I2C_SLV2_CTRL   0x2D
#define I2C_SLV3_ADDR   0x2E
#define I2C_SLV3_REG    0x2F
#define I2C_SLV3_CTRL   0x30
#define I2C_SLV4_ADDR   0x31
#define I2C_SLV4_REG    0x32
#define I2C_SLV4_DO     0x33
#define I2C_SLV4_CTRL   0x34
#define I2C_SLV4_DI     0x35
#define I2C_MST_STATUS  0x36
#define INT_PIN_CFG     0x37
#define INT_ENABLE      0x38
#define DMP_INT_STATUS  0x39
#define INT_STATUS      0x3A
#define ACCEL_XOUT_H    0x3B
#define ACCEL_XOUT_L    0x3C
#define ACCEL_YOUT_H    0x3D
#define ACCEL_YOUT_L    0x3E
#define ACCEL_ZOUT_H    0x3F
#define ACCEL_ZOUT_L    0x40
#define TEMP_OUT_H      0x41
#define TEMP_OUT_L      0x42
#define GYRO_XOUT_H     0x43
#define GYRO_XOUT_L     0x44
#define GYRO_YOUT_H     0x45
#define GYRO_YOUT_L     0x46
#define GYRO_ZOUT_H     0x47
#define GYRO_ZOUT_L     0x48

```



```

#define EXT_SENS_DATA_00 0x49
#define EXT_SENS_DATA_01 0x4A
#define EXT_SENS_DATA_02 0x4B
#define EXT_SENS_DATA_03 0x4C
#define EXT_SENS_DATA_04 0x4D
#define EXT_SENS_DATA_05 0x4E
#define EXT_SENS_DATA_06 0x4F
#define EXT_SENS_DATA_07 0x50
#define EXT_SENS_DATA_08 0x51
#define EXT_SENS_DATA_09 0x52
#define EXT_SENS_DATA_10 0x53
#define EXT_SENS_DATA_11 0x54
#define EXT_SENS_DATA_12 0x55
#define EXT_SENS_DATA_13 0x56
#define EXT_SENS_DATA_14 0x57
#define EXT_SENS_DATA_15 0x58
#define EXT_SENS_DATA_16 0x59
#define EXT_SENS_DATA_17 0x5A
#define EXT_SENS_DATA_18 0x5B
#define EXT_SENS_DATA_19 0x5C
#define EXT_SENS_DATA_20 0x5D
#define EXT_SENS_DATA_21 0x5E
#define EXT_SENS_DATA_22 0x5F
#define EXT_SENS_DATA_23 0x60
#define MOT_DETEC_STATUS 0x61
#define I2C_SLV0_DO      0x63
#define I2C_SLV1_DO      0x64
#define I2C_SLV2_DO      0x65
#define I2C_SLV3_DO      0x66
#define I2C_MT_DELAY_CTRL 0x67
#define SIGN_PATH_RESET  0x68
#define MOT_DETECT_CTRL  0x69
#define USER_CTRL        0x6A
#define PWR_MGMT_1        0x6B
#define PWR_MGMT_2        0x6C
#define DMP_BANK          0x6D
#define DMP_RW_PNT        0x6E
#define DMP_REG_1         0x70
#define DMP_REG_2         0x71
#define FIFO_COUNTH       0x72
#define FIFO_COUNTL       0x73
#define FIFO_R_W          0x74
#define WHO_AM_I          0x75
#define XA_OFFSET_H       0x77
#define XA_OFFSET_L       0x78
#define YA_OFFSET_H       0x7A
#define YA_OFFSET_L       0x7B
#define ZA_OFFSET_H       0x7D

```

```

#define ZA_OFFSET_L          0x7E
#include "math.h"
unsigned char MPU9250_Read(unsigned char address)
{
    int8 Data;
    MPU9250_I2C_Start();
    MPU9250_I2C_Wr( MPU9250_ADDRESS );
    MPU9250_I2C_Wr(address);
    MPU9250_I2C_Start();
    MPU9250_I2C_Wr( MPU9250_ADDRESS | 1 );
    Data=MPU9250_I2C_Rd(0);
    MPU9250_I2C_Stop();
    return Data;
}

void Mpu9250_Write(unsigned char address,unsigned char Data)
{
    MPU9250_I2C_Start();
    MPU9250_I2C_Wr( MPU9250_ADDRESS );
    MPU9250_I2C_Wr( address);
    MPU9250_I2C_Wr( Data);
    MPU9250_I2C_Stop();
}

unsigned char AK8963_Read(unsigned char address)
{
    int8 Data;
    AK8963_I2C_Start();
    AK8963_I2C_Wr( AK8963_ADDRESS );
    AK8963_I2C_Wr(address);
    AK8963_I2C_Start();
    AK8963_I2C_Wr(AK8963_ADDRESS | 1 );
    Data=AK8963_I2C_Rd(0);
    AK8963_I2C_Stop();
    return Data;
}

void AK8963_Write(unsigned char address,unsigned char Data)
{
    AK8963_I2C_Start();
    AK8963_I2C_Wr( AK8963_ADDRESS );
    AK8963_I2C_Wr( address);
    AK8963_I2C_Wr( Data);
    AK8963_I2C_Stop();
}

int16 AK8963_GetData(unsigned char address)
{
    int16 H=0,L=0;
    AK8963_I2C_Start();
    AK8963_I2C_Wr( AK8963_ADDRESS );
    AK8963_I2C_Wr(address);

```

```

    AK8963_I2C_Start();
    AK8963_I2C_Wr( AK8963_ADDRESS |1);
H=i2C_read();
L=i2C_read(0);
    MPU9250_I2C_Stop();
    return (H<<8)|L;
}
int16 Mpu9250_GetData(unsigned char address)
{
    int16 H=0,L=0;
    MPU9250_I2C_Start();
    MPU9250_I2C_Wr( MPU9250_ADDRESS );
    MPU9250_I2C_Wr(address);
    MPU9250_I2C_Start();
    MPU9250_I2C_Wr( MPU9250_ADDRESS |1);
    H=i2C_read();
    L=i2C_read(0);
    MPU9250_I2C_Stop();
    return (H<<8)|L;
}
void AK8963_asa()
{
    Mpu9250_Write(PWR_MGMT_1, 0x80);
    Mpu9250_Write(PWR_MGMT_1, 0x01);
    Mpu9250_Write(PWR_MGMT_2, 0x00);
    Mpu9250_Write(CONFIG, 0x03);
    Mpu9250_Write(SMPLRT_DIV, 0x04);
    Mpu9250_Write(INT_PIN_CFG, 0x22);
    Mpu9250_Write(INT_ENABLE, 0x01);
    AK8963_Write(AK8963_CNTL, 0x00);
    AK8963_Write(AK8963_CNTL, 0x0F);
}
void Mpu9250_Init()
{
    Mpu9250_Write(PWR_MGMT_1, 0x80);
    Mpu9250_Write(PWR_MGMT_1, 0x01);
    Mpu9250_Write(PWR_MGMT_2, 0x00);
    Mpu9250_Write(CONFIG, 0x04);
    Mpu9250_Write(SMPLRT_DIV, 0x04);
    Mpu9250_Write(GYRO_CONFIG,0x00);
    Mpu9250_Write(ACCEL_CONFIG,0x00);
    Mpu9250_Write(ACCEL_CONFIG2,0x00);
    Mpu9250_Write(INT_PIN_CFG, 0x22);
    Mpu9250_Write(INT_ENABLE, 0x01);
    AK8963_Write(AK8963_CNTL, 0x00);
    AK8963_Write(AK8963_CNTL, 0x16);
}

```

ANEXO 1 – DIAGRAMA ELÉTRICO IMU

Circuito Principal

Circuito Auxiliar