Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Caixa Preta para carros: proposta para calibração, coleta e fusão de dados

Paulo B. Teixeira Neto

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Ricardo Zelenovsky

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Caixa Preta para carros: proposta para calibração, coleta e fusão de dados

Paulo B. Teixeira Neto

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Ricardo Zelenovsky (Orientador)
CIC/UnB

Prof. Dr. Alexandre Ricardo Soares Romariz
ENE - UnB

Dr. Vinícius Lima
Membro externo

Hudson Pereira Ramos
Membro externo

do Curso de Engenharia da Computação

Brasília, 26 de maio de 2021

Dedicatória

Aos meus avós: Eliete Coutinho dos Santos e Ely Barradas dos Santos.

Agradecimentos

Agradeço a todos que me apoiaram neste trabalho de graduação. Meus pais, amigos e especialmente ao orientador Ricardo Zelenovsky que esteve sempre presente e disposto a ajudar.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Este projeto apresenta uma plataforma que coleta e armazena dados inerciais, de forma que eles possam ser extraídos e utilizados para a reconstrução da dinâmica da orientação de um veículo, auxiliando o trabalho da perícia de trânsito, que muitas vezes não pode confiar na utilização de dispositivos de coleta de dados já presentes nos veículos. O sistema desenvolvido faz o uso de uma unidade de medição inercial capaz de medir campo magnético, aceleração e giro, que são usados em métodos de fusão de dados para a estimativa da orientação. Em especial, este texto trata do motivo de se incluir o magnetômetro no sistema e aborda uma biblioteca que implementa métodos detalhados de *self-test* e calibração de sensores, com o fim de garantir a integridade dos dados obtidos.

Palavras-chave: Caixa Preta, unidade de medição inercial, fusão de dados sensoriais

Abstract

The goal for this thesis is presenting a platform that collects and stores inertial data that can be extracted and used by experts to reconstruct the vehicle's attitude. The device includes an inertial measurement unit that is able to collect acceleration, angular velocity and magnetic field which can be used to reconstruct the orientations by using sensor fusion algorithms. This text also presents why the system should use the magnetometer to provide correct data in all the three axis and also presents detailed methods for self-test and calibration that are used to make sure the data is consistent.

Keywords: Black box, Inertial Measurement Unit, Sensor Fusion

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ABS anti-lock braking system.

ADC analog-to-digital converter.

ARW angular random walk.

DC Direct Current.

EDR Event Data Recorder.

EEPROM electrically erasable programmable read-only memory.

GPS Global Positioning System.

I2C Inter-Integrated Circuit.

IMU Inertial Measurement Unit.

MEMS Microelectromechanical systems.

NED North, East, Down.

ROM Read only memory.

SPI Serial Peripheral Interface.

Capítulo 1

Introdução

Levantamentos da Organização Mundial da Saúde apontam que acidentes de trânsito causam 1,35 milhões de mortes por ano, sendo que 93% das fatalidades ocorrem em países de baixa e média renda [1]. No Brasil, este número chega a mais de 30 mil [2]. O país posiciona-se entre os dez com maior mortalidade nas ruas, rodovias e estradas.

Segundo o artigo 176 do Código de Trânsito Brasileiro [3], em caso de acidentes de trânsito com vítimas, é obrigatória a realização de perícia, que é uma investigação técnico-científica que analisa a dinâmica do acidente após o ocorrido, o reconstruindo a partir de provas como marcas de frenagem, fotografias, topografia, sinalização, entre outros [4]. O profissional envolvido é encarregado de escrever um laudo imparcial que poderá ser utilizado como prova em eventuais processos judiciais.

A análise pericial é extensa e complexa, sendo muitas vezes impossível a aquisição de dados precisos. Como exemplo de dificuldade nesse processo, a utilização de freios ABS reduz significativamente a existência de marcas de frenagem no asfalto, o que dificulta a estimação precisa da velocidade inicial do veículo, dado primordial para a reconstrução total da dinâmica do acidente. [5] Nesse contexto, o avanço da tecnologia permitiu a criação de dispositivos eletrônicos chamados de EDR, [6] do inglês *Event Data Recorder*, que são capazes de coletar continuamente dados inerciais dos veículos como velocidade, aceleração e giro, e armazená-los numa memória EEPROM após a detecção de uma colisão. Esses dados podem ser posteriormente extraídos e utilizados por profissionais para caracterizar acidentes. Dispositivos EDR também são informalmente chamados de “Caixas Pretas”, nome dado em alusão aos dispositivos de armazenamento de dados encontrados em aviões.

1.1 Definição do problema

Até o momento não existe no Brasil uma padronização rigorosa sobre o formato dos dados que dispositivos EDR, como o existente no *Airbag*, armazenam nos veículos. A extração desses dados não é um procedimento barato ou trivial, já que cada montadora os armazena como bem entende e não necessariamente divulga o procedimento de sua extração. [5] Isso é um sério problema para a comunidade *forense*, que não pode sempre contar com o auxílio desses dispositivos em suas análises.

1.2 Objetivos do projeto

1.2.1 Objetivos gerais

Com isso, o objetivo geral deste trabalho é propor um dispositivo eletrônico de baixo custo, externo à fabricação do veículo, que armazena dados inerciais num formato aberto e compreensível que pode ser utilizado numa gama de aplicações.

Este projeto é uma continuação do trabalho de mestrado de Vinícius de Oliveira Lima: "Proposta de Plataforma Inercial para Auxiliar na Perícia de Acidentes de Trânsito" [5] e dos projetos finais de graduação de Hudson Pereira Ramos e Vanessa Oliveira Lucena: "Proposta de plataforma inercial e simulador 3D para periciar acidentes de trânsito" [7] e Gabriela da Silva Lopes: "Caixa Preta para Veículos Automotivos" [8].

1.2.2 Objetivos específicos

Os objetivos deste texto em específico são:

- Apresentar a nova versão do dispositivo EDR desenvolvido, construída em circuito impresso.
- Garantir a integridade de dados inerciais a partir de procedimentos detalhados de *self-test* e calibrações.
- Introduzir o uso do magnetômetro ao sistema, mostrando sua importância em um processo de fusão de dados sensoriais.
- Garantir que os dados sensoriais obtidos estejam num formato legível que pode ser direcionado para outras aplicações.

1.3 Metodologia

Este projeto visa unir conhecimentos científicos com o fim de propor uma solução prática capaz de auxiliar o trabalho investigativo de acidentes de trânsito. A partir do levantamento bibliográfico de temas como medidas inerciais, parametrizações matemáticas de rotações e filtros de fusão de dados, é proposto um dispositivo EDR que coleta dados inerciais de veículos automotivos. O pós-processamento desses dados permite uma abordagem quantitativa, que possibilita portanto, a verificação experimental da viabilidade do dispositivo.

Os procedimentos realizados neste estudo foram:

1. Determinação do escopo do problema a ser resolvido;
2. Revisão bibliográfica buscando maior compreensão das tecnologias envolvidas;
3. Desenvolvimento do código embarcado do dispositivo de gravação de dados inerciais;
4. Desenvolvimento de simuladores e calibradores sensoriais utilizando o software Matlab;
5. Coleta de dados controlados, endereçando problemas específicos de cada sensor;
6. Aplicação de procedimentos de fusão sensorial para estimar orientação;
7. Conclusão analisando os resultados obtidos, levantando problemas a serem resolvidos em projetos futuros.

1.4 Estruturação do Documento

Este projeto é dividido em cinco capítulos:

O Capítulo 1 é a introdução que define o problema a ser resolvido e a estruturação do projeto.

O Capítulo 2 a fundamentação teórica que contextualiza a tecnologia e os métodos matemáticos utilizados.

O Capítulo 3 introduz a plataforma de coleta de dados inerciais desenvolvida, incluindo a configuração do dispositivo, sua organização de memória e como operá-lo para realizar calibragens, métodos de *self-test* e coleta de dados inerciais.

O Capítulo 4 mostra simulações controladas, mostrando todo o procedimento de *self-test*, calibração e simulações de rotações, comparando as respostas com e sem o uso do magnetômetro.

O Capítulo 5 finaliza o estudo com a conclusão e sugestões para projetos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo visa realizar uma introdução geral sobre os temas abordados no projeto. Serão introduzidos sensores de tecnologia MEMS (Sistemas microeletromecânicos), bem como a parametrização matemática de seus dados, métodos de calibração e porque devemos combinar dados de diferentes sensores para estimar a orientação de objetos.

2.1 Sensores MEMS

Sistemas microeletromecânicos (MEMS) são sistemas que combinam pequenos componentes eletrônicos com componentes não-eletrônicos. A interface analógica e digital coexiste no mesmo chip. A escala desses dispositivos varia de alguns micrômetros até alguns milímetros. Seu funcionamento consiste em microssensores que coletam dados a partir de fenômenos mecânicos, químicos ou magnéticos do ambiente, podendo encaminhar esses dados para micro-atuadores que são responsáveis pela conversão de energia em movimento [9]. Essa tecnologia é amplamente utilizada na indústria automobilística, de telecomunicações e na medicina devido a seu baixo custo e consumo energético. Neste projeto, com o fim de se coletar dados inerciais, foram utilizados três sensores de tipo MEMS: giroscópio, acelerômetro e magnetômetro, que se encontram no módulo MPU-9250 da Invensense Inc., que é uma unidade de medição inercial (IMU).

2.1.1 Giroscópio

Giroscópios são dispositivos utilizados para medição de velocidade angular. Os giroscópios MEMS utilizados no MPU-9250 são de estrutura vibratória cujo funcionamento consiste em determinar a taxa de rotação a partir da pseudo-força exercida pelo efeito Coriolis [10], que opera sobre sistemas de referência que são rotacionados em relação a um referencial inercial e causa uma vibração detectada por massas de prova capacitivas. O sinal é

então amplificado, demodulado e filtrado de forma a produzir uma tensão proporcional à velocidade angular do sensor, que no MPU-9250 é digitalizada em conversores analógico-digital (ADC) de 16 bits [11]. A Figura 2.1 [12] mostra como a força de Coriolis $F_{Coriolis}$ afeta a movimentação das placas capacitivas num sistema rotacionado com velocidade angular Ω_Z.

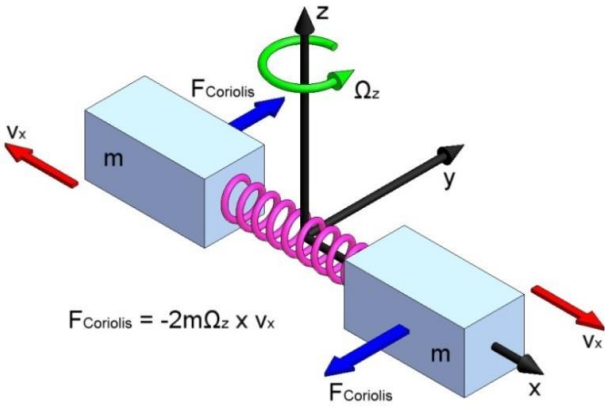


Figura 2.1: Força de Coriolis.

A saída de um giroscópio $y_{\omega,t}$ pode ser modelada como

$$y_{\omega,t} = \omega^b_{ib,t} + \delta^b_{\omega,t} + e^b_{\omega,t}, \tag{2.1}$$

sendo $\omega^b_{ib,t}$ a velocidade angular em torno de seu próprio eixo, $\delta^b_{\omega,t}$ um erro aditivo considerado como o *bias* que varia lentamente com o tempo e $e^b_{\omega,t}$ um ruído que normalmente é modelado como gaussiano.

2.1.2 Acelerômetro

Acelerômetros MEMS são dispositivos capazes de medir tanto a aceleração estática da gravidade ao se inclinar o sensor, como a aceleração dinâmica resultante de sua movimentação. Esses sensores são amplamente utilizados para medir inclinações, forças inerciais, impactos e vibrações.

Seu funcionamento consiste em pequenas massas de prova suspensas em cada eixo que se movem de acordo com as forças exercidas sobre o objeto. Essa movimentação é detectada diferencialmente por sensores capacitivos como ilustrados na Figura 2.2 [10]. No MPU-9250, os circuitos internos produzem então uma tensão proporcional à aceleração realizada, que é digitalizada em cada eixo por conversores analógico-digitais de 16 bits, com valores proporcionais ao campo gravitacional da Terra g.

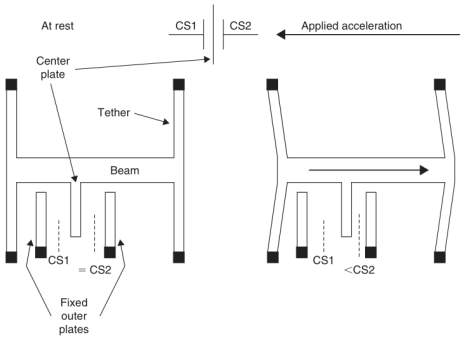


Figura 2.2: Acelerômetro MEMS.

A grandeza medida por esses dispositivos chama-se Força específica, que é definida por

$$f^b = R^{bn}(a^n_{ii} - g^n), \tag{2.2}$$

em que a^n_{ii} é a aceleração linear do sensor em relação ao referencial de navegação e g é o vetor da gravidade. R^{bn} indica a rotação do sistema para o referencial do objeto.

A saída do acelerômetro pode ser modelada por

$$y_{a,t} = f^b_t + \delta^b_{a,t} + e^b_{a,t}, \tag{2.3}$$

sendo f^b_t a força específica num instante t, $\delta^b_{a,t}$ o *bias* do acelerômetro e $e^b_{a,t}$ um ruído normalmente modelado como gaussiano.

2.1.3 Magnetômetro

Magnetômetros são sensores que detectam campo magnético. Os magnetômetros MEMS mais comuns utilizam-se do efeito Hall, que consiste na diferença de potencial gerada a partir de um campo magnético perpendicular à corrente elétrica do dispositivo. A Figura 2.3 [13] mostra como essa diferença de potencial é criada.

O MPU-9250 possui o magnetômetro AK8963, que possui sensores magnéticos nos eixos X, Y e Z, que possibilitam medidas numa escala de ± 4800uT. O sinal é digitalizado por Conversores Analógico-Digitais de 16 bits. [11]

A saída do magnetômetro pode ser modelada como

$$y_{m,t} = R^{bn}_t B \begin{pmatrix} cos(\delta) \\ 0 \\ sen(\delta) \end{pmatrix} + e^b_{m,t}, \tag{2.4}$$

sendo B o módulo do campo magnético, δ o ângulo de declinação magnética, R_t^{bn} a rotação do eixo do sensor para o eixo de navegação e $e_{m,t}^b$ um ruído gaussiano, que nesse dispositivo é consideravelmente maior do que no acelerômetro e giroscópio.

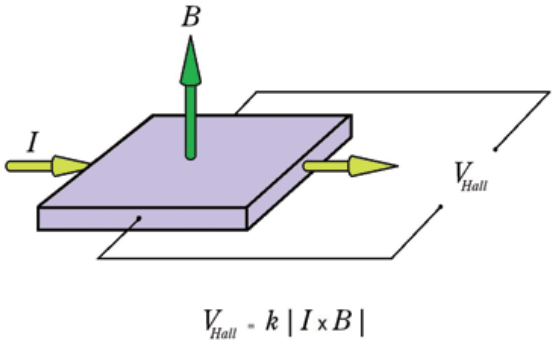


Figura 2.3: Efeito Hall.

2.1.4 Self-Test

O *Self-Test* é um mecanismo implementado em alguns sensores a nível de hardware que permite que os usuários realizem testes funcionais, verificando a integridade mecânica e funcional do dispositivo. Esse teste deve ser realizado sem a presença de movimento. Consiste em ativar o modo *self-test*, que faz com que uma atuação eletrônica mova a massa dos sensores para uma posição pré-estabelecida, sendo então a saída do sensor comparada com os resultados obtidos nos testes realizados na fabricação do dispositivo em questão. No MPU-9250, o *self-test* pode ser realizado no giroscópio, no acelerômetro e no magnetômetro AK8963 [14].

Self-Test do giroscópio e acelerômetro

A rotina de *self-test* para o IMU deve ser implementada a partir dos seguintes procedimentos:

1. Alterar a configuração de filtro digital passa-baixa do giroscópio e acelerômetro para 2. Essa opção é encontrada nos bits DLPF do registrador CONFIG. Isso faz com que a largura de banda dos filtros passa-baixa dos sensores mude para 92 HZ e a taxa de amostragem para 1 kHz;
2. Configurar a escala do giroscópio para $\pm 250°/s$ e a do acelerômetro para $\pm 2g$;
3. Esperar 250ms para estabilizar a configuração;

4. Com o *self-test* desligado, realizar a leitura de 200 medidas do giroscópio e acelerômetro, calculando a média entre elas. O que resulta em

$$\text{OS} = \begin{pmatrix} \text{GX\_OS} & \text{GY\_OS} & \text{GZ\_OS} & \text{AX\_OS} & \text{AY\_OS} & \text{AZ\_OS} \end{pmatrix}^T; \tag{2.5}$$

5. Habilitar o *self-test* nos três eixos do acelerômetro e giroscópio pelos registradores ACCEL_CONFIG e GYRO_CONFIG;

6. Realizar 200 leituras do giroscópio e acelerômetro, calculando a média entre elas. O que resulta em

$$\text{ST\_OS} = \begin{pmatrix} \text{GX\_ST\_OS} \\ \text{GY\_ST\_OS} \\ \text{GZ\_ST\_OS} \\ \text{AX\_ST\_OS} \\ \text{AY\_ST\_OS} \\ \text{AZ\_ST\_OS} \end{pmatrix}; \tag{2.6}$$

7. Calcular a resposta do *self-test*, que consiste na diferença entra a média das leituras com o *self-test* ativado e desativado:

$$\text{ST\_RESP} = \text{ST\_OS} - \text{OS} = \begin{pmatrix} \text{AXST} \\ \text{AYST} \\ \text{AZST} \\ \text{GXST} \\ \text{GYST} \\ \text{GZST} \end{pmatrix}; \tag{2.7}$$

8. Configurar o IMU para operação normal, voltando para as escalas previamente utilizadas.

O critério para se passar no teste é dado pela Tabela 2.1, em que ST_OTP é o valor de *self-test* de fábrica dado por

$$\text{ST\_OTP} = 2620/2^{\text{FS}} \times 1,01^{\text{ST\_CODE}-1}, \tag{2.8}$$

em que FS é o fator de escala e ST_CODE um valor calculado em fábrica que pode ser acessado pelos registradores de *self-test*.

Eixo	**Critério**
g_x	$\frac{GXST}{GXST\_OTP} > 0,5$
g_y	$\frac{GYST}{GYST\_OTP} > 0,5$
g_z	$\frac{GZST}{GZST\_OTP} > 0,5$
a_x	$0,5 < \frac{AXST}{AXST\_OTP} < 1,5$
a_y	$0,5 < \frac{AXST}{AXST\_OTP} < 1,5$
a_z	$0,5 < \frac{AXST}{AXST\_OTP} < 1,5$

Tabela 2.1: Critério de sucesso para o *self-test* do IMU

Self-Test do magnetômetro AK8963

Para a realização do *self-test* no magnetômetro, a seguinte rotina deve ser implementada:

1. Configurar o magnetômetro no modo *power-down*;
2. Habilitar o bit *self* de ASTC;
3. Habilitar o bit de *self-test* do magnetômetro;
4. Ler a saída $H = \begin{pmatrix} h_x & h_y & h_z \end{pmatrix}^T$;
5. Desabilitar o bit *self* de ASTC;
6. Aplicar o ajuste de sensibilidade

$$Hadj = H \times (\frac{(ASA - 128) \times 0,5}{128} + 1), \tag{2.9}$$

   em que ASA pode ser encontrado na ROM do AK8963.

Os critérios para o sucesso são dados pela Tabela 2.2.

	$\boldsymbol{h_x}$	$\boldsymbol{h_y}$	$\boldsymbol{h_z}$
critério	$-200 \leq h_x \leq 200$	$-200 \leq h_y \leq 200$	$-3200 \leq h_z \leq -800$

Tabela 2.2: Critério para o sucesso do *self-test* do magnetômetro

2.2 Calibração de sensores MEMS

Sensores MEMS possuem erros sistemáticos. Equipamentos de diferentes ou de mesmo fabricante podem apresentar diferentes respostas nas mesmas condições físicas. A acurácia de suas leituras é altamente dependente de sua devida calibração. Calibração é o processo utilizado para reduzir incertezas nas medidas sensoriais. Consiste em submeter o sensor a entradas conhecidas e encontrar os parâmetros que mais aproximam as leituras obtidas das leituras esperadas. Os métodos de calibração mais simples consistem em apenas corrigir *bias* e ganho sem assumir a não-linearidade e interdependência entre parâmetros. *Bias* refere-se ao *offset* DC que diferencia o dado lido para o dado real. Ganho refere-se à escala de medição.

2.2.1 Calibração do giroscópio

Como visto anteriormente na expressão 2.1, o giroscópio mede a velocidade angular $\omega^b_{ib,t}$ porém possui duas principais inacurácias: Um bias $\delta^b_{\omega,t}$ quase DC que varia lentamente com o tempo e um ruído branco $e^b_{\omega,t}$ de alta frequência causado pelo chamado passeio aleatório angular, no inglês: *Angle Random Walk* (ARW), que é causado por reações termo-elétricas do dispositivo e não pode ser removido apenas com a calibração.

Para obter uma estimativa do *bias*, basta calcular médias de leituras sensoriais enquanto o dispositivo está parado por alguns segundos:

$$\delta_\omega = \begin{pmatrix} \bar{g_x} & \bar{g_y} & \bar{g_z} \end{pmatrix}^T. \tag{2.10}$$

Cada leitura subsequente deve ser subtraída de δ_ω. Contudo, essa é apenas uma aproximação que pode ser utilizada como estimativa inicial, pois como visto, o *bias* varia com o tempo, mesmo que lentamente. Além disso, esta calibração não trata do ruído gaussiano, que quando integrado, aumenta o erro num fator proporcional à raiz quadrada do tempo [15]. Uma boa estimativa de giro deve depender da combinação de leituras com outros sensores e processos de filtragem de dados adequados.

2.2.2 Calibração do acelerômetro

O acelerômetro, assim como o giroscópio, possui erros de bias e passeio aleatório.

Com o sensor em estado estacionário, apenas a força peso é exercida e o módulo da

aceleração resultante deveria ser de 1g, que é a aceleração gravitacional exercida sobre o corpo. Porém, com o sensor descalibrado, esse valor é diferente. Esses erros são problemáticos para as estimativas de posição e velocidade, pois estas necessitam de um processo de integração chamado de *dead-reckoning*. Um *bias* constante na aceleração causará um erro que aumenta linearmente com o tempo para a estimativa da velocidade e quadraticamente durante a estimativa da posição. Essa imprecisão é normalmente chamada de *drift* de integração.

O processo de remoção do *bias* do acelerômetro é mais complexo do que o do giroscópio pois o sensor sofre efeitos da aceleração gravitacional. A medida corrigida do acelerômetro pode ser modelada como

$$a\_corr = (a - \delta_a) \times m_a, \tag{2.11}$$

em que a é a leitura sensorial da aceleração, δ_a^b o fator de *offset* e m_a o de escala.

Nesta seção, serão apresentados dois métodos para estimar δ_a^b e m_a. Ambos necessitam de medições sensoriais em estado estacionário e buscam com que o módulo da aceleração resultante seja igual a 1 g.

Calibração do acelerômetro por medição de seis pontos

É um método de calibração simples que consiste na realização de seis leituras com os eixos do sensor alinhados com o vetor da aceleração gravitacional e a partir dos valores esperados, obter a correção que transforma o módulo da aceleração em 1g [16]. São realizadas as seguintes leituras:

1. Leitura L1 com eixo x virado para cima.
2. Leitura L2 com eixo x virado para baixo.
3. Leitura L3 com eixo y virado para cima.
4. Leitura L4 com eixo y virado para baixo.
5. Leitura L5 com eixo z virado para cima.
6. Leitura L6 com eixo z virado para baixo.

A leitura sensorial do MPU-9250 é um valor de 16 bits com sinal. O ponto de aceleração zero em um determinado eixo é a média entre a leitura com o eixo virado para cima e a leitura com o eixo virado para baixo. Com isso os *offsets* podem ser estimados como

$$\begin{pmatrix} \delta_{ax} \\ \delta_{ay} \\ \delta_{az} \end{pmatrix} = \begin{pmatrix} \frac{(L1_x + L2_x)}{2} \\ \frac{(L3_y + L4_y)}{2} \\ \frac{(L5_z + L6_z)}{2} \end{pmatrix}. \tag{2.12}$$

A escala pode ser obtida a partir do conhecimento de que a diferença entre uma leitura com um determinado eixo voltado para cima e outra com o eixo voltado para baixo deve ser igual a 2 g. Então vale a expressão

$$\begin{pmatrix} \frac{1}{m_{ax}} \\ \frac{1}{m_{ay}} \\ \frac{1}{m_{az}} \end{pmatrix} = \begin{pmatrix} \frac{(L1_x - L2_x)}{2} \\ \frac{(L3_y - L4_y)}{2} \\ \frac{(L5_z - L6_z)}{2} \end{pmatrix}. \tag{2.13}$$

Para a aplicação deste método, é necessária alta precisão no posicionamento vertical do sensor, pois é considerado que o módulo do eixo orientado na vertical será 1g enquanto o dos outros eixos 0g, o que na prática, pode não ser uma boa aproximação sem o uso de equipamentos de calibragem especializados. A vantagem deste método é sua simplicidade, podendo inclusive ser implementado facilmente no Arduíno.

Calibração do acelerômetro por mínimos quadrados

A calibração do acelerômetro discutida anteriormente é simples de ser implementada, porém possui a limitação de que só são realizadas seis leituras e elas precisam de um alinhamento axial muito preciso difícil de ser obtido. O método de calibração por mínimos quadrados [16] oferece um modelo matemático que lida com esses problemas pois permite uma quantidade arbitrária de leituras e não requer que o sensor seja posicionado precisamente em posições específicas.

Um problema de mínimos quadrados é definido como:
Dado um conjunto de M pontos (x_i, y_i) com $1 \leq i \leq M$ e uma função $y = f(x, \beta)$, com β sendo um vetor com n valores, queremos encontrar os valores de β tal que

$$\beta = \arg\min \sum_{i=1}^{M} r_i^2, \tag{2.14}$$

sendo r_i o resíduo dado por

$$r_i = y_i - f(x_i, \beta). \tag{2.15}$$

Para este método de calibração, devem ser coletadas M medidas (a_{xi}, a_{yi}, a_{zi}) com o sensor parado em orientações distintas. Como o sensor não está em movimento, o módulo da aceleração medida em cada leitura deveria ser 1. Contudo, como as leituras realizadas não são ideais, é esperada a existência de erros. O erro para cada leitura pode ser obtido a partir de seu módulo como:

$$e_i = \left(\frac{a_{xi} - \delta_{ax}}{\frac{1}{m_{ax}}} \right)^2 + \left(\frac{a_{yi} - \delta_{ay}}{\frac{1}{m_{ay}}} \right)^2 + \left(\frac{a_{zi} - \delta_{az}}{\frac{1}{m_{az}}} \right)^2 - 1. \tag{2.16}$$

O objetivo deste método é encontrar os parâmetros β tais que

$$\beta = \begin{pmatrix} \delta_{ax} \\ \delta_{ay} \\ \delta_{az} \\ m_{ax} \\ m_{ax} \\ m_{ax} \end{pmatrix} = \arg\min \sum_{i=1}^{M} e_i^2. \tag{2.17}$$

Este pode ser caracterizado como um problema de mínimos quadrados não-linear e existem na literatura diversos métodos numéricos para sua resolução, como o método de Gauss-Newton ou o método de Levenberg-Marquardt. No Matlab, esse procedimento pode ser efetuado pela função *lsqnonlin* [17], que implementa o método de Levenberg-Marquardt.

2.2.3 Calibração do magnetômetro

Um magnetômetro ideal produz a saída

$$\mathrm{yi}_{m,t} = R_t^{bn} B \begin{pmatrix} cos(\delta) \\ 0 \\ sen(\delta) \end{pmatrix}. \tag{2.18}$$

Ao calcularmos o produto escalar $\mathrm{yi}_{m,t} \cdot \mathrm{yi}_{m,t}$ temos

$$\mathrm{yi}_{m,t} \cdot \mathrm{yi}_{m,t} = (\mathrm{yi}_{m,t})^T \mathrm{yi}_{m,t} = h_x^2 + h_y^2 + h_z^2 = B^2, \tag{2.19}$$

que é a equação de uma esfera posicionada no centro e que possui raio B. A componente de rotação é eliminada pois R^T indica a rotação inversa de forma que $(R_t^{bn})^T R_t^{bn} = I$.

Porém num sistema real, o campo magnético está sujeito a dois principais tipos de distorções: *Hard-Iron* e *Soft-Iron* que fazem com que a saída deixe de ser esférica e assuma a forma de um elipsóide [18]. Os métodos de calibração apresentados nesta seção têm como objetivo removê-las.

Distorções *hard-iron* são distorções aditivas causadas por objetos próximos ao magnetômetro que geram seu próprio campo magnético. Caso o magnetômetro seja rotacionado em todas as direções, a distorção *hard-iron* faz com que a esfera, que idealmente estaria posicionada no centro, seja deslocada. A distorção *Hard-Iron* pode ser modelada como um vetor de deslocamento V que possui 3 elementos e é somado ao campo ideal.

Distorções *soft-iron* são distorções vindas de materiais que não geram seu próprio campo magnético, porém são superfícies que alteram o campo magnético transmitido sobre elas. É um efeito comum em algumas superfícies metálicas. Ao rotacionarmos o magnetômetro em todas as direções, o resultado que idealmente seria uma esfera se deforma para um elipsóide. A distorção *soft-iron* pode ser modelada como uma matriz 3x3 W que multiplica o campo ideal.

O campo magnético que sofre com esses dois tipos de distorção pode ser modelado como

$$\mathrm{yd}_{m,t} = W R_t^{bn} B \begin{pmatrix} cos(\delta) \\ 0 \\ sen(\delta) \end{pmatrix} + V \tag{2.20}$$

e a Figura 2.4 [19] mostra seus efeitos.

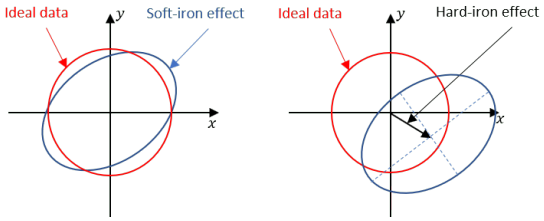


Figura 2.4: Distorções *Hard-iron* e *Soft-iron* visualizadas em duas dimensões.

Logo, as correções exercidas pelos procedimentos de calibração consistem em encontrar os valores que mais se aproximam de W^{-1} e V, de forma que o campo medido seja transformado de um elipsóide para uma esfera centralizada em zero, que é modelada pela Equação 2.21.

$$\mathrm{yi}_{m,t} = W^{-1}(\mathrm{yd}_{m,t} - V) \tag{2.21}$$

Método para calibração

A coleta dos dados deve ser realizada movendo-se o sensor lentamente em todas as direções possíveis. A correção da distorção *hard-iron* de cada eixo se dá ao se calcular o ponto médio das leituras:

$$V = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} \frac{(max(h_x)+min(h_x))}{2} \\ \frac{(max(h_y)+min(h_y))}{2} \\ \frac{(max(h_z)+min(h_z))}{2} \end{pmatrix} \tag{2.22}$$

A correção da distorção *soft-iron* pode ser realizada a partir de uma reescala ortogonal

$$W^{-1} = \begin{pmatrix} \frac{hd_{avg}}{\delta_{hdx}} & 0 & 0 \\ 0 & \frac{hd_{avg}}{\delta_{hdy}} & 0 \\ 0 & 0 & \frac{hd_{avg}}{\delta_{hdz}} \end{pmatrix}, \tag{2.23}$$

em que δ_{hd} é a metade da distância entre os pontos mínimos e máximos de cada eixo:

$$\delta_{hd} = \begin{pmatrix} \frac{(max(h_x) - min(h_x))}{2} \\ \frac{(max(h_y) - min(h_y))}{2} \\ \frac{(max(h_z) - min(h_z))}{2} \end{pmatrix} \tag{2.24}$$

e hd_{avg} é a média entre esses três valores:

$$hd_{avg} = \frac{\delta_{hdx} + \delta_{hdy} + \delta_{hdz}}{3} \tag{2.25}$$

Obtidos os valores de V e W^{-1}, a calibração é realizada pela expressão 2.21.

Este método é computacionalmente simples, oferece bons resultados e pode ser implementado no próprio Arduíno, porém não considera todas as propriedades do elipsóide. Existem métodos mais complexos, porém eles são computacionalmente mais intensivos.

2.3 Parametrizações de rotações

Esta seção tem o objetivo de introduzir parametrizações matemáticas para descrever rotações de objetos num espaço tridimensional. Esse processo de transformação de coordenadas é essencial para diversas áreas do conhecimento como robótica, controle aéreo e computação gráfica. Para a caixa preta, esses conceitos são usados extensivamente na calibração e na fusão de dados sensoriais. Duas parametrizações serão discutidas e comparadas: Ângulos de Euler e Quatérnios.

2.3.1 Ângulos de Euler

De acordo com o matemático Leonhard Euler, qualquer rotação pode ser descrita usando-se três ângulos (ϕ, θ, Ψ), que são respectivamente a rotação sobre os eixos x, y e z do objeto. A Figura 2.5 mostra a rotação de um sistema de coordenadas.

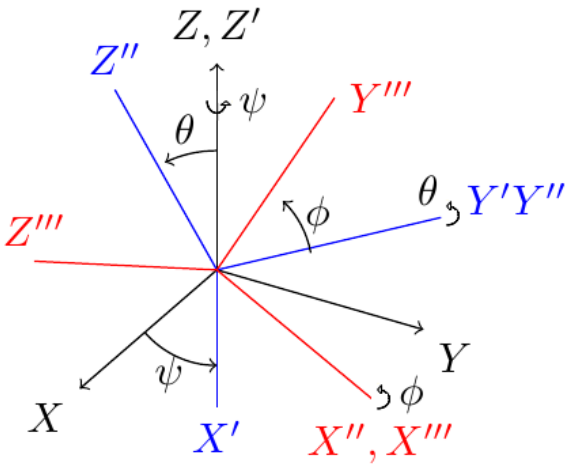


Figura 2.5: Ângulos de Euler.

A rotação de cada eixo pode ser descrita pelas matrizes:

$$
\begin{aligned}
R_x(\phi) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sen(\phi) \\ 0 & sen(\phi) & cos(\phi) \end{pmatrix}, \\
R_y(\theta) &= \begin{pmatrix} cos(\theta) & 0 & sen(\theta) \\ 0 & 1 & 0 \\ -sen(\theta) & 0 & cos(\theta) \end{pmatrix}, \\
\mathrm{R}_z(\psi) &= \begin{pmatrix} cos(\psi) & -sen(\psi) & 0 \\ sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}.
\end{aligned} \tag{2.26}
$$

Na engenharia é comum que as rotações em torno do eixo x, y e z sejam chamadas de *roll*, *pitch* e *yaw*.

Um vetor $V = \begin{pmatrix} vx & vy & vz \end{pmatrix}^T$ pode ser rotacionado nos ângulos (ψ, θ, ϕ) ao ser multiplicado por uma matriz de rotação R dada por:

$$R = (R_x(\phi))(R_y(\theta))(R_y(\psi)). \tag{2.27}$$

Existem doze ordens possíveis para descrever uma determinada rotação: z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y, x-y-z, y-z-x, z-x-y, x-z-y, z-y-x e y-x-z. O que muitas vezes torna essa representação ambígua e cria incompatibilidades em algoritmos.

Gimbal Lock

O principal problema em descrever rotações usando os ângulos de Euler é o *Gimbal Lock*, que é a perda de um dos graus de liberdade quando dois eixos se alinham paralelamente durante a rotação tridimensional.

Isso pode ser observado na rotação x-y-z se assumirmos $\theta = \pi/2$:

$$R = R_x(\phi)R_y(\theta)R_z(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sen(\phi) \\ 0 & sen(\phi) & cos(\phi) \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} cos(\psi) & -sen(\psi) & 0 \\ sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$R = \begin{pmatrix} 0 & 0 & 1 \\ sen(\phi+\psi) & cos(\phi+\psi) & 0 \\ -cos(\phi+\psi) & sen(\phi+\psi) & 0 \end{pmatrix}. \tag{2.28}$$

Qualquer que seja a alteração de ψ ou ϕ, a primeira linha e a última coluna de R não vão se alterar. O que mostra que o sistema perdeu a habilidade de se rotacionar em torno do eixo z, e também a de representar unicamente uma determinada rotação. Por conta desse problema, a representação por ângulos de Euler não é confiável para valores próximos de 90°, sendo comum que métodos de estimativa de orientação utilizem outras formas, como a de quatérnios, que será descrita a seguir.

2.3.2 Quatérnios

Quatérnios são números de 4 dimensões que estendem os números complexos. Foram descobertos pelo matemático William R. Hamilton. Podem ser escritos como

$$q = q_0 + q_1 i + q_2 j + q_3 k, \tag{2.29}$$

em que i,j e k são eixos complexos. que satisfazem a condição

$$i^2 = j^2 = k^2 = ijk = -1. \tag{2.30}$$

Operações algébricas com quatérnios

Assim como nos números complexos, as operações com quatérnios são simplificadas a partir das propriedades de suas unidades imaginárias.

Seja os quatérnios:

$$\begin{aligned} p &= p_0 + p_1 i + p_2 j + p_3 k \\ q &= q_0 + q_1 i + q_2 j + q_3 k \end{aligned} \tag{2.31}$$

A adição entre p e q é:

$$p + q = (q_0 + p_0) + (q_1 + p_1)i + (q_2 + p_2)j + (q_3 + p_3)k \tag{2.32}$$

e a multiplicação:

$$\begin{aligned} p * q = (q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3)+ \\ (q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2)i+ \\ (q_0p_2 + q_2p_0 - q_1p_3 + q_3p_1)j+ \\ (q_0p_3 + q_3p_0 + q_1p_2 - q_2p_1)k \end{aligned} \tag{2.33}$$

Conjugado, norma e quatérnio unitário

Seja o quatérnio $q = q_0 + q_1i + q_2j + q_3k$. O Conjugado de q, denotado por q^* é definido por

$$q^* = q_0 - q_1i - q_2j - q_3k, \tag{2.34}$$

a norma de q, denotada por $|q|$ é definida por

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \tag{2.35}$$

Um quatérnio de norma igual a 1 é definido como unitário. Ao dividir q por sua norma, encontramos o quatérnio unitário u_q chamado de versor de q:

$$u_q = \frac{q}{|q|} \tag{2.36}$$

Descrição de rotação utilizando quatérnios

Quatérnios podem ser utilizados para representar rotações de vetores tridimensionais. Seja um quatérnio unitário q em que

$$q = q_0 + q_1i + q_2j + q_3k = cos(\theta/2) + \hat{u}sen(\theta/2). \tag{2.37}$$

Um vetor $V_1 = \begin{pmatrix} v_{1x} & v_{1y} & v_{1z} \end{pmatrix}^T$ pode ser rotacionado no ângulo θ em torno do eixo de rotação $\hat{u}$ a partir da expressão

$$V_2 = qV_1q^*. \tag{2.38}$$

É possível verificar ao expandirmos a expressão 2.38 que a matriz de rotação M tal que $V_2 = MV_1$ é dada por

$$M = 2\begin{pmatrix} q_0^2 + q_1^2 - 0,5 & q_1q_2 - q_0q_3 & q_0q_2 + q_1q_3 \\ q_0q_3 + q_1q_2 & q_0^2 + q_2^2 - 0,5 & q_2q_3 - q_0q_1 \\ q_1q_3 - q_0q_2 & q_0q_1 + q_2q_3 & q_0^2 + q_3^2 - 0,5 \end{pmatrix}. \text{ [20]} \tag{2.39}$$

Diferente da matriz de rotação da expressão obtida por ângulos de Euler na Equação 2.27, para quatérnios não é necessário obter três matrizes de rotação, e não há necessidade de resolver funções trigonométricas, sendo portanto computacionalmente mais eficiente. Além disso, não existe o risco de *Gimbal Lock* durante interpolações. Por esses motivos, a notação de quatérnios é escolhida para diversos métodos de estimativa de orientação, como o filtro de Madgwick, que foi utilizado neste projeto.

2.4 Fusão de dados sensoriais

O objetivo da fusão de dados sensoriais é combinar a saída de diferentes sensores para construir um único modelo de dados. Seu objetivo é compensar as limitações de cada componente com dados de outros e fornecer uma saída mais acurada e que ofereça mais informações. Neste projeto, a fusão de dados é utilizada nos dados do acelerômetro, giroscópio, magnetômetro para produzir estimativas de orientação.

2.4.1 Limitações dos sensores

As leituras do giroscópio informam a velocidade angular de um objeto e o processo de integração (*dead-reckoning*) pode ser utilizado para estimar sua orientação em graus. Contudo, esse processo é limitado quando faz-se apenas uso do giroscópio pois apesar da calibração diminuir consideravelmente o *drift* de integração ao eliminar o bias constante, esse processo não endereça o ruído gaussiano gerado pela saída do dispositivo, que faz com que durante a integração, o erro cresça como explicado anteriormente.

Além disso, o giroscópio mede apenas a rotação em relação ao seu próprio eixo, sem qualquer conhecimento de sua orientação inicial, sendo impossível portanto estimar a orientação precisa em relação à Terra.

Uma prática comum adotada em unidades de medição inercial é a inclusão do acelerômetro, que por medir uma componente gravitacional, permite a estimativa geográfica dos eixos x e y [21]. Esse fato pode ser ilustrado ao representar a saída de um acelerômetro

parado $G_p = \begin{pmatrix} G_{px} & G_{py} & G_{pz} \end{pmatrix}^T$ normalizada em função de sua rotação de Euler *y-x-z*:

$$\frac{G_p}{|G_p|} = R_y(\theta)R_x(\phi)Rz(\psi)\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -sen(\theta)cos(\phi) \\ sen(\phi) \\ cos(\theta)cos(\phi) \end{pmatrix}. \tag{2.40}$$

Assim, os valores de ϕ e θ desta rotação podem ser obtidos:

$$\phi = arctan(\frac{G_{py}}{\sqrt{G_{px}^2 + G_{py}^2}}), \tag{2.41}$$

$$\theta = arctan(\frac{-G_{px}}{G_{pz}}). \tag{2.42}$$

O giroscópio continua sendo necessário por ser um sensor que não sofre com a interferência gravitacional. O acelerômetro não consegue diferenciar rotações de acelerações sobre o objeto e o processo de separação da componente gravitacional e da aceleração exercida pelo dispositivo não é trivial, sendo dependente de uma estimativa de orientação precisa que não é possível apenas com o acelerômetro, mas pode ser feita pela combinação dos dois sensores utilizando técnicas de fusão de dados.

A combinação entre o acelerômetro e o giroscópio não é suficiente para estimar a orientação completa do dispositivo nos três eixos, como ilustrado na expressão 2.40. Independente da técnica de fusão de dados utilizada para combinar aceleração e giro, ainda ocorrerá um *drift* de integração considerável no eixo z, pois não é fisicamente possível o acelerômetro estimar a orientação em torno desse eixo. A seção 4.3.3 mostra em detalhes essa limitação física, mas é simples verificar esse fato ao se colocar o dispositivo em cima de uma mesa, alinhando seu eixo z com o vetor da aceleração gravitacional. A leitura será $\begin{pmatrix} 0g & 0g & 1g \end{pmatrix}^T$ independente da direção que o sensor apontar. Para endereçar essa limitação, é comum o acréscimo do magnetômetro ao sistema, que é a abordagem realizada pelo módulo MPU-9250 utilizado neste projeto. Com o magnetômetro, o sistema passa a ter o conhecimento do norte magnético, que em conjunto com o acelerômetro e o giroscópio, fornece um sistema de referência completo com nove graus de liberdade que é capaz de estimar corretamente a orientação do sensor de acordo com o referencial da Terra. Para a fusão de dados, utiliza-se filtros como o de Kalman [22] ou o de Madgwick [23], que são abordados em detalhes pelo texto do aluno José Luiz Gomes Nogueira [24], que fez parte deste grupo e apresentará na mesma banca, focando no aspecto de pós-processamento dos dados com seu painel comparativo de métodos de filtragem. Neste trabalho, o pós-processamento ficou restrito à ambas versões do filtro de Madgwick: A que apenas utiliza aceleração e giro, e a que também inclui o magnetômetro.

Capítulo 3

Proposta de dispositivo para aquisição de dados inerciais

Neste capítulo será apresentado o dispositivo de coleta e armazenamento de dados, detalhando seus modos de funcionamento e como operá-lo. É também abordada a escolha dos componentes utilizados, a estrutura do software embarcado, como os dados são organizados em memória e como eles são exportados para o devido pós-processamento.

3.1 Estrutura do Dispositivo

Foi utilizado neste projeto um dispositivo chamado de "Caixa Preta - Versão 1.0", que pode ser classificado como um EDR (Event Data Recorder), pois seu objetivo é coletar dados inerciais capazes de reconstruírem a dinâmica de um veículo.

As principais contribuições deste projeto foram:

- Agora o dispositivo foi concebido numa placa de circuito impresso.
- Aumentou-se a quantidade de memória do sistema, permitindo mais tempo de coleta de dados. Além disso, usa-se a memória EEPROM do Arduíno para armazenar persistentemente dados de configuração e calibração;
- Agora faz-se uso do magnetômetro. Como explicado na seção 2.4.1 seu uso é indispensável para que a orientação seja estimada corretamente em todos os eixos.
- Foram desenvolvidas rotinas de *self-test* e calibração de dispositivos, que podem ser executadas no modo de operação "Calibração de Fábrica".

A figura Figura 3.1 ilustra o equipamento em questão e seus esquemáticos encontram-se no anexo I.

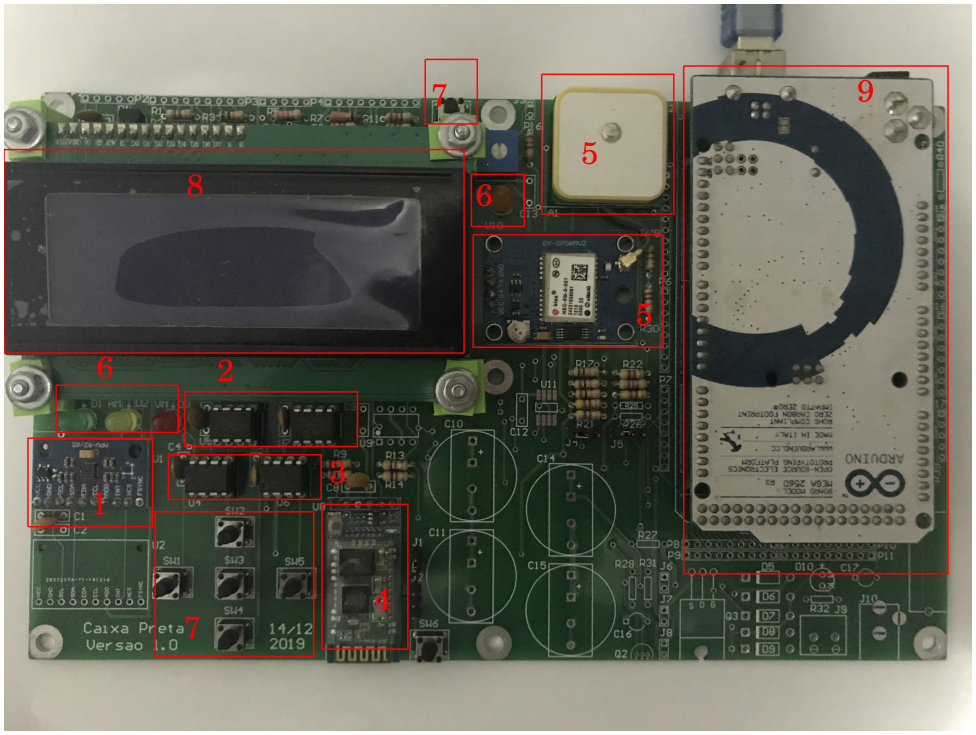


Figura 3.1: Caixa preta.

Os principais componentes do circuito são:

1. **Módulo MPU-9250:** Seu uso foi preferível ao MPU-6050 utilizado em projetos anteriores, pois apesar de também possuir a unidade de medição inercial que coleta aceleração e giro, este módulo inclui integrado o magnetômetro AK8963, oferecendo maior praticidade na configuração e gerência do barramento de dados utilizado para a operação do dispositivo.

2. **Duas memórias SRAM 23LC1024:** Usadas para armazenamento volátil de dados. Cada unidade consegue armazenar 128kB. São controladas por comunicação SPI.

3. **Duas memórias EEPROM 24AA1025:** Para armazenamento persistente dos dados. Cada unidade oferece 128kB de armazenamento. São controladas por comunicação I2C.

4. **Módulo *bluetooth*:** Para auxiliar na comunicação com outros dispositivos.

5. **Módulo de GPS:** Para auxiliar na estimativa da posição e velocidade do veículo.

6. **Quatro LEDs:** Utilizados na depuração da placa.
7. **Seis botões:** Para a seleção de opções e operação dos modos.
8. **LCD:** Para auxiliar na operação e seleção de modos sem a necessidade de um monitor serial.
9. **Arduíno Mega 2560:** Módulo que possui a unidade de processamento ATmega2560. Responsável por coordenar todos os outros componentes citados.

3.2 Organização dos dados

O sistema é separado em três diferentes memórias: Memórias SRAM e EEPROMs externas e a EEPROM interna do ATmega2560. As memórias SRAM, que oferecem rápido acesso e alta vida útil são responsáveis pelo armazenamento volátil imediato de dados que são coletados durante a operação da Caixa Preta. As memórias EEPROM externas são responsáveis pelo armazenamento não volátil de dados que são transferidos da SRAM em eventos específicos como acidentes. A memória EEPROM do ATmega2560 foi utilizada para armazenar os dados coletados na chamada Calibração de Fábrica, que armazena dados sobre o dispositivo que não serão frequentemente alterados pelo usuário.

3.2.1 Uso das memórias SRAM e EEPROM externas

Os dois dispositivos SRAM 23LC1024 de 128 kB foram combinados para trabalharem como uma única memória volátil de 256 kB. A partir do endereço fornecido, o sistema decide qual memória irá acessar. O mesmo foi feito com os dispositivos EEPROM 24AA1025 para compor a memória não volátil do sistema.

Os dados foram organizados de acordo com a tabela Tabela 3.1. A ideia é que na ocorrência de um acidente, até então simulado por um botão, todos os seus dados sejam transferidos para a memória não volátil externa, também de 256 kB. Porém, neste projeto, priorizou-se a escrita dos dados da SRAM na porta serial, que pode ser lida diretamente por ferramentas de manipulação e análise de dados como o Matlab.

Finalidade	**Faixa de memória**	**Quantidade de mensagens**
Dados do MPU-9250	00000_{16} a $37E5F_{16}$	12.720
Configurações	$37E60_{16}$ a $37FFF_{16}$	-
Dados do GPS	38000_{16} a $3FFFF_{16}$	256

Tabela 3.1: Organização da SRAM

Na seção de 00000_{16} a $37E6F_{16}$, são armazenados os dados inerciais de aceleração, giro e campo magnético: $\begin{pmatrix} a_{xi} & a_{yi} & a_{zi} & g_{xi} & g_{yi} & g_{zi} & h_{xi} & h_{yi} & h_{zi} \end{pmatrix}^T$ que ocupam 18 bytes

por leitura. Ao todo, este espaço permite o armazenamento de 12.720 leituras, o que numa taxa de operação de 100Hz possibilita 127,2 segundos de coleta de dados.

A seção de $37E60_{16}$ a $37FFF_{16}$ armazena dados de configuração e de calibração ao ligar. Esses dados, que também devem ser escritos na EEPROM após o acidente, devem informar especificações importantes como se os dispositivos passaram pelos procedimentos *self-test* e calibração de fábrica, a data e hora do acidente, o limiar de disparo que caracteriza acidentes e as posições de memória que indicam o início e o fim da aquisição de dados, o que permite reconstruir a ordem de coleta dos dados. Também são armazenados nessa seção os parâmetros relativos à calibração ao ligar, que é um procedimento realizado no início da operação de coleta de dados que calcula médias sensoriais do giroscópio e acelerômetro e podem ser usados para estimação de *bias* num eventual pós processamento. Além disso, as médias calculadas na calibração ao ligar podem ser comparadas com as obtidas na calibração de fábrica, o que pode sinalizar a necessidade de um novo processo de calibração.

A seção de 38000_{16} a $3FFFF_{16}$ é onde os dados do GPS são armazenados. Cada mensagem possui 128 bytes, então é possível gravar 256 mensagens de dados.

3.2.2 Uso da EEPROM do ATmega2560

O microprocessador ATmega2560 possui uma EEPROM de 4kB que foi utilizada para armazenar dados que o usuário não vai alterar com frequência. A calibração de fábrica é o procedimento que escreve nessa memória vários dados de calibração sensoriais, médias, somatórios e resultados de *self-tests*. O modo de operação 5 - Calibração de Fábrica, guia o usuário na coleta desses dados. O apêndice B mostra o código em C que define a proposta atual de disposição dos dados da memória.

3.3 Configurações dos dispositivos

Esta seção mostra como os dispositivos de coleta de dados foram configurados. O código completo das configurações encontra-se no Apêndice B.

3.3.1 Configuração do MPU-9250

Configuração do IMU

O MPU-9250 possui integrado o MPU-6050, que é uma unidade de medição inercial que possui um acelerômetro e um giroscópio, ambos de 3 eixos e com saída de 16 bits. O acelerômetro pode ser Configurado nas escalas: $\pm 2g$, $\pm 4g$, $\pm 8g$ e $\pm 16g$, enquanto o giroscópio pode ser configuradas as escalas: $\pm 250°/s$, $\pm 500°/s$, $\pm 1000°/s$ e $\pm 2000°/s$.

Foi criada uma rotina de configuração cujo objetivo é ativar o IMU em um estado conhecido que é descrito pela Tabela 3.2. A largura de banda foi selecionada de forma que fosse a menor possível a fim de reduzir a quantidade de ruídos. Uma possível desvantagem para essa abordagem é que por reduzir a quantidade de informação, a resposta do sistema pode demorar mais para se aproximar do valor correto. A comunicação com o dispositivo é realizada via I^2C pelo endereço 68_{16}.

	Largura de banda(Hz)	***Delay***(ms)	**Escala**	**Taxa** (Hz)
Acelerômetro	5,05	32,48	$\pm 2g$	100,00
Giroscópio	5,00	33,48	$\pm 250°/s$	100,00

Tabela 3.2: Configuração do giroscópio e acelerômetro do MPU-9250

Magnetômetro

A configuração do magnetômetro é mais complexa do que a do IMU por ser um dispositivo que apesar de estar no mesmo módulo, é acessado por outro endereço I2C. Existem duas formas de acessar o magnetômetro. A primeira é habilitando o MPU-9250 como mestre e configurando sua interface I2C interna para realizar a comunicação. A segunda é desabilitando o modo mestre e habilitando o modo "Bypass I2C", que como ilustrado na Figura 3.2, permite a comunicação direta de nossa biblioteca para o endereço do magnetômetro, que é $0C_{16}$.

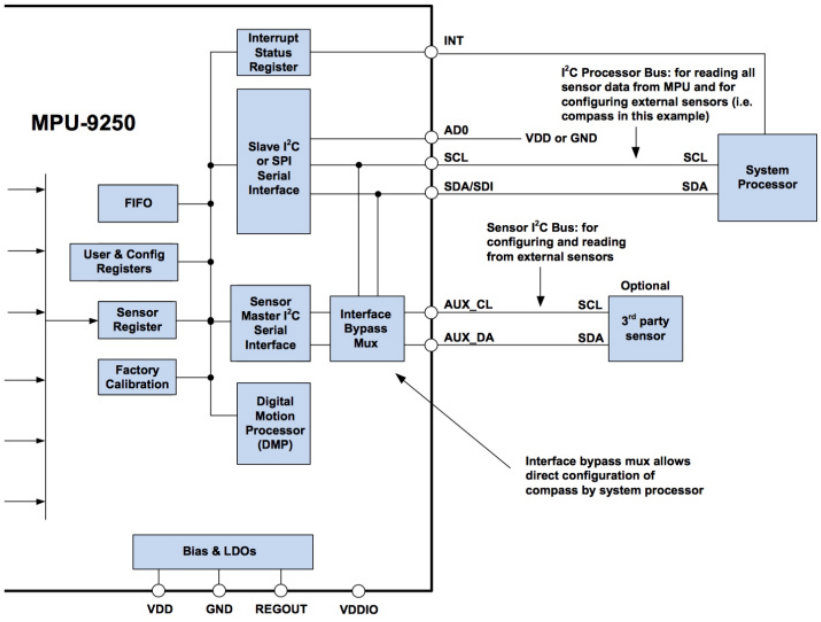


Figura 3.2: Habilitou-se o modo bypass para a comunicação direta com o magnetômetro.

O segundo método foi o escolhido por fornecer mais controle sobre a comunicação I2C.

A Figura 3.3 [25] mostra os modos de operação do magnetômetro :

8.3.6. CNTL1: Control1

Addr	Register name	D7	D6	D5	D4	D3	D2	D1	D0
Write/read register									
0AH	CNTL1	0	0	0	BIT	MODE3	MODE2	MODE1	MODE0
	Reset	0	0	0	0	0	0	0	0

MODE[3:0]: Operation mode setting
"0000": Power-down mode
"0001": Single measurement mode
"0010": Continuous measurement mode 1
"0110": Continuous measurement mode 2
"0100": External trigger measurement mode
"1000": Self-test mode
"1111": Fuse ROM access mode
Other code settings are prohibited

BIT: Output bit setting
"0": 14-bit output
"1": 16-bit output

Figura 3.3: Modos de operação do magnetômetro.

O magnetômetro foi configurado para o modo contínuo 2, que coleta dados na frequência de 100 Hz, equivalente à taxa de coleta escolhida para o IMU. A saída foi configurada para 16 bits. Sua escala é fixa de $\pm 4800 \mu T$ e sua saída deve ser corrigida com o ajuste de sensibilidade dado pela expressão 2.9. Sendo H a leitura do magnetômetro e os valores de ASA são obtidos nos registradores 10h, 11h e 12h que podem ser lidos ao alterar o modo de funcionamento para FUSE-ROM.

3.4 Estrutura do software embarcado

O funcionamento do circuito embarcado na caixa preta é dividido em modo de teste e modo de operação, que são executados em laço após a configuração inicial do circuito, que é a etapa em que são configurados *timers*, protocolos, botões, LCD, comunicação serial, IMU, magnetômetro e GPS. O modo de teste é inicializado quando se liga o dispositivo enquanto o botão central é pressionado como mostra a Figura 3.4. O sistema faz uso de dois *timers* do ATMega2560: O *timer* 1, responsável por uma interrupção de 100Hz que verifica se o LCD precisa de atualização e também verifica as saídas seriais UART0 e UART2, além de coordenar os conversores analógico-digital, que são usados por exemplo no acionamento dos botões. O *timer* 2 coordena toda a operação do LCD numa frequência de 25.000 Hz.

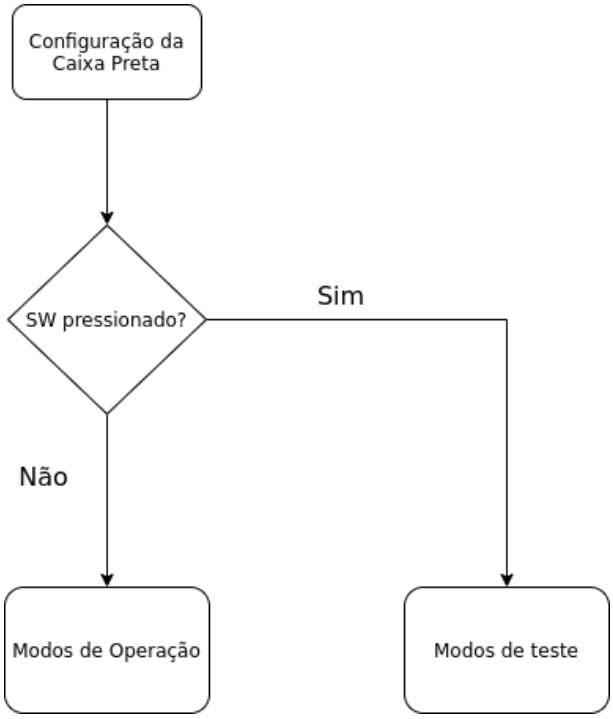


Figura 3.4: Funcionamento da caixa preta.

Os modos podem ser selecionados pelos botões da caixa preta ou por um comando serial da forma t<n> ou o<n> para selecionar o n-ésimo modo de teste ou operação.

3.4.1 Modos de teste

Os modos de teste são os responsáveis pelo teste da integridade dos dispositivos. Todas as funcionalidades foram inicialmente implementadas nesse modo antes de serem usadas de forma definitiva no modo de operação. A biblioteca foi desenvolvida de forma a permitir a criação rápida de novos testes. Como exemplo, a Figura 3.5 mostra a interface do dispositivo ao selecionar e executar o modo de teste que efetua o *self-test* do magnetômetro.

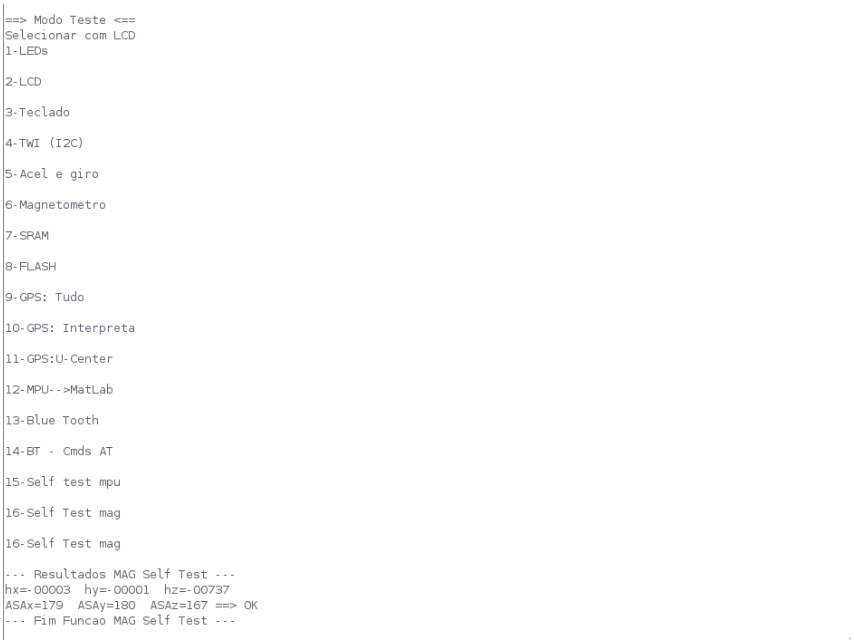


Figura 3.5: Utilizando o modo de teste.

3.4.2 Modos de operação

Os modos de operação são os modos principais de funcionamento da caixa preta.

Calibração de fábrica

A calibração de fábrica é o modo de operação 5, que é acessível ao escrever o comando o5 no monitor serial ou selecionando a quinta opção do modo de operação a partir do LCD. É o modo que popula a memória EEPROM do Arduíno. O usuário receberá instruções no terminal serial para efetuar a devida calibração. Os procedimentos são:

1. Gravação de dados de cabeçalho como local, data e frequência de amostragem;
2. Começar a coletar medidas inerciais do MPU-9250;
3. Gravação de médias, primeiras e últimas medidas dos dados. Essas informações permitem uma análise de erro intrínseco dos sensores. Esses dados também podem ser usados na calibração do giroscópio;
4. É realizado o *self-test* do IMU e do magnetômetro;
5. É sinalizado na EEPROM que esses valores foram calculados e também são gravados os valores obtidos como resposta;

6. É executado o procedimento de calibração de acelerômetro por seis pontos descritos na Seção 2.2.2 e os seis dados de leitura obtidos são salvos na EEPROM. O usuário é instruído a apontar cada eixo para cima e para baixo e pressionar o botão da caixa preta para realizar as leituras. Cada leitura, na prática foi implementada como uma média de 30 leituras com o fim de reduzir eventuais picos. Como os dados foram obtidos após botões serem pressionados, também foi necessário ignorar as primeiras 160 leituras, pois existe uma pequena oscilação nos sensores ocasionada durante o aperto do botão, o que pode interferir com o cálculo dos parâmetros de correção;
7. É executado o procedimento de calibração de magnetômetro descrito na seção 2.2.3. O usuário é instruído a mover o magnetômetro lentamente descrevendo uma esfera. Os valores gravados na EEPROM são os seis valores mínimos e máximos que são usados nas Expressões 2.22 e 2.24.

Calibrações complexas com o Matlab

As calibrações sensoriais executadas pela calibração de fábrica são simples pois foram implementadas no próprio Arduíno, que possui limitações de processamento e memória. Com isso, foram criados os modos de operação 7 e 8 que coletam dados usados na calibração do magnetômetro e acelerômetro respectivamente, os enviando na porta serial, para que sejam recebidos e manipulados pelo Matlab em procedimentos de calibração mais complexos. Um dos métodos de calibração de acelerômetro implementados foi o dos mínimos quadrados descrito na seção 2.2.2. Já para o magnetômetro foi implementado uma calibração que utiliza a função "magcal" da toolbox de fusão de dados do Matlab, que utiliza uma modelagem mais complexa que considera a natureza elíptica da saída descalibrada, a transformando numa esfera de acordo com o descrito na seção 2.2.3. O apêndice A mostra os algoritmos de Matlab descritos.

Operação de coleta de dados

A coleta de dados por todos os sensores: IMU, magnetômetro e GPS é executada pelo modo de operação 1, acessado no monitor serial pelo comando o1 ou selecionando o primeiro modo de operação pelo LCD. O primeiro procedimento realizado é o de preparação, que realiza a calibração ao ligar, coletando os dados descritos na seção 3.2.1 e armazenando na seção de configuração da memória SRAM.

A aquisição de dados se inicia ao pressionar o botão central ou enviando o comando S para a porta serial. É então realizada a coleta de dados sensoriais até que a memória seja totalmente preenchida ou até que a aquisição seja interrompida pelo usuário..

Após a aquisição dos dados, imprime-se os dados da memória SRAM na saída serial separados com os delimitadores descritos na Tabela 3.3.

Delimitador	**Descrição**
#[m ... m]#	Dados de aquisição do MPU-9250
#[g ... g]#	Dados do GPS
#[l ... l]#	Dados com os resultados da Calibração ao Ligar
#[f ... f]#	Dados da calibração de fábrica

Tabela 3.3: Formato do arquivo gerado pela caixa-preta

Capítulo 4

Análise dos dados

Este capítulo mostra toda a operação da plataforma desenvolvida. São abordados os procedimentos de *self-test*, calibração, aquisição e pós-processamento dos dados em duas simulações de rotações controladas.

4.1 Aplicação do self-test

O primeiro procedimento a ser executado é o *self-test* pois é ele que garante a integridade dos sensores. Para a unidade de medição inercial, foi realizado o procedimento descrito na seção 2.1.4. cuja saída é ilustrada pela Figura 4.1, que indica que o giroscópio e o acelerômetro atenderam as condições da Tabela 2.1.

```
15-Self test mpu

medias sem self test
        +00335 +00553 +18677 +00063 +00068 +00042

medias com self test
        +07397 +06986 +29136 +19720 +22323 +24987

Self test response
        +07062 +06433 +10459 +19657 +22255 +24945

Factory self test code (nos registradores)
        +101 +092 -124 -051 -040 -029

Self Test value (st_opt)
        +7086.60 +6479.56 +9647.18 +19946.22 +22253.36 +24827.37

Porcentagens self_test_response / self_test_value
        +0.9965 +0.9928 +1.0841 +0.9854 +1.0000 +1.0047

MPU Self Test Passou
```

Figura 4.1: *Self-Test do giroscópio e acelerômetro.*

Em seguida foi realizado o *self-test* do magnetômetro cujo precedimento foi explicado na seção 2.1.4. A saída é ilustrada pela Figura 4.2, que indica que o magnetômetro passou.

```
16-Self Test mag

--- Resultados MAG Self Test ---
hx=+00003  hy=-00001  hz=-00743
ASAx=179  ASAy=180  ASAz=167 ==> OK
--- Fim Funcao MAG Self Test ---


MAG Self Test Passou
```

Figura 4.2: *Self-Test do magnetômetro.*

Os testes de todos os sensores presentes no MPU-9250 passaram, o que indica que eles são adequados para o uso e pode-se prosseguir para as próximas etapas.

4.2 Obtenção de parâmetros de calibração

Apesar dos resultados obtidos na seção 4.1 indicarem que os sensores estão funcionando corretamente, ainda existem erros associados às medidas obtidas que serão acumulados durante os processos de estimativa de orientação caso não sejam corrigidos.

Obtenção de parâmetros de calibração do giroscópio

Foram coletadas mil leituras com o giroscópio parado em cima de uma mesa. A Figura 4.3 mostra essas medidas puras. O esperado era que as medidas dos três eixos fossem centralizadas em $y = 0$, porém nota-se um bias δ_ω, que pode ser aproximado como a média das leituras obtidas em cada eixo:

$$\delta_\omega = \begin{pmatrix} \bar{gx} \\ \bar{gy} \\ \bar{gz} \end{pmatrix} = \begin{pmatrix} 66,583 \\ 66,710 \\ 44,185 \end{pmatrix} \tag{4.1}$$

A Figura 4.4 mostra a saída do giroscópio após $\delta\omega$ ser subtraído de suas leituras.

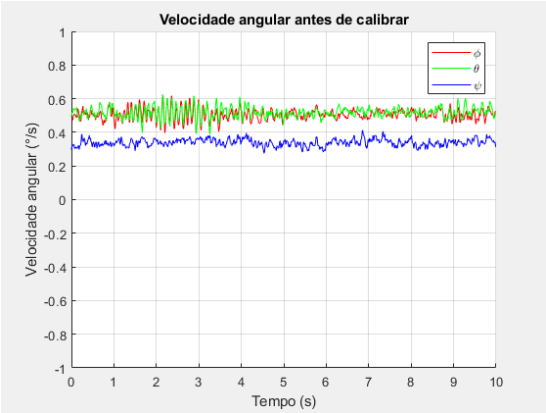


Figura 4.3: Giroscópio antes de calibrar

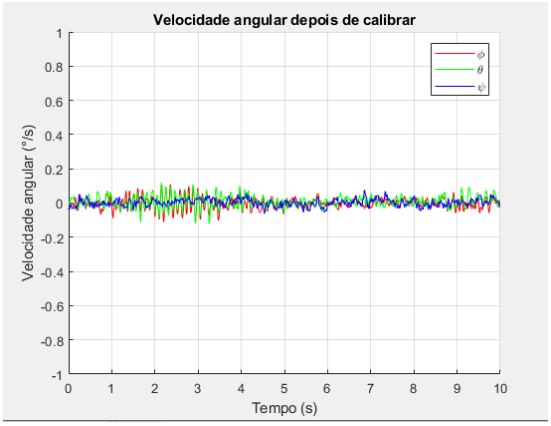


Figura 4.4: Giroscópio depois de calibrar

Obtenção de parâmetros de calibração do acelerômetro

Assim como para o giroscópio, foram coletadas mil medidas com o dispositivo parado em cima de uma mesa. A Figura 4.5 mostra os dados obtidos. Nota-se que há um erro considerável, pois o módulo $a_x^2 + a_y^2 + a_z^2$ esperado para um sensor parado é 1, e nesta saída, o valor é visivelmente maior, pois o eixo z sozinho já ultrapassa este valor e há também um *offset* visível nos eixos x e y.

Foram realizados os dois procedimentos de calibração discutidos na seção 2.2.2. O primeiro procedimento foi a calibração por seis pontos, que gerou os parâmetros de correção de *offset* e escala

$$\begin{aligned} \delta_{a1} &= \begin{pmatrix} 179,26 \\ 321,78 \\ 2148,34 \end{pmatrix}, \\ m_{a1} &= \begin{pmatrix} 0,000061 \\ 0,000061 \\ 0,000061 \end{pmatrix}, \end{aligned} \tag{4.2}$$

enquanto o segundo procedimento foi o dos mínimos quadrados que gerou os parâmetros

$$\begin{aligned} \delta_{a2} &= \begin{pmatrix} 178,883319 \\ 313,812427 \\ 2184,195236 \end{pmatrix}, \\ m_{a2} &= \begin{pmatrix} 0,000061 \\ 0,000061 \\ 0,000061 \end{pmatrix}. \end{aligned} \tag{4.3}$$

Ao aplicar os parâmetros δ_{a1}, m_{a1}, δ_{a2} e m_{a2} na Equação 2.11, as saídas corrigidas são

as das figuras 4.6 e 4.7. Ambos os procedimentos melhoraram as leituras e não houve diferença significativa entre eles, apesar da calibração em seis pontos ser menos precisa.

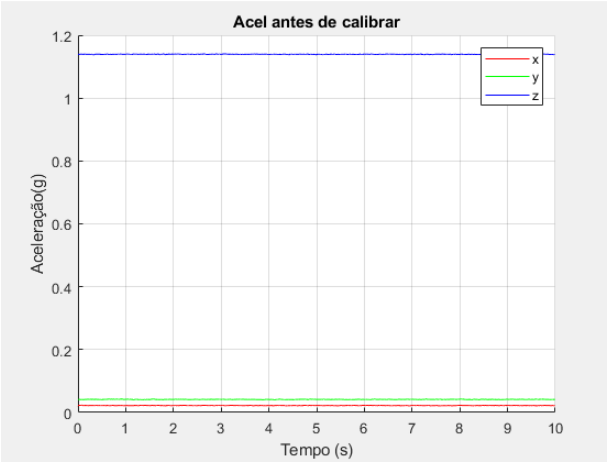


Figura 4.5: *Dados do acelerômetro antes da calibração.*

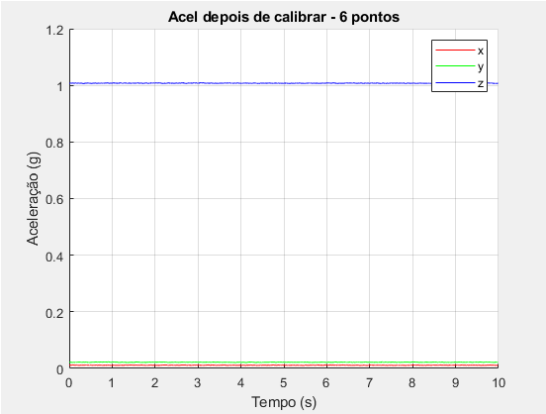


Figura 4.6: Saída do acelerômetro quando aplicada a calibração em seis pontos

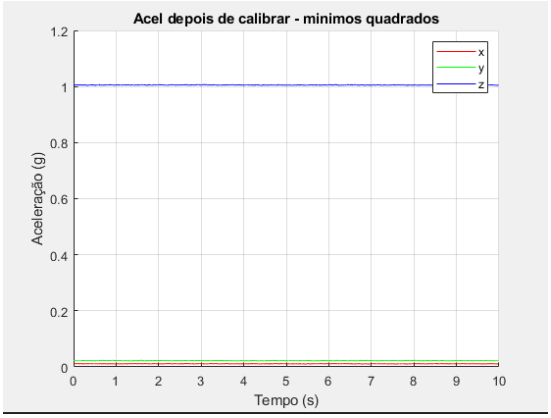


Figura 4.7: Saída do acelerômetro quando a aplicada a calibração por mínimos quadrados

Obtenção de parâmetros de calibração do magnetômetro

Para o magnetômetro, os dados foram coletados ao rotacionar a caixa preta lentamente em diversas orientações com o intuito de descrever uma esfera centralizada em (0,0,0). Ou seja, os pares (h_x, h_y), (h_x, h_z) e (h_y, h_z) precisam descrever círculos centralizados em $(0, 0)$. Porém a Figura 4.8 mostra que esses pares descrevem formas elípticas deslocadas do centro, o que mostra os efeitos das distorções *hard-iron* e *soft-iron* discutidas. Foi aplicado então o método de calibração descrito na seção 2.2.3. Os valores obtidos de *offset* e escala foram

$$\delta_h = \begin{pmatrix} 97,1367187 \\ 114,296875 \\ -255,820312 \end{pmatrix}, \qquad m_h = \begin{pmatrix} 1,0421361643 & 0 & 0 \\ 0 & 0,9165464165 & 0 \\ 0 & 0 & 1,0533187009 \end{pmatrix}, \tag{4.4}$$

que se aplicados na Equação 2.21, convertendo devidamente seus valores para μT, obtém-se o gráfico da figura 4.9, cujos valores condizem com a realidade, pois o módulo do campo magnético da América do Sul é de fato em torno de 30 μT. [26]

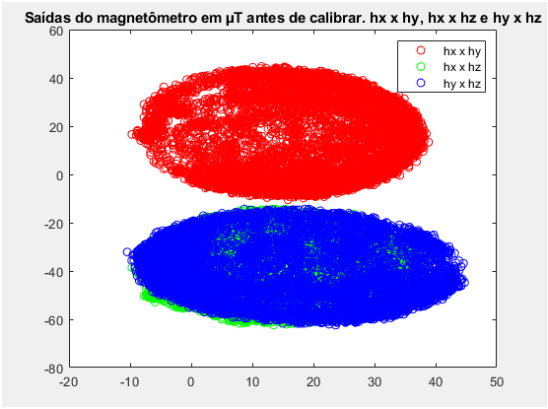


Figura 4.8: Saída do magnetômetro descalibrado

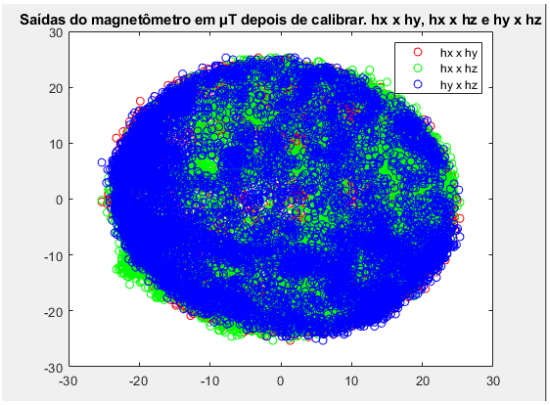


Figura 4.9: Saída do magnetômetro calibrado

4.3 Simulando rotações

Nesta seção são realizadas duas simulações de rotações utilizando o filtro de Madgwick, comparando as orientações obtidas com e sem o uso do magnetômetro. A primeira simulação visa utilizar o dispositivo para estimar a orientação de acordo com o referencial da Terra. A segunda visa forçar o aumento do *bias* do giroscópio para demonstrar a necessidade do magnetômetro para a estimação do eixo z. Todos os dados sensoriais foram corrigidos com a metodologia discutida na seção 4.2 antes de serem processados.

4.3.1 Alinhando os eixos dos sensores

O filtro de Madgwick exige que os sensores estejam alinhados, o que não é o caso do MPU-9250 como mostra a Figura 4.10.

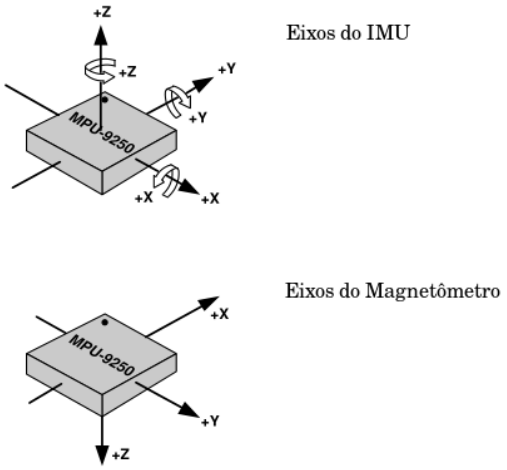


Figura 4.10: *Eixos do IMU e magnetômetro.*

Sendo

$$\mathrm{M_{input}} = \begin{pmatrix} g_x^m \\ g_y^m \\ g_z^m \\ a_x^m \\ a_y^m \\ a_z^m \\ h_x^m \\ h_y^m \\ h_z^m \end{pmatrix} \tag{4.5}$$

a entrada do filtro de Madgwick, a correção de eixos utilizada foi:

- $a_x^m = -ax$;
- $a_y^m = -ay$;
- $g_x^m = -gx$;
- $g_y^m = -gy$;
- $h_x^m = -hy$;
- $h_y^m = -hx$;
- $h_z^m = -hz$.

Assim, ao utilizar o filtro completo, a orientação zero é a que o eixo $-x$ do IMU aponta para o norte, o eixo $-z$ aponta para baixo e o eixo y aponta para leste, ou seja, passa a ser possível analisar os eixos ($-x$, y, $-z$) do IMU como um sistema de coordenadas NED.

4.3.2 Simulação 1 - estimando a orientação geográfica

Esta é uma simulação de 120 segundos, que consiste em:

1. Apontar a caixa preta para oeste por 20 segundos;
2. Apontar a caixa preta para norte por 20 segundos;
3. Apontar a caixa preta para leste por 20 segundos;
4. Apontar a caixa preta para oeste por 20 segundos;
5. Apontar a caixa preta para norte por 20 segundos;
6. Apontar a caixa preta para oeste por 20 segundos.

Seu objetivo é verificar a precisão do dispositivo num intervalo de tempo longo e se é possível estimar uma orientação geográfica.

Utilizando apenas o IMU

A saída da versão IMU do filtro, que usa apenas acelerômetro e giroscópio, é dada pela Figura 4.11. Percebe-se que o gráfico não mostra diretamente a orientação geográfica. Sua curva descreve toda a rotação do eixo z em termos da orientação inicial do sensor. Isso era esperado pois como explicado na seção 2.4.1, o giroscópio mede apenas a velocidade angular em torno de seu próprio eixo sem obter qualquer informação geográfica, enquanto o acelerômetro, apesar de conseguir estimar orientações em torno dos eixos x e y, não detecta orientações em torno do eixo gravitacional. Apesar dessa limitação, ainda é possível reconstruir a rotação realizada desde que a orientação inicial seja conhecida. Isso ocorre pois não houveram movimentos bruscos e não houve tempo para o acúmulo de muitos erros no giroscópio, que foi o responsável por estimar a rotação em torno de z, descrita pela curva azul.

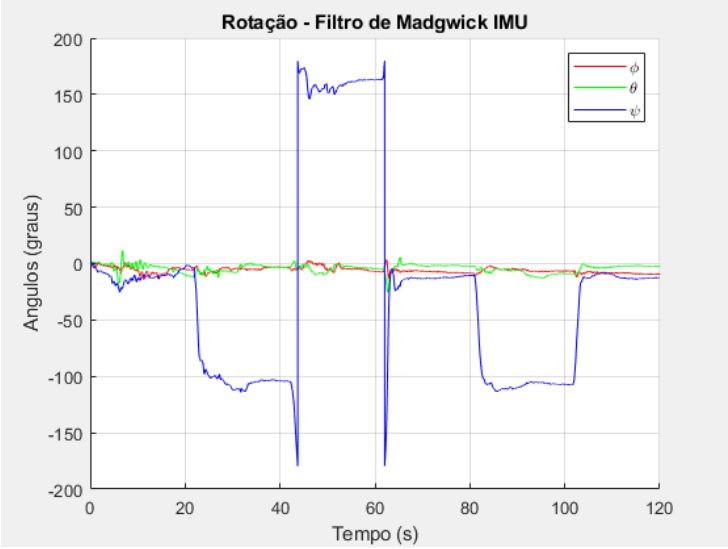


Figura 4.11: *Simulação 1 apenas com o acelerômetro e giroscópio.*

Acrescentando o magnetômetro

A saída utilizando o filtro completo, que inclui o magnetômetro é descrita pela Figura 4.12. Nota-se que em torno de 10 segundos o filtro rotaciona em torno do eixo z em aproximadamente 90° no sentido anti-horário, o que pela regra da mão direita indica que convergiu para a direção inicial correta que é oeste. Contudo, nota-se um maior ruído nos eixos x e y que vem de interferências no sinal do magnetômetro, que é notoriamente mais ruidoso do que os outros sensores utilizados.

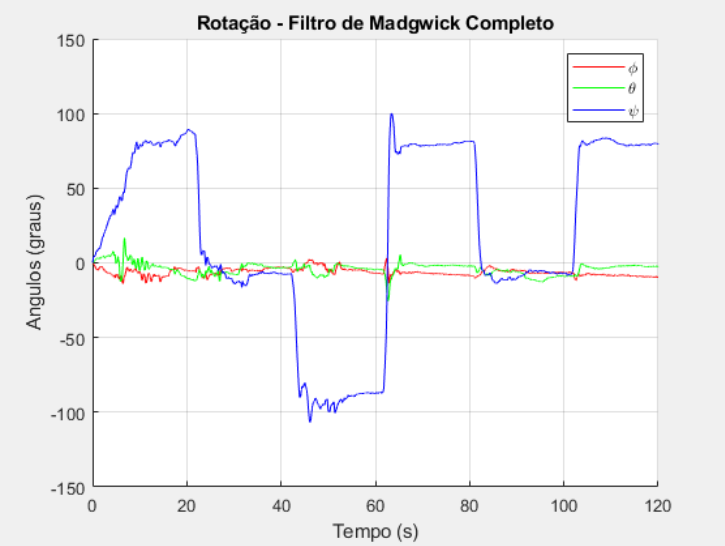


Figura 4.12: *Simulação 1 utilizando o filtro completo.*

4.3.3 Simulação 2 - forçando o bias do giroscópio

Esta é uma simulação de 90 segundos que consiste em:

1. Rotacionar o dispositivo rápida e desordenadamente por 20 segundos;
2. Apontar o dispositivo para norte por 20 segundos;
3. Rotacionar novamente por 20 segundos;
4. Apontar o dispositivo para norte por 20 segundos;
5. rotacionar por 5 segundos;
6. Apontar para norte por mais 5 segundos.

Seu objetivo é analisar a acurácia do sistema em movimentos bruscos que aceleram o acúmulo de *drift* do giroscópio.

Utilizando apenas o IMU

A orientação gerada pelo filtro sem o magnetômetro é dada pela Figura 4.13. Nota-se que cada vez que os movimentos bruscos ocorrem, o eixo z se afasta mais de seu valor inicial, mesmo que o dispositivo seja sempre posicionado no mesmo lugar. Isso não ocorre nos eixos x e y, que são posicionados corretamente pois o acelerômetro possui a gravidade como referência, porém como o eixo z é estimado apenas com a integração da velocidade angular do giroscópio, o erro sensorial acumulado pelos movimentos bruscos torna o uso de um sistema sem magnetômetro inviável para estimar orientações geográficas.

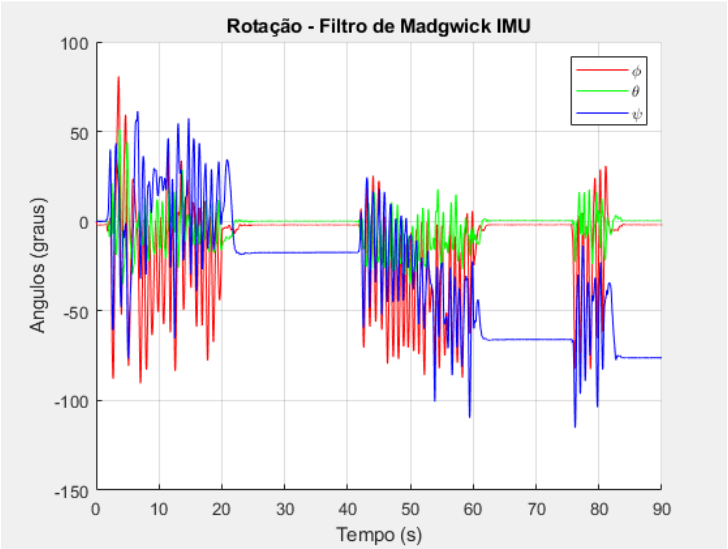


Figura 4.13: *Simulação 2 utilizando apenas o IMU.*

Acrescentando o magnetômetro

A orientação gerada pelo filtro de Madgwick completo é dada pela Figura 4.14. Verica-se que, apesar da rotação em torno do eixo z ser mais ruidosa devido à natureza do magnetômetro, o *drift* que existia foi corrigido mesmo após a ocorrência de movimentações bruscas, o que torna esta abordagem mais robusta para sistemas em que ocorrem movimentos imprevisíveis, o que é o caso de um veículo automotivo.

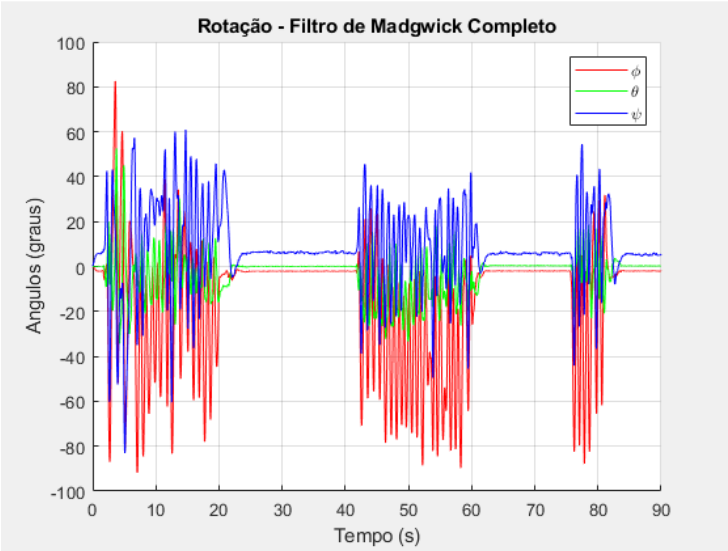


Figura 4.14: *Simulação 2 utilizando o filtro completo.*

Capítulo 5

Conclusão

Este projeto apresentou uma nova versão de um sistema de obtenção de dados inerciais capaz de auxiliar o trabalho pericial de trânsito. Buscou-se fornecer uma ampla biblioteca que cobre toda a parte de calibração, testes, coleta e exportação de dados sensoriais, que em conjunto com o painel comparativo de filtros criado pelo aluno José Luiz Gomes Nogueira [24], que também fez parte do projeto e apresentará na mesma banca, tem-se um sistema completo capaz de coletar e pós-processar dados. Neste texto, os focos principais foram garantir a integridade dos dados com mecanismos detalhados de *self-test* e calibração, além do acréscimo do magnetômetro ao sistema, analisando a viabilidade de seu uso. Os resultados obtidos mostraram que a fusão de dados que inclui o magnetômetro não só estima corretamente a orientação geográfica do dispositivo, como é a única forma de coletar dados corretos em todos os eixos, já que um sistema apenas com giroscópio e acelerômetro falham em descrever a orientação em torno do eixo z, especialmente em sistemas com movimentações bruscas imprevisíveis.

5.1 Perspectivas futuras

Como contribuições futuras, propõe-se a estimativa da posição, que com a biblioteca implementada até então, só pode ser feita com o sistema de posicionamento global (GPS) ou integrando duas vezes os dados do acelerômetro após a remoção de sua componente gravitacional. O problema da primeira forma é que a coleta de dados pelo GPS é lenta, não sendo o suficiente para caracterizar um acidente de trânsito. O problema da segunda forma até então era que a dupla integração dos dados do acelerômetro causam um grande *drift* de integração, tornando rapidamente a estimativa de posição inviável. Grande parte desse erro é causado pela componente gravitacional que não teria sido completamente removida devido às limitações que a ausência do magnetômetro trazia ao sistema, criando um erro que aumenta com o tempo devido à integração. Com a melhoria que o magnetômetro

trouxe no processo de fusão de sensores, agora será possível estimar a orientação do eixo gravitacional de forma mais precisa. Porém, apenas remover a componente gravitacional do sinal da aceleração não remove o ruído do acelerômetro, que também cresce devido à integração. Sugere-se então uma prática adotada na indústria que é a utilização de um filtro de Kalman estendido que inclui como estados os dados recebidos do GPS [27], que não sofrem com *drift* em grandes intervalos de tempo, permitindo então a melhoria da estimação se combinados com os dados do MPU-9250.

Referências

[1] *World health organization. road traffic injuries. 2020.* https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, acesso em 15 jan. 2021. 1

[2] *PORTAL DO TRÂNSITO: em 2020, 80 pessoas morreram por dia em consequência de acidente de trânsito no país.* https://www.portaldotransito.com.br/noticias/em-2020-80-pessoas-morreram-por-dia-em-consequencia-de-acidente-de-transito-no-pais/, acesso em 15 jan. 2021. 1

[3] *DETRAN. código de trânsito brasileiro - ctb.* https://detran.to.gov.br/legislacao/outros/codigo-de-transito-brasileiro-ctb/, acesso em 15 jan. 2021. 1

[4] VILLAMARIM, Tuany Caldas: *Migakgas. como os peritos avaliam acidentes de trânsito.* https://www.migalhas.com.br/depeso/288084/como-os-peritos-avaliam-um-acidente-de-transito, acesso em 15 jan. 2021. 1

[5] LIMA, Vinícius De Oliveira: *Proposta de plataforma inercial para auxiliar na perícia de acidentes de trânsito.* 2016. 1, 2

[6] *LARSON, aaron. what is an automobile black box. 2018.* https://www.expertlaw.com/library/accidents/auto_black_boxes.html, acesso em 15 jan. 2021. 1

[7] RAMOS, Hudson Pereira; LUCENA, Vanessa Oliveira: *Proposta de plataforma inercial e simulador 3d para periciar acidentes de trânsito. trabalho de conclusão de curso (bacharelado em engenharia mecatrônica), Universidade de Brasília, Brasília.* 2017. 2

[8] LOPES, Gabriela da Silva: *Caixa preta para veículos automotivos, Universidade de Brasília, Brasília.* 2018. 2

[9] PARTNERSHIP, PRIME Faraday: *An introduction to MEMS (micro-electromechanical systems) - Loughborough University.* 5

[10] ZUMBAHLEN, Hank: *Linear Circuit Design Handbook*, volume 1993. INC., AnalogDevices, 1986. 5, 6

[11] INC., Invensense: *MPU-9250 Product Specification.* 2016. 6, 7

[12] PARTNERSHIP, PRIME Farada: *Recognition of elementary upper limb movements in nomadic environment - University of Southampton.* 6

[13] *Magnetometer basics for mobile phone applications 2012.* https://www.electronicproducts.com/magnetometer-basics-for-mobile-phone-applications/, acesso em 21 maio 2021. 7

[14] INC., Invensense: *MPU-9250 accelerometer, gyroscope and compass self-test implementation.* 2013. 8

[15] *Imu specifications.* https://www.vectornav.com/resources/imu-specifications, acesso em 20 mai. 2021. 11

[16] *Accelerometer calibration iii: Improving accuracy with least-squares and the gauss-newton method.* https://chionophilous.wordpress.com/2011/08/26/accelerometer-calibration-iii-improving-accuracy-with-least-squares-and-the-gauss-newton-method/, acesso em 17 mai. 2021. 12, 13

[17] *Solve nonlinear least-squares (nonlinear data-fitting) problems.* https://www.mathworks.com/help/optim/ug/lsqnonlin.html, acesso em 20 mai. 2021. 14

[18] OZYAGCILAR, Talat: *Pcalibrating an ecompass in the presence of hard- and soft-iron interference.* 2015. 14

[19] *Magnetometer calibration coefficients matlab.* https://www.mathworks.com/help/nav/ref/magcal.html, acesso em 15 jan. 2021. 15

[20] BEN-ARI, Moti: *A tutorial on euler angles and quaternions - weizmann institute of science.* 2017. 20

[21] INC., Invensense: *Tilt sensing using a three-axis accelerometer.* 2013. 20

[22] *Kalman filter tutorial.* https://www.kalmanfilter.net/, acesso em 20 mai. 2021. 21

[23] MADGWICK, Sebastian O.H.: *An efficient orientation filter for inertial and inertial/magnetic sensor arrays.* 2010. 21

[24] NOGUEIRA, José Luiz Gomes: *Caixa preta para carros: Comparação de métodos de estimativa para inclinação usando acelerômetro, giroscópio e magnetômetro, Universidade de Brasília, Brasília.* 2021. 21, 42

[25] INC., Invensense: *MPU-9250 Register Map and Descriptions Revision 1.6.* 2015. 27

[26] *Earth's magnetic field.* https://web.ua.es/docivis/magnet/earths_magnetic_field2.html, acesso em 20 mai. 2021. 36

[27] Manon Kok, Jeroen D. Hol, Thomas B. Schon: *Using inertial sensors for position and orientation estimation, foundations and trends in signal processing: Vol. 11: No. 1-2, pp 1-153.* 2017. 43

Apêndice A

Scripts de Matlab para manipulação de dados

Calibração da aceleração por quadrados mínimos

```
1
2  % Calibra  o do acelerometro pelo metodo dos quadrados m nimos
3  x0 = [0 0 0 1 1 1];
4
5  fh=@(x)leastSquareFun(x,samples);
6  [x,resnorm] = lsqnonlin(fh,x0);
7
8  accel_offset(1)=x(1);
9  accel_offset(2)=x(2);
10 accel_offset(3)=x(3);
11
12 accel_scale(1)=1/x(4);
13 accel_scale(2)=1/x(5);
14 accel_scale(3)=1/x(6);
15
16
17 function F = leastSquareFun(x,samples)
18 k = 1:9;
19 F = ((samples(k,1)-x(1))/x(4)).^2 + ((samples(k,2)-x(2))/x(5)).^2 + ((
       samples(k,3)-x(3))/x(6)).^2 -1;
20 end
```

Calibração do acelerômetro com o método dos seis pontos

```
1
2  % Calibra  o do acelerometro pelo metodo dos seis pontos
3
4  azplus  = [ mean(ax(1:qtd_amostras)) mean(ay(1:qtd_amostras)) mean(az(1:
       qtd_amostras))];
5  ayplus  = [ mean(ax(qtd_amostras+1:2*qtd_amostras)) mean(ay(qtd_amostras
       +1:2*qtd_amostras)) mean(az(qtd_amostras+1:2*qtd_amostras))];
6  ayminus = [ mean(ax(2*qtd_amostras+1:3*qtd_amostras)) mean(ay(2*
       qtd_amostras+1:3*qtd_amostras)) mean(az(2*qtd_amostras+1:3*qtd_amostras
       ))];
7  axminus = [ mean(ax(3*qtd_amostras+1:4*qtd_amostras)) mean(ay(3*
       qtd_amostras+1:4*qtd_amostras)) mean(az(3*qtd_amostras+1:4*qtd_amostras
       ))];
8  axplus  = [ mean(ax(4*qtd_amostras+1:5*qtd_amostras)) mean(ay(4*
       qtd_amostras+1:5*qtd_amostras)) mean(az(4*qtd_amostras+1:5*qtd_amostras
       ))];
9  azminus = [ mean(ax(5*qtd_amostras+1:6*qtd_amostras)) mean(ay(5*
       qtd_amostras+1:6*qtd_amostras)) mean(az(5*qtd_amostras+1:6*qtd_amostras
       ))];
10
11 samples=[axplus;ayplus;ayminus;azminus;azplus;axminus];
12
13
14
15 %dados iniciais
16 accel_offset=[0 0 0];
17 accel_scale=[1 1 1];
18 GRAVITY=9.80665;
19
20 %Realiza a calibra  o
21 % [accel_offset, accel_scale] = calib_accel_6_points(axplus, axminus,
       ayplus, ayminus, azplus, azminus);
22 [accel_offset, accel_scale] = calib_accel_6_points(axplus, axminus, ayplus
       , ayminus, azplus, azminus);
23
24 % Escreve no arquivo os dados de calibra  o
```

```
fileID  = fopen([directory '\calib_acel.txt'],'w');
fprintf(fileID , '%f\n', accel_offset(1));
fprintf(fileID , '%f\n', accel_offset(2));
fprintf(fileID , '%f\n', accel_offset(3));
fprintf(fileID , '%f\n', accel_scale(1));
fprintf(fileID , '%f\n', accel_scale(2));
fprintf(fileID , '%f\n', accel_scale(3));
fclose(fileID);


function [accel_offset, accel_scale] = calib_accel_6_points(axplus,axminus
    ,ayplus,ayminus,azplus,azminus)
    accel_offset(1)=(axplus(1)+axminus(1))/2;
    accel_offset(2)=(ayplus(2)+ayminus(2))/2;
    accel_offset(3)=(azplus(3)+azminus(3))/2;

    accel_scale(1)=1/((axplus(1)-axminus(1))/2);
    accel_scale(2)=1/((ayplus(2)-ayminus(2))/2);
    accel_scale(3)=1/((azplus(3)-azminus(3))/2);
end
```

Calibração do giroscópio

```

% Realiza a calibra  o do Giroscpio
% Utiliza o teste 12 para coletar os dados
% Apenas calcula medias e escreve num arquivo
% Escreve no arquivo calib_giro.txt
% O arquivo mil_leituras.m mostra como ler e converter os dados de
    calibragem

% limpa tela e variaveis
close all;
clear;
clc;

% diretrio do script
```

```
14 if(~isdeployed)
15   cd(fileparts(which(mfilename)));
16 end
17 directory = pwd;
18
19 fprintf(1,'Teste 12\n');
20 fprintf(1,'O Matlab recebe dados da da Caixa Preta.\n');
21 fprintf(1,'Deixe o sensor parado por alguns segundos e a calibra  o sera
       feita automaticamente.\n');
22 fprintf(1,'Em caso de erro, a porta serial pode estar travada.\n');
23 fprintf(1,'Neste caso use "fclose(instrfind)" para fechar porta serial.\n'
     );
24
25
26 %Par metros
27 fa=100;          %Frequ ncia de amostragem em Hz
28 ta=1/fa;         %Intervalo entre amostras (periodo)
29
30 %Escalas
31 esc_ac=2;
32 esc_giro=250;
33
34 %Amostras
35 qtd_amostras = 1000;
36
37 %Abre porta serial (n  o pode estar aberta na IDE do arduino
38 sid=serial('COM6','Baudrate',115200);
39 % sid=serial('/dev/ttyACM0','Baudrate',115200);
40
41 fopen(sid);
42 if (sid==-1)
43     fprintf(1,'Nao abriu COM6.\n');
44 %     fprintf(1,'Nao abriu /dev/ttyACM0.\n');
45     return;
46 end
47
48 x1=0;
```

```
49 x2=0;
50
51 gx = zeros(1000,1);
52 gy = gx;
53 gz = gx;
54
55 pause(2);
56
57 %acessa o teste 12 mandando o c    d i g o  pelo serial
58 fprintf(sid,'t12\r\n');
59
60 while x1~='[' || x2~='m'
61     x1=x2;
62     x2=fread(sid,1);
63 end
64
65 fprintf(1,'\nIniciando  r e c e p   o de dados...\n');
66 pause(1);
67
68 t=fread(sid,1);
69 t=fscanf(sid,'%d');
70 t=fscanf(sid,'%f');
71
72 while t~=55555
73     t=fscanf(sid,'%d');
74 end
75
76 uax=0;  % ltimo  ax
77 uay=0;  % ltimo  ay
78 uaz=0;  % ltimo  az
79
80 ugx=0;
81 ugy=0;
82 ugz=0;
83
84 uhx=0;
85 uhy=0;
```

```
uhz=0;
ix=1;

ch = 0;
% while uax~=22222 || uay~=22222

cnt = 10;
for i = 1:qtd_amostras

    if mod(i,100) == 0
       fprintf(1,'%d\n',cnt);
        cnt=cnt-1;
    end

    %endere o e indice
    ch = fscanf(sid,'%d');
    ch = fscanf(sid,'%d');

    %aceleracao
    ch=fscanf(sid,'%d');
    ch=fscanf(sid,'%d');
    ch=fscanf(sid,'%d');

    %giro
    ugx = fscanf(sid,'%d');
    ugy = fscanf(sid,'%d');
    ugz = fscanf(sid,'%d');

    %campo magnetico
    ch = fscanf(sid,'%d');
    ch = fscanf(sid,'%d');
    ch = fscanf(sid,'%d');

    gx(ix)=ugx;
    gy(ix)=ugy;
    gz(ix)=ugz;

```

```
123     ix=ix+1;
124 end
125 
126 ix=ix-1;
127 fprintf(1,'\nTeminou recep  o de dados.\n');
128 fprintf(sid,'x\r\n');
129 fclose(sid);
130 fprintf(1,'Recebidas %d leituras por eixo.\n',ix);
131 fprintf(1,'Dura  o %.2f segundos.\n',ix/fa);
132 %close all;
133 
134 total_leituras = size(gx,1);
135 fprintf('Total de leituras: %d\n', total_leituras);
136 
137 % Converter giros em "graus/seg"
138 % gx=esc_giro*(gx/32767);
139 % gy=esc_giro*(gy/32767);
140 % gz=esc_giro*(gz/32767);
141 
142 intervalo = 0.01; %10ms
143 eixoX = 0:length(gx)-1;
144 eixoX = eixoX * intervalo;
145 
146 
147 % Offset de calibra  o do giro
148 gx_offset = mean(gx);
149 gy_offset = mean(gy);
150 gz_offset = mean(gz);
151 
152 % Escreve no arquivo os dados de calibra  o
153 fileID  = fopen([directory '\calib_giro.txt'],'w');
154 fprintf(fileID , '%f\n', gx_offset);
155 fprintf(fileID , '%f\n', gy_offset);
156 fprintf(fileID , '%f\n', gz_offset);
157 fclose(fileID);
158 
159 fprintf(1,"Pronto!\n")
```

Calibração simples para o magnetômetro

```
1
2  % Calibra  o Magnetometro - metodo apenas corrigindo as escalas de cada
3  % eixo. A matriz de escala e diagonal
4  % L  dados do Magnet metro pela porta serial usando o modo opera6
5  % fclose(instrfind) -->> fechar porta
6  % Os dados escritos no arquivo  s  o
7  %%%%%%%%%%%%%%%%%%%%%%%%%%%%
8  % offsetx
9  % offsety
10 % offsetz
11 % escalaxx
12 % escalaxy
13 % escalaxz
14 % escalayx
15 % escalayy
16 % escalayz
17 % escalazx
18 % escalazy
19 % escalazz
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
21 %
22 % Para obter o valor, use a express o
23 % sendo H = [
24 %             hx1 hy1 hz1
25 %             hx2 hy2 hz2
26 %             ...
27 %             hxn hyn hzn
28 %           ]
29 % H_Calibrado = (H - offset)*escala
30 %
31 % UNIDADES
32 % Para converter para alguma unidade:
33 % uT: Hx * 4912.0f / 32760.0
34 % G:  Hx * (4912.0f / 32760.0)/100
35 %
```

```
36 % O arquivo mil_leituras.m exemplifica como ler e converter os dados de
       calibragem
37 % ------------------------------------
38
39 clear all;
40 clear;
41 clc;
42
43 % diret rio do script
44 if(~isdeployed)
45   cd(fileparts(which(mfilename)));
46 end
47 directory = pwd;
48
49 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50 % Usuario selecionar sua janela
51 janela=5;       %Tamanho da janela em segundos
52 passo=0.5;      %Passo da janela em segundos
53 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54
55 fprintf(1,'Opera 7\n');
56 fprintf(1,'O Matlab vai receber dados do Magnet metro da Caixa Preta.\n')
       ;
57 fprintf(1,'Rotacione lentamente para todas as dire  es poss veis.\n');
58 fprintf(1,'Em caso de erro, a porta serial pode estar travada.\n');
59 fprintf(1,'Neste caso use "fclose(instrfind)" para fechar porta serial.\n'
       );
60
61 %Par metros
62 fa=100;         %Frequ ncia de amostragem em Hz
63 ta=1/fa;        %Intervalo entre amostras (periodo)
64
65
66 %Calculos para a janela, em amostras
67 tam=janela*fa;   %Tamanho da janela em nr de amostras
68 pa=fa*passo;    %Passo em nr de amostras
69
```

```
sid=serial('COM6','Baudrate',115200);
fopen(sid);
if (sid==-1)
    fprintf(1,'Nao abriu COM6.\n');
    return;
end

x1=0;
x2=0;

hx=zeros(1,tam);
hy=hx;
hz=hx;
eixo=0:ta:janela-ta;

%teste2
% plot(eixo,hx,'b');
pause(2);

% Inicia o modo opera 7
fprintf(sid,'o7\r\n');

while x1~='#' | x2~='['
    x1=x2;
    x2=fread(sid,1);
    fprintf(1,'%c',x2);
end

%Chegou sinal, iniciar recep  o de dados
fprintf(1,'==> Padr o esperado.');
fprintf(1,'\nIniciando recep  o de dados...\n');
pause(1);

asax=fscanf(sid,'%d',10);
asay=fscanf(sid,'%d',10);
asaz=fscanf(sid,'%d',10);

```

```
% ASA precisa ser sem sinal, mas a caixa preta esta enviando com sinal
asax=typecast(int8(asax),'uint8');
asay=typecast(int8(asay),'uint8');
asaz=typecast(int8(asaz),'uint8');

asaxC = 1+ ((double(asax) - 128.0)*0.5)/128.0;
asayC = 1+ ((double(asay) - 128.0)*0.5)/128.0;
asazC = 1+ ((double(asaz) - 128.0)*0.5)/128.0;

figure(1);
uhx=0;  % ltimo  hx
uhy=0;  % ltimo  hy
uhz=0;  % ltimo  hz
ix=1;

hold on;
scatter(hx,hy,'Or') ;
scatter(hx,hz,'Og') ;
scatter(hy,hz,'Ob') ;
grid;
while true
    uhx = fscanf(sid,'%d',10);
    uhy = fscanf(sid,'%d',10);
    uhz = fscanf(sid,'%d',10);
    hx(ix)=uhx;
    hy(ix)=uhy;
    hz(ix)=uhz;
    if hx(1,ix) == 22222 && hx(1,ix) == 22222
        ix=ix+1;
        break;
    end
    if mod(ix,pa) == 0
        scatter(hx, hy, 'r');
        scatter(hx, hz, 'g');
        scatter(hy, hz, 'b');
        title('magnetometro');
        legend('xy', 'xz', 'yz');
```

```
        drawnow;
    end
    ix=ix+1;
end
hold off;

%por algum motivo o primeiro hz esta sempre dando zero. Ignorar ele
hz(1,1) = hz(1,2);

ix=ix-1;
fprintf(1,'\nTeminou recep  o de dados.\n');
fprintf(sid,'x\r\n');
fclose(sid);
fprintf(1,'Recebidas %d leituras por eixo.\n',ix);
fprintf(1,'Dura  o %.2f segundos.\n',ix/fa);
%close all;

% Remover a marca final "22222" de todos os eixos
% Repete a pen ltima leitura
hx(1,ix)=hx(1,ix-1);
hy(1,ix)=hy(1,ix-1);
hz(1,ix)=hz(1,ix-1);

%h corrigido com o ajuste da sensibilidade
hxASA = hx*asaxC;
hyASA = hy*asayC;
hzASA = hz*asazC;

magData = [hxASA' hyASA' hzASA'];

[ h_off, h_sc ] = calibracao_simples(magData);
% [ h_off, h_sc ] = calibracao_lq(magData);

% Aplicar a calibra  o
magDataCalibrated = (magData - h_off)*h_sc;

%escreve em arquivo offsets e escalas
```

```
fName = [directory '\calib_mag.txt'];
fid=fopen(fName,'w');
%Verificar se abriu o arquivo
if (fid==-1)
    fprintf(1,'Nao abriu arquivo [%s]. Parar!\n',fName);
    return;
end

fprintf(fid, '%f\n', h_off(1));
fprintf(fid, '%f\n', h_off(2));
fprintf(fid, '%f\n', h_off(3));

fprintf(fid, '%f\n', h_sc(1));
fprintf(fid, '%f\n', h_sc(2));
fprintf(fid, '%f\n', h_sc(3));
fprintf(fid, '%f\n', h_sc(4));
fprintf(fid, '%f\n', h_sc(5));
fprintf(fid, '%f\n', h_sc(6));
fprintf(fid, '%f\n', h_sc(7));
fprintf(fid, '%f\n', h_sc(8));
fprintf(fid, '%f\n', h_sc(9));
fclose(fid);

%Plota

%converte para uT
magData = magData*4912.0 / 32760;
magDataCalibrated=magDataCalibrated*4912.0 / 32760;

hold on;
figure('Name','Final uT antes de calibrar');
plot(magData(:,1),magData(:,2),'or',magData(:,1),magData(:,3),'og',magData
   (:,2),magData(:,3),'ob');
legend("hx x hy", "hx x hz", "hy x hz");
hold off;

hold on;
```

```
figure('Name','Final apos calibragem');
plot(magDataCalibrated(:,1),magDataCalibrated(:,2),'or', ...
     magDataCalibrated(:,1),magDataCalibrated(:,3),'og', ...
     magDataCalibrated(:,2),magDataCalibrated(:,3),'ob');
legend("hx x hy", "hx x hz", "hy x hz");
hold off;

function [ h_off, h_sc ] = calibracao_simples(magData)
    % Hard Iron - Remover offset
    hx_off=(max(magData(:,1))+min(magData(:,1)))/2;
    hy_off=(max(magData(:,2))+min(magData(:,2)))/2;
    hz_off=(max(magData(:,3))+min(magData(:,3)))/2;

    % Soft Iron - Corrigir a escala
    hx_avg=(max(magData(:,1))-min(magData(:,1)))/2;
    hy_avg=(max(magData(:,2))-min(magData(:,2)))/2;
    hz_avg=(max(magData(:,3))-min(magData(:,3)))/2;









    avg_h=(hx_avg+hy_avg+hz_avg)/3;

    hx_sc=avg_h/hx_avg;
    hy_sc=avg_h/hy_avg;
    hz_sc=avg_h/hz_avg;

    h_off = [ hx_off hy_off hz_off ];
    h_sc = [
              hx_sc  0       0       ;
              0      hy_sc   0       ;
              0      0       hz_sc
            ];
end

function [ h_off, h_sc ] = calibracao_lq(magData)

```

```
hxq = magData(:,1).*magData(:,1);
hyq = magData(:,2).*magData(:,2);
hzq = magData(:,3).*magData(:,3);
Y=-hxq;
PSI=[hyq hzq magData(:,1) magData(:,2) magData(:,3) ones(size(magData,1)
    ,1)];
TETA= inv(PSI'*PSI)*PSI'*Y;
a=TETA(1);
b=TETA(2);
c=TETA(3);
d=TETA(4);
e=TETA(5);
f=TETA(6);

xc=-1*c/2;      %Eq3
yc=-1*d/(2*a);  %Eq1 e Eq4
zc=-1*e/(2*b);  %Eq5 e Eq2
xrq=(xc*xc+a*yc*yc+b*zc*zc)-f;   %Eq6
yrq=xrq/a;      %Eq1
zrq=xrq/b;      %Eq2

h_off=[xc yc zc];
h_sc=[1/xrq 0 0; 0 1/yrq 0; 0 0 1/zrq];


end
```

Calibração do magnetômetro utilizando a função *magcal* do matlab

```
% Calibra  o Magnetometro - Metodo retornando a matriz de escala
    completa
% Utiliza a fun  o do matlab magcal
%  L  dados do Magnet metro pela porta serial usando o modo opera6
% fclose(instrfind) --> fechar porta
% Os dados escritos no arquivo  s  o
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% offsetx
```

```
% offsety
% offsetz
% escalaxx
% escalaxy
% escalaxz
% escalayx
% escalayy
% escalayz
% escalazx
% escalazy
% escalazz
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Para obter o valor, use a express o
% sendo H = [
%             hx1 hy1 hz1
%             hx2 hy2 hz2
%             ...
%             hxn hyn hzn
%           ]
% H_Calibrado = (H - offset)*escala
%
% UNIDADES
% Para converter para alguma unidade:
% uT: Hx * 4912.0f / 32760.0
% G:  Hx * (4912.0f / 32760.0)/100
%
% O arquivo mil_leituras.m exemplifica como ler e converter os dados de
   calibragem
% ---------------------------------------

clear all;
clear;
clc;

% diret rio do script
if(~isdeployed)
```

```
44    cd(fileparts(which(mfilename)));
45  end
46  directory = pwd;
47
48  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49  % Usuario selecionar sua janela
50  janela=5;       %Tamanho da janela em segundos
51  passo=0.5;      %Passo da janela em segundos
52  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53
54  fprintf(1,'Opera 7\n');
55  fprintf(1,'O Matlab vai receber dados do Magnet metro da Caixa Preta.\n')
       ;
56  fprintf(1,'Rotacione lentamente para todas as dire  es poss veis.\n');
57  fprintf(1,'Em caso de erro, a porta serial pode estar travada.\n');
58  fprintf(1,'Neste caso use "fclose(instrfind)" para fechar porta serial.\n'
       );
59
60  %Par metros
61  fa=100;         %Frequ ncia de amostragem em Hz
62  ta=1/fa;        %Intervalo entre amostras (periodo)
63
64
65  %Calculos para a janela, em amostras
66  tam=janela*fa;   %Tamanho da janela em nr de amostras
67  pa=fa*passo;    %Passo em nr de amostras
68
69  sid=serial('COM6','Baudrate',115200);
70  fopen(sid);
71  if (sid==-1)
72      fprintf(1,'Nao abriu COM6.\n');
73      return;
74  end
75
76  x1=0;
77  x2=0;
78
```

```
hx=zeros(1,tam);
hy=hx;
hz=hx;
eixo=0:ta:janela-ta;

%teste2
% plot(eixo,hx,'b');
pause(2);

% Inicia o modo opera 7
fprintf(sid,'o7\r\n');

% Esperar sinal da Caixa Preta
fprintf(1,'\nPor favor, selecione opera 6 na Caixa Preta.\n');
while x1~='#' | x2~='['
    x1=x2;
    x2=fread(sid,1);
    fprintf(1,'%c',x2);
end

%Chegou sinal, iniciar recep  o de dados
fprintf(1,'==> Padr o esperado.');
fprintf(1,'\nIniciando recep  o de dados...\n');
pause(1);

asax=fscanf(sid,'%d',10);
asay=fscanf(sid,'%d',10);
asaz=fscanf(sid,'%d',10);

% ASA precisa ser sem sinal, mas a caixa preta esta enviando com sinal
asax=typecast(int8(asax),'uint8');
asay=typecast(int8(asay),'uint8');
asaz=typecast(int8(asaz),'uint8');

asaxC = 1+ ((double(asax) - 128.0)*0.5)/128.0;
asayC = 1+ ((double(asay) - 128.0)*0.5)/128.0;
asazC = 1+ ((double(asaz) - 128.0)*0.5)/128.0;
```

```
116
117 figure(1);
118 uhx=0;  % ltimo  hx
119 uhy=0;  % ltimo  hy
120 uhz=0;  % ltimo  hz
121 ix=1;
122
123 hold on;
124 scatter(hx,hy,'Or') ;
125 scatter(hx,hz,'Og') ;
126 scatter(hy,hz,'Ob') ;
127 grid;
128 while true
129     uhx = fscanf(sid,'%d',10);
130     uhy = fscanf(sid,'%d',10);
131     uhz = fscanf(sid,'%d',10);
132     hx(ix)=uhx;
133     hy(ix)=uhy;
134     hz(ix)=uhz;
135     if hx(1,ix) == 22222 && hx(1,ix) == 22222
136         ix=ix+1;
137         break;
138     end
139     if mod(ix,pa) == 0
140         scatter(hx, hy, 'r');
141         scatter(hx, hz, 'g');
142         scatter(hy, hz, 'b');
143         title('magnetometro');
144         legend('xy', 'xz', 'yz');
145         drawnow;
146     end
147     ix=ix+1;
148 end
149 hold off;
150
151 %por algum motivo o primeiro hz esta sempre dando zero. Ignorar ele
152 hz(1,1) = hz(1,2);
```

```
153
154 ix=ix-1;
155 fprintf(1,'\nTeminou recep  o de dados.\n');
156 fprintf(sid,'x\r\n');
157 fclose(sid);
158 fprintf(1,'Recebidas %d leituras por eixo.\n',ix);
159 fprintf(1,'Dura  o %.2f segundos.\n',ix/fa);
160 %close all;
161
162 % Remover a marca final "22222" de todos os eixos
163 % Repete a pen ltima leitura
164 hx(1,ix)=hx(1,ix-1);
165 hy(1,ix)=hy(1,ix-1);
166 hz(1,ix)=hz(1,ix-1);
167
168 %h corrigido com o ajuste da sensibilidade
169 hxASA = hx*asaxC;
170 hyASA = hy*asayC;
171 hzASA = hz*asazC;
172
173 magData = [hxASA' hyASA' hzASA'];
174
175 %Faz a calibra  o
176 [A,b,expMFS]  = magcal(magData);
177 magDataCorrected = (magData-b)*A;
178
179 %escreve em arquivo offsets e escalas
180 fName = [directory '\calib_mag.txt'];
181 fid=fopen(fName,'w');
182 %Verificar se abriu o arquivo
183 if (fid==-1)
184     fprintf(1,'Nao abriu arquivo [%s]. Parar!\n',fName);
185     return;
186 end
187 fprintf(fid, '%f\n', b(1));
188 fprintf(fid, '%f\n', b(2));
189 fprintf(fid, '%f\n', b(3));
```

```
fprintf(fid, '%f\n', A(1));
fprintf(fid, '%f\n', A(2));
fprintf(fid, '%f\n', A(3));
fprintf(fid, '%f\n', A(4));
fprintf(fid, '%f\n', A(5));
fprintf(fid, '%f\n', A(6));
fprintf(fid, '%f\n', A(7));
fprintf(fid, '%f\n', A(8));
fprintf(fid, '%f\n', A(9));
fclose(fid);

% Plota convertendo para micro Tesla
escala = 4912.0 / 32760;
magData = magData*escala;
[A,b,expMFS]  = magcal(magData);
magDataCorrected = (magData-b)*A;
de = HelperDrawEllipsoid;
de.plotCalibrated(A,b,expMFS,magData,magDataCorrected,'Auto');
```

Algorítmo criado para as simulações usando o filtro de madgwick

```
% Aplica o filtro de madgwick a partir dos dados gerados pelo le_e_grava.m

% Limpa dados
close all;
clear;
clc;

% diret rio do script
if(~isdeployed)
  cd(fileparts(which(mfilename)));
end
directory = pwd;

madgwickPath = [directory '\quaternion_library'];
addpath(madgwickPath);

```

```
17 fprintf(1,'Os dados ser o lidos de leituras.txt\n');
18
19 %Par metros
20 fa=100;          %Frequ ncia de amostragem em Hz
21 ta=1/fa;         %Intervalo entre amostras (periodo)
22
23 %Escalas
24 esc_giro = 250/32767; % transformar giro em   /s
25 esc_mag  = 4912.0/32760.0; % transformar mag em uT
26 esc_ac   = 9.80665; %transformar acel em m/s2
27
28 %L   do arquivo
29 fid  = fopen([directory '\leituras.txt'],'r');
30 leituras=fscanf(fid,'%f');
31 fclose(fid);
32 qtd_leituras = size(leituras,1)/9;
33
34 %coloca as leituras nas vari veis
35 ax = zeros(qtd_leituras,1);
36 ay = zeros(qtd_leituras,1);
37 az = zeros(qtd_leituras,1);
38
39 gx = zeros(qtd_leituras,1);
40 gy = zeros(qtd_leituras,1);
41 gz = zeros(qtd_leituras,1);
42
43 hx = zeros(qtd_leituras,1);
44 hy = zeros(qtd_leituras,1);
45 hz = zeros(qtd_leituras,1);
46
47 for i=1:qtd_leituras
48     ax(i) = leituras(1+9*(i-1));
49     ay(i) = leituras(2+9*(i-1));
50     az(i) = leituras(3+9*(i-1));
51
52     gx(i) = leituras(4+9*(i-1));
53     gy(i) = leituras(5+9*(i-1));
```

```
    gz(i) = leituras(6+9*(i-1));

    hx(i) = leituras(7+9*(i-1));
    hy(i) = leituras(8+9*(i-1));
    hz(i) = leituras(9+9*(i-1));
end


% Calibra dados

% Acelerometro m/s
fid  = fopen([directory '\calib_acel.txt'],'r');
calibAccel=fscanf(fid,'%f');
fclose(fid);
accel = [(ax-calibAccel(1))*calibAccel(4)*esc_ac (ay-calibAccel(2))*
    calibAccel(5)*esc_ac (az-calibAccel(3))*calibAccel(6)*esc_ac ];

% Giroscopio   /s
fid  = fopen([directory '\calib_giro.txt'],'r');
calibGiro=fscanf(fid,'%f');
fclose(fid);

gyro=[ (gx-calibGiro(1))*esc_giro (gy-calibGiro(2))*esc_giro (gz-calibGiro
    (3))*esc_giro ];


% Magnetmetro uT
fid = fopen([directory '\calib_mag.txt'],'r');
calibMag=fscanf(fid,'%f');
fclose(fid);
h_off = calibMag(1:3)';
h_sc  = reshape(calibMag(4:12),3,3);
magData=[hx hy hz];
mag= (magData-h_off)*h_sc*esc_mag;
% mag= magData*esc_mag;

% Configura eixo X
```

```
89  intervalo = 0.01; %10ms
90  eixoX = 0:length(ax)-1;
91  eixoX = eixoX * intervalo;
92
93  %corrige eixos
94  accel(:,1)=accel(:,1)*-1;
95  accel(:,2)=accel(:,2)*-1;
96
97  gyro(:,1)=gyro(:,1)*-1;
98  gyro(:,2)=gyro(:,2)*-1;
99
100 a = mag(:,1);
101 mag(:,1)=mag(:,2);
102 mag(:,2)=a;
103 mag(:,1)=mag(:,1)*-1;
104 mag(:,2)=mag(:,2)*-1;
105 mag(:,3)=mag(:,3)*-1;
106
107
108 % Cria o filtro
109 % mag = mag/100;
110 % filtro de madgwick (imu)
111 AHRS_IMU = MadgwickAHRS('SamplePeriod', 0.01, 'Beta', 0.1);
112 quaternion_imu = zeros(length(eixoX), 4);
113 for t = 1:length(eixoX)
114     AHRS_IMU.UpdateIMU(gyro(t,:) * (pi/180), accel(t,:));       % as
        unidades do giroscopio devem ser em radiano
115     quaternion_imu(t, :) = AHRS_IMU.Quaternion;
116 end
117
118 euler = quatern2euler(quaternConj(quaternion_imu)) * (180/pi);  % use
    conjugate for sensor frame relative to Earth and convert to degrees.
119 figure('Name', 'Angulos');
120 hold on;
121 plot(eixoX, euler(:,1), 'r');
122 grid;
123 plot(eixoX, euler(:,2), 'g');
```

```
plot(eixoX, euler(:,3), 'b');
title('Rota  o - Filtro de Madgwick IMU');
xlabel('Tempo (s)');
ylabel('Angulos (graus)');
legend('\phi', '\theta', '\psi');
hold off;

% %filtro de madgwick (completo)



AHRS_COMPLETO =  MadgwickAHRS('SamplePeriod', 0.01, 'Beta', 0.1);
quaternion_completo = zeros(length(eixoX), 4);

for t = 1:length(eixoX)
    AHRS_COMPLETO.Update(gyro(t,:) * (pi/180), accel(t,:), mag(t,:))% as
        unidades do giroscopio devem ser em radiano
    quaternion_completo(t, :) = AHRS_COMPLETO.Quaternion;
end
euler = quatern2euler(quaternConj(quaternion_completo)) * (180/pi);
figure('Name', 'Angulos2');
hold on;
plot(eixoX, euler(:,1), 'r');
grid;
plot(eixoX, euler(:,2), 'g');
plot(eixoX, euler(:,3), 'b');
title('Rota  o - Filtro de Madgwick Completo');
xlabel('Tempo (s)');
ylabel('Angulos (graus)');
legend('\phi', '\theta', '\psi');
hold off;
```

Apêndice B

Códigos de Arduíno

B.1 Definições utilizadas na caixa preta

Incluem os registradores utilizados e o atual mapa de memória.

```
1  // Defs.h
2  // Defines usados na Caixa Preta
3  // 20/05/2021
4
5  #define TRUE  1
6  #define FALSE 0
7
8  #define CR 0xD    //Carriage Return
9  #define LF 0xA    //Line Feed
10 #define SPC 0x20  //Espa o
11
12 // BITS - defini  es das posi  es
13 #define BIT0  0x01
14 #define BIT1  0x02
15 #define BIT2  0x04
16 #define BIT3  0x08
17 #define BIT4  0x10
18 #define BIT5  0x20
19 #define BIT6  0x40
20 #define BIT7  0x80
21
22 // Endere os na SRAM
```

```
23 #define MPU_ADR_INI 0x00000L  //In cio  rea  MPU (12.720 msg x 18 =
       228.960 Bytes) = 12,7 seg
24 #define MPU_ADR_FIM 0x37E60L  //Fim     rea  MPU (228.960)
25 #define CXP_ADR_INI 0x37E60L  //In cio  rea  de conficura  o da Caixa
       Preta (sobra 416 bytes)
26 #define CXP_ADR_FIM 0x38000L  //Fim     rea  de conficura  o da Caixa
       Preta
27 #define GPS_ADR_INI 0x38000L  //In cio  rea  GPS (256 msg x 128 = 32.768)
        = 25,6 seg
28 //#define GPS_ADR_FIM 0x40000L  //Fim     rea  GPS
29 #define GPS_ADR_FIM 0x40000L-GPS_PASSO  //Fim     rea  GPS
30 #define MPU_PASSO   18        //Tamanho de uma mensagem do MPU (Acel, Giro
       , Mag)
31 #define GPS_PASSO   128       //Tamanho da mensagem do GPS (gps_dados[])
32
33 // OPERA  O - Principais par metros
34 #define OP_FREQ       SAMPLE_RT_100Hz //Freq de opera  o do MPU e tb
       para Calibra  o ao Ligar:
35 #define OP_ESC_ACEL   ACEL_FS_8G      //Acel: escala para opera  o
36 #define OP_ESC_GIRO   GIRO_FS_2000    //Giro: escala para opera  o
37
38 #define OP_QTD_MED_AG   8             //Calibra  o ao Ligar: quantidade
       de medidas por eixo acel e giro
39 #define OP_QTD_MED_MG  64             //Calibra  o ao Ligar: quantidade
       de medidas por eixo mag
40
41 // C digos usados para indicar sim/n o , passou/falhou
42 #define COD_SIM  0x5353     //(21.331) 2x ASCII(S) Afirmativo
43 #define COD_NAO  0x4E4E     //(20.046) 2x ASCII(N) Negativo
44
45 // Limiares para indicar Acidente
46 #define LIMIAR_AX  4     //4g
47 #define LIMIAR_AY  4     //4g
48 #define LIMIAR_AZ  4     //4g
49 #define LIMIAR_GX  1000 //1.000 gr/s
50 #define LIMIAR_GY  1000 //1.000 gr/s
51 #define LIMIAR_GZ  1000 //1.000 gr/s
```

```
52
53 // SRAM (FLASH) — Uso na Opera  o — Posi  es para guardar par metros
54 #define OP_OK      CXP_ADR_INI  //Fez calibra  o ao ligar? 0x4E4E=NN=
       N o (pronta) 0x5353=SS=Sim — Repeti  o do ASCII
55 #define OP_BATEU   OP_OK+2      //Bateu? 0x4E4E=NN=N o (pronta) 0x5353=SS
       =Sim — Repeti  o do ASCII
56 #define OP_ST_OK   OP_BATEU+2   //Passou no Self Test 0x4E4E=n o e 0x5353
       =sim
57 #define OP_STH_OK  OP_ST_OK+2   //Magnet metro Passou no Self Test 0x4E4E
       =n o e 0x5353=sim
58 #define OP_CF_OK   OP_STH_OK+2  //Existe calibra  o de F brica (CF) 0
       x4E4E=n o e 0x5353=sim
59 #define OP_CFH_OK  OP_CF_OK+2   //Magent metro fez calibra  o de
       F brica (CF) 0x4E4E=n o e 0x5353=sim
60
61 // Calibra  o Acelera  o e Giro ao ligar o carro
62 #define OPC_FREQ_AG OP_CFH_OK+2   //Freq de amostragem usada na
       calibra  o Acel e Giro
63 #define OPC_BW_AG   OPC_FREQ_AG+2 //Banda passante usada na calibra  o
       Acel e Giro
64 #define OPC_ESC_AC  OPC_BW_AG+2   //Escala do Acelerometro usada na
       calibra  o
65 #define OPC_ESC_GI  OPC_ESC_AC+2  //Escala do Giroscopio usada na
       calibra  o
66 #define OPC_QTD_AG  OPC_ESC_GI+2  //QTd de medidas para a calibra  o
       Acel e Giro
67 #define OPC_AX      OPC_QTD_AG+2  //Calibra  o AX
68 #define OPC_AY      OPC_AX+2      //Calibra  o AY
69 #define OPC_AZ      OPC_AY+2      //Calibra  o AZ
70 #define OPC_TP      OPC_AZ+2      //Calibra  o TP
71 #define OPC_GX      OPC_TP+2      //Calibra  o GX
72 #define OPC_GY      OPC_GX+2      //Calibra  o GY
73 #define OPC_GZ      OPC_GY+2      //Calibra  o GZ
74 // Calibra  o Magnet metro ao ligar o carro
75 #define OPC_QTD_MG  OPC_GZ+2      //QTd de medidas para a calibra  o
       Magn
76 #define OPC_ESC_MG  OPC_QTD_MG+2  //Escalas acel (MSB) e giro (LSB)
```

```
77  #define OPC_HX      OPC_ESC_MG+2  //Calibra  o HX
78  #define OPC_HY      OPC_HX+2      //Calibra  o HY
79  #define OPC_HZ      OPC_HY+2      //Calibra  o HZ
80
81  // Par metros para opera  o: escala e limiar
82  #define OP_FREQ_AG  OPC_HZ+2      //Freq de amostragem usada na opera  o
        Acel e Giro
83  #define OP_BW_AG    OP_FREQ_AG+2  //Banda passante usada na opera  o
       Acel e Giro
84  #define OP_ESC_AC   OP_BW_AG+2    //Escala do Acelerometro usada na
       opera  o
85  #define OP_ESC_GI   OP_ESC_AC+2   //Escala de giro para operacao
86  #define OP_ESC_MG   OP_ESC_GI+2   //Escala de magnet para operacao
87  #define OP_LIM_AX   OP_ESC_MG+2   //AX - Limiar para disparo (valor
       absoluto)
88  #define OP_LIM_AY   OP_LIM_AX+2   //AY - Limiar para disparo (valor
       absoluto)
89  #define OP_LIM_AZ   OP_LIM_AY+2   //AZ - Limiar para disparo (valor
       absoluto)
90  #define OP_LIM_GX   OP_LIM_AZ+2   //GX - Limiar para disparo (valor
       absoluto)
91  #define OP_LIM_GY   OP_LIM_GX+2   //GY - Limiar para disparo (valor
       absoluto)
92  #define OP_LIM_GZ   OP_LIM_GY+2   //GZ - Limiar para disparo (valor
       absoluto)
93  // Quem disparou
94  #define OP_MPU_ADR  OP_LIM_GZ+2   //(32 bits) Endereco MPU no momento do
       disparo
95  #define OP_GPS_ADR  OP_MPU_ADR+4  //(32 bits) Endereco GPS no momento do
       disparo
96  #define OP_DISP_TP  OP_GPS_ADR+4  //Temperatura no instante do disparo
97  #define OP_DISP_AX  OP_DISP_TP+2  //AX disparou (SS=Sim e NN=Nao)
98  #define OP_DISP_AY  OP_DISP_AX+2  //AY disparou (SS=Sim e NN=Nao)
99  #define OP_DISP_AZ  OP_DISP_AY+2  //AZ disparou (SS=Sim e NN=Nao)
100 #define OP_DISP_GX  OP_DISP_AZ+2  //GX disparou (SS=Sim e NN=Nao)
101 #define OP_DISP_GY  OP_DISP_GX+2  //GY disparou (SS=Sim e NN=Nao)
102 #define OP_DISP_GZ  OP_DISP_GY+2  //GZ disparou (SS=Sim e NN=Nao)
```

```
103 #define OP_BRK      OP_DISP_GZ+2  //Aquisi  o foi interrompida (S=Sim e
    NN=N o)
104 #define OP_ULT_ADR  OP_BRK+2      // ltimo  endere o gravado pelo MPU
105
106 // Data e hora do acidente
107 #define OP_AC_DATA  OP_ULT_ADR+4    //ddmmyy0 --> Data do acidente, vem do
     GPS
108 #define OP_AC_HORA  OP_AC_DATA+8    //hhmmss.sss0 --> hora do acidente,
    vem do GPS
109 #define OP_VAZIO    OP_AC_HORA+12   //Vazio
110
111 // CALIBRA  O DE F BRICA
112 const char *CF_HOJE = "24/07/20"; //Data para Configura  o de F brica
113 const char *CF_BSB = "Brasilia";  //Data para Configura  o de F brica
114 #define G_PADRAO    9.80665       //1g padr o
115 #define G_BSB       9.7808439     //Ac. gravidade em Bras lia
116 #define CF_ESPERA   10          //Tempo de espera (seg) antes de calibrar
    o MPU
117 #define CF_FREQ     100           //Calibra  o de F brica: Freq de
    amostragem do MPU
118 #define CF_QTD_MED  256         //Calibra  o de F brica: quantidade de
    medidas por eixo
119
120 // Endere os da EEPROM
121 // Usada para guardar Calibra ao de F brica
122 // Posi  es para Data, Local e Acelera  o
123 #define EEPROM_TAM      4096  //Tamanho da EEPROM (12 bits de endere os)
124 #define CF_COD_OK       0x5353            //C digo indica calibra  o
     j   fita, SIM = SS = 0x5353
125
126 #define CF_OK           0                 //0 - J  fez calibra  o? SS =
     SIM
127 #define CF_DATA         CF_OK+2           //2 - Data da configura  o
128 #define CF_LOCAL        CF_DATA+14        //10 - String com Local da
    configura  o
129 #define CFG_PADRAO      CF_LOCAL+32       //30 - String com Acelera  o
    da gravidade padr o
```

```
130 #define CFG_LOCAL       CFG_PADRAO+16     //40 - String com Acelera  o
    da gravidade local
131 #define CFG_PADRAO_BIN  CFG_LOCAL+16      //50 - Inteiro com Acelera  o
    da gravidade padr o
132 #define CFG_LOCAL_BIN   CFG_PADRAO_BIN+2  //52 - Inteiro com Acelera  o
    da gravidade local
133 #define CF_WHO          CFG_LOCAL_BIN+2   //54 - Resposta ao Who am I em
    decimal
134 //////// Calibra  o --- Posi  es para guardar par metros e resultados
     das m dias das medidas
135 #define CF_FA           CF_WHO+2          //56 - Freq de amostragem usada
    na calibra  o
136 #define CF_BW           CF_FA+2           //58 - Banda passante usada na
    calibra  o
137 #define CF_QTD          CF_BW+2           //5A - Quantidade de medidas por
     eixo
138 #define CF_ESC_AC       CF_QTD+2          //5C - Escala usada para o
    acelerometro
139 #define CF_ESC_GI       CF_ESC_AC+2       //5E - Escala usada para o
    giroscopio
140 #define CF_AX           CF_ESC_GI+2       //60
141 #define CF_AY           CF_AX+2           //62
142 #define CF_AZ           CF_AY+2           //64
143 #define CF_TP           CF_AZ+2           //66
144 #define CF_GX           CF_TP+2           //68
145 #define CF_GY           CF_GX+2           //6A
146 #define CF_GZ           CF_GY+2           //6C
147 //////// Somat rios --- Posi  es para guardar a m dia das medidas
148 #define CF_AX_SOMA     CF_GZ+2            //6E
149 #define CF_AY_SOMA     CF_AX_SOMA+4       //70
150 #define CF_AZ_SOMA     CF_AY_SOMA+4       //74
151 #define CF_TP_SOMA     CF_AZ_SOMA+4       //78
152 #define CF_GX_SOMA     CF_TP_SOMA+4       //7C
153 #define CF_GY_SOMA     CF_GX_SOMA+4       //80
154 #define CF_GZ_SOMA     CF_GY_SOMA+4       //84
155 //////// Calibra  o --- Posi  es para guardar a Primeira e  ltima  das
     medidas e a m dia
```

```
156 #define CF_AX_PRI      CF_GZ_SOMA+4       //88
157 #define CF_AY_PRI      CF_AX_PRI+2        //8A
158 #define CF_AZ_PRI      CF_AY_PRI+2        //8C
159 #define CF_TP_PRI      CF_AZ_PRI+2        //8E
160 #define CF_GX_PRI      CF_TP_PRI+2        //90
161 #define CF_GY_PRI      CF_GX_PRI+2        //92
162 #define CF_GZ_PRI      CF_GY_PRI+2        //94
163 #define CF_AX_ULT      CF_GZ_PRI+2        //96
164 #define CF_AY_ULT      CF_AX_ULT+2        //98
165 #define CF_AZ_ULT      CF_AY_ULT+2        //9A
166 #define CF_TP_ULT      CF_AZ_ULT+2        //9C
167 #define CF_GX_ULT      CF_TP_ULT+2        //9E
168 #define CF_GY_ULT      CF_GX_ULT+2        //A0
169 #define CF_GZ_ULT      CF_GY_ULT+2        //A2
170 //////// Self Test --- Posi  es para guardar resultados do self-test
171 #define CF_ST_OK       CF_GZ_ULT+2        //A4 - Passou no self test? TRUE
    /FALSE
172 #define CF_ST_OFF_AX   CF_ST_OK+2
173 #define CF_ST_OFF_AY   CF_ST_OFF_AX+2
174 #define CF_ST_OFF_AZ   CF_ST_OFF_AY+2
175 #define CF_ST_OFF_GX   CF_ST_OFF_AZ+2
176 #define CF_ST_OFF_GY   CF_ST_OFF_GX+2
177 #define CF_ST_OFF_GZ   CF_ST_OFF_GY+2
178
179 #define CF_ST_ON_AX    CF_ST_OFF_GZ+2
180 #define CF_ST_ON_AY    CF_ST_ON_AX+2
181 #define CF_ST_ON_AZ    CF_ST_ON_AY+2
182 #define CF_ST_ON_GX    CF_ST_ON_AZ+2
183 #define CF_ST_ON_GY    CF_ST_ON_GX+2
184 #define CF_ST_ON_GZ    CF_ST_ON_GY+2
185
186 #define CF_ST_REG_AX   CF_ST_ON_GZ+2    //ax - Reg de self-test (16 bits
    mas usa apenas 8 bits)
187 #define CF_ST_REG_AY   CF_ST_REG_AX+2   //ay - Reg de self-test (16 bits
    mas usa apenas 8 bits)
188 #define CF_ST_REG_AZ   CF_ST_REG_AY+2   //az - Reg de self-test (16 bits
    mas usa apenas 8 bits)
```

```
189 #define CF_ST_REG_GX   CF_ST_REG_AZ+2   //gx - Reg de self-test (16 bits
    mas usa apenas 8 bits)
190 #define CF_ST_REG_GY   CF_ST_REG_GX+2   //gy - Reg de self-test (16 bits
    mas usa apenas 8 bits)
191 #define CF_ST_REG_GZ   CF_ST_REG_GY+2   //gz - Reg de self-test (16 bits
    mas usa apenas 8 bits)
192
193 #define CF_ST_TOL_AX   CF_ST_REG_GZ+2   //ax - Resultado self-test (<14%)
    (16 bits mas usa apenas 8 bits)
194 #define CF_ST_TOL_AY   CF_ST_TOL_AX+2   //ay - Resultado self-test (<14%)
    (16 bits mas usa apenas 8 bits)
195 #define CF_ST_TOL_AZ   CF_ST_TOL_AY+2   //az - Resultado self-test (<14%)
    (16 bits mas usa apenas 8 bits)
196 #define CF_ST_TOL_GX   CF_ST_TOL_AZ+2   //gx - Resultado self-test (<14%)
    (16 bits mas usa apenas 8 bits)
197 #define CF_ST_TOL_GY   CF_ST_TOL_GX+2   //gy - Resultado self-test (<14%)
    (16 bits mas usa apenas 8 bits)
198 #define CF_ST_TOL_GZ   CF_ST_TOL_GY+2   //gz - Resultado self-test (<14%)
    (16 bits mas usa apenas 8 bits)
199
200 #define CF_MAG_OK     CF_ST_TOL_GZ+2   //J  fez calibra  o do
    Magnet metro? COD_SIM ou COD_NAO
201 #define CF_STH_OK     CF_MAG_OK+2      //Magnet metro passou no Self Test
    ?
202 #define CF_STH_HX     CF_STH_OK+2      //Leitura de HX durante o Self Test
    ?
203 #define CF_STH_HY     CF_STH_HX+2      //Leitura de HY durante o Self Test
    ?
204 #define CF_STH_HZ     CF_STH_HY+2      //Leitura de HZ durante o Self Test
    ?
205 #define CF_HX_ASA     CF_STH_HZ+2      //hx - ASA = Ajuste gravado na ROM
    (16 bits mas usa apenas 8 bits) Negativo?
206 #define CF_HY_ASA     CF_HX_ASA+2      //hy - ASA = Ajuste gravado na ROM
    (16 bits mas usa apenas 8 bits) Negativo?
207 #define CF_HZ_ASA     CF_HY_ASA+2      //hz - ASA = Ajuste gravado na ROM
    (16 bits mas usa apenas 8 bits) Negativo?
```

```
208 #define CF_HX_OFF     CF_HZ_ASA+2      //hx - Offset multiplicado por 10,
    para dar precis o
209 #define CF_HY_OFF     CF_HX_OFF+2      //hy - Offset multiplicado por 10,
    para dar precis o
210 #define CF_HZ_OFF     CF_HY_OFF+2      //hz - Offset multiplicado por 10,
    para dar precis o
211 #define CF_HX_ESC     CF_HZ_OFF+2      //hx - Escala multiplicada por 10,
    para dar precis o
212 #define CF_HY_ESC     CF_HX_ESC+2      //hx - Escala multiplicada por 10,
    para dar precis o
213 #define CF_HZ_ESC     CF_HY_ESC+2      //hx - Escala multiplicada por 10,
    para dar precis o
214 //dados usados na calibra  o do acelerometro
215 #define CF_AX_X_CIMA     CCF_HZ_ESC+2     //valor de ax quando o eixo X
    aponta para cima
216 #define CF_AX_X_BAIXO    CF_AX_X_CIMA+2   //valor de ax quando o eixo X
    aponta para baixo
217 #define CF_AX_Y_CIMA     CF_AX_X_BAIXO+2  //valor de ay quando o eixo Y
    aponta para cima
218 #define CF_AX_Y_BAIXO    CF_AX_Y_CIMA+2   //valor de ay quando o eixo Y
    aponta para baixo
219 #define CF_AX_Z_CIMA     CF_AX_Y_BAIXO+2  //valor de az quando o eixo Z
    aponta para cima
220 #define CF_AX_Z_BAIXO    CF_AX_Z_CIMA+2   //valor de az quando o eixo Z
    aponta para baixo
221 //dados usados na calibra  o do magnet metro
222 #define CF_HX_MIN    CF_AX_Z_BAIXO+2      // valor m nimo de hx
223 #define CF_HX_MAX    CF_HX_MIN+2          // valor m ximo de hx
224 #define CF_HY_MIN    CF_HX_MAX+2          // valor m nimo de hy
225 #define CF_HY_MAX    CF_HY_MIN+2          // valor m nimo de hy
226 #define CF_HZ_MIN    CF_HY_MAX+2          // valor m nimo de hz
227 #define CF_HZ_MAX    CF_HZ_MIN+2          // valor m ximo de hz
228
229
230
231 // LCD Constantes para os bits de controle
232 #define LCD_BL  8 //Back Light
```

```
233 #define LCD_RW  4 //R/#W
234 #define LCD_RS  2 //RS
235 #define LCD_E   1 //Enable
236
237 // LCD - Bits para indicar qual linha mudou
238 #define LCD_LINHA0 BIT0 //Linha 0 foi alterada
239 #define LCD_LINHA1 BIT1 //Linha 1 foi alterada
240 #define LCD_LINHA2 BIT2 //Linha 0 foi alterada
241 #define LCD_LINHA3 BIT3 //Linha 1 foi alterada
242
243 // LCD - Setas Esqeuerda e Direita
244 #define LCD_SETA_DIR  0x7E  //
245 #define LCD_SETA_ESQ  0x7F  //
246
247 // Timer 1
248 #define FREQ_T1 100  //Freq de interrup  o do timer 1
249
250 // Timer 2
251 #define FREQ_T2 1000  //Freq de interrup  o do timer 1
252 //#define FREQ_T2 5000  //Freq de interrup  o do timer 1
253 //#define FREQ_T2 25000  //Freq de interrup  o do timer 1
254
255 // TESTE
256 #define TESTE_TOT 17     //Modos de teste: 1, 2 , ..., 17
257 #define TESTE_0    0     //Opera
258 #define TESTE_1    1     //LEDs
259 #define TESTE_2    2     //LCD
260 #define TESTE_3    3     //Teclado
261 #define TESTE_4    4     //TWI
262 #define TESTE_5    5     //Acel e giro
263 #define TESTE_6    6     //Magnetometro
264 #define TESTE_7    7     //SRAM
265 #define TESTE_8    8     //FLAH
266 #define TESTE_9    9     //GPS Tudo
267 #define TESTE_10  10     //GPS RMC GSA
268 #define TESTE_11  11     //GPS U-Center
269 #define TESTE_12  12     //MPU->Matlab
```

```
270 #define TESTE_13  13     //BlueTooth
271 #define TESTE_14  14     //Livre
272 #define TESTE_15  15     //Livre
273 #define TESTE_16  16     //Livre
274 #define TESTE_17  17     //Livre
275
276 // OPERA
277 #define OPERA_TOT  9     //Modos de teste: 1, 2 , ..., 9
278 #define OPERA_0    0     //Teste
279 #define OPERA_1    1     //Aqusi  o de Dados
280 #define OPERA_2    2     //Livre
281 #define OPERA_3    3     //Livre
282 #define OPERA_4    4     //Livre
283 #define OPERA_5    5     //Calibra  o de F brica
284 #define OPERA_6    6     //Calibra  o do Magnet metro
285 #define OPERA_7    7     //Livre
286 #define OPERA_8    8     //Livre
287 #define OPERA_9    9     //Livre
288
289 // TECLADO - Parametros para leitura das chaves
290 #define  SW_FILA_TAM  10  // tamanho da fila do teclado
291 #define  SW_TOL     20    // tolerancia para identificar chave
292 // Codigos para as chaves
293 #define  SW_NADA    7
294 #define  SW_INF     6
295 #define  SW_DIR     4
296 #define  SW_SUP     3
297 #define  SW_ESQ     2
298 #define  SW_SEL     0
299 #define  SW_SEQ1    8
300 #define  SW_SEQ2    9
301 #define  SW_NAOSEI 10
302
303 // Aleat - Constantes para o gerador pseudo-aleat rio
304 #define ALEAT_SUGEST_M   53  //m sugerido
305 #define ALEAT_SUGEST_D   109 //d sugerido
306 #define ALEAT_SUGEST_U   13  //u sugerido (semente)
```

```
307
308
309 // GPS
310 // Identificar tipos de mensagens
311 #define GPS_NADA  0
312 #define GPS_RMC   1
313 #define GPS_VTG   2
314 #define GPS_GGA   3
315 #define GPS_GSA   4
316 #define GPS_GSV   5
317 #define GPS_GLL   6
318 #define GPS_MSG_TAM 200   //Tamanho max de uma msg do GPS
319 #define GPS_DADOS_TAM 128   //Tamanho do vetor para os dados extra dos do
        GPS
320
321 // Marcar posi  o de cada um dos par metros guardados em gps_dados[
       GPS_DADOS_TAM]
322 #define GPS_STATUS     0                 //2 bytes
323 #define GPS_HORA      (GPS_STATUS+2)    //11 bytes
324 #define GPS_DATA      (GPS_HORA+11)     //7 bytes
325 #define GPS_LAT       (GPS_DATA+7)      //11 bytes
326 #define GPS_NS        (GPS_LAT+11)      //2 bytes
327 #define GPS_LONG      (GPS_NS+2)        //12 bytes
328 #define GPS_EW        (GPS_LONG+12)     //2 bytes
329 #define GPS_VEL_NOS   (GPS_EW+2)        //8 bytes
330 #define GPS_CURSO     (GPS_VEL_NOS+8)   //8 bytes
331 #define GPS_PDOP      (GPS_CURSO+8)     //6 bytes
332 #define GPS_HDOP      (GPS_PDOP+6)      //6 bytes
333 #define GPS_VDOP      (GPS_HDOP+6)      //6 bytes
334 #define GPS_VEL_KPH   (GPS_VDOP+6)      //7 bytes
335 #define GPS_VEL_UN    (GPS_VEL_KPH+7)   //2 bytes
336 #define GPS_FIX       (GPS_VEL_UN+2)    //2 bytes ?sem uso? n o me
       lembro por que
337 #define GPS_QTD_SAT   (GPS_FIX+2)       //3 bytes
338 #define GPS_ALT       (GPS_QTD_SAT+3)   //7 bytes
339 #define GPS_ALT_UN    (GPS_ALT+7)       //2 bytes
340 #define GPS_ADR_SRAM  (GPS_ALT_UN+2)    //5 bytes
```

```
341
342 /////////////// MPU 6050 - Constantes
343 #define MPU_ADR  0x68  //Endere o MPU-6050
344 #define MPU_EWR  0xD0  //MPU para escrita (0x68<<1)
345 #define MPU_ERD  0xD1  //MPU para leitura (0x68<<1 + 1)
346 #define MPU9250_WHO 0x73  //MPU Who am I
347
348 //Escalas para Giros pio
349 #define GIRO_FS_250  0   // +/- 250 graus/seg
350 #define GIRO_FS_500  1   // +/- 500 graus/seg
351 #define GIRO_FS_1000 2   // +/- 1000 graus/seg
352 #define GIRO_FS_2000 3   // +/- 2000 graus/seg
353
354 //Escalas para Acelerometro
355 #define ACEL_FS_2G  0   // +/- 2g
356 #define ACEL_FS_4G  1   // +/- 4g
357 #define ACEL_FS_8G  2   // +/- 8g
358 #define ACEL_FS_16G 3   // +/- 16g
359
360 // Valores para o Sample Rate, Registrador SMPLRT_DIV
361 // Considerando Taxa = 1kHz (Registrador CONFIG)
362 #define SAMPLE_RT_1kHz     0   // 1.000/(0+1) = 1000
363 #define SAMPLE_RT_500Hz    1   // 1.000/(1+1) = 500
364 #define SAMPLE_RT_333Hz    2   // 1.000/(2+1) = 333,33
365 #define SAMPLE_RT_250Hz    3   // 1.000/(3+1) = 250
366 #define SAMPLE_RT_200Hz    4   // 1.000/(4+1) = 200
367 #define SAMPLE_RT_166Hz    5   // 1.000/(5+1) = 166,66
368 #define SAMPLE_RT_142Hz    6   // 1.000/(6+1) = 142,85
369 #define SAMPLE_RT_125Hz    7   // 1.000/(7+1) = 125
370 #define SAMPLE_RT_111Hz    8   //1.000 /(8+1) = 111,11
371 #define SAMPLE_RT_100Hz    9   //1.000 /(9+1) = 100
372
373 // Registradores do MPU-9250 que foram usados
374
375 //adicionados 26 set
376 #define SELF_TEST_X_GYRO      0x00
377 #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 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 SELF_TEST_X      0x0D
#define SELF_TEST_Y      0x0E
#define SELF_TEST_Z      0x0F
#define SELF_TEST_A      0x10
#define SMPLRT_DIV       0x19
#define CONFIG           0x1A
#define GYRO_CONFIG      0x1B
#define ACCEL_CONFIG     0x1C
#define ACCEL_CONFIG_2   0x1D //adicionado 26 set
#define FIFO_EN          0x23
#define INT_PIN_CFG      0x37
#define INT_ENABLE       0x38
#define INT_STATUS       0x3A
#define ACCEL_XOUT_H     0x3B
#define TEMP_OUT_H       0x41
#define USER_CTRL        0x6A
#define PWR_MGMT_1       0x6B
#define FIFO_COUNTH      0x72
#define FIFO_COUNTL      0x73
#define FIFO_R_W         0x74
#define WHO_AM_I         0x75

/////////////// MPU 9250 - Magnet metro
#define MAG_I2C_ADDR    0x0C //endere o i2c do magnetometro
#define MAG_I2C_ADDR_WR 0x18 //0x0c << 1
```

```
415 #define MAG_I2C_ADDR_RD 0x19 //(0x0c << 1) + 1
416 #define MAG_WHO         0x48  //MAG Who am I
417 // Registradores
418 #define MAG_CNTL_1      0x0A  //Controle 1
419 #define MAG_CNTL_2      0x0B  //(RSV) Controle 2
420 #define MAG_ASTC        0x0C  //Self Test
421 #define MAG_XOUT_L      0x03  //MAG XL seq:[XL XH YL YH ZL ZH]
422 #define MAG_ASAX        0x10 //end. reg. ASAX do magnetometro
423 #define MAG_ASAY        0x11 //end. reg. ASAY do magnetometro
424 #define MAG_ASAZ        0x12 //end. reg. ASAZ do magnetometro
425 #define MAG_ST1         0x02 //end. reg. ST1 do magnetometro (DRDY)
426 #define MAG_ST2         0x09 //end. reg. ST2 do magnetometro (HOFL)
427
428 //
    ////////////////////////////////////////////////////////////////////////////////
429 ////////////// CONSTANTES DE BAIXO N VEL
    ////////////////////////////////////////
430 //
    ////////////////////////////////////////////////////////////////////////////////
431
432 // SRAM 23LC1024
433
434 #define SRAM_TAM_CHIP 0x20000L  //M ximo por chip
435
436 // Instru  es da SRAM
437 #define SRAM_READ    3   //Ler dado da mem ria
438 #define SRAM_WRITE   2   //Escrever dado da mem ria
439 #define SRAM_RDMR    5   //Ler Registrador de Modo
440 #define SRAM_WRMR    1   //Escrever no Registrador de Modo
441
442 // Registrador de Modo: Modos de Opera  o
443 #define SRAM_MODO_BYTE   0x00  //Modo Byte
444 #define SRAM_MODO_PAG    0x80  //Modo P gina
445 #define SRAM_MODO_SEQ    0x40  //Modo Sequencial
446
```

```
447 // Velocidades SPI, verificar o dobrador (SPI2X)
448 #define SPI_125K   0 //SCL=125KHz, SPI2X=0
449 #define SPI_250K   1 //SCL=250KHz, SPI2X=0
450 #define SPI_500K   2 //SCL=500KHz, SPI2X=1
451 #define SPI_1M     3 //SCL=1MHz,   SPI2X=0
452 #define SPI_2M     4 //SCL=2MHz    SPI2X=1
453 #define SPI_4M     5 //SCL=4MHz    SPI2X=0
454 #define SPI_8M     6 //SCL=8MHz    SPI2X=1
455
456 #define MISO  50  //Master Input
457 #define MOSI  51  //Master Output
458 #define SCK   52  //Sa da do rel gio
459 #define CS0   49  //(PL0) Controla o estado do #CS0 (0x0 0000 -> 0x1 FFFF)
460 #define CS1   48  //(PL1) Controla o estado do #CS1 (0x2 0000 -> 0x3 FFFF)
461 #define CS2   47  //(PL2) Controla o estado do #CS2
462 #define SS    53  //Para Mestre    preciso SS como sa da
463
464 // LCD
465 #define NRL 4   //Qtd de linhas do LCD
466 #define NRC 20  //Qtd de colunas do LCD
467
468
469 ////////////////////////////////////////////////////////////
470 ////////////////////// FILAS ///////////////////////////////
471 ////////////////////////////////////////////////////////////
472
473 // UART0(Arduini) e UART2(Bluetooth) integradas
474 #define SERI_FILA_TAM 50   //Tamanho da fila circular de entrada serial
475 #define SERO_FILA_TAM 200   //Tamanho da fila circular de sa da
476
477 // GPS - Serial 3
478 #define GPS_TX_FILA_TAM 10 //Tamanho da fila circular de TX
479 #define GPS_RX_FILA_TAM 300 //Tamanho da fila circular de RX
480
481 // Endere os da FLASH 24LC1025 (128 KB) TWI
482 #define FLASH1_ADR     0x50  //FLASH1
483 #define FLASH2_ADR     0x51  //FLASH2
```

```
484 #define FLASH1_ADR_B0  FLASH1_ADR+0  //FLASH1, Bloco 0: 64KB (0x00000 -> 0
    x0FFFF)
485 #define FLASH1_ADR_B1  FLASH1_ADR+4  //FLASH1, Bloco 1: 64KB (0x10000 -> 0
    x1FFFF)
486 #define FLASH2_ADR_B0  FLASH2_ADR+0  //FLASH2, Bloco 0: 64KB (0x00000 -> 0
    x0FFFF)
487 #define FLASH2_ADR_B1  FLASH2_ADR+4  //FLASH2, Bloco 1: 64KB (0x10000 -> 0
    x1FFFF)
488 #define FLASH_PAG   128              //Tamanho da p gina para grava  o
489
490 /////////////// TWI - C digos de Status
491 #define TWI_START_OK      8     //Start OK
492 #define TWI_START_REP_OK  0x10  //Start Repetido OK
493 #define TWI_SLA_WR_ACK    0x18  //EET enviado e ACK recebido
494 #define TWI_SLA_WR_NACK   0x20  //EET enviado e NACK recebido
495 #define TWI_TX_DATA_ACK   0x28  //Dado enviado e ACK recebido
496 #define TWI_SLA_RD_ACK    0x40  //EER enviado e ACK recebido
497 #define TWI_SLA_RD_NACK   0x48  //EER enviar e NACK recebido
498 #define TWI_RX_DATA_NACK  0x58  //Dado recebido e NACK gerado
499 #define TWI_RX_DATA_ACK   0x50  //Dado recebido e ACK gerado
500 #define TWI_TMI_OUT       10000 //Time out
501
502 // C digos de erro no trabalho com TWI
503 #define TWI_ERRO_1  1   //Erro ao gerar START
504 #define TWI_ERRO_2  2   //Erro ao gerar START Repetido
505 #define TWI_ERRO_3  3   //Erro Escravo Receptor endere ado (ER) n o
    enviou ACK
506 #define TWI_ERRO_4  4   //Erro Escravo Transmissor endere ado (ET) n o
    enviou ACK
507 #define TWI_ERRO_5  5   //Erro Escravo Receptor (ER) n o enviou ACK ap s
     envio do dado
508 #define TWI_ERRO_6  6   //Erro ao receber um dado do Escravo Transmissor (
    ET) e gerar um ACK
509 #define TWI_ERRO_7  7   //Erro ao receber um dado do Escravo Transmissor (
    ET) e gerar um NACK
510 #define TWI_ERRO_8  8   //Erro ao esperar TWINT - Timeout esperando TWINT
    ir para 1
```

B.2 Configuração do IMU do MPU-9250

```
// Colocar o MPU num estado conhecido
// Taxa = 1 kHz, Banda: Acel=5.05 Hz e Giro=5 Hz. Delay Acel = 32.48ms.
   Delay Giro = 33.48
// Taxa de amostragem =  taxa/(1+SMPLRT_DIV) = 1k/10 = 100Hz
//Escalas acel = +/2g e giro = +/-250 gr/s
void mpu_config(void) {

  // Despertar MPU, Rel gio = PLL do Giro-x
  mpu_wr(PWR_MGMT_1, 0x01);
  delay(200);       //200ms - Esperar PLL estabilizar

  // Definir escalas
  mpu_escalas(GIRO_FS_250, ACEL_FS_2G);   //Escalas acel = +/2g e giro =
     +/-250 gr/s

  // 6 => Liga o filtro passa-baixa do giros c pio e temperatura para para
      5Hz
  // Delay giro = 33.48ms, Taxa giro = 1Khz; Delay temperatura 18.6ms
  mpu_wr(CONFIG, 6);

  // 6 => Liga o filtro passa-baixa do aceler metro para para 5.05Hz.
     Delay= 32.48ms
  mpu_wr(ACCEL_CONFIG_2, 6);

  // 9 ==> Taxa de amostragem =  taxa/(1+SMPLRT_DIV) = 1k/10 = 100Hz
  mpu_wr(SMPLRT_DIV, SAMPLE_RT_100Hz);  //Taxa de amostragem = 100 Hz
  //mpu_wr(SMPLRT_DIV, SAMPLE_RT_500Hz);  //Taxa de amostragem = 500 Hz
}
```

B.3 Configuração do Magnetômetro

```
// Inicializar Magnet metro
void mpu_mag_config(void){
  mpu_wr(USER_CTRL, 0x00);          //Desab. modo mestre no mpu
```

```
  mpu_wr(INT_PIN_CFG, 0x02);        //Hab.o bypass I2C
  mpu_wr_mg_reg(MAG_CNTL_1, 0x00); //Magnetometo Power Down
  delay(100);                       //Espera trocar de modo
  mpu_wr_mg_reg(MAG_CNTL_1, 0x16); //??? Mag. Modo Continuo, 100Hz e 16
     bits
  delay(100);                       //espera trocar de modo
}
```

B.4 calibração do acelerômetro

```
//acel_calibra Obtem os valores necessarios para calibrar o acelerometro
//v_out[6] = [   ax_x_cima,
//               ax_x_baixo,
//               ay_y_cima,
//               ay_y_baixo,
//               az_z_cima,
//               az_z_baixo,
//             ]
//
//   % o seguinte algoritmo do matlab realiza a calibra  o
//    accel_offset(1)=(ax_x_cima + ax_x_baixo)/2;
//    accel_offset(2)=(ay_y_cima + ay_y_baixo)/2;
//    accel_offset(3)=(az_z_cima + az_z_baixo)/2;
//
//    accel_scale(1)=1/((ax_x_cima - ax_x_baixo)/2);
//    accel_scale(2)=1/((ay_y_cima - ay_y_baixo)/2);
//    accel_scale(3)=1/((az_z_cima - az_z_baixo)/2);
//
//    ax_calibrado=(ax-accel_offset(1))*accel_scale(1);
//    ay_calibrado=(ay-accel_offset(2))*accel_scale(2);
//    az_calibrado=(az-accel_offset(3))*accel_scale(3);
byte acel_calibra(int* v_out, byte prn){

  int vt[7];
  long aux[6];

```

```
27    char *msg1="Erro Who am I = ";
28    char *msg_calibr[6] = { "X para cima",
29                            "X para baixo",
30                            "Y para cima",
31                            "Y para baixo",
32                            "Z para cima",
33                            "Z para baixo"};
34
35    byte whoami;
36    int qtd_medidas=30;
37
38    ser_str("Calibra  o do acelerometro.\n");
39    ser_str("Posicione o sensor de acordo com as instru  es.\n");
40    ser_str("Ao posicionar, pressione qualquer bot o para realizar a medida
          .\n");
41
42    mpu_acorda();
43    mpu_config();
44    whoami=mpu_whoami();
45    if (whoami != MPU9250_WHO){
46  //    lcd_str(1,0,msg1); lcd_dec16unz(1,16,whoami);
47      ser_str(msg1);     ser_dec16unz(whoami);  ser_crlf(1);
48      delay(1000);
49      return FALSE;
50    }
51
52    delay(1000);
53
54  //  lcd_apaga_lin(1);
55  //  lcd_apaga_lin(2);
56  //  lcd_apaga_lin(3);
57  //  lcd_str(1,0,"Calib do acel: ");
58
59    int quit=FALSE;
60    byte x=0;
61    int cnt = qtd_medidas;
62    for(int i = 0 ; i < 6 ; i++){
```

```
63      //Inicia coleta de dados
64
65      ser_str(msg_calibr[i]); ser_crlf(1);
66
67      //espera usuario apertar botao para medir, ou terminar mandando X pelo
            serial
68      while(TRUE){
69        while (TRUE){
70          if (seri_tira(&x)==FALSE)   break;
71          if (x=='x' || x=='X')       quit=TRUE;
72        }
73        if ( sw_tira(&x) == TRUE) break;
74        if(quit == TRUE) break;
75      }
76
77      if(quit == TRUE) return FALSE;
78
79      //inicia as leituras
80      mpu_int();
81
82      //faz algumas leituras para estabilizar e evitar o movimento de
           apertar o bot o
83      int l=160;
84      while(l--){
85        while (mpu_dado_ok == FALSE);
86        mpu_rd_ac_tp_gi(vt);  //Ler MPU
87      }
88
89      aux[0]=aux[1]=aux[2]=0;
90      cnt = qtd_medidas;
91
92      while(cnt--){
93          while (mpu_dado_ok == FALSE);   //Agaurdar MPU a 100 Hz (10 ms)
94          mpu_dado_ok=FALSE;
95
96          mpu_rd_ac_tp_gi(vt);  //Ler MPU
97
```

```
98          aux[0]+=vt[0];
99          aux[1]+=vt[1];
100         aux[2]+=vt[2];
101
102       }
103
104       aux[0]/=qtd_medidas;
105       aux[1]/=qtd_medidas;
106       aux[2]/=qtd_medidas;
107
108       v_out[i] = (int)aux[(int)floor(i/2)];
109
110       mpu_des_int();
111   }
112
113   ser_str("dados coletados.\n");
114   if(prn){
115     ser_str("Valores que ser o usados na calibragem do acelerometro:");
          ser_crlf(1);
116     ser_str("ax:");  ser_crlf(1);
117     ser_dec16(v_out[0]); ser_crlf(1);
118     ser_dec16(v_out[1]); ser_crlf(1);
119
120     ser_str("ay:");  ser_crlf(1);
121     ser_dec16(v_out[2]); ser_crlf(1);
122     ser_dec16(v_out[3]); ser_crlf(1);
123
124     ser_str("az:");  ser_crlf(1);
125     ser_dec16(v_out[4]); ser_crlf(1);
126     ser_dec16(v_out[5]); ser_crlf(1);
127   }
128   return TRUE;
129 }
```

B.5 calibração do magnetômetro

```
1  // mag_calibra obtem os valores necessarios para calibrar o magnetometro
2  // asa[3] = [asax, asay, asaz] s o os ajustes de sensibilidade
3  // h_extr[6] = [hx_min, h_max, hy_min, h_max, hz_min, hz_max]
4  // Esses s o todos os dados para realizar uma calibra o de
       magnet metro simples no p s processamento:
5  //
6  // O algoritmo de matlab para calibrar   :
7  //     % Hard Iron - Remover offset
8  //     hx_off=(hx_max + hx_min)/2;
9  //     hy_off=(hy_max + hy_min)/2;
10 //     hz_off=(hz_max + hz_min)/2;
11 //
12 //     % Soft Iron - Corrigir a escala
13 //     hx_avg_delta=(hx_max - hx_min)/2;
14 //     hy_avg_delta=(hy_max - hy_min)/2;
15 //     hz_avg_delta=(hz_max - hz_min)/2;
16 //
17 //     avg_h_delta=(hx_avg_delta + hy_avg_delta + hz_avg_delta)/3;
18 //
19 //     hx_sc=avg_h_delta/hx_avg_delta;
20 //     hy_sc=avg_h_delta/hy_avg_delta;
21 //     hz_sc=avg_h_delta/hz_avg_delta;
22 //
23 //     h_off = [ hx_off hy_off hz_off ];
24 //     h_sc  = [
25 //                 hx_sc  0       0       ;
26 //                 0      hy_sc   0       ;
27 //                 0      0       hz_sc
28 //             ];
29 //    magDataCalibrated = (magData - h_off)*h_sc;
30 byte mag_calibra(byte* asa, int* h_extr, byte prn ){
31
32   byte mag_st,who;
33   int vetor[3];
34
35   h_extr[0] = 32767;
36   h_extr[1] = -32768;
```

```
37    h_extr[2] = 32767;
38    h_extr[3] = -32768;
39    h_extr[4] = 32767;
40    h_extr[5] = -32768;
41
42
43    mpu_config();          //MPU configurar
44    mpu_mag_config();      //MAG configurar
45
46    who = mag_whoami();
47    if (who != MAG_WHO){
48  //    lcd_str(1,13,"whoami nao encontrado");  //MPU N o respondendo
49      ser_str("Magnet metro n o encontrado\n");
50      return FALSE;
51    }
52
53    ser_str("Calibra o do Magnet metro.\n\n");
54    ser_str("Quando a calibra o iniciar, movimente a caixa preta
          lentamente em todas as dire es poss veis.\n");
55    ser_str("O procedimento encerrado ao pressionar qualquer bot o da
          caixa preta.\n");
56    ser_str("Inicia em 5 segundos...\n");
57
58  //  lcd_apaga_lin(1);
59  //  lcd_apaga_lin(2);
60  //  lcd_apaga_lin(3);
61  //  lcd_str(1,0,"Calibr mag.");
62  //  lcd_str(2,0,"Inicia em 5 seg...");
63
64    delay(5000);
65    ser_str("Coleta de dados iniciou...\n");
66
67    // Iniciar coleta
68
69    // coleta o ajuste de sensibilidade
70    mpu_mag_rd_rom(asa);
71
```

```
72    // Habilitar interrup  o MPU (Dado Pronto)
73    mpu_sample_rt(SAMPLE_RT_100Hz);
74    mpu_int();
75
76  //  lcd_str(1,0,"Coletando dados...");
77  //  lcd_str(2,0,"Rotacione lentamente");
78  //  lcd_str(3,0,"e finalize com botao");
79    while(TRUE){
80      while (mpu_dado_ok == FALSE);   //Agaurdar MPU a 100 Hz (10 ms)
81      mpu_dado_ok=FALSE;
82      mag_st=mpu_rd_mg_out(vetor);
83
84      if (mag_st==1){        //Tudo certo
85  //      lcd_char(2, 5,'1');
86  //      lcd_char(2,15,'0');
87
88       //atualiza valores maximos e minimos
89       if(vetor[0] < h_extr[0]) h_extr[0] = vetor[0];
90       if(vetor[0] > h_extr[1]) h_extr[1] = vetor[0];
91       if(vetor[1] < h_extr[2]) h_extr[2] = vetor[1];
92       if(vetor[1] > h_extr[3]) h_extr[3] = vetor[1];
93       if(vetor[2] < h_extr[4]) h_extr[4] = vetor[2];
94       if(vetor[2] > h_extr[5]) h_extr[5] = vetor[2];
95
96     }
97     else if(mag_st==0){    //Dado n  o pronto
98  //      lcd_char(2, 5,'0');
99  //      lcd_char(2,15,'0');
100    }
101
102    else if(mag_st==2){    //Sensor Overflow
103 //      lcd_char(2, 5,'1');
104 //      lcd_char(2,15,'1');
105    }
106
107     //if (sw_tira(&who))     break;
```

```
108     if (fim_qqtec_x() == TRUE)  break;  //qq Tecla o letra x para
            finalizar
109   }
110
111   mpu_des_int();
112 //  lcd_apaga_lin(1);
113 //  lcd_apaga_lin(2);
114 //  lcd_apaga_lin(3);
115 //  lcd_str(2,0,"Fim da coleta");
116   ser_str("Fim da coleta de dados.\n");
117
118   if(prn){
119     ser_str("Valores de calibragem do magnet metro:");  ser_crlf(1);
120     ser_str("ASA:");  ser_crlf(1);
121     ser_dec8u(asa[0]);    ser_crlf(1);
122     ser_dec8u(asa[1]);    ser_crlf(1);
123     ser_dec8u(asa[2]);    ser_crlf(1);
124     ser_str("Hx:");  ser_crlf(1);
125     ser_dec16(h_extr[0]); ser_crlf(1);
126     ser_dec16(h_extr[1]); ser_crlf(1);
127     ser_str("Hy");  ser_crlf(1);
128     ser_dec16(h_extr[2]); ser_crlf(1);
129     ser_dec16(h_extr[3]); ser_crlf(1);
130     ser_str("Hz");  ser_crlf(1);
131     ser_dec16(h_extr[4]); ser_crlf(1);
132     ser_dec16(h_extr[5]); ser_crlf(1);
133   }
134
135   return TRUE;
136
137 }
```

B.6 Self-Test do IMU

```
1 // MPU: Realizar Self-Test (ST), prn = imprimir resultados?
2 // Retorna: TRUE  se passou no teste
```

```
3  //          FALSE se falhou no teste
4  // baseado no documento AN-MPU-9250A-03 MPU-9250 Accel Gyro and Compass
       Self-Test Implementation v1 0_062813.pdf
5  byte mpu_self_test_2(byte prn = FALSE) {
6
7    float ST_OPT[6];           // self test value de f brica
8    long  sum[6];              // registrador auxilar para calcular a soma
9    int   aux[6];              // registrador auxiliar para leituras
10   int   mediaNoST[6];        // media das 200 medidas sem o self test
11   int   mediaST[6];          // media das 200 medidas com o self test
12   int   selfTestResponse[6]; // self test response (medias com self test
         - medias sem self test)
13   byte  ST_CODE[6];          // factory self test code (usado para
         calcular o self test value)
14   float porcentagens[6];     // porcentagem ao dividir selfTestResponse
         por ST_OPT (determina se st passou)
15
16   int   qtd = 200;           // quantos valores coletar para calcular a
         m dia
17   bool  passou = true;       // valor que retorna no final da fun  o (
         passou ou n o?)
18
19   /* 3.0 Procedimento */
20   // 1. Configura es necess rias para o self test
21   mpu_wr(SMPLRT_DIV, 0x00);
22   mpu_wr(CONFIG, 0x02);        // Girosc pio: mudando DLPF para config 2.
         taxa do giroscopio 1 kHz e DLPF  92 Hz
23   mpu_wr(ACCEL_CONFIG_2, 0x02); // Acelerometro: taxa do acelerometro 1
         kHz banda 92 Hz
24
25   //configura escalas do giroscopio e acelerometro para o recomendado para
         o self test
26   // +/- 2g e +/-250gr/seg
27   mpu_escalas(GIRO_FS_250,ACEL_FS_2G);
28   delay(250); //Aguardar cofigura  o estabilizar
29
```

```
30  // 2. Com o self test desligado, ler 200 medidas do giroscpio e
        acelerometro e armazenar as m dias
31  // em mediaNoST [ax, ay, az, gx, gy, gz]
32  for (int i = 0; i<6; i++)   sum[i]=0; //Zerar acumulador
33  mpu_int(); // Habilitar interrup o
34  mpu_dado_ok=FALSE;
35  //200 medidas sem self test
36  for (int i=0; i<qtd; i++){
37    while(mpu_dado_ok == FALSE);
38    mpu_dado_ok = FALSE;
39    mpu_rd_ac_gi(aux);
40    for (int j=0; j<6; j++) sum[j] += aux[j];
41  }
42  // Calcular as m dias sem o self test
43  for (int i=0; i<6; i++) mediaNoST[i] = sum[i]/qtd;
44
45  // 3. habilitar self test nos 3 eixos do acelerometro e giroscpio
46  mpu_wr(ACCEL_CONFIG, 0xE0);
47  mpu_wr(GYRO_CONFIG,  0xE0);
48
49  // 4. delay para as oscila  es estabilizarem
50  delay(25);
51
52  // 5. Com o self test ligado, ler 200 medidas do giroscpio e
        acelerometro e armazenar as m dias
53  // em mediaST [ax, ay, az, gx, gy, gz]
54  for (int i = 0; i<6; i++)   sum[i]=0; //Zerar acumulador
55  //200 medidas com self test
56  for (int i=0; i<qtd; i++){
57    while(mpu_dado_ok == FALSE);
58    mpu_dado_ok = FALSE;
59    mpu_rd_ac_gi(aux);
60    for (int j=0; j<6; j++) sum[j] += aux[j];
61  }
62  for (int i=0; i<6; i++) mediaST[i] = sum[i]/qtd;
63
64  // 6. calculando as respostas para o self test
```

```
65  for(int i = 0 ; i < 6 ; i++){
66    selfTestResponse[i] = mediaST[i] - mediaNoST[i];
67  }
68
69  /* 3.1 Configurar giro e acel para opera  o normal */
70  mpu_wr( ACCEL_CONFIG, 0x00);
71  mpu_wr( GYRO_CONFIG,  0x00);
72  delay(25);  // Delay a while to let the device stabilize
73  //acertar as escalas
74  mpu_escalas(GIRO_FS_250,ACEL_FS_8G);
75  //tem que rodar o mpu_config denovo?
76
77  /* 3.2 crit rios para passar no self test */
78
79  // 1. lendo factory Self-Test Code do giroscopio e acelerometro
80  // X-axis accel self-test
81  ST_CODE[0] = mpu_rd(SELF_TEST_X_ACCEL);
82  // Y-axis accel self-test
83  ST_CODE[1] = mpu_rd(SELF_TEST_Y_ACCEL);
84  // Z-axis accel self-test
85  ST_CODE[2] = mpu_rd(SELF_TEST_Z_ACCEL);
86  // X-axis gyro self-test
87  ST_CODE[3] = mpu_rd(SELF_TEST_X_GYRO);
88  // Y-axis gyro self-tes
89  ST_CODE[4] = mpu_rd(SELF_TEST_Y_GYRO);
90  // Z-axis gyro self-test
91  ST_CODE[5] = mpu_rd(SELF_TEST_Z_GYRO);
92
93  // 2. calculando factory self-test value a partir do factory self test
       code
94  // FT[Xa]
95  ST_OPT[0] = (float)(2620/1<<GIRO_FS_250)*(pow(1.01 ,((float)ST_CODE[0] -
        1.0) ));
96  // FT[Ya]
97  ST_OPT[1] = (float)(2620/1<<GIRO_FS_250)*(pow(1.01 ,((float)ST_CODE[1] -
        1.0) ));
98  // FT[Za]
```

```
99    ST_OPT[2] = (float)(2620/1<<GIRO_FS_250)*(pow(1.01 ,((float)ST_CODE[2] -
          1.0) ));
100   // FT[Xg]
101   ST_OPT[3] = (float)(2620/1<<GIRO_FS_250)*(pow(1.01 ,((float)ST_CODE[3] -
          1.0) ));
102   // FT[Yg]
103   ST_OPT[4] = (float)(2620/1<<GIRO_FS_250)*(pow(1.01 ,((float)ST_CODE[4] -
          1.0) ));
104   // FT[Zg]
105   ST_OPT[5] = (float)(2620/1<<GIRO_FS_250)*(pow(1.01 ,((float)ST_CODE[5] -
          1.0) ));
106
107   for(int i = 0 ; i < 6 ; i++){
108     porcentagens[i] = (float)selfTestResponse[i]/ST_OPT[i];
109   }
110
111   // 3. Determinando a condi  o de passar no teste
112   // X-gyro: (GXST / GXST_OTP) > 0.5
113   // Y-gyro (GYST / GYST_OTP) > 0.5
114   // Z-gyro (GZST / GZST_OTP) > 0.5
115   // X-Accel 0.5 < (AXST / AXST_OTP) < 1.5
116   // Y-Accel 0.5 < (AYST / AYST_OTP) < 1.5
117   // Z-Accel 0.5 < (AZST / AZST_OTP) < 1.5
118
119   for(int i = 0 ; i < 3 ; i++){
120     //testando giro
121     if(porcentagens[i+3] <= 0.5) passou = false;
122     //testando acel
123     if(!((porcentagens[i] > 0.5) && (porcentagens[i] < 1.5))) passou =
          false;
124   }
125
126
127
128   return passou;
129
130 }
```

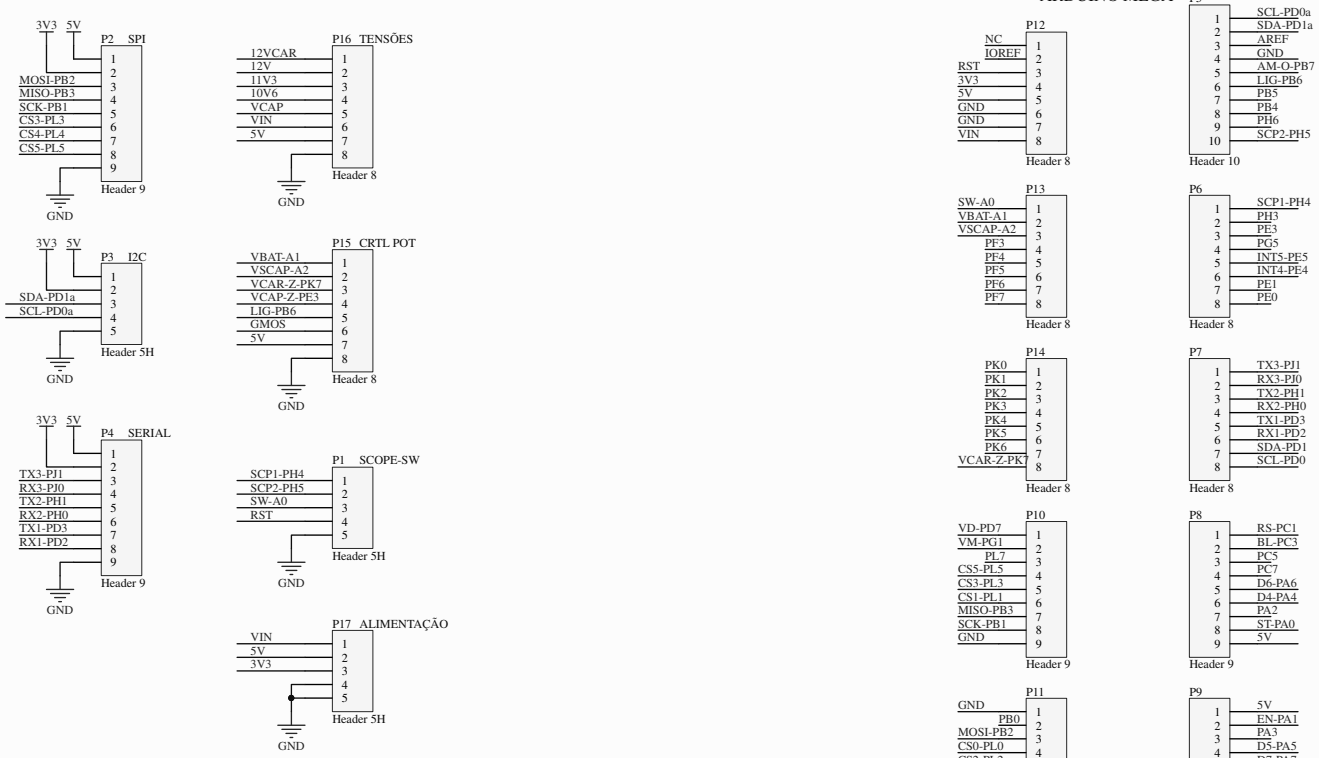
B.7 Self-Test do Magnetômetro

```
/ MAG: Realizar Self-Test (ST), prn = imprimir resultados?
// Retorna: TRUE  se passou no teste
//          FALSE se falhou no teste
// vetor[ hx hy hz ] --> espa o para 3 inteiros
byte mpu_mag_self_test(int *vetor, byte prn) {
  byte vet[6],ok;
  int aux[3];
  byte asa[3];

  mpu_wr_mg_reg(MAG_CNTL_1, 0x00);  //(1) MODE=0, Magnetometro Power Down
  delay(100);
  mpu_wr_mg_reg(MAG_ASTC, 0x64);    //(2) SELF=1
  delay(100);
//  mpu_wr_mg_reg(MAG_CNTL_1, 0xC);   //(3) BIT=1 (16 bits) e MODE=8 (self
    test)
  mpu_wr_mg_reg(MAG_CNTL_1, 0x18);   //(3) BIT=1 (16 bits) e MODE=8 (self
     test)
  delay(100);

  byte r = 0;
  r = mpu_rd_mg_reg(MAG_ST1) &1;
  while(r&1 == 0){
    r = mpu_rd_mg_reg(MAG_ST1) &1;
  }


  mpu_rd_mg_blk(MAG_XOUT_L, vet, 6);
  aux[0] = (int)((int)(vet [1] << 8) | vet[0]);    //Montar Mag X
  aux[1] = (int)((int)(vet [3] << 8) | vet[2]);    //Montar Mag Y
  aux[2] = (int)((int)(vet [5] << 8) | vet[4]);    //Montar Mag Z
  mpu_wr_mg_reg(MAG_ASTC, 0);       //(2) SELF=0
  mpu_wr_mg_reg(MAG_CNTL_1, 0x00);  //(1) MODE=0, Magnetometro Power Down

  // ASA: Ajuste de sensibilidade
  mpu_mag_rd_rom(asa);    //asa[0]=ASAx, asa[1]=ASAy, asa[2]=ASAz,
```

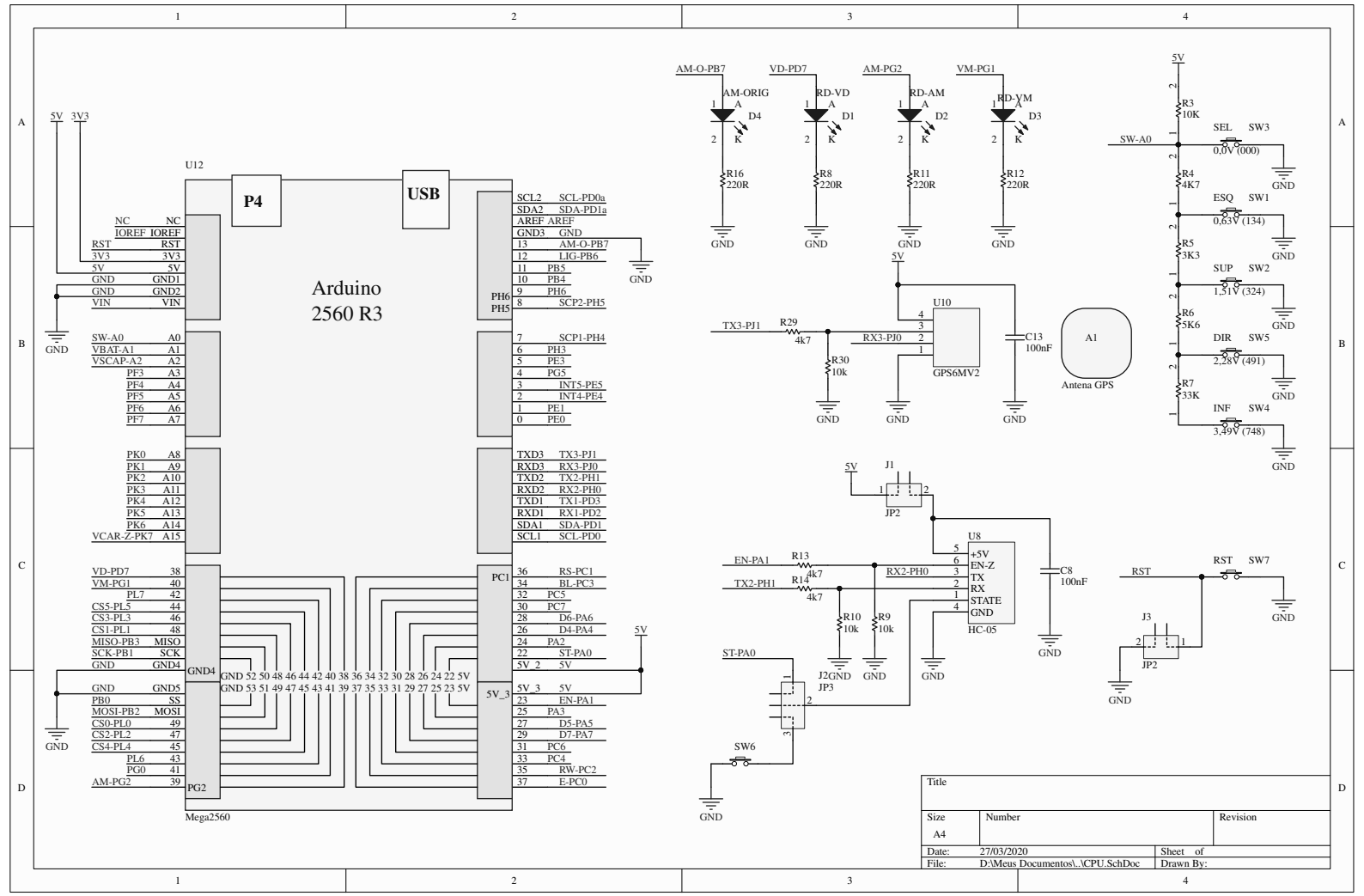
```
34
35
36    vetor[0]=(float)aux[0]*( 1+((float)asa[0]-128)/256.);
37    vetor[1]=(float)aux[1]*( 1+((float)asa[1]-128)/256.);
38    vetor[2]=(float)aux[2]*( 1+((float)asa[2]-128)/256.);
39
40
41    ok=TRUE;
42    if ( (vetor[0] <=  -200) || (vetor[0] >=  200))  ok=FALSE;  //hx
43    if ( (vetor[1] <=  -200) || (vetor[1] >=  200))  ok=FALSE;  //hy
44    if ( (vetor[2] <= -3200) || (vetor[2] >= 3200))  ok=FALSE;  //hz
45
46    return ok;
47  }
```

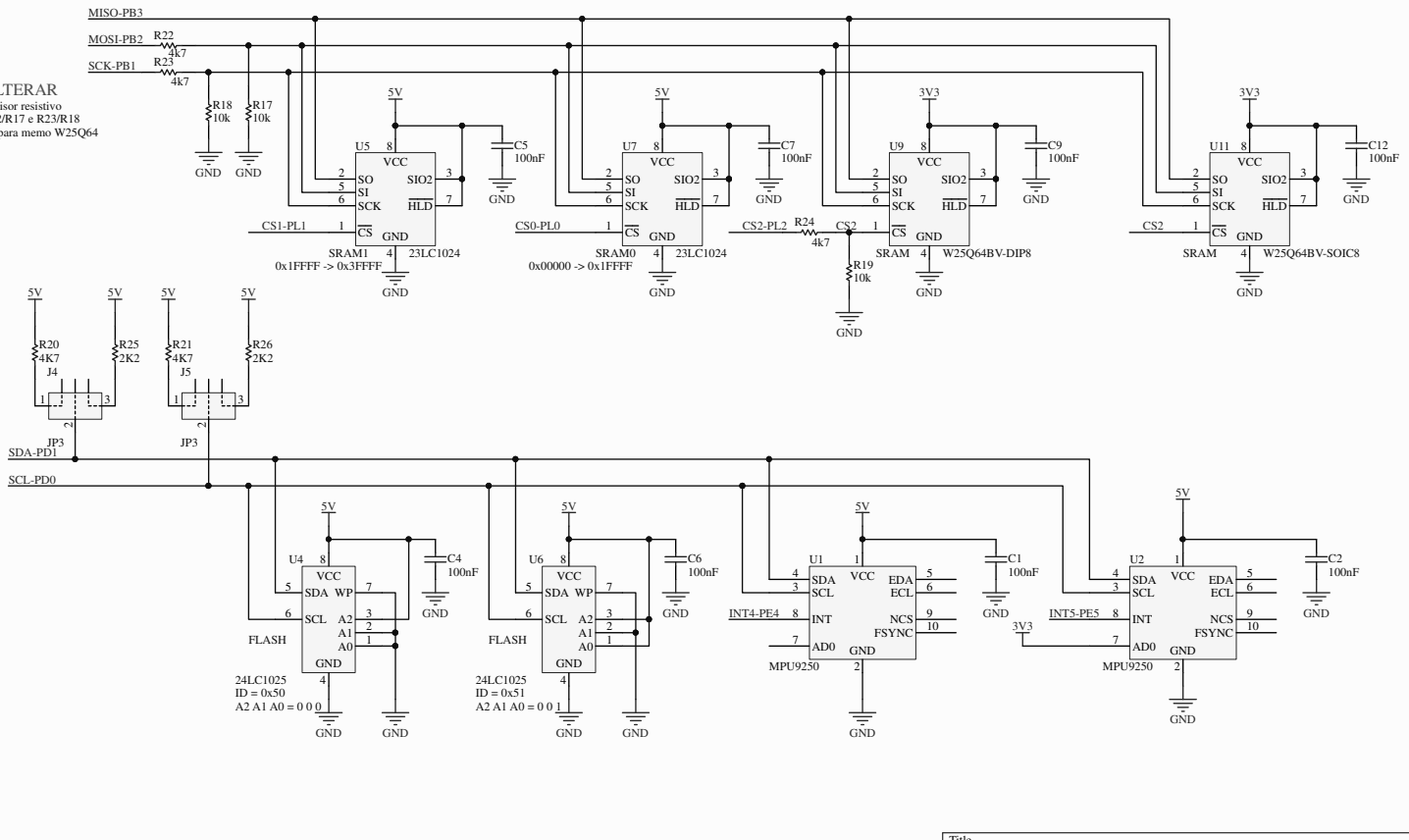
Anexo I

Esquematicos

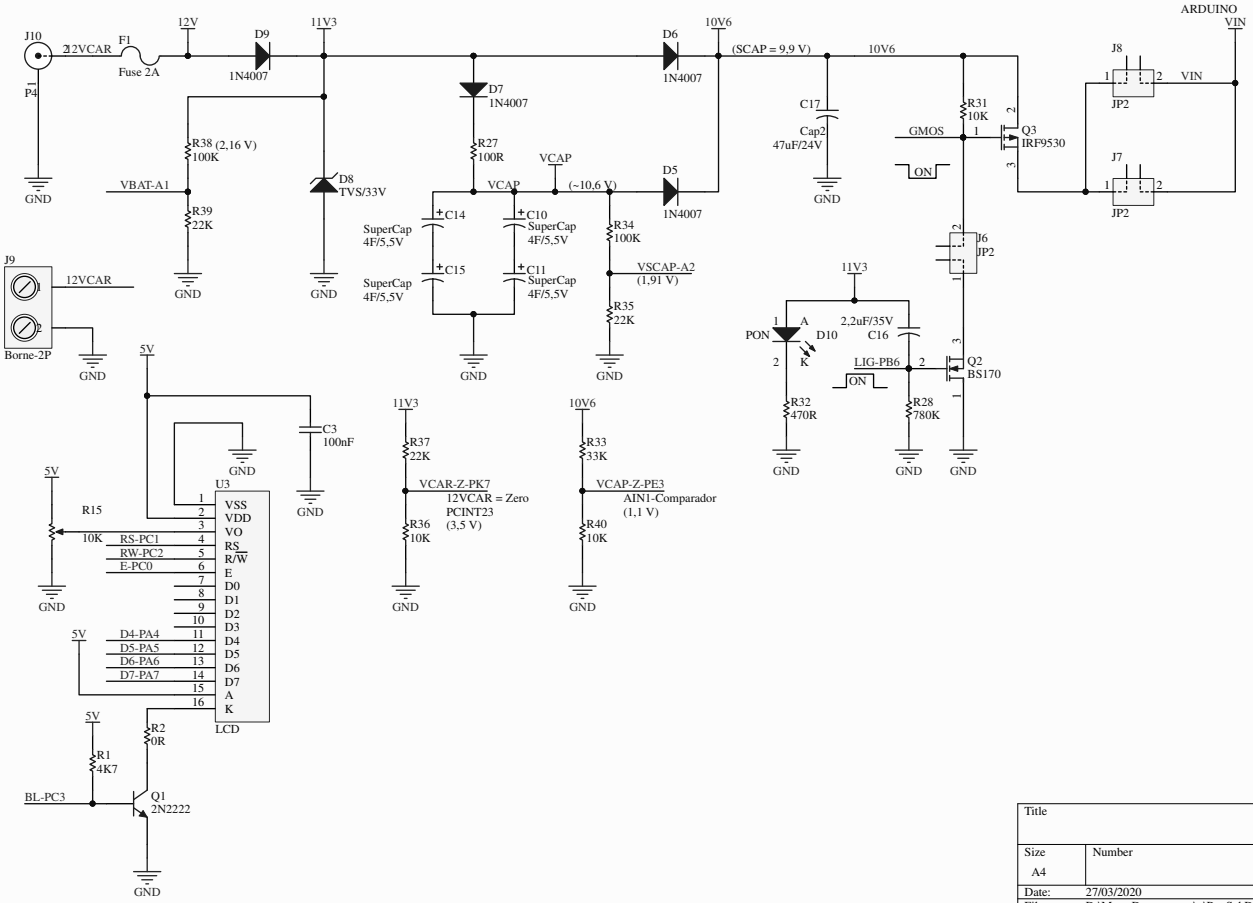


Title		
Size: A4	Number	Revision
Date: 27/03/2020		Sheet of
File: D:\Meus Documentos\..\Conect.SchDoc		Drawn By:


U12
P4
USB
Arduino
2560 R3
Mega2560
PG2
PC1
GND4
5V_3
AM-O-PB7
VD-PD7
AM-PG2
VM-PG1
AM-ORIG
RD-VD
RD-AM
RD-VM
D4
D1
D2
D3
R16 220R
R8 220R
R11 220R
R12 220R
U10
GPS6MV2
A1
Antena GPS
C13 100nF
TX3-PJ1
RX3-PJ0
R29 4k7
R30 10k
U8
HC-05
C8 100nF
EN-PA1
TX2-PH1
RX2-PH0
R13 4k7
R14 4k7
R10 10k
R9 10k
J1
JP2
J2
JP3
J3
ST-PA0
SW6
SW-A0
R3 10K
R4 4K7
R5 3K3
R6 5K6
R7 33K
SEL SW3
0,0V (000)
ESQ SW1
0,63V (134)
SUP SW2
1,51V (324)
DIR SW5
2,28V (491)
INF SW4
3,49V (748)
RST SW7
Title
Size A4
Number
Revision
Date: 27/03/2020
Sheet of
File: D:\Meus Documentos\..\CPU.SchDoc
Drawn By:



Title		
Size: A4	Number	Revision
Date: 27/03/2020		Sheet of
File: D:\Meus Documentos\..\Memo.SchDoc		Drawn By:



Title		
Size: A4	Number	Revision
Date: 27/03/2020		Sheet of
File: D:\Meus Documentos\..\Pot.SchDoc		Drawn By: