UNIVERSIDADE PAULISTA

RAFAEL JOSÉ MOTA OCARIZ

SISTEMA DE DETERMINAÇÃO DE VERTICALIDADE UTILIZANDO UNIDADE DE MEDIDAS INERCIAIS

SÃO JOSÉ DOS CAMPOS

2018

RAFAEL JOSÉ MOTA OCARIZ

SISTEMA DE DETERMINAÇÃO DE VERTICALIDADE UTILIZANDO UNIDADE DE MEDIDAS INERCIAIS

Trabalho de conclusão de curso apresentado como parte das atividades para obtenção do título de Bacharel do curso de Engenharia da Computação da Universidade Paulista, UNIP, no campus de São José dos Campos.

Orientador: Prof. Dr. Rogério Moreira Cazo.

SÃO JOSÉ DOS CAMPOS

2018

RAFAEL JOSÉ MOTA OCARIZ

SISTEMA DE DETERMINAÇÃO DE VERTICALIDADE UTILIZANDO UNIDADE DE MEDIDAS INERCIAIS

Trabalho de conclusão de curso apresentado como parte das atividades para obtenção do título de Bacharel do curso de Engenharia da Computação da Universidade Paulista, UNIP, no campus de São José dos Campos.

Aprovado em:

BANCA EXAMINADORA

_____ , ___ / ___ / _____

Prof. Mestre Pedro Carlos da Silva Euphrasio

Universidade Paulista - UNIP

_____ , ___ / ___ / _____

Prof. Dr. Rogério Moreira Cazo

Universidade Paulista - UNIP

_____ , ___ / ___ / _____

Prof. Mestre André Yoshimi Kusumoto

Universidade Paulista - UNIP

DEDICATÓRIA

Dedico este trabalho à minha família e aos meus amigos que sempre estiveram ao meu lado.

AGRADECIMENTOS

Agradeço primeiramente a DEUS por ser o pilar principal na minha vida, também a minha mãe, irmã e colegas que me apoiaram durante esta jornada e ao professor Rogério Moreira Cazo pelas suas orientações, incentivos e colaborações para a conclusão deste trabalho.

“O sucesso é ir de fracasso em fracasso sem perder o entusiasmo.”

Winston Churchill

RESUMO

Determinar a verticalidade é essencial em foguetes, robôs, aviões, em sistemas de mapeamento topográfico, etc. Este trabalho apresenta uma solução capaz de realizar tal determinação. Este projeto foi idealizado a partir da necessidade de se determinar a verticalidade das pernas de um robô durante seu movimento de caminhada. Para uma abordagem inicial só foi determinada a verticalidade em um corpo em repouso. Para atender a este objetivo utilizou-se conhecimentos de ângulos de Euler, aplicação de acelerômetros, engenharia de software e computação embarcada. Foram traçados objetivos específicos que auxiliaram a guiar o desenvolvimento do projeto. Foi realizado um levantamento a respeito das opções existentes para cada componente sensível e selecionado o que melhor atendia às necessidades do projeto. Foram feitos dois programas de computador para o projeto, sendo que um foi utilizado durante os testes e outro é o programa final projetado para resolver o problema proposto. Para fins de visualização foi utilizada uma tela com tecnologia de interação por toque como interface homem máquina. Durante os testes os sensores foram devidamente caracterizados e o software e equações deduzidas foram testados de diversas formas. Por fim concluiu-se que o projeto atendeu a todos os objetivos específicos e solucionou o problema sugerido no tema, que é determinar a verticalidade.

Palavras-chave: verticalidade, acelerômetros, ângulos, Euler, caracterização.

ABSTRACT

Determining verticality is essential in rockets, robots, airplanes, topographic mapping systems, etc. This work presents a solution capable of making such determination. This project was conceived from the need to determine the verticality of the legs of a robot during its walking movement. For an initial approach only, the verticality was determined in a body at rest. To meet this goal, we used knowledge of Euler angles, application of accelerometers, software engineering and embedded computing. Specific objectives were set out to help guide the development of the project. A survey was made regarding the options available for each sensitive component and selected which best met the needs of the project. Two computer programs were made for the project, one was used during the tests and another is the final program designed to solve the proposed problem. For visualization purposes, a touch screen technology was used as the human machine interface. During the tests the sensors were properly characterized, and the software and equations deduced were tested in several ways. Finally, it was concluded that the project met all the specific objectives and solved the problem suggested in the theme, which is to determine the verticality.

Keywords: verticality, accelerometers, angles, Euler, characterization.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Um robô humanoide possui inúmeros sistemas de controle internos, um desses é o controle de postura, responsável por manter o robô em pé e estável e para isso é necessário controlar a atitude de várias peças que o compõe. Ainda assim, antes de projetar um sistema que controle a atitude é necessário saber como determinar a mesma.

A determinação de atitude tem por objetivo informar a um usuário o sentido e direção de apontamento de um objeto. Essa determinação, ou medição, é feita por meio de cálculos realizados sobre valores obtidos por blocos de sensores conhecidos como IMU (Unidade de Medidas Inerciais). Atualmente existem sistemas que determinam a verticalidade por meio de laser, porém com aplicabilidade distinta da sugerida para o projeto em questão.

O sistema a laser citado não informa a verticalidade do objeto, o dado obtido por meio de sua medição é se o corpo a que ele está fixado está ou não em posição vertical. Diante da necessidade levantada de determinar com precisão a atitude de um objeto atrelado a um fato secundário de que o mesmo deve possuir um baixo custo de produção. Outra necessidade apresentada é que, independente da atitude do objeto medido, o que o sistema de um robô precisa é da sua variação com a vertical, no caso a verticalidade do mesmo. Portanto, buscou-se ao longo deste trabalho responder ao seguinte problema: é possível determinar a verticalidade de um objeto mantendo o baixo custo e com precisão aceitável?

Para resolver o problema proposto foi traçado como objetivo geral projetar um sistema capaz de determinar a verticalidade de um objeto utilizando dados coletados por uma unidade de medidas inerciais. A fim de atingir o mesmo foram traçados os seguintes objetivos específicos:

- Especificar uma IMU de baixo custo capaz de atender ao projeto;
- Elaborar um código para Arduino que faça a aquisição de dados dos sensores para caracterização dos mesmos;
- Realizar os ensaios de caracterização dos sensores;
- Definir as equações necessárias para converter os dados obtidos com os sensores nos ângulos de variação do objeto medido com a vertical;
- Modelar e programar o software responsável por calcular a verticalidade do objeto medido;
- Programar interface homem máquina a ser impressa numa tela LCD (*Liquid Crystal Display*) com suporte à interação via toque na tela para apresentação de resultados.

Diante do ambiente competitivo por criar um sistema robótico com movimentos mais orgânicos e precisos e da necessidade de diversos sistemas de determinação de atitude distribuídos por um robô surge à proposta deste trabalho de apresentar uma solução barata e que forneça dados de verticalidade a fim de facilitar os cálculos posteriores em supersistemas que os utilizarão como entrada.

Para o desenvolvimento do presente trabalho foram utilizadas pesquisas bibliográficas, foi elaborada a modelagem e implementação de software embarcado e foram realizados ensaios em laboratório. A pesquisa bibliográfica baseou-se em livros, publicações científicas e artigos da internet na área de sensores, engenharia de controle e cálculo. A modelagem de software foi desenvolvida em sua totalidade dentro do ambiente fornecido pelo software *Enterprise Architect* e os ensaios em laboratório foram realizados utilizando os equipamentos existentes no LICS (Laboratório de Identificação, Navegação, Controle e Simulação) presente no Instituto de Aeronáutica e Espaço (IAE).

1.1. Trabalhos relacionados

Foram selecionados alguns trabalhos realizados em áreas correlatas ou que em algum ponto do mesmo exista uma atividade relacionada ao projeto aqui exposto.

O primeiro trabalho relacionado é o publicado por Ferreira et. al. (2008), assemelhando-se a este trabalho no uso de ângulos de Euler como ponto de partida para execução dos cálculos de determinação de atitude. Outro trabalho relacionado, porém com uma abordagem distinta, é o de Kalantari et. al. (2010). No decorrer de seu trabalho ele determina a vertical a partir de imagens obtidas por câmeras. Por fim um terceiro e último trabalho relacionado é a dissertação de mestrado de Oliveira (2013) onde ele calcula a atitude de um nanossatélite usando uma terceira técnica por meio de quatérnios.

1.2. Organização do documento

Com a finalidade de apresentar todos os pontos pertinentes do projeto, este documento foi dividido em cinco capítulos. O primeiro destes e no caso este é responsável por introduzir e contextualizar o tema proposto, apresentar os objetivos e descrever a metodologia utilizada no desenvolvimento da pesquisa. O segundo capítulo traz toda a fundamentação teórica necessária para que o leitor entenda os conceitos utilizados durante o trabalho e conheça os equipamentos utilizados para montagem do protótipo e desenvolvimento das atividades. No terceiro capítulo o leitor encontrará informações mais detalhadas dos materiais utilizados no projeto, assim como os motivos que levaram à escolha destes materiais. Ainda neste capítulo será exposta com maiores detalhes toda a metodologia de desenvolvimento utilizada, assim como a modelagem do software embarcado desenvolvida. No quarto capítulo estão descritos minuciosamente os testes realizados e a análise e aplicação dos resultados dos mesmos de forma a atingir os objetivos traçados. No quinto e último capítulo o autor apresenta a conclusão obtida após o seguimento de todas as atividades do projeto. Neste capítulo final também foram apontadas algumas sugestões de trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

É de suma importância para um trabalho acadêmico que este esteja bem embasado. Partindo deste princípio, durante este capítulo serão introduzidos de forma clara e objetiva conceitos cruciais para compreensão do desenvolvimento e do produto final.

2.1. Verticalidade

Verticalidade é definida pelo dicionário da língua portuguesa como a qualidade daquilo que é vertical, ou seja, a qualidade do alinhamento com a vertical de um corpo. Então, para determinar-se a verticalidade, ou seja, o quão alinhado com a vertical o corpo medido está, deve-se antes determinar o que será compreendido como vertical.

2.1.1. Vertical

Wikipédia (2013) descreve em seu artigo a vertical como a linha imaginária posicionada no centro do campo de visão do observador formada pela gravidade. Ele ilustra esta linha imaginária como a reta formada por uma corda fixada a um prumo, muito utilizado por pedreiros. Quando este se estabiliza a sua corda fica em posição perfeitamente vertical.

2.2. Unidade de medidas inerciais

Siciliano & Khatib (2008) descreve uma unidade de medidas inerciais (IMU) como um dispositivo formado por diversos sensores como acelerômetros, girômetros e magnetômetros com a finalidade de estimar a posição relativa, velocidade, orientação e aceleração de um veículo em movimento. Como para esta primeira abordagem foi definido que será realizada a determinação da verticalidade do corpo em repouso apenas os dados acelerométricos serão utilizados.

2.2.1. Acelerômetros

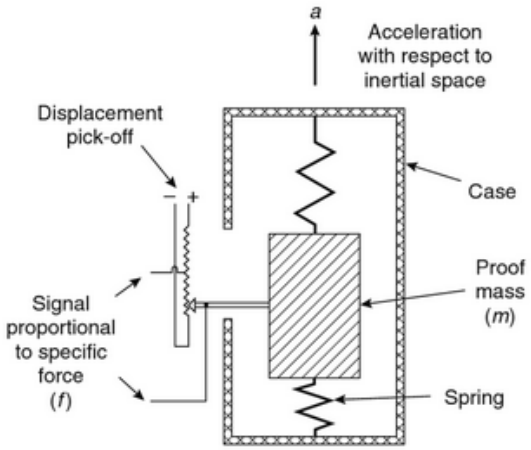
Segundo Aggarwal et. al. (2010), acelerômetros são sensores que medem a força específica em um plano de referência, que pode ser utilizado para determinar a aceleração de um corpo em movimento.

Baseando-se na definição de acelerômetro apresentada pode-se concluir que este sensor tem a capacidade de indicar ao usuário a aceleração da gravidade atuante no corpo ou uma componente da mesma caso o sensor não esteja alinhado com a mesma.

Logo, com os levantamentos realizados, também pode-se concluir que uma tríade de acelerômetros, montados em posição ortogonal, é capaz de fornecer as três componentes cartesianas da aceleração da gravidade atuantes no corpo medido.

O método utilizado por um acelerômetro para realizar tal medição é detalhado por Titterton & Weston (2004), eles explicam que uma pequena massa instalada no interior do sensor é conectada a uma mola. A deformação gerada na mola devidos às forças aplicadas na massa são traduzidas pelo sistema do acelerômetro e enviadas ao usuário. A Figura 1 ilustra o sistema descrito.

Figura 1 - Um acelerômetro simples.



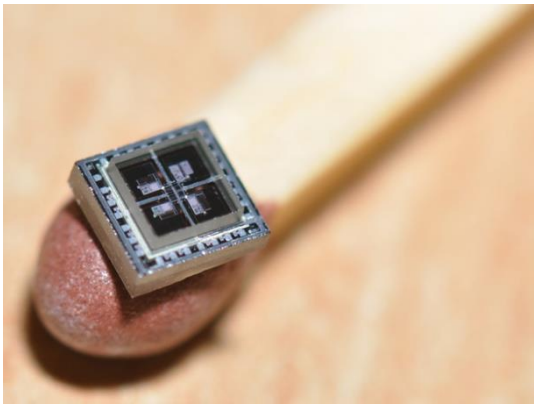
Fonte: Titterton & Weston (2004)

2.2.2. Sistemas micro eletromecânicos

Devido a necessidade de se manter um baixo custo foi feita uma pesquisa por sensores que atendessem tal requisito. Aggarwal et. al. (2010) apresenta a tecnologia dos sistemas micro eletromecânicos (MEMS) como algo emergente e com grande potencial. Suas principais qualidades são o baixo custo, baixo consumo energético e fácil disponibilidade.

Aggarwal et. al. (2010) também ilustra o método de construção dos MEMS como a integração de elementos mecânicos, eletrônicos e sensores em um substrato comum baseado em silicone. A Figura 2 apresenta um dispositivo MEMS.

Figura 2 - MEMS.



Fonte: Ehrlich (2014)

2.3. Método de representação dos ângulos

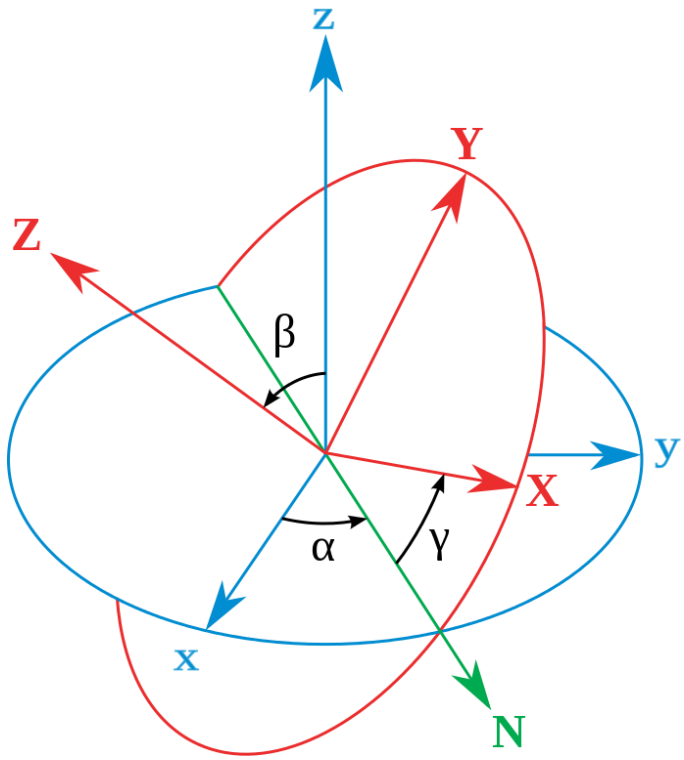
O método mais comum para representar ângulos é por meio de ângulos de Euler. Wie (2008) descreve a utilização de ângulos de Euler para determinar a orientação de um corpo por meio do uso de três rotações sucessivas. Estas rotações devem sempre respeitar a ordem

de rotação de cada eixo. Existem doze conjuntos de rotações conhecidas e cada uma delas possui equações específicas.

Siciliano & Khatib (2008) apresentam os ângulos de Euler como três ângulos independentes que especificam a orientação de um corpo em rotação em referência a um corpo fixo.

Na Figura 3 pode ser visto a representação mais comum dos ângulos de Euler α, β, e γ.

Figura 3 - Representação dos ângulos de Euler.



Fonte: Wikipédia (2014)

2.3.1. Referência

"Em física, sistema de coordenadas de referência ou referencial é utilizado para se medir e registrar as grandezas físicas [...] Cada observador deve escolher um referencial para que se possa realizar suas medidas." (Wikipédia, 2018).

O conceito de referência é muito importante, pois todos os dados medidos devem ser feitos em relação a algo. No caso esse algo é a referência, assim como visto na citação anterior. Neste projeto foram utilizadas duas referências, uma medida e outra estimada. A medida é a aceleração da gravidade, esta que representa uma linha imaginária no sentido vertical. A estimada é o plano horizontal, este não pode ser medido diretamente, mas pode ser estimado por meio da ausência da aceleração da gravidade numa medição com acelerômetros.

2.3.2. Aplicação dos ângulos de Euler

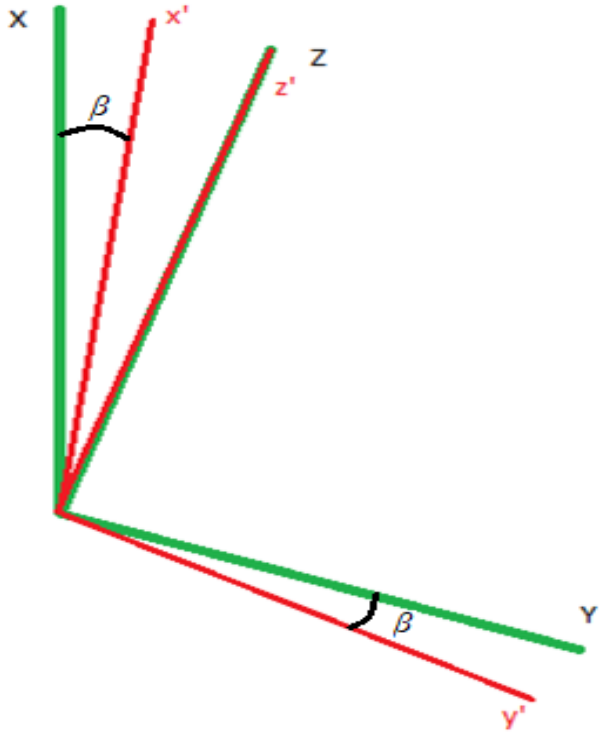
Como descrito anteriormente, o sistema proposto utiliza apenas duas referências, logo não se pode utilizar a teria de ângulos de Euler com três rotações, visto que torna impossível realizar comparações com uma referência que não existe.

Utilizando a metodologia apresentada por Titterton & Weston (2004), definiu-se o sistema de rotações na sequência zy'.

O motivo para tal é que a tríade acelerométrica montada no protótipo está alinhada com o eixo X para cima, Y para o leste e Z para norte. Como não há uma referência para o norte não serão utilizadas rotações no eixo x".

No sistema desenvolvido serão determinados os ângulos α e β conforme ilustrados nas Figuras 4 e 5.

Figura 4 - Representação da rotação no eixo Z.



Fonte: Autor

Utilizando a Figura 4 como guia para os cálculos referentes à primeira rotação, obtém-se que:

$$X = x'cos\beta - y'sen\beta$$

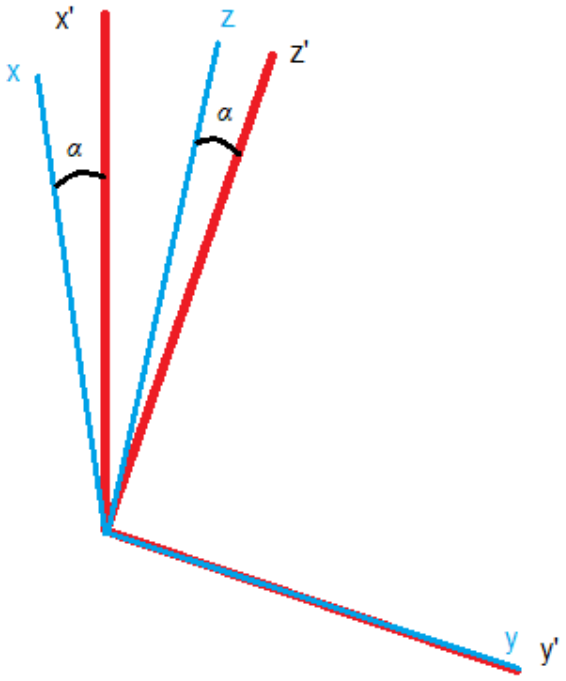
$$Y = x'sen\beta + y'cos\beta$$

$$Z = z'$$

Com base nessas equações pode-se montar a matriz que representa a primeira rotação no sistema proposto:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} cos\beta & -sen\beta & 0 \\ sen\beta & cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Figura 5 – Representação da rotação no eixo y'



Fonte: Autor

Seguindo o mesmo processo efetua-se os cálculos de rotação no eixo *y'*. Obtém-se as equações:

$$x' = xcos\alpha + zsen\alpha$$

$$y' = y$$

$$z' = -xsen\alpha + zcos\alpha$$

Montando a matriz da segunda rotação atinge-se o resultado:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} cos\alpha & 0 & sen\alpha \\ 0 & 1 & 0 \\ -sen\alpha & 0 & cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Logo, a matriz que relaciona a origem, no caso a referência, com a orientação final do corpo medido é dada por:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} cos\beta & -sen\beta & 0 \\ sen\beta & cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\alpha & 0 & sen\alpha \\ 0 & 1 & 0 \\ -sen\alpha & 0 & cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Logo:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} cos\beta cos\alpha & -sen\beta & cos\beta sen\alpha \\ sen\beta cos\alpha & cos\beta & sen\beta sen\alpha \\ -sen\alpha & 0 & cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Como o sistema referência é constituído pela vertical e pelo plano horizontal, pode-se definir que a matriz de origem é:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix}$$

Onde g é a aceleração da gravidade local. Utilizando o mesmo tipo de dados para definir a orientação do corpo medido, obtém-se os dados em unidade de engenharia medidos pelos acelerômetros:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix}$$

Onde g_x, g_y e g_z são os dados medidos pelos sensores.

Porém estas equações ainda não informam os ângulos de variação do corpo com a vertical, ou seja, a verticalidade. Para isso deve-se montar um sistema linear de equações para determinar o valor de α e β. Por meio da matriz obtida e substituindo os valores conforme relacionado anteriormente, encontra-se o seguinte sistema linear de equações:

$$\begin{cases} g = g_x cos\beta cos\alpha - g_y sen\beta + g_z cos\beta sen\alpha \\ 0 = g_x sen\beta cos\alpha + g_y cos\beta + g_z sen\beta sen\alpha \\ 0 = -g_x sen\alpha + g_z cos\alpha \end{cases}$$

Como é possível observar, a terceira equação é a única que possui uma única variável, tornando simples a definição do valor da mesma:

$$g_z cos\alpha = g_x sen\alpha$$

$$\frac{g_z}{g_x} = \frac{sen\alpha}{cos\alpha} = tg\alpha$$

Logo:

$$\alpha = tg^{-1}\left(\frac{g_z}{g_x}\right)$$

Como o valor da aceleração da gravidade local varia conforme a latitude e altitude, então o mesmo não pode ser utilizado com precisão em qualquer lugar, então a primeira

equação não pode ser utilizada sem antes definir uma equação que defina a aceleração da gravidade local. Para evitar a necessidade de tal cálculo então pode-se utilizar a segunda equação:

$$sen\beta(g_x cos\alpha + g_z sen\alpha) + g_y cos\beta = 0$$

$$\frac{sen\beta}{cos\beta} = tg\beta = \frac{-g_y}{g_x cos\alpha + g_z sen\alpha}$$

Logo:

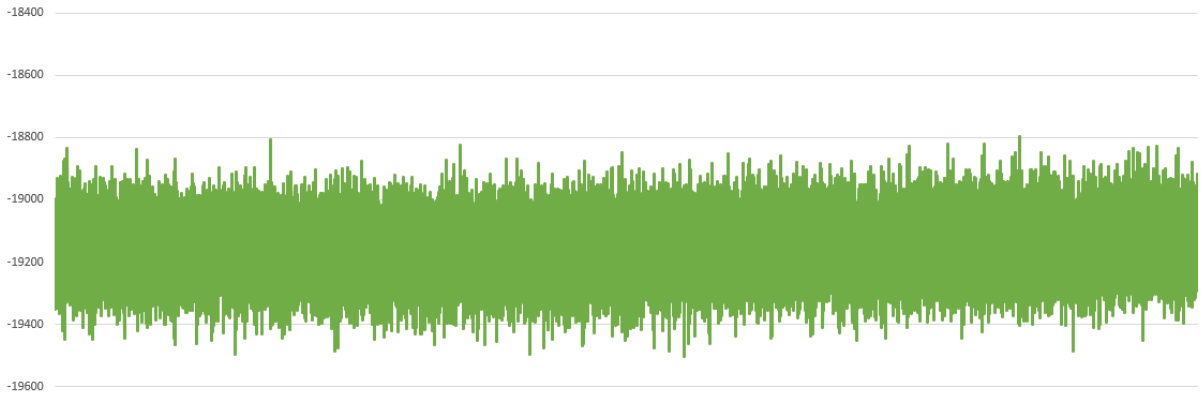
$$\beta = tg^{-1}\left(\frac{-g_y}{g_x cos\alpha + g_z sen\alpha}\right)$$

2.4. Ruído

Com a finalidade de aumentar a precisão do cálculo para determinação da verticalidade deve-se aumentar a precisão dos dados medidos, para isso deve-se entender o que é ruído.

Silver (2012) descreve o ruído como aquilo que distrai o usuário da verdade. Observando a Figura 6 pode-se claramente identificar o ruído.

Figura 6 - Dados medidos por um acelerômetro.



Fonte: Autor

O ruído é o que impede o usuário de obter o real valor medido pelo sensor, os dados ilustrados na Figura 6 apresentam ao sistema uma medição muito ampla, reduzindo grandemente a precisão da mesma.

A fim de aumentar a precisão do sistema é necessário adotar uma metodologia para tratamento de ruídos. Aggarwal et. al. (2010) apresenta a técnica da variância de Allan como a mais comum para a modelagem e determinação das características do ruído aleatório subjacente a um sensor MEMS.

2.4.1. Tratamento de ruídos

Segundo obtido no Wikipédia (2016), existem algumas variações na forma de calcular a variância de Allan. Para este trabalho foi selecionada a variância M-Sample (M amostras) dada pela seguinte equação:

$$\sigma_y^2(M,T,\tau) = \frac{1}{M-1}\left\{\sum_{i=0}^{M-1}\left[\frac{x(iT+\tau)-x(iT)}{\tau}\right]^2 - \frac{1}{M}\left[\sum_{i=0}^{M-1}\frac{x(iT+\tau)-x(iT)}{\tau}\right]^2\right\}$$

Onde:

- x(t) é o ângulo de fase em radianos medido no tempo t ou a frequência fracionária média medida na amostra;
- M é o número de amostras;
- T é o tempo entre cada amostra;
- τ é o tempo de duração de cada amostra. Geralmente T e τ são iguais.

No caso será utilizada a frequência fracionária média para x(t), esta é obtida utilizando a seguinte formulação matemática:

$$\overline{y}(t,\tau) = \frac{x(t+\tau)-x(t)}{\tau}$$

Nesta equação x(t) é a média das medições realizadas no período t.

Porém, para atenuação do ruído será utilizado o desvio padrão de Allan, este calculado por meio da variância obtida:

$$\sigma_y = \sqrt{\sigma_y^2} = \sqrt{\frac{1}{M-1}\left\{\sum_{i=0}^{M-1}\left[\frac{x(iT+\tau)-x(iT)}{\tau}\right]^2 - \frac{1}{M}\left[\sum_{i=0}^{M-1}\frac{x(iT+\tau)-x(iT)}{\tau}\right]^2\right\}}$$

Pinheiro et. al. (2009) afirma que em uma distribuição normal, aproximadamente 95% dos dados medidos estão numa amplitude de duas vezes o desvio padrão usando a média como origem. O desvio padrão de Allan obtido é sempre menor que o valor do desvio padrão clássico, isso se deve ao fato de que se utiliza valores de médias por período e não os valores medidos. Ainda assim a amplitude de duas vezes o desvio padrão será utilizado, porém com a finalidade de atenuar o ruído. Logo os dados medidos pelos sensores serão dados pelas seguintes condicionais:

$$Se\ g_{medido} > \overline{g_{medido}} + 2\sigma_y$$
$$então\ g_{medido} = \overline{g_{medido}} + 2\sigma_y$$

E:

$$Se\ g_{medido} < \overline{g_{medido}} - 2\sigma_y$$
$$então\ g_{medido} = \overline{g_{medido}} - 2\sigma_y$$

Após realizada a atenuação do ruído é então calculada uma nova média dos valores atenuados e esta média será entendida como o valor real da componente da aceleração da gravidade medida pelo sensor.

2.5. Caracterização de sensores

Outro passo necessário a fim de obter melhores medições de dados dos sensores é a caracterização dos mesmos. Por este processo entende-se como o levantamento de características dos sensores por meio de análise de dados obtidos em ensaios. Vale ressaltar que os dados levantados nos ensaios de caracterização serão utilizados para realizar os cálculos de tratamento de ruído discutidos anteriormente. Neste projeto serão estimados somente os valores de drift, fator de escala e ponto zero dos sensores acelerométricos.

Nyce (2016) descreve drift como a variação na medição realizada pelo sensor que acontece com o tempo mesmo que os dados do ambiente não sofram alteração. Como exemplo, num sensor de posição o drift é a diferença apresentada na posição num determinado tempo mesmo que o objeto medido não tenha mudado de posição.

A técnica para obtenção do valor de drift será simples: os dados obtidos serão particionados e a média dos valores da primeira partição serão comparada com a média dos valores da última partição. Será realizada uma análise no comportamento da medição do sensor para determinar se o mesmo apresenta período de drift mais severo ou se o drift é constante desde o início do período de operação do sensor.

Lawrence (2001) afirma que o fator de escala é a relação entre o valor de output do sensor e o valor medido. Porém, para calcular o fator de escala adequado é necessário definir o ponto zero do sensor, para isso coleta-se dados do valor máximo positivo e do valor máximo negativo, a média destes valores é considerado o valor zero do sensor e baseando-se nele pode-se obter a amplitude do valor bruto medido para realizar o cálculo de fator de escala. Logo:

$$\bar{g} = \frac{g_{máx} + g_{min}}{2}$$

$$f_{escala} = \frac{g_{máx} - \bar{g}}{g_{ambiente}}$$

Onde:

- $\bar{g}$ é o valor do ponto zero do sensor;
- f_{escala} é o valor do fator de escala;
- $g_{ambiente}$ é $o\ valor$ da gravidade do ambiente;
- $g_{máx}$ é o valor máximo medido pelo acelerômetro num estado de repouso, este valor coincide com o valor de aceleração da gravidade no local.

Vale ressaltar que todos os dados de características que serão obtidos são fornecidos pelo fabricante com uma certa tolerância. A caracterização realizada visa apenas confirmar os dados recebidos por meio de datasheet.

Para obtenção dessas características são coletados dados por meio do teste conhecido como ensaio de quatro posições.

2.5.1. Ensaio de quatro posições

Lawrence (2001) elucida a respeito deste ensaio que, apesar de seu nome, posiciona o sensor em seis posições distintas. Isso se deve ao fato de ser necessário obter dados de medição máxima e mínima para cada um dos três eixos ortogonais.

Outra variação deste ensaio apresentado pelo mesmo autor solicita vinte e uma posições para o ensaio. A quantidade de posições aumenta pois ele adiciona, além das posições com eixos alinhados, posições desalinhadas a trinta, quarenta e cinco e sessenta graus, respectivamente, para cada eixo, máximo e mínimo. Esta segunda variação do ensaio de quatro posições visa também avaliar o alinhamento dos sensores em sua integração mecânica na tríade. Como o sensor a ser utilizado é integrado dentro do chip MEMS então não há a necessidade de realizar esta variação do ensaio com maior coleta de dados.

O ambiente necessário para a garantia de sucesso do ensaio é rigoroso. O utilizado foi o fornecido pelo Laboratório de Identificação, Navegação, Controle e Simulação (LICS) presente no Instituto de Aeronáutica e Espaço (IAE). Foi utilizado um desempeno de granito classe 0, o que garante sua planicidade, posicionado sobre um bloco sísmico, evitando assim influências externas de vibração. O laboratório tem temperatura constante de 20°C com precisão de 0.5°C.

Foi decidido aplicar a atenuação de ruído pelo método apresentado anteriormente antes de realizar qualquer levantamento de característica do sensor, garantindo assim uma medição mais precisa para definição das propriedades solicitadas.

2.6. Engenharia de Software

Segundo Rezende (2005), engenharia de software é a arte de construir, com base no conhecimento científico e empírico, conjuntos ou subconjuntos de sistemas computacionais conhecidos como programas de computador ou software.

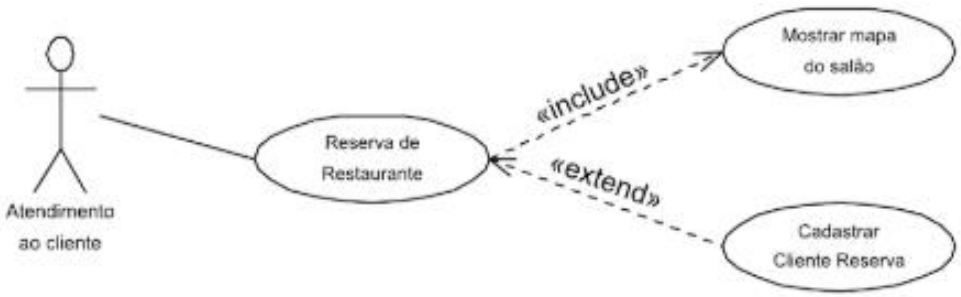
Segundo Carvalho e Chiossi (2001), a engenharia de software é uma disciplina constituída por metodologias e ferramentas a serem aplicadas a partir da percepção do problema até o momento em que o sistema desenvolvido se torna operacional, objetivando solucionar problemas relacionados ao processo de desenvolvimento e ao produto de software.

Para o projeto de software a ser idealizado para o projeto serão utilizados alguns diagramas introduzidos pela engenharia de software. Nos subcapítulos seguintes serão apresentados tais diagramas e seus conceitos a fim de elucidar suas funcionalidades.

2.6.1. Diagrama de casos de uso

Melo (2010) descreve este diagrama como uma ferramenta para expressar as fronteiras do sistema, ou seja, os elementos do sistema que são visíveis aos atores que interagem com ele. A Figura 7 ilustra tal diagrama.

Figura 7 - Representação de um diagrama de casos de uso.



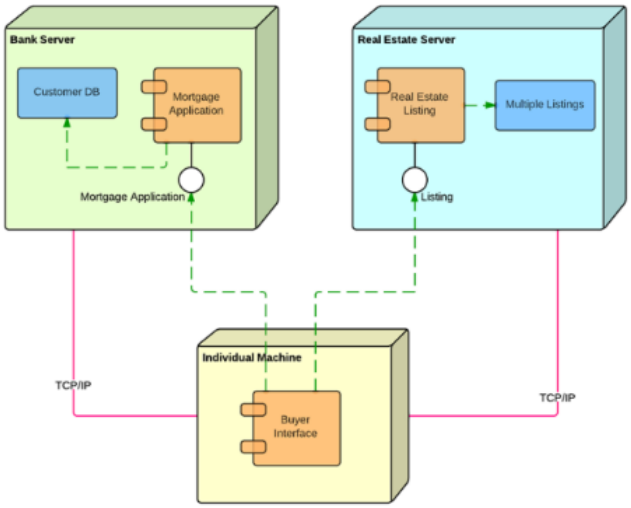
Fonte: Melo (2010)

O diagrama de casos de uso não tem como objetivo apresentar detalhes complexos de implementação do software.

2.6.2. Diagrama de implementação ou implantação

Bezerra (2015) dá suma importância ao uso do diagrama de implementação durante a fase de elaboração do projeto. Ele também descreve a construção deste diagrama, onde se desenham estereótipos gráficos que fazem menção a elementos físicos do sistema e dentro desses elementos cartões com funcionalidades internas a eles. Cada elemento físico possui linhas de conexão entre eles e o mesmo vale para as funções, mostrando canais de comunicação e quais processos conversam entre si. A Figura 8 apresenta graficamente este tipo de diagrama.

Figura 8 - Representação de um diagrama de implementação

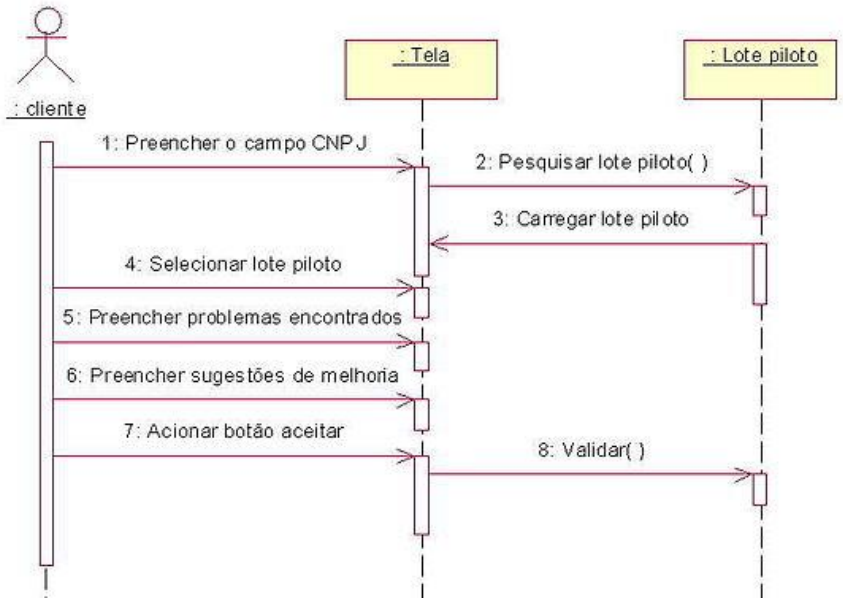


Fonte: Lucid Software (2018)

2.6.3. Diagrama de sequência

Guedes (2018) apresenta este diagrama como do tipo comportamental, sua principal função é buscar, da melhor forma possível, ilustrar a sequência de eventos que ocorrem durante um determinado processo, indicando as mensagens a serem enviadas e quais elementos estão envolvidos. Ele ainda cria uma ligação entre este diagrama e o diagrama de casos de uso, afirmando que para cada caso de uso deve haver uma sequência definida. A Figura 9 apresenta este diagrama.

Figura 9 - Representação de um diagrama de sequência.



Fonte: Wikipédia (2018)

3. DESENVOLVIMENTO

Este capítulo visa elucidar o leitor a respeito dos materiais utilizados e os critérios de seleção dos mesmos, a montagem do protótipo, o projeto de software por meio dos diagramas e a metodologia de desenvolvimento visando testes e otimização.

3.1. Materiais

Nesta seção serão apresentados os principais materiais utilizados para montagem do protótipo, sejam estes materiais físicos, ou seja, hardware ou softwares que foram utilizados para desenvolvimento do código e do projeto. Também será apresentada uma justificativa por trás da seleção de cada componente e ferramenta.

3.1.1. Arduino

Segundo o Wikipédia (2017) o Arduino é uma plataforma de prototipagem eletrônica e de placa única com portas digitais, analógicas, PWM e tudo programado geralmente em C/C++ dentro do ambiente de desenvolvimento fornecido pela própria equipe de projetistas do Arduino.

Existem vários modelos do Arduino existentes no mercado, para atender as necessidades do sistema proposto o Arduino selecionado deve possuir uma quantidade de memória flash superior. Durante o processo de compra o Arduino a ser utilizado apareceram três opções com melhor fator custo/benefício.

A Tabela 1 apresenta uma comparação dos dados pertinentes referentes a cada uma dessas variantes do Arduino. Os dados foram obtidos por meio do site oficial do Arduino (2018) e os preços por meio do site Mercado Livre.

Tabela 1 - Comparação de dados entre versões de Arduino.

	Arduino UNO	**Arduino Due**	**Arduino MEGA 2560**
Preço médio na internet (R$)	55,00	110,00	70,00
Memória Flash (KB)	32	512	256
Clock (MHz)	16	84	16
SRAM (KB)	2	96	8

Fonte: Autor

Como pode ser observado o Arduino Due é uma opção extremamente mais poderosa que os demais, porém seu custo é mais elevado, inviabilizando-o. O Arduino UNO apesar de mais barato possui uma quantidade de memória flash e memória RAM muito baixa, ficando muito a desejar para as necessidades do projeto. A versão com custo médio mais adequado e que apresenta uma configuração que atende às demandas do projeto foi o Arduino MEGA 2560. A variante selecionada possui o processador ATMEL ATmega2560 e arquitetura RISC. A Figura 10 ilustra o componente selecionado.

Figura 10 - Arduino MEGA 2560.

Fonte: Souza (2014)

3.1.2. Unidade de medidas inerciais

Foram encontradas duas variantes de IMU no mercado, uma contendo termômetro, acelerômetros e girômetros e outra contendo ainda um magnetômetro, todos tri-axiais. A Tabela 2 apresenta as duas unidades e suas características mais relevantes para a análise e seleção de componente. Os dados presentes na tabela foram extraídos do datasheet de ambos os componentes no site da InvenSense (2018) e os preços obtidos pelo site Mercado Livre.

Tabela 2 - Comparação entre as versões das IMU.

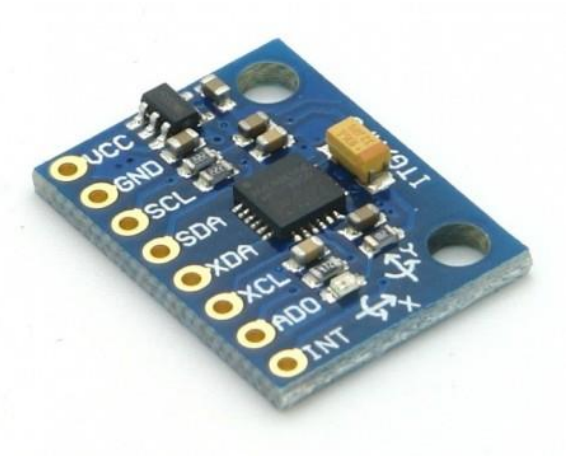
	MPU-9250	**MPU-6050**
Preço médio na internet (R$)	40,00	19,00
Precisão do acelerômetro (%)	±3	±3
Velocidade de envio de dados (Hz)	4000	1000
Atenuação de ruído automática	Sim	Não
Alteração de sensibilidade vs. temperatura (%/°C)	±0.026	±0.02
Outros sensores	Girômetro e magnetômetro	Girômetro

Fonte: Autor

Como já mencionado anteriormente, para esta primeira abordagem será utilizado apenas o acelerômetro. A precisão de ambos é a mesma, o projeto prevê aplicação de técnicas de tratamento de ruído e o custo no MPU-6050 é inferior. Estas foram as principais justificativas que levaram à seleção deste componente ao invés do MPU-9250.

A Figura 11 apresenta a IMU MPU-6050 integrada numa placa de circuito impresso em sua forma comercializada conhecida como GY-521. Casa haja a posterior necessidade, o MPU-6050 tem a capacidade de fazer fusão se sensor com um magnetômetro. As informações consideradas mais pertinentes a respeito deste componente estão disponíveis no Anexo A.

Figura 11 - MPU-6050 integrado na placa comercial GY-521.



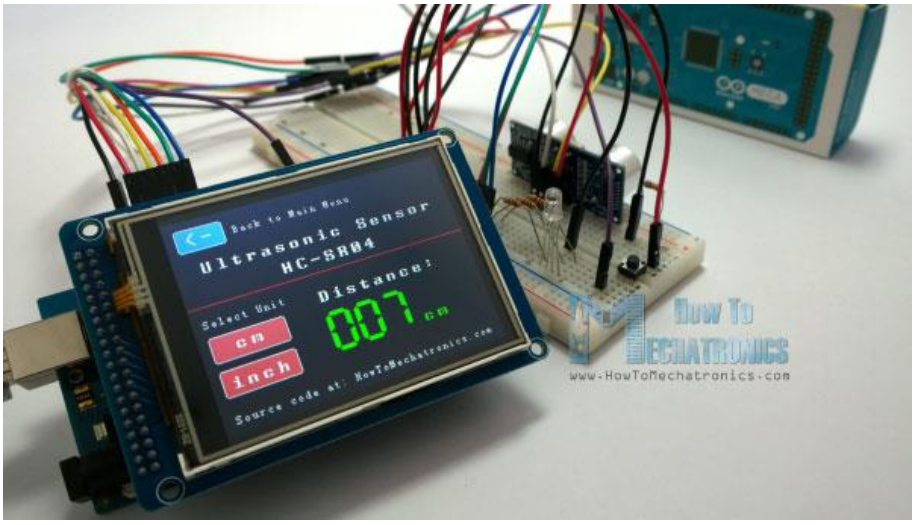
Fonte: Ivanov (2017)

3.1.3. Tela LCD

Para facilitar a visualização dos resultados obtidos foi selecionada uma tela LCD com tecnologia de toque na tela que fosse de fácil integração com o Arduino selecionado e de baixo custo. Realizando uma pesquisa de mercado foi encontrada a tela com interface de integração, esta conhecida como shield, específica para o modelo de Arduino selecionado. Infelizmente este componente é o mais caro do protótipo. Para uma aplicação laboratorial existem outras soluções para acesso aos resultados da determinação de verticalidade que não envolvem o uso de uma tela LCD e com custo muito baixo e até nulo.

A tela selecionada e com custo médio de R$180,00 é a TFT-320QVT com tecnologia de toque na tela resistiva e a respectiva interface para integração no Arduino foi a TFT LCD MEGA Shield V2.2 com valor de R$22,00 produzida e vendida pela Elecfreaks. A Figura 12 apresenta a tela selecionada em funcionamento.

Figura 12 - Tela LCD selecionada em funcionamento.

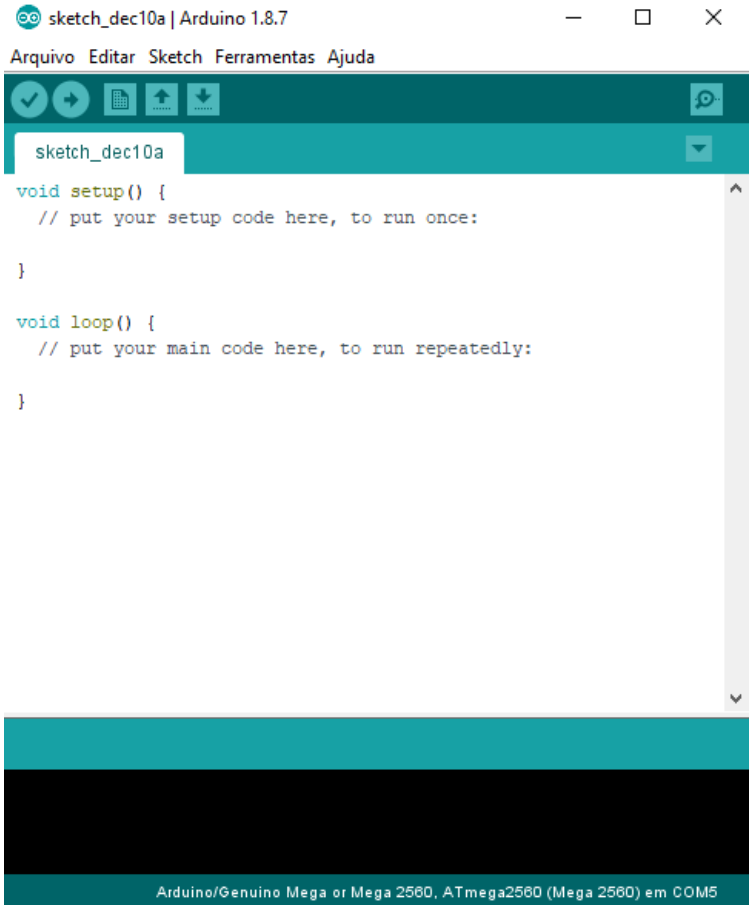


Fonte: Dejan (2015)

3.1.4. Arduino IDE

O site oficial do Arduino fornece uma IDE para programação e gravação do Arduino, esta é escrita em JAVA e baseada no software Processing e outros softwares de código aberto. A versão utilizada para desenvolvimento do código neste projeto foi a 1.8.7. A Figura 13 apresenta a janela inicial de desenvolvimento deste software.

Figura 13 - Página inicial do Arduino IDE.



Fonte: Autor

Esta IDE foi selecionada devido ao fato de ser o ambiente de desenvolvimento oficial para o Arduino.

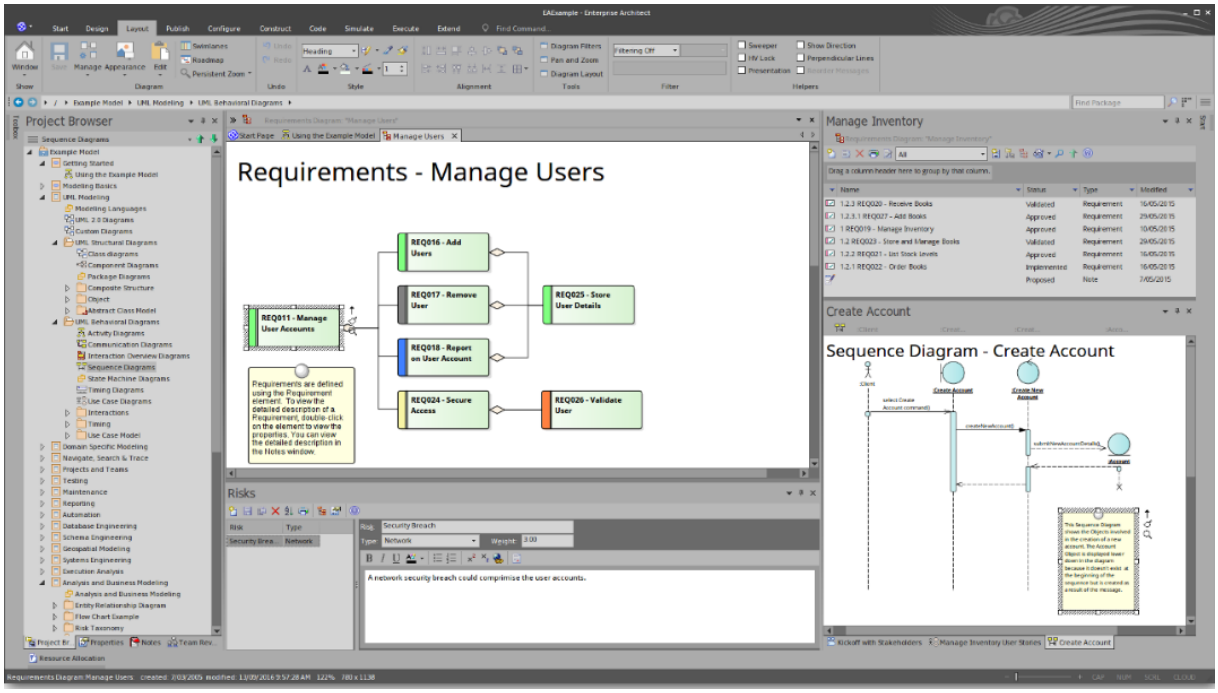
3.1.5. Enterprise Architect

Software desenvolvido pela Sparx Systems para fazer modelagem, projeto, construção, teste e implantação de sistemas, sejam estes computacionais, de software, empresariais, etc.

Neste software é possível desenvolver diagramas UML para visualização dinâmica e até mesmo gerar automaticamente inúmeros diagramas baseando-se nos já modelados. É possível ainda realizar simulações e gerar automaticamente o código caso o software tenha sido bem projetado e modelado por meio dos diagramas disponíveis.

A Figura 14 mostra a janela deste software com um diagrama de sequência e um diagrama de requisitos.

Figura 14 - Janela do Enterprise Architect.



Fonte: Autor

O critério de seleção desta ferramenta de modelagem e projeto é a expertise do autor, que fez um curso de uso da ferramenta em dezembro de 2017 no Instituto Nacional de Pesquisas Espaciais (INPE) ministrado por professor contratado por empresa externa. Além disso o autor a utiliza em seu dia-a-dia de trabalho no Instituto de Aeronáutica e Espaço, logo, é uma ferramenta de fácil acesso e uso para o mesmo.

3.2. Metodologia de desenvolvimento

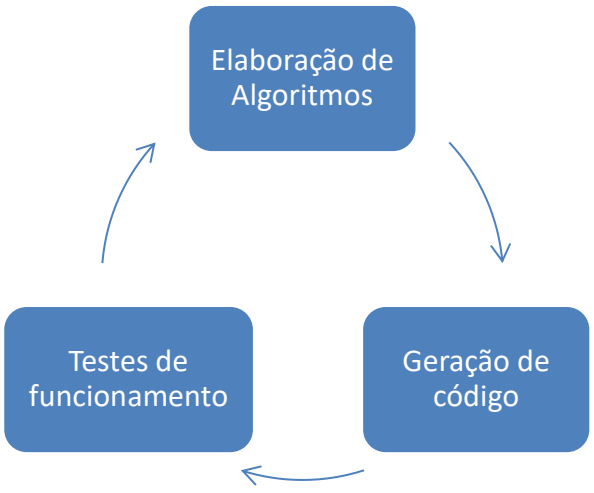
O processo de desenvolvimento foi dividido em etapas:

- Planejamento: nesta etapa inicial todo o escopo do trabalho foi definido e os processos para a obtenção do resultado final desejado foi arquitetado;
- Seleção dos materiais e ferramentas: discutido anteriormente neste capítulo, foi feita uma pesquisa dos materiais disponíveis no mercado e levantado seu fator custo/benefício para atender às necessidades do projeto;
- Montagem: nesta etapa foi feita a montagem eletrônica e mecânica dos componentes selecionados. Esta etapa será tratada com maiores detalhes posteriormente neste capítulo;
- Caracterização dos sensores: após selecionados todos os materiais, se deu início ao processo de caracterização dos sensores. Como já apresentado anteriormente o teste para aquisição de dados e levantamento de características dos acelerômetros foi o ensaio de quatro posições. Nesta etapa foram levantadas as características de variância e desvio padrão de Allan, ponto zero do sensor, drift e fator de escala;
- Projeto de software: detalhado posteriormente no decorrer deste capítulo, foi feito um projeto de software de forma a organizar e facilitar o processo de desenvolvimento do

software embarcado do Arduino para que o protótipo atinja os objetivos dispostos anteriormente;

- Programação do Arduino: com o protótipo montado e o projeto de software concluído foi realizada a implementação do código na IDE do Arduino. Após o fim da programação o código foi embarcado. Os códigos elaborados para aquisição de dados dos sensores e o código final implementando a interface gráfica e execução das funções do projeto de software estão dispostos no Apêndice A;
- Testes de funcionamento: com o protótipo montado e o Arduino já carregado com o código, foram realizados testes de funcionamento do código. Estes testes serão detalhados posteriormente no capítulo separado para o mesmo;
- Otimização do produto: esta etapa marca o início e fim do ciclo de otimização do desenvolvimento do produto final. Após obtidos os resultados dos testes de funcionamento, estes servem como entrada para realizar um refinamento no projeto de software, caso os cálculos estabelecidos não estejam atendendo adequadamente aos objetivos do projeto os mesmos podem ser analisados e refinados. A Figura 15 apresenta um fluxograma que ilustra o ciclo de otimização.

Figura 15 - Ciclo de desenvolvimento e otimização do sistema.



Fonte: Autor

3.3. Montagem do protótipo

A integração do protótipo foi feita em duas etapas:

- Montagem do sistema eletrônico;
- Montagem do sistema mecânico.

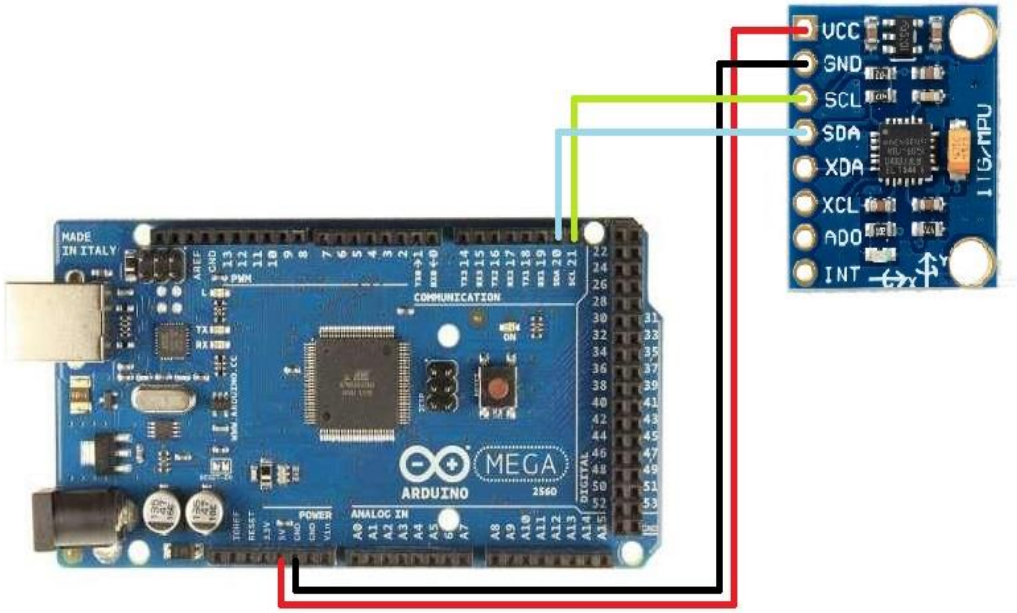
A montagem do sistema eletrônico foi bem simplificada graças à shield para integração da tela LCD ao Arduino. Infelizmente essa shield impede o uso dos bornes do Arduino após instalada, sendo necessário a instalação de uma régua de bornes na shield e as conexões feitas entre o sensor e o Arduino foram feitas também via shield. A Figura 16 apresenta o esquema elétrico desenhado para conexão do sensor ao Arduino, esta mesma

ligação foi implementada na shield. Já a Figura 17 apresenta todo o sistema eletrônico integrado.

É possível observar o sensor montado na placa de interface. Um dos benefícios desta configuração de montagem foi a facilidade de enrijecer sua posição na montagem mecânica, impedindo assim que o sensor forneça ao sistema ruídos devido a folgas de montagem.

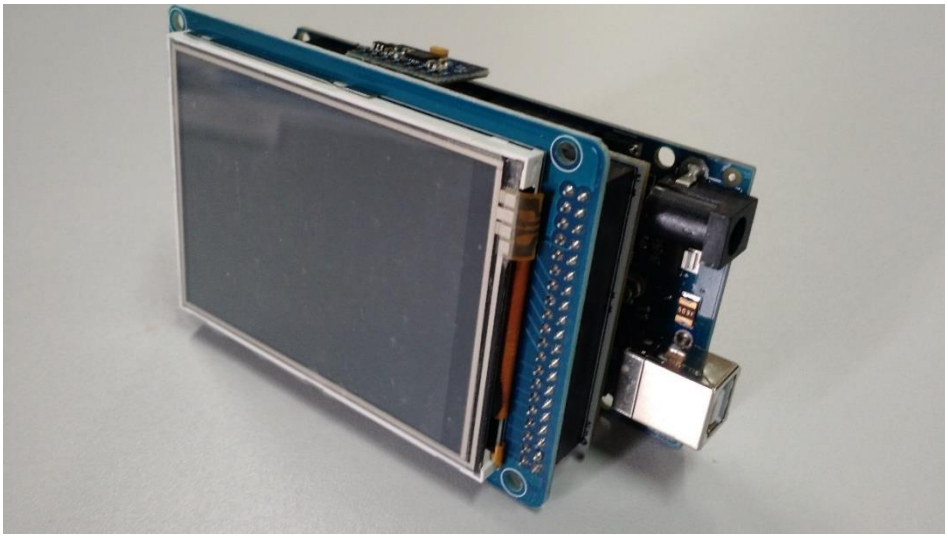
Não foi projetada uma alimentação via pilhas para o protótipo, logo o mesmo necessita estar conectado a um computador via USB (Universal Serial Bus) para que funcione.

Figura 16 - Esquema de conexão do sensor no Arduino.



Fonte: Autor

Figura 17 - Montagem completa do sistema eletrônico.

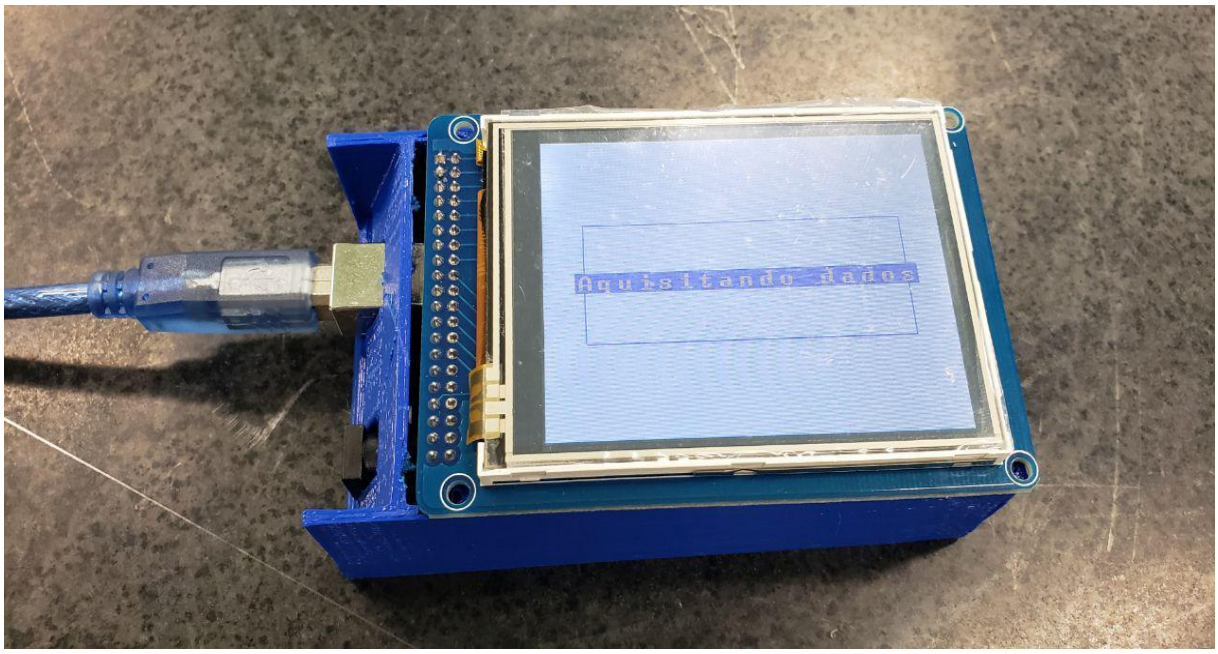


Fonte: Autor

A montagem mecânica foi a mais simplificada possível. Foi realizado o projeto de um case (caixa) e este foi impresso em 3D. Este processo de fabricação foi selecionado devido a sua facilidade e baixo custo. O desenho técnico do case projetado está no Apêndice B.

A montagem eletrônica foi então encaixada dentro da caixa. A tela LCD e o sensor foram parafusados na caixa. A Figura 18 ilustra o protótipo montado dentro da caixa de plástico.

Figura 18 - Montagem mecânica.



Fonte: Autor

3.4. Projeto de Software

Nesta seção serão apresentados e detalhados os diagramas elaborados no processo de engenharia de software. Foram selecionados alguns diagramas que tem aplicação coerente com o tipo de software desenvolvido, como o diagrama de casos de uso, diagrama de implementação, diagrama de sequência e um diagrama de fluxo para o algoritmo. Foi feito também um esboço da interface gráfica de forma a ter um objetivo claro durante a programação da interface a ser impressa na tela LCD.

3.4.1. Diagrama de caso de uso

O sistema tem dois atores interagindo com o mesmo de maneiras distintas:

- O usuário realizando requisições para o sistema, no caso estas requisições são a solicitação da medição da verticalidade ou apenas zerar a medição realizada;
- O sensor, este interage com o sistema fornecendo dados para o algoritmo que está sendo executado.

As funções executadas pelo usuário incluem a execução do algoritmo para serem concluídas, isso será visto com mais detalhes posteriormente.

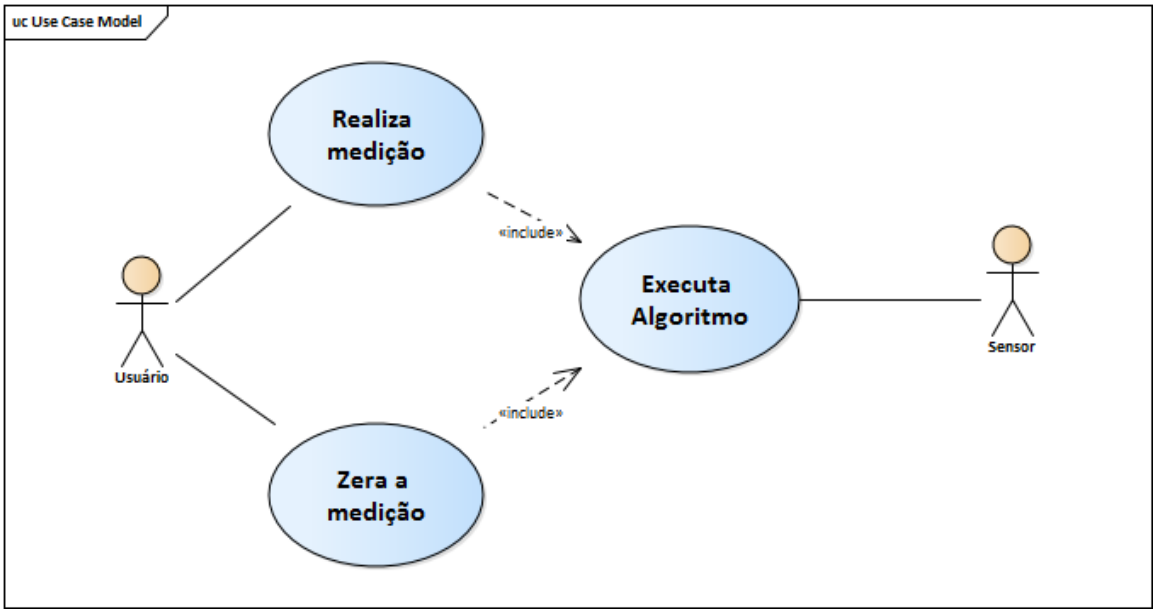
Estas funcionalidades foram selecionadas para evitar de haver erros de medição ou leitura. Caso o usuário esqueça se fez ou não a medição ele pode simplesmente solicitar que o sistema zere os valores medidos e depois realize uma nova medição. Vale ressaltar que não é necessário zerar os dados medidos para realizar a nova medição.

Por medição entende-se o processo de determinar os ângulos de variação do corpo com o sentido vertical, ou seja, sua verticalidade.

A unidade de memória não entrou como ator neste diagrama pois todos os elementos utilizados além dos dados lidos dos sensores são gerados dentro do próprio algoritmo.

A Figura 19 ilustra o caso de uso projetado para o sistema.

Figura 19 - Diagrama de caso de uso.



Fonte: Autor

3.4.2. Diagrama de implantação

Com o objetivo de facilitar a visualização de onde estão sendo executadas as funções do sistema, foi elaborado este diagrama.

O protótipo contém três elementos de hardware:

- O Arduino, que é o ambiente de execução de todo o processamento;
- O sensor, que faz a aquisição de dados do ambiente;
- A tela LCD, que é o dispositivo de interface com o usuário.

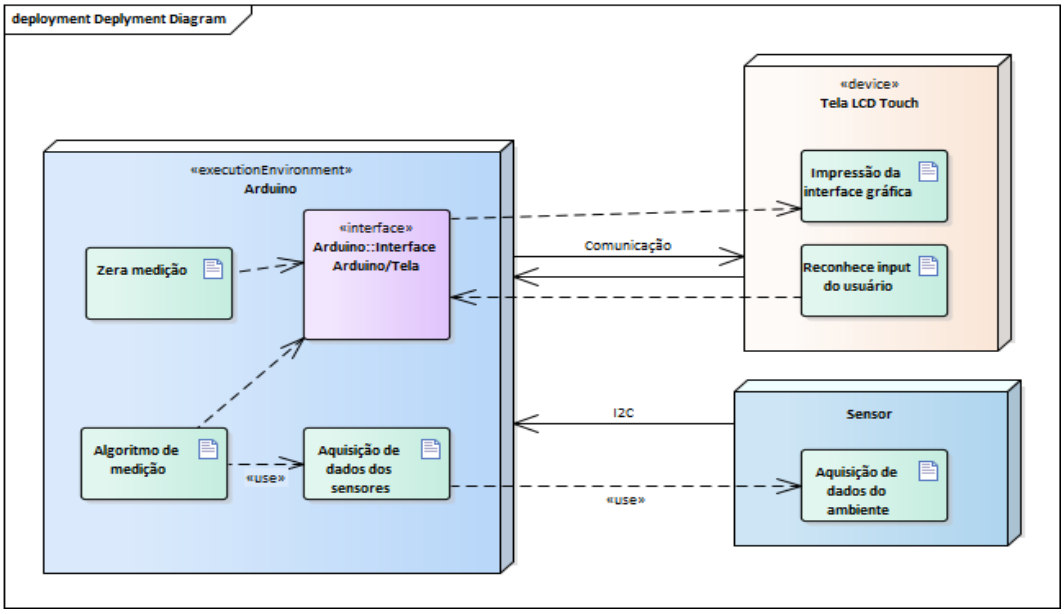
A função embutida no sensor é a de coletar os dados acelerométricos do ambiente. Como o MPU-6050 é uma IMU ele fornece também dados giométricos e de temperatura. Outra função existente é a de enviar estes dados ao Arduino por meio da comunicação via I^2C (Inter-Integrated Circuit), que nada mais é que um canal de comunicação FIFO (First-in First-out) digital.

A tela tem a função de imprimir a interface gráfica ao usuário e de coletar suas requisições via toque. O controle do sistema está embutido na tela, visto que o mesmo não executa nada sem uma solicitação do usuário, assim como visto no diagrama de caso de uso anterior.

O Arduino possui a função de executar o algoritmo que determina a verticalidade do sistema com base nos dados recebidos pelo sensor e os envia para a tela LCD para serem impressos ao usuário. É no Arduino que todo o processamento de computação gráfica é feito para atualizar a interface gráfica visualizada na tela. Foi definida uma função de interface interna no Arduino responsável por chamar a execução de uma das duas funções disponíveis ao usuário.

A Figura 20 ilustra toda essa funcionalidade explanada.

Figura 20 - Diagrama de implantação.



Fonte: Autor

3.4.3. Diagrama de sequência

Complementando o diagrama de caso de uso, o diagrama de sequência apresenta, de forma linear, como o sistema executa cada função e quais os elementos envolvidos.

Pode-se observar, pela Figura 21, que o sistema inicialmente passa por um processo de inicialização. Este processo é inerente do sistema e a única interferência com o usuário é o fato do mesmo energizar o sistema. É devido ao fato de este processo não ser algo configurado pelo programa que ele não se encontra no diagrama de caso de uso anteriormente apresentado.

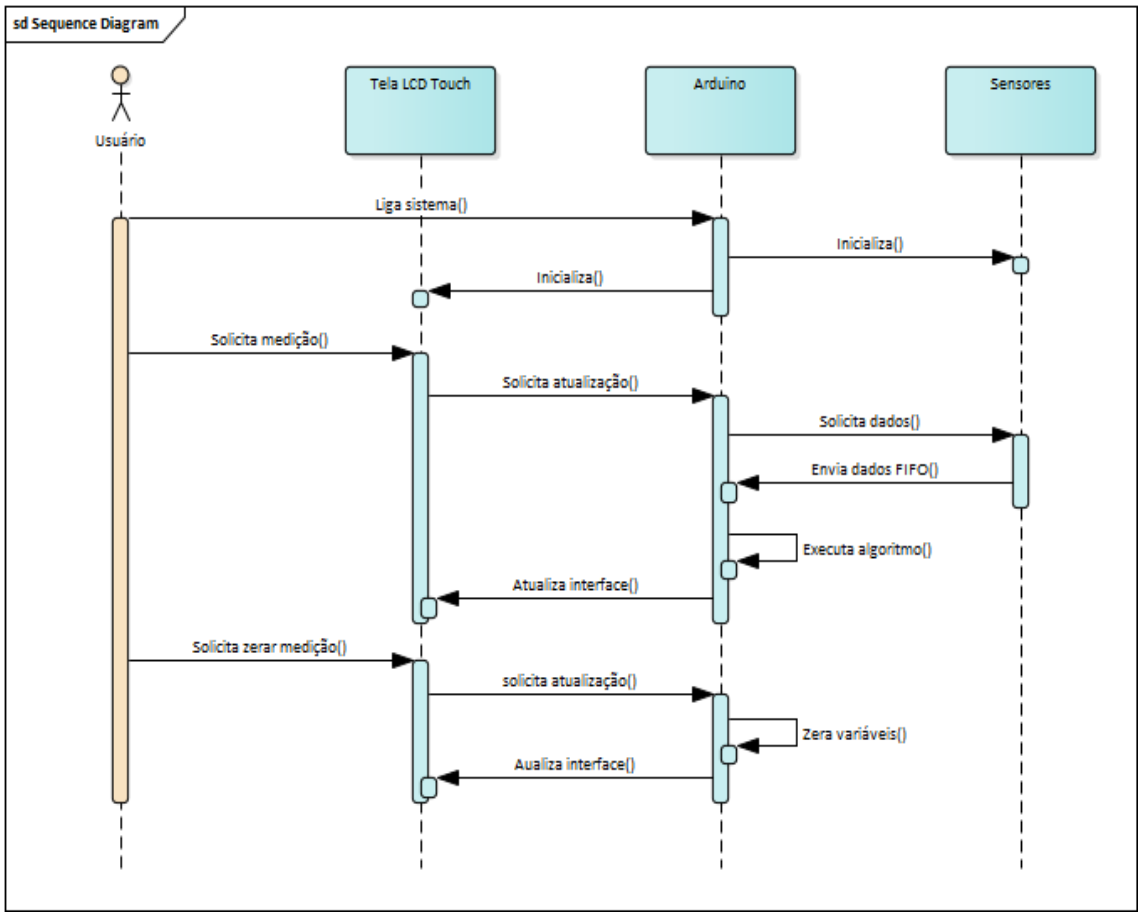
Logo em seguida o sistema fica em standby aguardando alguma solicitação do usuário. A interação do usuário é feita por meio de toque na tela, esta reconhece o toque e, caso haja solicitação, ela informa automaticamente o Arduino.

Quando o usuário solicita a realização de uma medição a tela LCD informa o Arduino disparando a requisição que executa o algoritmo de determinação de verticalidade.

No diagrama aparece uma solicitação de dados por parte do Arduino para os sensores, porém isso não acontece fisicamente. Se o buffer do sensor estiver vazio o mesmo coleta dados do ambiente, porém, como o Arduino não utiliza estes dados fora da execução do algoritmo, o buffer fica constantemente cheio. Quando é solicitada uma atualização a primeira função do algoritmo é ler os dados no buffer do sensor e esvaziá-lo, com isso o sensor realiza uma nova leitura dos dados do ambiente enchendo o buffer novamente e o Arduino mais uma vez faz esta leitura. Este processo será explanado com mais detalhes posteriormente. Com o Arduino utilizando os dados dos sensores ele pode executar seu algoritmo para determinar a verticalidade do sistema. Ao fim ele atualiza a interface gráfica para que o usuário possa visualizar os valores calculados.

Se a solicitação do usuário é o de zerar as variáveis que contém dos dados calculados ou medidos, então a tela informa o Arduino disparando assim a requisição de zerar as variáveis. A função atribui às variáveis o valor zero e logo em seguida atualiza a interface gráfica.

Figura 21 - Diagrama de sequência.



Fonte: Autor

3.4.4. Algoritmo

Para detalhar o processo do software que realizará a determinação da verticalidade foi modelado um algoritmo usando o fluxograma.

O algoritmo tem início com uma entrada manual (ou externa) do usuário solicitando a medição. Logo que a solicitação é realizada o índice do vetor que aloca os valores de medição de cada eixo do acelerômetro tri-axial é zerado.

Com esse índice zerado o Arduino então faz a primeira leitura dos dados dos acelerômetros, salva os dados no vetor correspondente, acrescenta um ao índice e então verifica se salvou dados suficiente para realizar com a maior precisão possível a determinação da verticalidade. Essa quantidade de dados foi limitada pelo tamanho da memória do Arduino selecionado a 400 conjuntos de dados. Concluída a coleta dos dados, o sistema então calcula a média dos dados medidos para g_x, g_y e g_z.

O sistema então inicia a segunda etapa do processo, que é a etapa de atenuação do ruído por meio do uso do desvio padrão de Allan previamente calculado durante o ensaio de caracterização dos sensores. Os novos valores para as medições com ruído atenuado de g_x, g_y e g_z são então salvas na memória usando o mesmo vetor usado durante a coleta de dados. Uma nova média é então calculada para cada um dos três eixos e salva na memória usando a mesma variável utilizada para salvar as médias inicialmente.

Este reaproveitamento de variáveis reduz os riscos de o sistema utilizar valores incorretos para os cálculos devido a algum erro de programação e também reduz o uso de memória flash do Arduino.

Com os dados finais de média para os três eixos o sistema então determina-se o valor de α utilizando a equação definida anteriormente durante a fundamentação teórica. Obtendo o valor de α e salvando-o na memória por meio de uma variável pode-se então determinar o valor de β usando a equação enunciada juntamente com a de determinação de α.

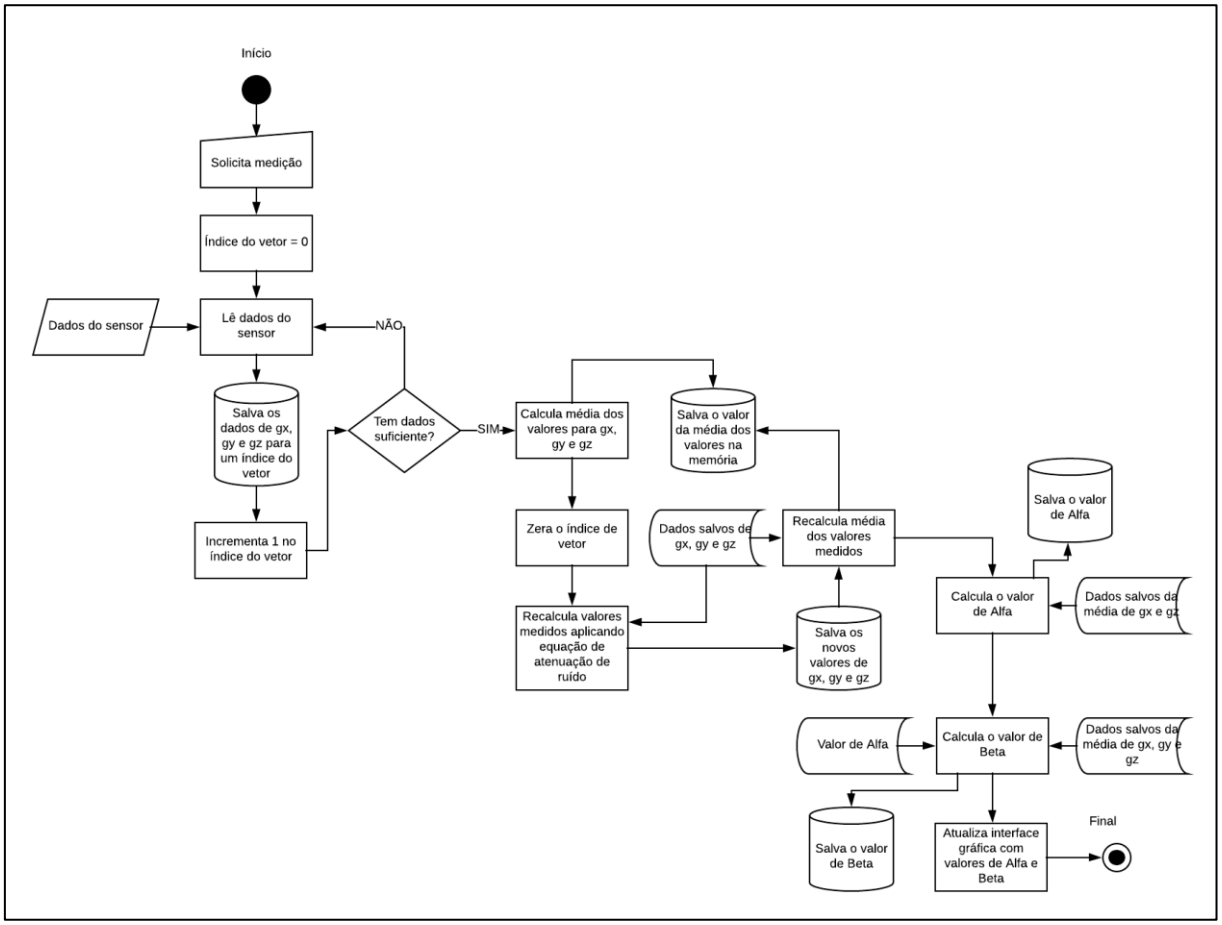
Com os valores de α e β o sistema então pode atualizar a interface gráfica apresentando ao usuário a variação do corpo em relação à vertical, ou verticalidade, calculada.

Todo o processo descrito pode ser observado na Figura 22.

No fluxograma pode-se notar que os dados dos sensores são recebidos de uma fonte externa ao algoritmo. Como já ilustrado anteriormente no diagrama de caso de uso onde o sensor é um ator que interage com o algoritmo.

Outro detalhe que vale ressaltar é que em nenhum momento o sistema exclui os dados medidos, logo estas variáveis são sempre sobrescritas quando um novo processo é executado. Os únicos valores que podem ser alterados fora do algoritmo são os calculados para α e β na função que zera os valores medidos.

Figura 22 - Algoritmo de cálculo de variação da verticalidade



Fonte: Autor

3.4.5. Esboço da interface gráfica

A Figura 23 ilustra o esboço elaborado para a interface gráfica a ser impressa na tela.

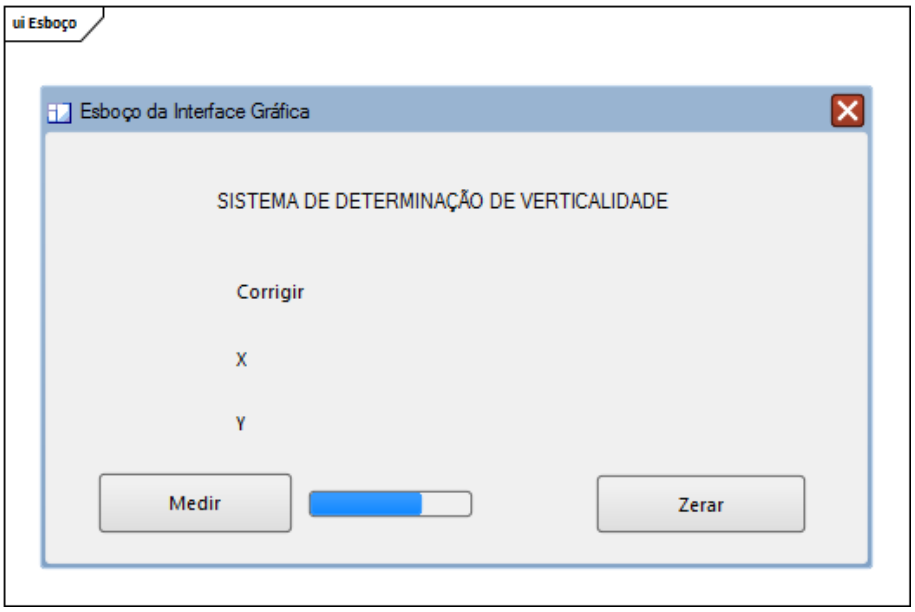
Como é um esboço então a reprodução exata do que está contido no mesmo talvez não seja feita, porém todas as informações e o layout apresentado serão respeitados.

As principais informações apresentadas na tela são os valores para α e β, na interface nomeados de X e Y.

Também estão dispostos, na parte inferior da tela, dois botões com as funções de realizar a medição e de zerar a medição, assim como solicitado no diagrama de caso de uso. Para a realização da medição foi sugerido a implementação de uma barra de carregamento para que o usuário saiba qual a porcentagem do processo de determinação de verticalidade o sistema já executou.

Foi também inserido um cabeçalho contendo o nome do Projeto.

Figura 23 - Esboço da interface gráfica.



Fonte: Autor.

4. TESTES E RESULTADOS

Neste capítulo serão apresentados os testes realizados, os resultados obtidos e será apresentada a análise dos resultados conforme a necessidade a que cada teste se propõe a atender.

4.1. Testes

Foram realizados dois tipos de testes:

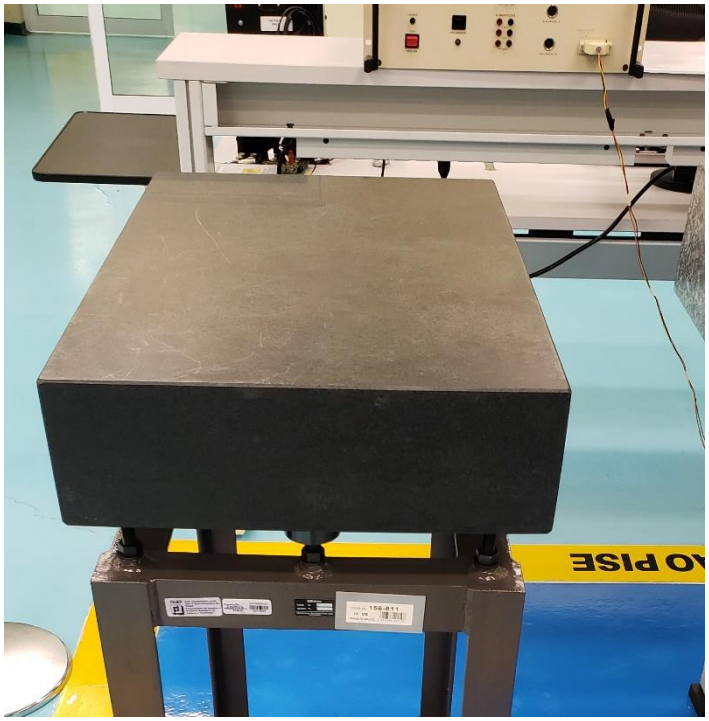
- Ensaio de quatro posição para aquisição de dados que foram utilizados para a caracterização dos sensores;
- Teste de funcionamento de software embarcado (FSE): este teste visa garantir que o software projetado está funcionando adequadamente.

Todos os testes e ensaios foram realizados no LICS.

4.1.1. Ensaio de quatro posições

Para a realização deste ensaio foi utilizado uma mesa desempeno de granito classe 0 como já descrito anteriormente. A temperatura do laboratório foi fixada em 20°C e o alinhamento do desempeno aferido com precisão de 0°0’1”, o que é desprezível devido à precisão do sistema sendo desenvolvido. É possível visualizar o desempeno utilizado na Figura 24.

Figura 24 - Desempeno de granito utilizado.

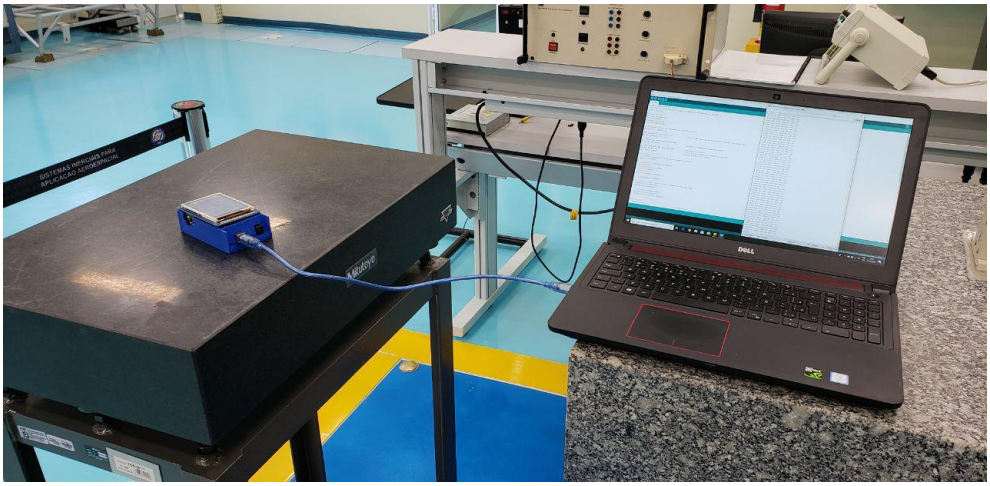


Fonte: Autor

Neste ensaio foram coletados dados acelerométricos dos três eixos em cada posição, porém só foram utilizados os dados obtidos do eixo relevante à cada posição.

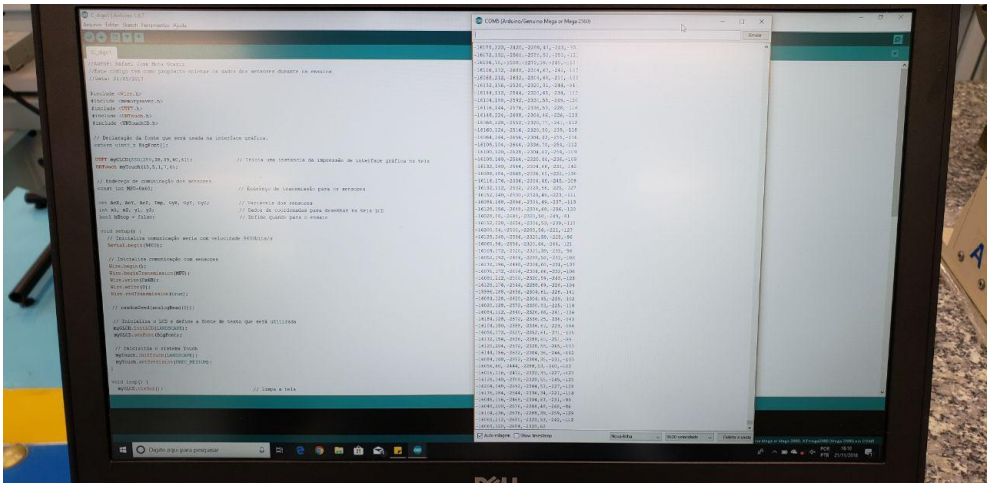
As Figuras 25 e 26 mostram o procedimento de ensaio sendo realizado.

Figura 25 - Realizando coleta de dados no eixo X positivo.



Fonte: Autor

Figura 26 - Dados sendo coletados pelo monitor serial da IDE do Arduino.



Fonte: Autor

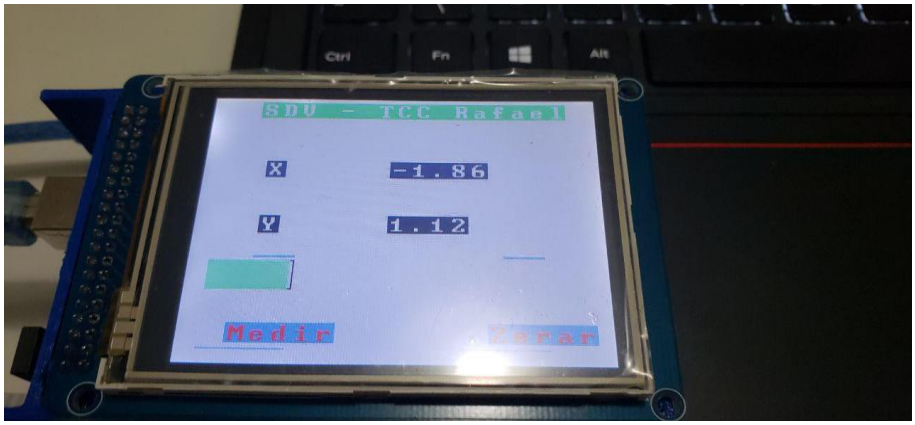
4.1.2. Teste FSE

O teste FSE foi feito em duas etapas.

A primeira etapa foi diretamente no protótipo, onde foram testadas as funcionalidades de interação do usuário com o sistema, ou seja, se o sistema recebe adequadamente as solicitações de zerar o sistema e de realizar a determinação da verticalidade do corpo medido.

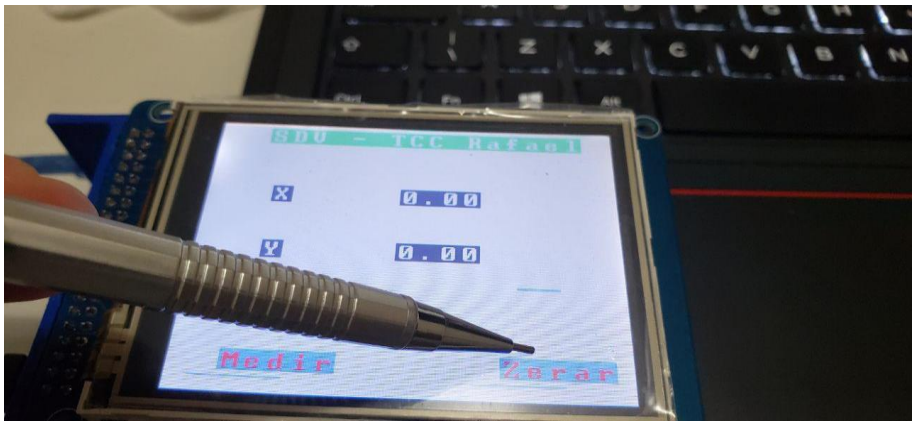
As Figuras 27 e 28 ilustram esta primeira etapa do teste FSE, observe pelas imagens que o sistema reconheceu o toque na tela adequadamente e imprimiu na tela os dados calculados pelo sistema.

Figura 27 - Testando botão para realizar medição.



Fonte: Autor

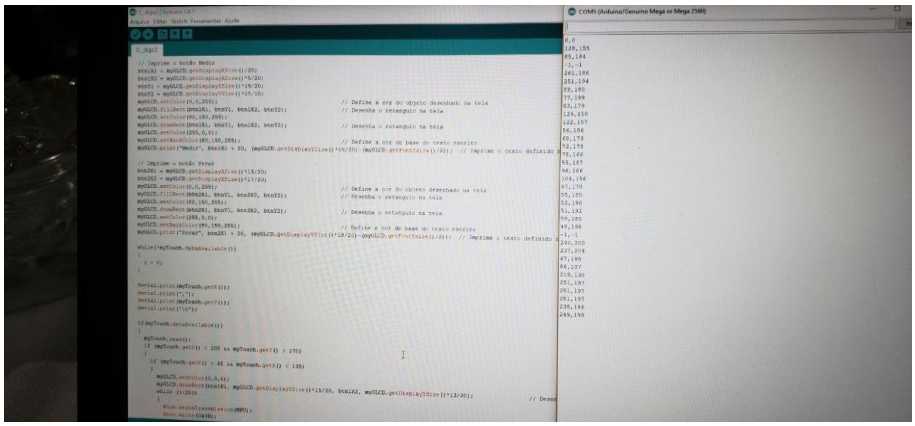
Figura 28 - Testando botão que zera as medições.



Fonte: Autor

A segunda etapa foi por meio de interação entre o protótipo e o computador, onde cada etapa do código executado realizava uma impressão no console indicando sua execução ou o resultado de sua execução. A Figura 29 ilustra esta segunda etapa do teste FSE.

Figura 29 - Teste por meio de impressão de texto de referência no console.



Fonte: Autor

4.2. Resultados

Após a realização de todos os testes, os dados coletados foram tratados e analisados de forma. Os resultados obtidos após cada teste podem ser vistos com mais detalhes nos subcapítulos seguintes.

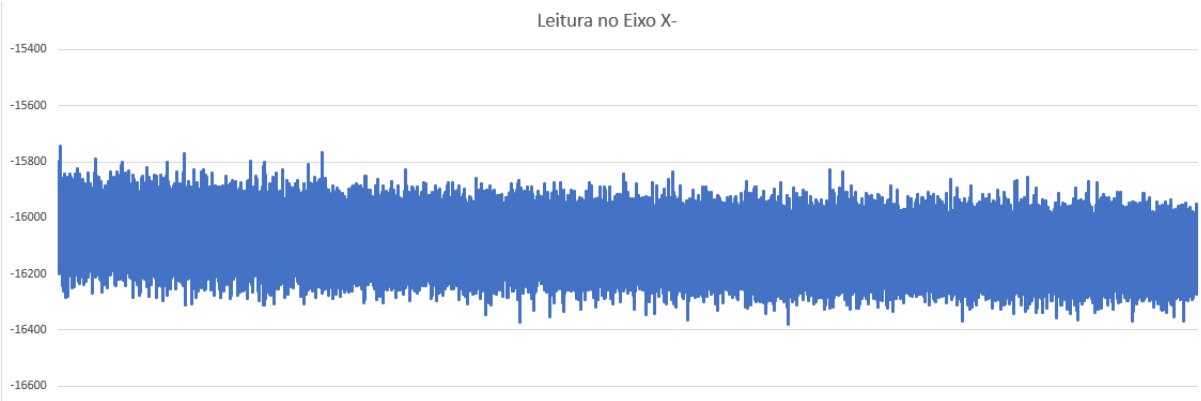
4.2.1. Ensaio de quatro posições

As figuras 30 a 35 apresentam os dados brutos obtidos no ensaio de quatro posições para cada um dos eixos medidos.

Figura 30 - Dados obtidos na posição X positiva.

Fonte: Autor

Figura 31 - Dados obtidos na posição X negativa.

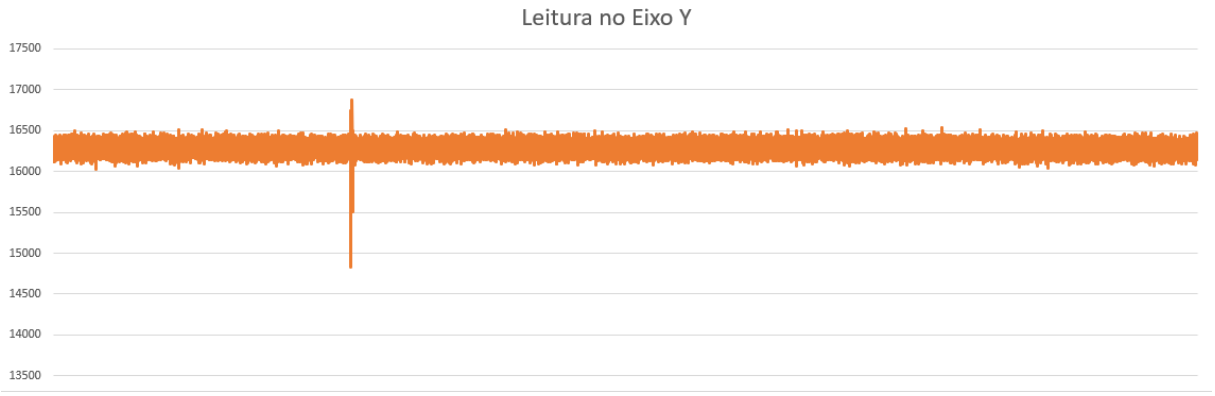


Fonte: Autor

O tempo de amostragem utilizado em todas as medições foi de uma hora. Como a frequência de amostragem foi de 30Hz, foram obtidos no total cento e oito mil conjuntos de dados para cada posição. Como dito anteriormente, só foram utilizados os valores medidos pelo sensor referente aos eixos alinhados com as posições do ensaio.

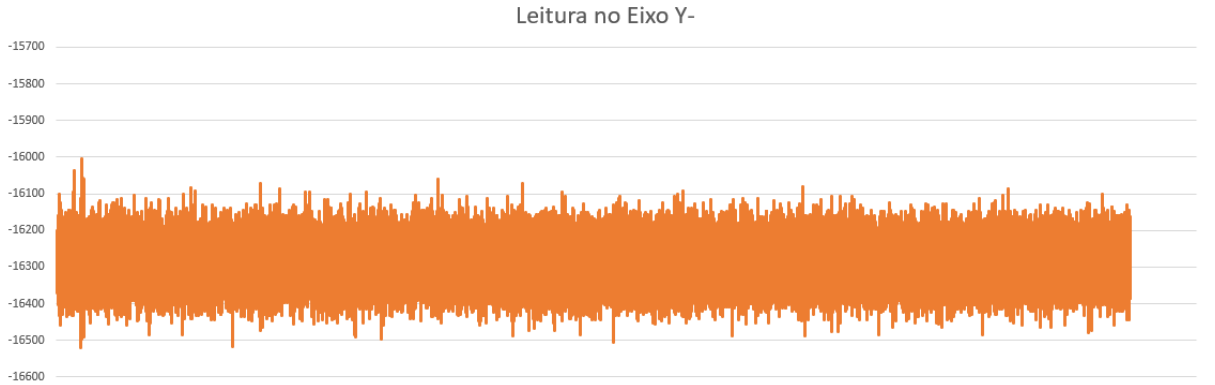
Na figura 32 é possível visualizar um spike na medição, isso deve ser causado por alguma interferência operacional no bloco sísmico durante a coleta dos dados.

Figura 32 - Dados obtidos na posição Y positiva.



Fonte: Autor

Figura 33 - Dados obtidos na posição Y negativa.

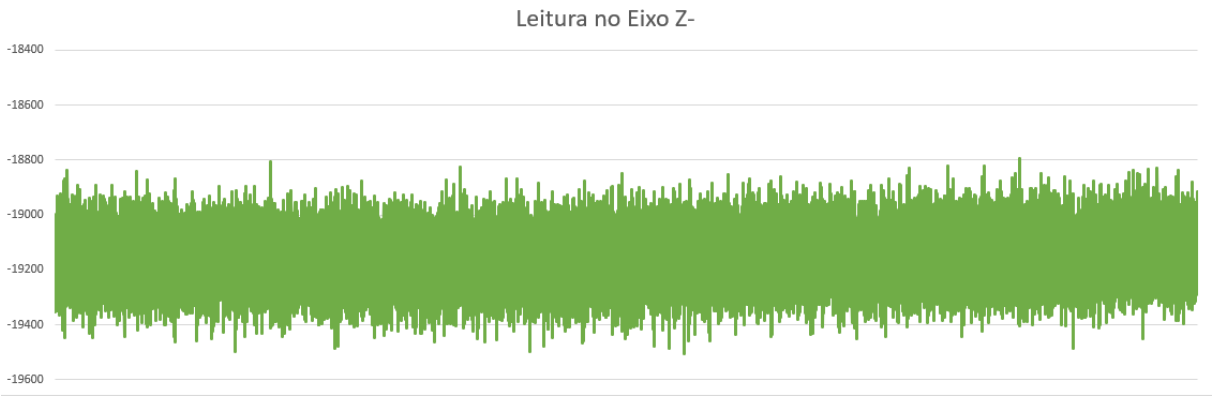


Fonte: Autor

Figura 34 - Dados obtidos na posição Z positiva.

Fonte: Autor

Figura 35 - Dados obtidos na posição Z negativa.



Fonte: Autor

Com os dados em mãos foi possível obter o valor da variância de Allan para cada eixo. Cada medição foi dividida em doze amostras de cinco minutos cada. Como o ruído de cada eixo comporta-se da mesma forma, independente no sentido, visto que é o mesmo sensor e o ruído é inerente ao sensor e não ao sentido da medição, então a variância de Allan foi calculada utilizando-se os dados dos eixos positivos apenas.

Com isso, aplicando-se a fórmula para calcular a variância de Allan, obteve-se os seguintes valores:

$$\sigma_x^2 = 259{,}8045$$

$$\sigma_y^2 = 327{,}4516$$

$$\sigma_z^2 = 1467{,}83$$

Já foi dito anteriormente que o cálculo da variância de Allan era apenas um passo para obter o real valor a ser utilizado no tratamento de ruídos: o desvio padrão de Allan. Com base nos valores obtidos foram calculados os seguintes valores para o desvio padrão de Allan:

$$\sigma_x = 16{,}118$$

$$\sigma_y = 18{,}096$$

$$\sigma_z = 38{,}312$$

Após realizada a atenuação do ruído de todos os dados foi calculada a média dos valores por período de amostragem para realizar o cálculo do drift em porcentagem. Os dados obtidos de drift em uma hora foram os seguintes:

$$drift_x = -0{,}52\%/h$$

$$drift_y = 0{,}04\%/h$$

$$drift_z = 0{,}12\%/h$$

Foram calculados valores de ponto zero do sensor para cada um dos períodos da amostragem e no fim foi calculado um valor médio, os valores de ponto zero finais encontrados são os seguintes:

$$g_x(0) = 344{,}968$$

$$g_y(0) = -2{,}197$$

$$g_z(0) = -2565{,}7$$

Vale ressaltar que, para um uso mais extenso dos sensores há a necessidade de compensar o drift dos mesmos para o valor de ponto zero também de forma a não haver erros na amplitude dos dados coletados do ambiente.

Finalmente, considerando a amplitude do sinal com os valores obtidos pode-se calcular o fator de escala dos sensores. Porém, antes é necessário determinar o valor da gravidade em São José dos Campos em g.

$$g_{nível\ do\ mar} = 1g = 9{,}81 m/s^2$$

$$g_{São\ José\ dos\ Campos} = \frac{9{,}7962 m}{s^2} = 0{,}9986 g$$

Agora é possível obter os valores de fator de escala:

$$f_{escala\ x} = \frac{16782{,}34 - 344{,}968}{0{,}9986} \cong 16400$$

$$f_{escala\ y} = \frac{16083{,}07 - (-2{,}197)}{0{,}9986} \cong 16108$$

$$f_{escala\ z} = \frac{14024{,}2 - (-2565{,}7)}{0{,}9986} \cong 16613$$

Com isso obteve-se todos as características necessárias dos sensores para atender às necessidades do projeto.

4.2.2. Teste de funcionamento de software

A Tabela 3 apresenta todos os testes de funcionamento de software realizados e seus respectivos resultados.

Tabela 3 - Testes FSE realizados e seus resultados.

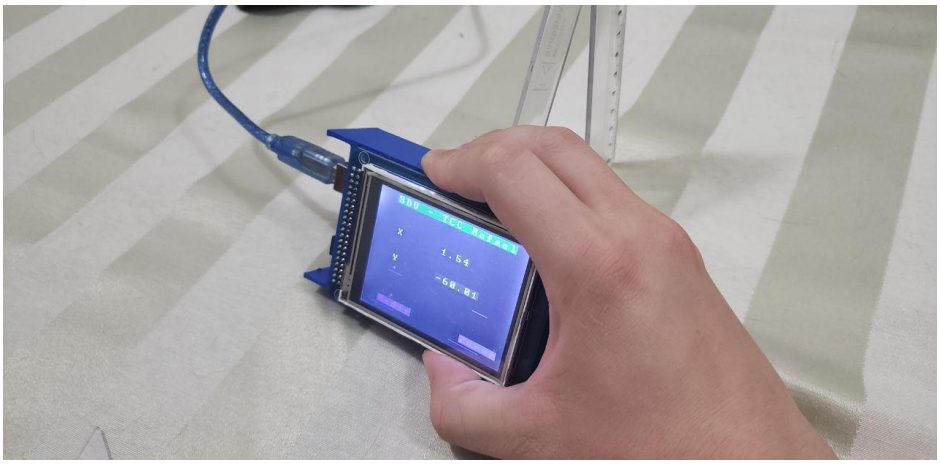
Teste	Resultado	Comentários
Teste de conexão dos sensores	Sucesso	A comprovação de que a conexão foi realizada com sucesso é o fato de que os dados dos sensores foram lidos com sucesso.
Teste de inicialização da tela LCD I	Falha	O mapeamento dos pinos estava incorreto, após uma pesquisa minuciosa no datasheet do shield da tela LCD foram identificados os pinos adequados e reconfigurada a criação da instancia

		da tela no Arduino.
Teste de inicialização da tela LCD II	Sucesso	Após a devida reconfiguração a tela inicializou com sucesso e todos os testes de impressão de círculos, bitmaps e retângulos foram realizados com sucesso.
Teste de impressão de elementos na tela	Sucesso	Todos os objetos foram impressos adequadamente na tela LCD
Teste de reconhecimento de toque na tela I	Falha	Os pinos da configuração da instância do objeto que reconhece o toque na tela estavam incorretos, impedindo assim que qualquer toque pudesse ser reconhecido adequadamente.
Teste de reconhecimento de toque na tela II	Falha	As coordenadas do toque na tela e de impressão são diferentes, logo o sistema não reconhecia nenhum comando solicitado de toque na tela.
Teste de reconhecimento de toque na tela III	Sucesso	Após realizar um mapeamento adequado das coordenadas dos botões para toque e realizadas as configurações no código o toque na tela funcionou adequadamente.
Teste de funcionamento da barra de carregamento	Sucesso	A barra de carregamento informando ao usuário o estágio da medição funcionou adequadamente.
Teste do botão “Zerar”	Sucesso	-
Teste do botão “Medir”	Sucesso	-
Teste do cálculo de determinação de variação da vertical I	Falha	Valor medido não era coerente com a posição do corpo.
Teste do cálculo de determinação de variação da vertical II	Sucesso	Valor obtido após revisão das equações possui maior coerência. Devido a limitação de uso de apenas 400 dados o sistema se tornou impreciso

Fonte: Autor

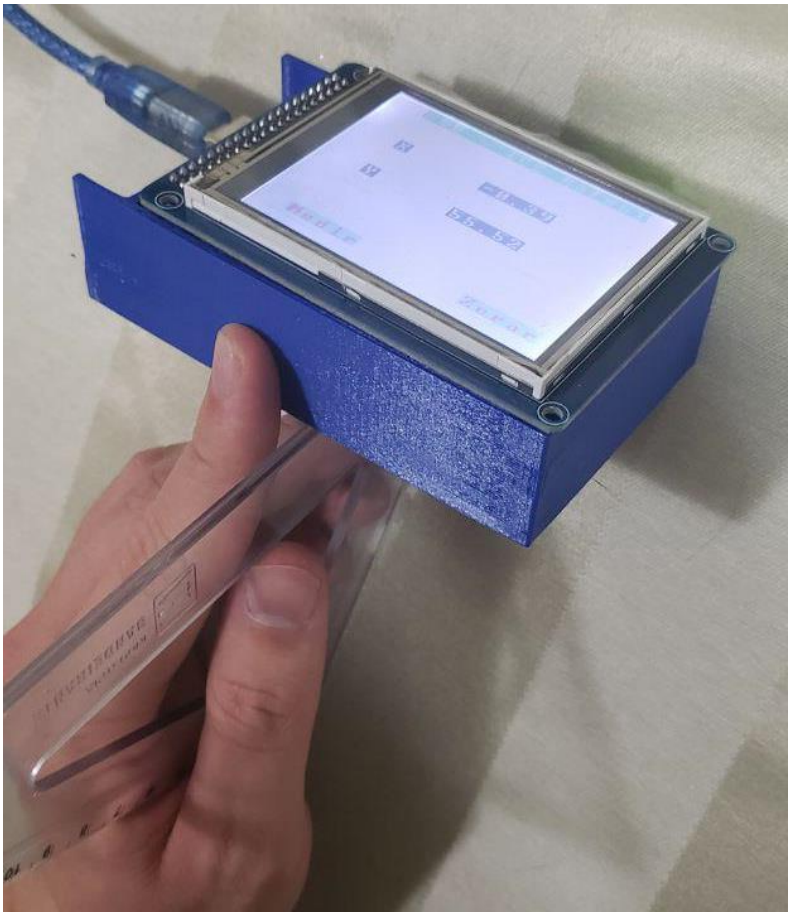
As Figuras 36 a 43 ilustram o teste de cálculo de determinação da verticalidade II e é possível ler na tela do protótipo os valores medidos de inclinação. Para X, ou valor de β, foram realizados testes a 30° e 60° e para Y, ou valor de α, foram realizados testes a 45° e 60°. Os esquadros utilizados como referência podem ser vistos na Figura 44.

Figura 36 - Cálculo de Inclinação Y - 60º negativos.



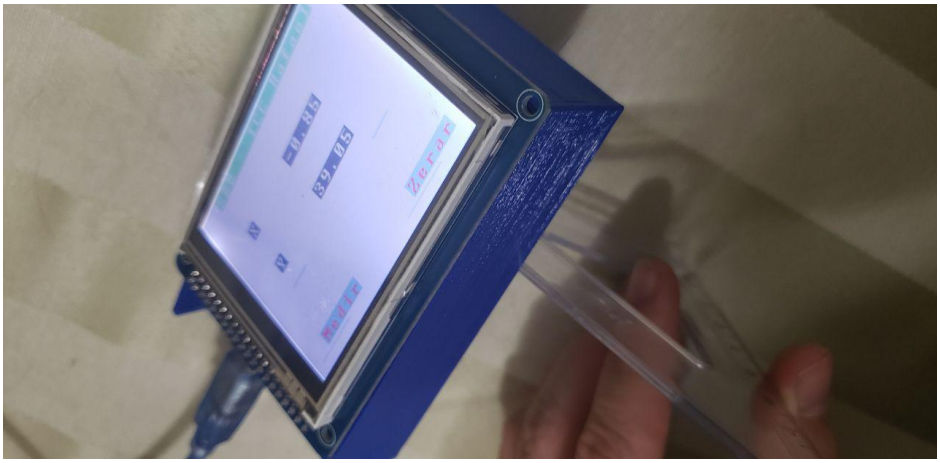
Fonte: Autor

Figura 37 - Cálculo de Inclinação Y - 60º positivos.



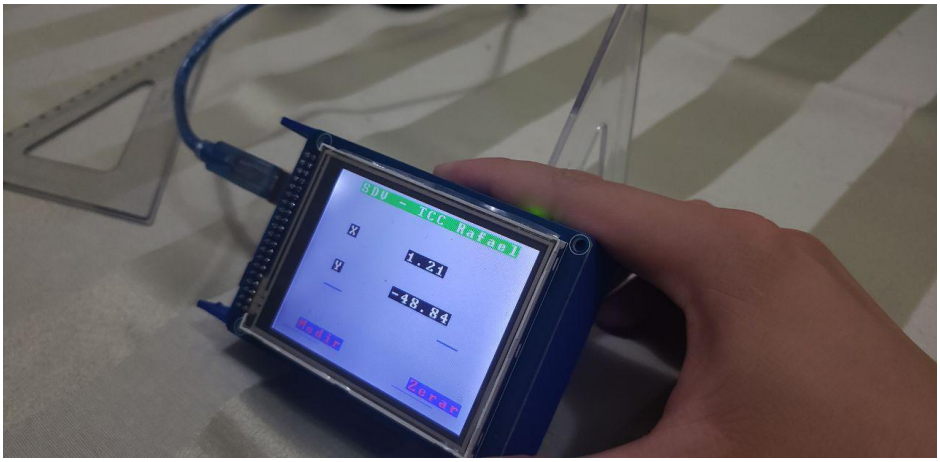
Fonte: Autor

Figura 38 - Cálculo de Inclinação Y - 45º positivos.



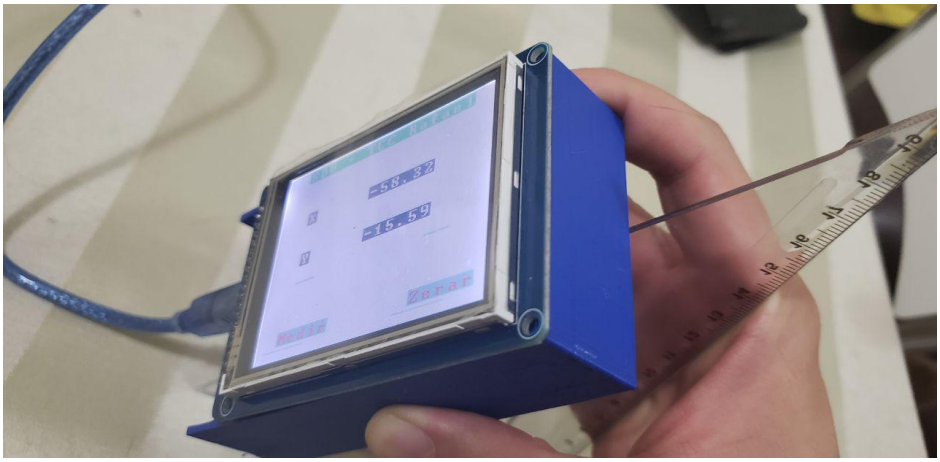
Fonte: Autor

Figura 39 - Cálculo de Inclinação Y - 45º negativos.



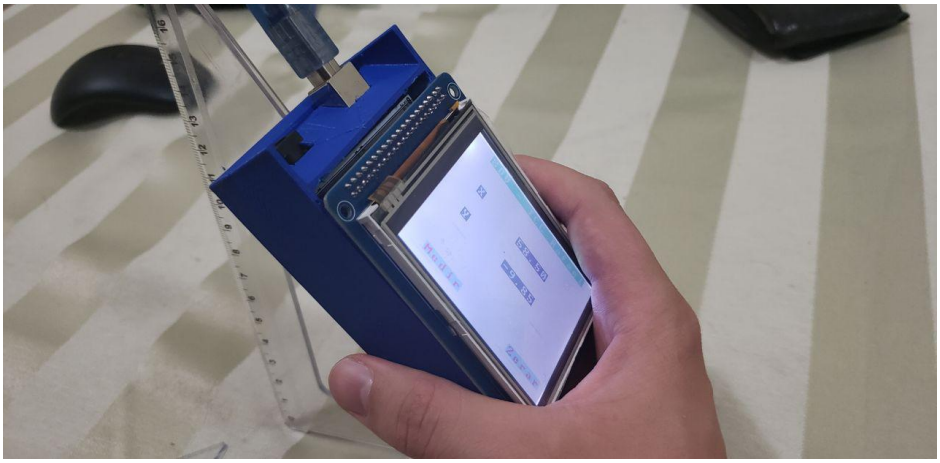
Fonte: Autor

Figura 40 - Cálculo de Inclinação X - 60º negativos.



Fonte: Autor

Figura 41 - Cálculo de Inclinação X - 60º positivos.

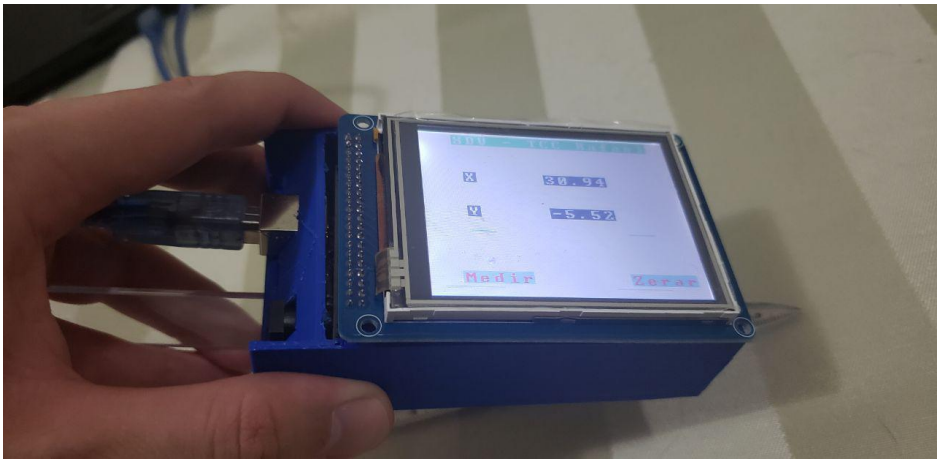


Fonte: Autor

Figura 42 - Cálculo de Inclinação X - 30º negativos.

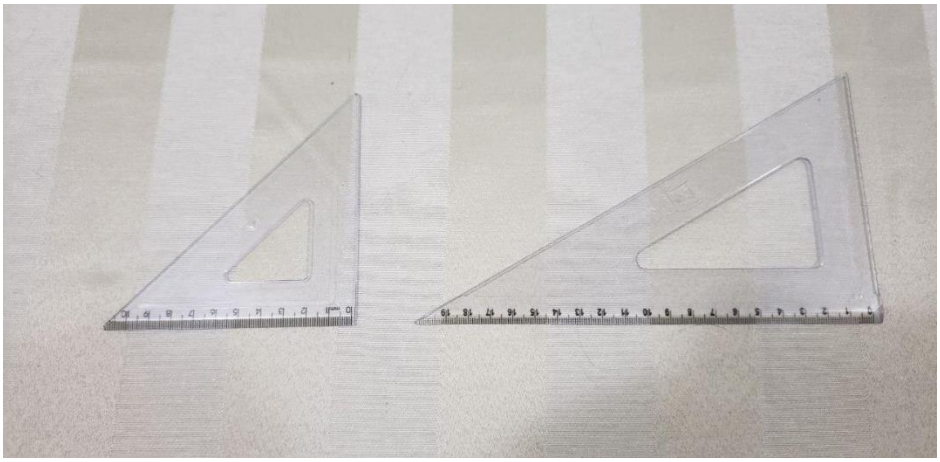
Fonte: Autor

Figura 43 - Cálculo de Inclinação X - 30º positivos.



Fonte: Autor

Figura 44 - Transferidores 45/45 e 30/60 utilizados como referência.



Fonte: Autor

5. CONCLUSÃO

Após as análises dos resultados apresentados previamente, o sistema provou que, ainda que com certa imprecisão, realiza a função a qual foi projetado com sucesso.

Quanto ao cumprimento dos objetivos específicos:

- A IMU escolhida para o protótipo atendeu às necessidades do projeto e teve um custo muito acessível;
- Foi realizado um código simples, presente no Apêndice A, que realizou a aquisição dos dados dos sensores para o processo de caracterização;
- Foi realizado o ensaio de quatro posições a fim de coletar dados para a caracterização e esses dados foram utilizados com sucesso e as características obtidas;
- As equações necessárias para converter os dados acelerométricos obtidos para os ângulos α e β de variação da vertical foram deduzidas com sucesso utilizando ângulos de Euler;
- O software foi devidamente projetado e implementado utilizando técnicas de engenharia de software;
- A interface impressa na tela foi programada com sucesso e por meio de testes foi comprovado o funcionamento adequado.

Notou-se que a precisão do sistema foi prejudicada por alguns fatores que são objeto de estudos, mas podem ser:

- Erros no processo de caracterização;
- Erros de ortogonalidade entre os sensores;
- Necessidade de se utilizar mais que 400 conjuntos de dados durante os cálculos de determinação de verticalidade de forma a ser menos influenciado pelo ruído.

5.1. Trabalhos futuros sugeridos

Como prosseguimento do trabalho aqui iniciado são sugeridos dois possíveis temas a serem trabalhados:

- Otimização do sistema de forma a reduzir a imprecisão por meio de análise de ortogonalidade, implementação de matriz de correção de ortogonalidade, uso de sensores mais precisos, etc.
- Rastreamento de atitude de forma dinâmica utilizando os dados de girômetros, o que será uma continuação direta deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

AGGARWAL, Priyanka et. al. *MEMS-Based Integrated Navigation*. Norwood: Artech House, 2010.

BEZERRA, Eduardo. *Princípios de análise e projeto de sistemas com UML*. 3ª ed. Rio de Janeiro: Elsevier, 2015.

CARVALHO, Ariadne M. B. R.; CHIOSSI, Thelma C. dos S. Introdução à engenharia de software. Campinas: Editora da Unicamp, 2001.

Compare board specs. ARDUINO. Disponível em: <https://www.arduino.cc/en/products.compare>. Acesso em: 01 de dez. 2018.

Diagrama de sequência. WIKIPÉDIA. Disponível em: <https://pt.wikipedia.org/wiki/Diagrama_de_sequência>. Acesso em: 01 de dez. 2018.

DEJAN. Arduino TFT LCD touch screen tutorial. HOW TO MECHATRONICS. 2015. Disponível em: <https://howtomechatronics.com/tutorials/arduino/arduino-tft-lcd-touch-screen-tutorial/>. Acesso em: 20 de nov. 2017.

EHRLICH, Gabriela. *Tiny MEMS*. IEC-Tech. 2014. Disponível em: <https://iecetech.org/issue/2014-01/Tiny-MEMS>. Acesso em: 05 de dez. 2015.

FERREIRA, Alexandre J. et. al. Procedimento experimental para determinação de atitude de satélites artificiais. Science and Engineering Journal, Brasil, volume 17, pág. 57-64, dezembro de 2008.

GUEDES, Gilleanes T. A. *UML 2: Uma abordagem prática*. 3ª ed. São Paulo: Novatec, 2018.

INVENSENSE. *MPU-6000 and MPU-6050 product specification revision 3.4*. Sunnyvale, 2013. 52p.

–. MPU-9250 product specification revision 1.1. San Jose, 2016. 42p.

IVANOV. *Como ligar acelerómetro e giroscópio GY-521 MPU-6050 ao Arduino?*. 2017. Disponível em: <http://www.arduinoetecnologia.com.br/projetos/como-ligar-acelerometro-e-giroscopio-gy-521-mpu-6050-ao-arduino/>. Acesso em: 01 de dez. 2018.

LUCID SOFTWARE. *O que é um diagrama de implem*entação. LUCID SOFTWARE. Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-implementacao-uml>. Acesso em: 01 de dez. 2018.

KALANTARI, Mahzad et. al. A new solution to the relative orientation problem using only 3 points and the vertical direction. Journal of mathematical imaging and vision. 9 de nov. 2010 - Caderno 39 - Pág. 259-268.

LAWRENCE, Anthony. *Modern inertial technology – Navigation, guidance, and control*. New York: Edward Brothers Inc. 2001.

MELO, Ana C. Desenvolvendo aplicações com UML 2.2 – Do conceitual à implementação. 3ª ed. Rio de Janeiro: Brasport, 2010.

NYCE, David S. *Position sensors*. New Jersey: John Wuley & Sons Inc, 2016.

OLIVEIRA, Gabriel F. *Um sistema de determinação e controle de atitude de baixo custo para o primeiro nanossatélite Ucraniano, UYS-1*. 2013. 73f. Dissertação (Mestrado em Engenharia Elétrica) – Universidade de Brasília, Brasília, 2013.

PINHEIRO, João I. D. et. al. *Estatística básica – A arte de trabalhar com dados*. Rio de Janeiro: Elsevier, 2009.

REZENDE, Denis A. Engenharia de software e sistemas de informação. 3ª ed. Rio de Janeiro: Brasport, 2005.

SICILIANO, Bruno; KHATIB, Oussama. *Handbook of robotics*. Berlin: Springer, 2008.

SOUZA, Fábio. *Saiba mais sobre a placa Arduino MEGA 2560*. 2014. EMBARCADOS. Disponível em: <https://www.embarcados.com.br/arduino-mega-2560/>. Acesso em: 20 de out. 2017.

SILVER, Nate. *The signal and the noise*. New York: The Penquin Press, 2012.

TITTERTON, D. H.; WESTON, J. L. *Strapdown inertial navigation technology*. 2ª ed. Reston: The Institution of Electrical Engineers, 2004.

WIE, Bong. *Space vehicles dynamic and control*. 2ªed. Ames: American Institute of Aeronautics and Astronautics, 2004.

WIKIPÉDIA. *Allan variance*. WIKIPÉDIA. Disponível em: <https://en.wikipedia.org/wiki/Allan_variance>. Acesso em: 10 de ago. 2017.

–. Ângulos de Euler. WIKIPÉDIA. Disponível em: <https://pt.wikipedia.org/wiki/Ângulos_de_Euler>. Acesso em: 15 de dez. 2014.

–. *Arduino*. WIKIPÉDIA. Disponível em: <https://pt.wikipedia.org/wiki/Arduino>. Acesso em: 05 de nov. 2017.

–. *Referencial*. WIKIPÉDIA. Disponível em: <https://pt.wikipedia.org/wiki/Referencial>. Acesso em: 01 de dez. 2018.

–. *Vertical*. WIKIPÉDIA. Disponível em: <https://pt.wikipedia.org/wiki/Vertical>. Acesso em: 23 de jun. 2016.

ANEXO A

Primary Differences between MPU-6000 and MPU-6050

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I^2C, SPI	I^2C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500µA
- Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I^2C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I^2C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 *g* shock tolerant
- 400kHz Fast Mode I^2C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/ √Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.4 Electrical Specifications, Continued

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
SERIAL INTERFACE						
SPI Operating Frequency, All Registers Read/Write	MPU-6000 only, Low Speed Characterization		100 ±10%		kHz	
	MPU-6000 only, High Speed Characterization		1 ±10%		MHz	
SPI Operating Frequency, Sensor and Interrupt Registers Read Only	MPU-6000 only		20 ±10%		MHz	
I^2C Operating Frequency	All registers, Fast-mode			400	kHz	
	All registers, Standard-mode			100	kHz	
I^2C ADDRESS	AD0 = 0		1101000			
	AD0 = 1		1101001			
DIGITAL INPUTS (SDI/SDA, AD0, SCLK/SCL, FSYNC, /CS, CLKIN)						
V_{IH}, High Level Input Voltage	MPU-6000	0.7*VDD			V	
	MPU-6050	0.7*VLOGIC			V	
V_{IL}, Low Level Input Voltage	MPU-6000			0.3*VDD	V	
	MPU-6050			0.3*VLOGIC	V	
C_I, Input Capacitance			< 5		pF	
DIGITAL OUTPUT (SDO, INT)						
V_{OH}, High Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000	0.9*VDD			V	
	R_{LOAD}=1MΩ; MPU-6050	0.9*VLOGIC			V	
V_{OL1}, LOW-Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000			0.1*VDD	V	
	R_{LOAD}=1MΩ; MPU-6050			0.1*VLOGIC	V	
$V_{OL.INT1}$, INT Low-Level Output Voltage	OPEN=1, 0.3mA sink Current			0.1	V	
Output Leakage Current	OPEN=1		100		nA	
t_{INT}, INT Pulse Width	LATCH_INT_EN=0		50		µs	

6.5 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Typical	Units	Notes
Primary I^2C I/O (SCL, SDA)				
V_{IL}, LOW-Level Input Voltage	MPU-6000	-0.5 to 0.3*VDD	V	
V_{IH}, HIGH-Level Input Voltage	MPU-6000	0.7*VDD to VDD + 0.5V	V	
V_{hys}, Hysteresis	MPU-6000	0.1*VDD	V	
VIL, LOW Level Input Voltage	MPU-6050	-0.5V to 0.3*VLOGIC	V	
VIH, HIGH-Level Input Voltage	MPU-6050	0.7*VLOGIC to VLOGIC + 0.5V	V	
Vhys, Hysteresis	MPU-6050	0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	3mA sink current	0 to 0.4	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	3	mA	
	V_{OL} = 0.6V	5	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	
Auxiliary I^2C I/O (AUX_CL, AUX_DA)	**MPU-6050: *AUX_VDDIO*=0**			
V_{IL}, LOW-Level Input Voltage		-0.5V to 0.3*VLOGIC	V	
V_{IH}, HIGH-Level Input Voltage		0.7*VLOGIC to VLOGIC + 0.5V	V	
V_{hys}, Hysteresis		0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	VLOGIC > 2V; 1mA sink current	0 to 0.4	V	
V_{OL3}, LOW-Level Output Voltage	VLOGIC < 2V; 1mA sink current	0 to 0.2*VLOGIC	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	1	mA	
	V_{OL} = 0.6V	1	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	

APÊNDICE A

```
//Autor: Rafael José Mota Ocariz
//Este código tem como propósito coletar os dados dos sensores durante os ensaios
//Data: 21/05/2017

#include <Wire.h>
#include <memorysaver.h>
#include <UTFT.h>
#include <URTouch.h>
#include <URTouchCD.h>

// Declaração da fonte que será usada na interface gráfica.
extern uint8_t BigFont[];

UTFT myGLCD(SSD1289,38,39,40,41);               // Inicia uma instancia da impressão de interface gráfica na tela

// Endereço de comunicação dos sensores
const int MPU=0x68;                             // Endereço de transmissão para os sensores

int AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;          // Variáveis dos sensores
int x1, x2, y1, y2;                             // Dados de coordenadas para desenhar na tela LCD
bool bStop = false;                             // Define quando para o ensaio

int max = 4350, min = 3500;

void setup() {
  // Inicializa comunicação seria com velocidade 9600bits/s
  Serial.begin(9600);

  // Inicializa comunicação com sensores
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);

  // randomSeed(analogRead(0));

  // Inicializa o LCD e define a fonte de texto que será utilizada
  myGLCD.InitLCD(LANDSCAPE);
  myGLCD.setFont(BigFont);
}

void loop() {
  myGLCD.clrScr();                              // Limpa a tela
  myGLCD.fillScr(VGA_WHITE);                    // Define o plano de fundo para branco

  myGLCD.setColor(VGA_BLUE);                    // Define a cor do objeto desenhado na tela
  myGLCD.setBackColor(VGA_BLUE);                // Define a cor de base do texto escrito

  // Definir coordenadas de desenho do retangulo
  x1 = myGLCD.getDisplayXSize()/10;
```

```
  x2 = myGLCD.getDisplayXSize()*9/10;
  y1 = myGLCD.getDisplayYSize()*3/10;
  y2 = myGLCD.getDisplayYSize()*7/10;
  myGLCD.drawRect(x1, y1, x2, y2);              // Desenha o retangulo na tela

  myGLCD.setColor(255,130,0);                   // Define cor do texto
  if (bStop == false)
  {
    myGLCD.print("Aquisitando dados", (myGLCD.getDisplayXSize()/2)-135, (myGLCD.getDisplayYSize()/2)-(myGLCD.getFontYsize()/2));  // Imprime o texto definido no centro da tela
  } else
  {
    myGLCD.print("Aquisicao de dados finalizada", CENTER, CENTER);  // Imprime o texto definido no centro da tela
  }

  while(bStop == false)
  {
    Wire.beginTransmission(MPU);
    Wire.write(0x3B);
    Wire.endTransmission(false);

    Wire.requestFrom(MPU, 14, true);

    AcX = Wire.read()<<8|Wire.read();
    AcY = Wire.read()<<8|Wire.read();
    AcZ = Wire.read()<<8|Wire.read();
    Tmp = Wire.read()<<8|Wire.read();
    GyX = Wire.read()<<8|Wire.read();
    GyY = Wire.read()<<8|Wire.read();
    GyZ = Wire.read()<<8|Wire.read();

    Serial.print(AcX); Serial.print(",");
    Serial.print(AcY); Serial.print(",");
    Serial.print(AcZ); Serial.print(",");
    Serial.print(Tmp); Serial.print(",");
    Serial.print(GyX); Serial.print(",");
    Serial.print(GyY); Serial.print(",");
    Serial.print(GyZ); Serial.print("\n");
  }
}
```

```
//Autor: Rafael José Mota Ocariz
//Este código tem como propósito determinar a verticalidade e imprimir na tela para o usuário
//Data: 10/12/2018

#include <Wire.h>
#include <memorysaver.h>
#include <UTFT.h>
#include <URTouch.h>
#include <URTouchCD.h>
#include <math.h>

// Declaração da fonte que será usada na interface gráfica.
extern uint8_t BigFont[];

UTFT myGLCD(SSD1289,38,39,40,41);               // Inicia uma instancia da impressão de interface gráfica na tela
URTouch myTouch(6,5,4,3,2);                     // Inicia uma instancia do touch

// Endereço de comunicação dos sensores
const int MPU=0x68;                             // Endereço de transmissão para os sensores

float AcX[400], AcY[400], AcZ[400], Tmp, GyX, GyY, GyZ, Time;          // Variáveis dos sensores
float sumAcX = 0, sumAcY = 0, sumAcZ = 0, sumGyX = 0, sumGyY = 0, sumGyZ = 0;               // Soma das variáveis
int x1, x2, y1, y2;                             // Dados de coordenadas para desenhar na tela LCD
int btn1X1, btn1X2, btnY1, btnY2;               // Coordenadas do botão Medir
int btn2X1, btn2X2;                             // Coordenadas do botão Zerar

int i = 0;                                      // indice dos dados medidos no processo de medição

float medidoX = 0, medidoY = 0;                 // Os dados impressos ao usuário
float alfa, beta;                               // Os angulos de Euler calculados
float precision = 0.01f;                        // precisão cobrada dos cálculos

const float DPALLANX = 16.12f;
const float DPALLANY = 18.09f;
const float DPALLANZ = 38.31f;

void setup() {
  // Inicializa comunicação seria com velocidade 9600bits/s
  Serial.begin(9600);

  // Inicializa comunicação com sensores
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);

  // randomSeed(analogRead(0));

  // Inicializa o LCD e define a fonte de texto que será utilizada
  myGLCD.InitLCD(LANDSCAPE);
  myGLCD.setFont(BigFont);
```

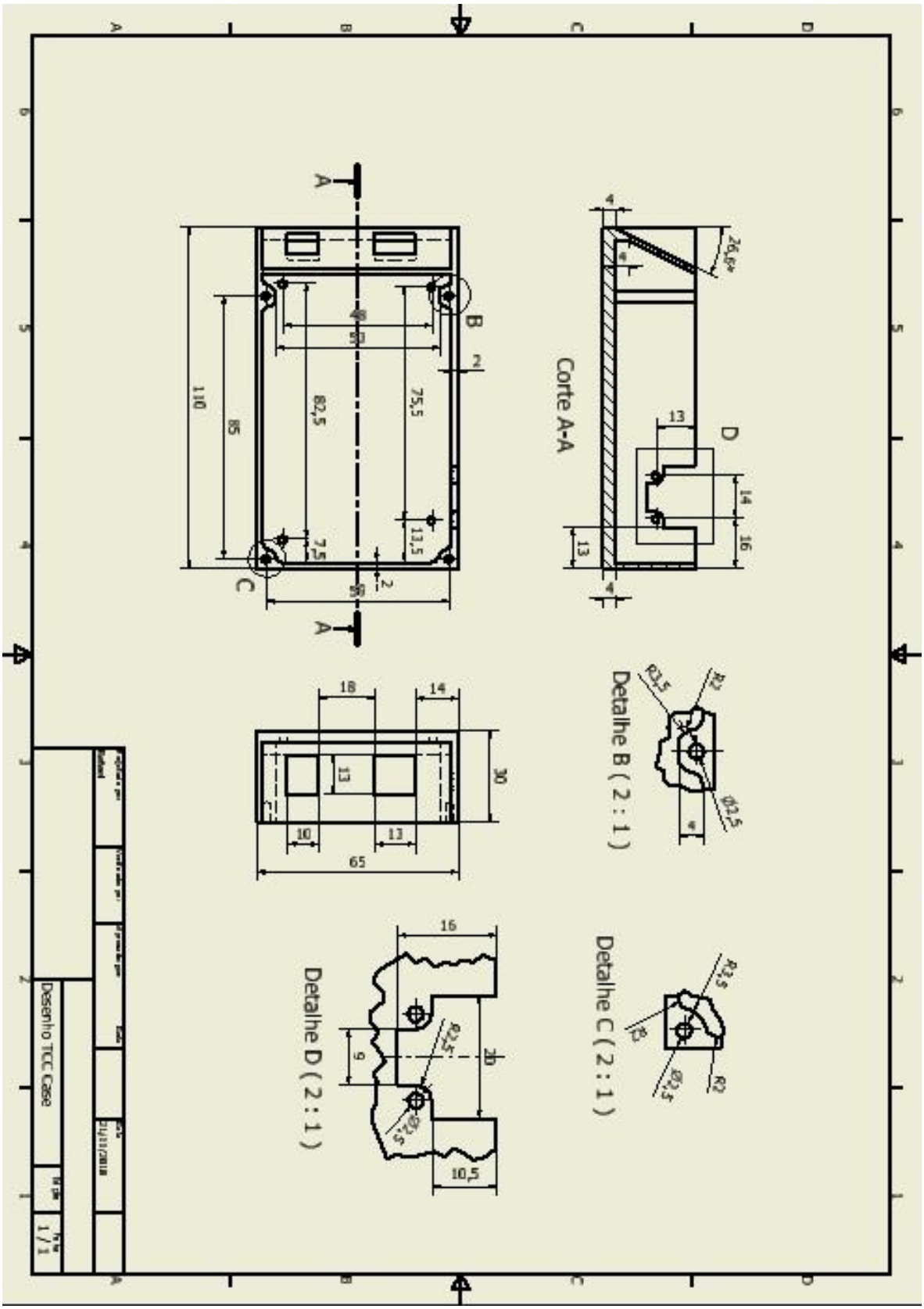
```
  // Inicializa o sistema Touch
  myTouch.InitTouch(LANDSCAPE);
  myTouch.setPrecision(PREC_HI);
}

void loop() {
  myGLCD.clrScr();                        // Limpa a tela
  myGLCD.fillScr(210,210,210);                // Define o plano de fundo para branco
  myGLCD.setBackColor(0,255,0);

  // Impressão do cabeçalho na tela
  myGLCD.setColor(0,255,0);
  myGLCD.fillRect(0,0,myGLCD.getDisplayXSize(), myGLCD.getDisplayYSize()/10);
  myGLCD.setColor(255,255,255);
  myGLCD.print("SDV - TCC Rafael", myGLCD.getDisplayXSize()/2 - 120,
myGLCD.getDisplayYSize()/20-(myGLCD.getFontYsize()/2));

  // Impressão dos dados
  myGLCD.setBackColor(0,0,0);
  myGLCD.print("X", 50, (myGLCD.getDisplayYSize()/5)+myGLCD.getFontYsize());
  myGLCD.printNumF(medidoX, 2, 150,
(myGLCD.getDisplayYSize()/5)+myGLCD.getFontYsize());
  myGLCD.print("Y", 50, (myGLCD.getDisplayYSize()*11/20)-myGLCD.getFontYsize());
  myGLCD.printNumF(medidoY, 2, 150, (myGLCD.getDisplayYSize()*11/20)-
myGLCD.getFontYsize());

  // Imprime o botão Medir
  btn1X1 = myGLCD.getDisplayXSize()/20;
  btn1X2 = myGLCD.getDisplayXSize()*5/20;
  btnY1 = myGLCD.getDisplayYSize()*19/20;
  btnY2 = myGLCD.getDisplayYSize()*17/10;
  myGLCD.setColor(0,0,255);                   // Define a cor do objeto desenhado na tela
  myGLCD.fillRect(btn1X1, btnY1, btn1X2, btnY2);          // Desenha o retangulo na tela
  myGLCD.setColor(80,150,255);
  myGLCD.drawRect(btn1X1, btnY1, btn1X2, btnY2);          // Desenha o retangulo na tela
  myGLCD.setColor(255,0,0);
  myGLCD.setBackColor(80,150,255);                // Define a cor de base do texto escrito
  myGLCD.print("Medir", btn1X1 + 20, (myGLCD.getDisplayYSize()*18/20)-
(myGLCD.getFontYsize()/2));  // Imprime o texto definido no centro da tela

  // Imprime o botão Zerar
  btn2X1 = myGLCD.getDisplayXSize()*13/20;
  btn2X2 = myGLCD.getDisplayXSize()*17/20;
  myGLCD.setColor(0,0,255);                   // Define a cor do objeto desenhado na tela
  myGLCD.fillRect(btn2X1, btnY1, btn2X2, btnY2);          // Desenha o retangulo na tela
  myGLCD.setColor(80,150,255);
  myGLCD.drawRect(btn2X1, btnY1, btn2X2, btnY2);          // Desenha o retangulo na tela
  myGLCD.setColor(255,0,0);
  myGLCD.setBackColor(80,150,255);                // Define a cor de base do texto escrito
  myGLCD.print("Zerar", btn2X1 + 20, (myGLCD.getDisplayYSize()*18/20)-
(myGLCD.getFontYsize()/2));  // Imprime o texto definido no centro da tela

  while(!myTouch.dataAvailable())
  {
    i=0;
```

```
  }

  if(myTouch.dataAvailable())
  {
   myTouch.read();

   // Verifica se o usuário tocou em um botão
   if (myTouch.getY() < 205 && myTouch.getY() > 170)
   {
    if (myTouch.getX() > 45 && myTouch.getX() < 135)
    {
     myGLCD.setColor(0,0,0);
     // Desenha o retangulo na tela para a barra de carregamento
     myGLCD.drawRect(btn1X1, myGLCD.getDisplayYSize()*15/20, btn1X2,
myGLCD.getDisplayYSize()*13/20);
     while (i<400)
     {
      Wire.beginTransmission(MPU);
      Wire.write(0x3B);
      Wire.endTransmission(false);

      Wire.requestFrom(MPU, 14, true);

      AcX[i] = Wire.read()<<8|Wire.read();
      AcY[i] = Wire.read()<<8|Wire.read();
      AcZ[i] = Wire.read()<<8|Wire.read();
      Tmp = Wire.read()<<8|Wire.read();
      GyX = Wire.read()<<8|Wire.read();
      GyY = Wire.read()<<8|Wire.read();
      GyZ = Wire.read()<<8|Wire.read();
      Time = millis();

      sumAcX += AcX[i];
      sumAcY += AcY[i];
      sumAcZ += AcZ[i];

      myGLCD.setColor(100,255,100);
      myGLCD.fillRect(btn1X1, myGLCD.getDisplayYSize()*15/20, btn1X1 + (btn1X2-
btn1X1)*i/1200, myGLCD.getDisplayYSize()*13/20);   // Desenha a barra de carregamento

      i++;
     }

     i=0;
     sumAcX = (sumAcX/400);
     sumAcY = (sumAcY/400);
     sumAcZ = ((sumAcZ/400)-(-2540));

     while (i<400)
     {
      if (AcX[i] > (sumAcX + 2*DPALLANX))
      {
       AcX[i] = sumAcX + 2*DPALLANX;
      } else if (AcX[i] < (sumAcX - 2*DPALLANX))
      {
```

```
          AcX[i] = sumAcX - 2*DPALLANX;
        }

        if (AcY[i] > (sumAcY + 2*DPALLANY))
        {
          AcY[i] = sumAcY + 2*DPALLANY;
        } else if (AcY[i] < (sumAcY - 2*DPALLANY))
        {
          AcY[i] = sumAcY - 2*DPALLANY;
        }

        if (AcZ[i] > (sumAcZ + 2*DPALLANZ))
        {
          AcZ[i] = sumAcZ + 2*DPALLANZ;
        } else if (AcZ[i] < (sumAcZ - 2*DPALLANZ))
        {
          AcZ[i] = sumAcZ - 2*DPALLANZ;
        }

        myGLCD.setColor(100,255,100);
        myGLCD.fillRect(btn1X1, myGLCD.getDisplayYSize()*15/20, btn1X1 + (btn1X2-
btn1X1)*(400+i)/1200, myGLCD.getDisplayYSize()*13/20);   // Desenha a barra de carregamento

        i++;
       }
       i = 0;

       while (i<400)
       {
        sumAcX += AcX[i];
        sumAcY += AcY[i];
        sumAcZ += AcZ[i];

        myGLCD.setColor(100,255,100);
        myGLCD.fillRect(btn1X1, myGLCD.getDisplayYSize()*15/20, btn1X1 + (btn1X2-
btn1X1)*(800+i)/1200, myGLCD.getDisplayYSize()*13/20);   // Desenha a barra de carregamento
        i++;
       }
       i = 0;

       sumAcX = (((sumAcX/400)-344.98f)/16400)*(-1);
       sumAcY = (((sumAcY/400)-(-2.917f))/16108)*(-1);
       sumAcZ = (((sumAcZ/400)-(-2565.7f))/16613)*(-1);

       alfa = atan(sumAcZ/sumAcX);
       beta = atan(-sumAcY/((sumAcX*cos(alfa))+(sumAcZ*sin(alfa))));

       alfa = alfa*57.2958f;          // Convertendo radianos para graus
       beta = beta*57.2958f;          // Convertendo radianos para graus

       medidoX = beta;
       medidoY = alfa;

      } else if(myTouch.getX() > 215 && myTouch.getX() < 300)
      {
```

```
        medidoX = 0;
        medidoY = 0;
      }
    }
  }
}
```

APÊNDICE B


Corte A-A
Detalhe B (2 : 1)
Detalhe C (2 : 1)
Detalhe D (2 : 1)
Desenho TCC Case
110
85
82,5
75,5
65
26,6°
1 / 1