UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE UM MECANISMO EMBARCADO DE AUXÍLIO À PILOTAGEM DE AEROMODELOS

ALEXANDRE ZENDRON

BLUMENAU
2012

2012/1-04

ALEXANDRE ZENDRON

PROTÓTIPO DE UM MECANISMO EMBARCADO DE AUXÍLIO À PILOTAGEM DE AEROMODELOS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Miguel Alexandre Wisintainer, Mestre - Orientador

BLUMENAU
2012

2012/1-04

PROTÓTIPO DE UM MECANISMO EMBARCADO DE AUXÍLIO À PILOTAGEM DE AEROMODELOS

Por

ALEXANDRE ZENDRON

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Miguel Alexandre Wisintainer, Mestre – Orientador, FURB
Membro:	Prof. Antonio Carlos Tavares, Mestre – FURB
Membro:	Prof. Francisco Adell Péricas, Mestre – FURB

Blumenau, 09 de julho de 2012

Dedico este trabalho a minha família, namorada e amigos, especialmente aqueles que me ajudaram, direta ou indiretamente na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

Aos meus pais, Erico e Marieane, que sempre estiveram presentes me apoiando e incentivando.

Aos meus amigos, irmão e principalmente minha namorada, Camila, por terem tido paciência, entenderem meus momentos de ausência e pela marcação cerrada para a conclusão deste trabalho.

Ao meu orientador, Miguel Alexandre Wisintainer, por ter acompanhado e incentivado a elaboração deste trabalho.

O único homem que está isento de seus erros, é aquele que não arrisca acertar.

Albert Einstein

RESUMO

Este trabalho descreve o desenvolvimento de um *software* embarcado que é capaz de interromper a queda de um aeromodelo. O *software* utiliza duas placas para o funcionamento, uma possui sensores inerciais e angulares, acelerômetros e girômetros, enquanto a outra é responsável por realizar os cálculos para interromper a queda do aeromodelo. O calculo utilizado para integrar as medidas dos sensores e remover os ruídos dos mesmos é o filtro de Kalman. Em conjunto com o desenvolvimento do projeto é feito um programa simulador responsável por mostrar no computador gráficos com a relação entre medidas reais dos acelerômetros e medias após a aplicação do filtro de Kalman e o posicionamento do aeromodelo no plano tridimensional.

Palavras-chave: Filtro de Kalman. Acelerômetro. Girômetro. Aeromodelo. Interromper queda.

ABSTRACT

This work describes the firmware development that is capable to stop the fall of an model airplane. This firmware works with two boards, one of them have the inertial sensors like accelerometer and gyroscope while the other one is responsible for the calculations to interrupt the model airplane fall. The calculation used to join the sensors measures and reduce the noise is the Kalman filter. In order to the project development a simulation software was build to present the graphics of the measures calculated by de Kalman filter, and the tridimensional position of the model airplane.

Key-words: Kalman filter. Accelerometer. Gyroscope. Model airplane. Fall interrupt.

LISTA DE ILUSTRAÇÕES

Figura 1 – Triedro – Eixos de movimentação 16
Figura 2 – Superfícies de Controle 17
Figura 3 – Sinal PWM 20
Figura 4 – Servomecanismo 21
Figura 5 – ArduPilot 22
Figura 6 – Atomic IMU 23
Figura 7 – Processing IDE 24
Figura 8 - Servomecanismo ligado ao receptor 26
Figura 9 – FMA Direct 29
Figura 10 – Casos de uso 31
Figura 11 – Diagrama de seqüência 32
Figura 12 – Fluxograma do mecanismo embarcado 33
Figura 13 – Diagrama de classes programa de simulação 34
Figura 14 – Diagrama de classes mecanismo embarcado 34
Figura 15 – Tela de configuração da placa Atomic IMU 36
Quadro 1 – Comandos da placa Atomic IMU 37
Figura 16 – Fluxograma Atomic IMU 37
Figura 17 – Cadeia de caracteres 38
Figura 18 – Circulo trigonométrico no plano cartesiano 39
Figura 19 – Representação dos eixos X, Y e Z no aeromodelo 40
Figura 20 – Posicionamento do aeromodelo conforme resposta da função da primeira coluna 40
Figura 21 – Erro da função `Atan2(x,y)` 41
Figura 22 – Fórmula do estado do sistema 42
Quadro 2 – Implementação da fórmula da Figura 22 42
Figura 23 – Fórmula do calculo do ganho 43
Quadro 3– Trecho do código de atualização do ganho do sistema 43
Figura 24 – Fórmula da atualização da matriz de covariância 43
Quadro 4 – Atualização da matriz de covariância 44
Figura 25 – Fórmula de estimativa do estado 44
Quadro 5 – Implementação da fórmula da Figura 25 44

Quadro 6 – Chamada da função do filtro de Kalman 44
Figura 26 – Horizonte virtual 45
Figura 27 – Modelos tridimensionais 46
Figura 28 – Gráfico de variação das medições 46
Quadro 7 – Código do método `setup()` 47
Quadro 8 – Processo de leitura da amostra dos sensores 47
Quadro 9 – Função `getPosiçãoProfundor()` 49
Quadro 10 – Chamada do método `kalman()` 50
Quadro 11 – Método `kalman()` 50
Figura 29 – Conexões entre as placas Atomic IMU e ArduPilot 51
Figura 30 – Centro de gravidade 52
Figura 31 – Montagem da placa Atomic IMU no aeromodelo 53
Figura 32 – Esquema de ligação do circuito original 54
Figura 33 – Esquema de ligação do circuito com a placa Atomic IMU 54
Figura 34 – Posição comando aileron 55
Figura 35 – Posição do comando profundor 55
Figura 36 – Chave responsável por acionar o mecanismo 56
Figura 37 – Posicionamento do aeromodelo 57
Figura 38 – Gráfico eixo x com pouco ruído 58
Figura 39 – Gráfico eixo x com muito ruído 58
Quadro 12 – Características do protótipo desenvolvido e trabalhos correlatos 59

LISTA DE SIGLAS

3D – Três Dimensões

A/D – Analógico / Digital

ASCII – *American Standard Code for Information Interchange*

BCC – Curso de Ciência da Computação – Bacharelado

CC – Corrente Continua

CPU – *Center Processing Unit*

DOF – *Degrees Of Freedom*

DSC – Departamento de Sistemas e Computação

E/S – Entada / Saída

FM – *Frequency Modulation*

GPS – *Global Positioning System*

IDE – *Integrated Development Enviroment*

IMU – *Inertial Measuremet Unit*

MIT - *Massachusetts Institute of Technology*

PWM – *Pulse-Width Modulation*

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

ROM – *Read Only Memory*

SCV – Sistema de Controle de Vôo

SNI – Sistema de Navegação Inercial

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS DO TRABALHO 14
1.2 ESTRUTURA DO TRABALHO 14
2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 AEROMODELO 15
2.1.1 Eixos do aeromodelo 15
2.2 NAVEGAÇÃO 16
2.2.1 Superfícies de controle 17
2.2.2 Navegação inercial 17
2.3 RECURSOS 18
2.3.1 Micro controlador 18
2.3.1.1 ATMEGA328 19
2.3.2 Sensores / atuadores 20
2.3.3 ArduPilot 21
2.3.4 Atomic IMU 6 Degrees Of Freedom (DOF) 22
2.3.5 Processing IDE 23
2.3.6 Arduino IDE 24
2.4 CONTROLE REMOTO 25
2.4.1 Transmissor 25
2.4.2 Receptor 25
2.5 AUTOMAÇÃO 26
2.5.1 Sistemas embarcados 26
2.5.2 Sistemas de controle 27
2.5.3 Sistema de malha fechada 27
2.5.4 Filtro de Kalman 28
2.6 TRABALHOS CORRELATOS 28
3 DESENVOLVIMENTO DO PROTÓTIPO 30
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 30
3.2 ESPECIFICAÇÃO 31
3.2.1 Casos de uso 31

3.2.2 Diagrama de sequência 31
3.2.3 Fluxograma 33
3.2.3.1 Diagrama de classes 33
3.3 IMPLEMENTAÇÃO 35
3.3.1 Técnicas e ferramentas utilizadas 35
3.3.1.1 Hardware utilizado 35
3.3.1.2 `Atan2(x,y)` 38
3.3.1.3 Algoritmo de Kalman 41
3.3.1.4 Desenvolvimento do programa de simulação 45
3.3.1.5 Desenvolvimento do software embarcado 48
3.3.2 Operacionalidade da implementação 51
3.3.2.1 Conexões da placa ArduPilot e Atomic IMU 51
3.3.2.2 Montagem da placa Atomic IMU no aeromodelo 52
3.3.2.3 Conexões do mecanismo embarcado x receptor FM 53
3.3.2.4 Verificar a orientação das superfícies de controle 55
3.3.2.5 Ativação do mecanismo durante o voo 56
3.4 RESULTADOS E DISCUSSÃO 56
4 CONCLUSÕES 60
4.1 EXTENSÕES 61
REFERÊNCIAS BIBLIOGRÁFICAS 62

1 INTRODUÇÃO

Desde a década de quarenta os sistemas de navegação, em especial os Sistemas de Navegação Inercial (SNI) tornaram-se importantes componentes em aplicações científicas e militares (ROCHA, 2006).

Com a miniaturização dos componentes eletrônicos e o incremento do poder de processamento dos computadores, tornou-se possível o desenvolvimento de componentes inerciais de baixo custo, possibilitando seu uso em aeronaves não tripuladas, desempenhando um papel muito importante em muitas outras aplicações não militares (SAMPAIO, 2006, p. 19).

Beneficiando-se das inovações constantes da mecatrônica, universidades e empresas que atuam na área aeronáutica criam e aprimoram produtos com aplicações nas mais diversas áreas. A aeronáutica foi uma grande beneficiada com o avanço destas tecnologias, tendo em vista que o emprego de tecnologias é limitado devido a restrições de peso, dimensões e custo.

Sistemas de navegação inercial são responsáveis por fornecer informações precisas e confiáveis de localização em três Dimensões (3D), em especial quando aplicados em aeronaves que normalmente apresentam considerável instabilidade de equilíbrio quando comparadas a veículos terrestres ou aquáticos (LANARI; BORGES, 2007, p. 1). Em conseqüência deste fato, é grande o interesse de desenvolvimento de sistemas capazes de fornecer informações de posição em tempo real a partir de componentes de baixo custo.

O fato de ser possível identificar em tempo real a posição em que a aeronave se encontra no espaço, permite o desenvolvimento de um sistema capaz de agir sobre as superfícies de controle de uma aeronave a cada instante.

Levando em consideração as informações descritas acima, opta-se por utilizar um aeromodelo para aplicação de um sistema capaz de interromper a queda de uma aeronave, pois este dispõe das mesmas características de uma aeronave, porém em escala reduzida.

O projeto irá dispor de sensores inerciais como girômetros e acelerômetros fixados no centro de gravidade do aeromodelo, bem como um dispositivo responsável por interpretar as informações fornecidas pelos sensores, identificando assim o erro em relação à posição de equilíbrio do aeromodelo, atuando a cada instante nas superfícies de controle do mesmo até que a estabilidade seja atingida, trabalhando de forma semelhante a um piloto automático. O acionamento deste dispositivo será feito pelo piloto através do acionamento de uma chave

contida no rádio controle, este responsável por controlar o aeromodelo.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um sistema que seja capaz de interromper a queda do aeromodelo a partir do acionamento do mecanismo pelo piloto, estabilizando o mesmo e tendo como referências de posicionamento leituras providas por sensores fixados no mesmo.

Os objetivos específicos do trabalho são:

a) capturar o posicionamento angular do aeromodelo;
b) interpretar o posicionamento do aeromodelo para proceder com a tomada de decisão para a correção do nivelamento;
c) acionar os servo mecanismos para proceder com a correção de nível do aeromodelo.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em quatro capítulos. No capítulo dois é descrita toda a fundamentação teórica utilizada como referência durante a elaboração e desenvolvimento do protótipo, apresenta também os recursos de *hardware* e *software* utilizados como a plataforma de desenvolvimento Arduíno e Processing, as placas ArduPilot e Atomic IMU, escolhidas para o desenvolvimento do protótipo e por fim são apresentados alguns trabalhos correlatos existentes no mercado atualmente.

No capitulo três é detalhado todo o desenvolvimento do protótipo, as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

Por fim no capítulo quatro são apresentadas as conclusões e sugestões para a extensão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos que embasam e qualificam o desenvolvimento do protótipo. A seção 2.1 explana de forma breve o que é um aeromodelo. Na seção 2.2 é descrito o conceito de navegação. Na seção 2.3 são abordados os recursos utilizados para o desenvolvimento do protótipo, como ferramentas de desenvolvimento e as placas controladoras utilizadas para a elaboração do mesmo. Na seção 2.4 é explanado o modo como o piloto interage com o aeromodelo. Na seção 2.5 são apresentados elementos de automação, entre eles são citados sistemas de malha fechada e filtro de Kalman. E por fim na seção 2.6 são apresentados trabalhos correlatos ao tema proposto.

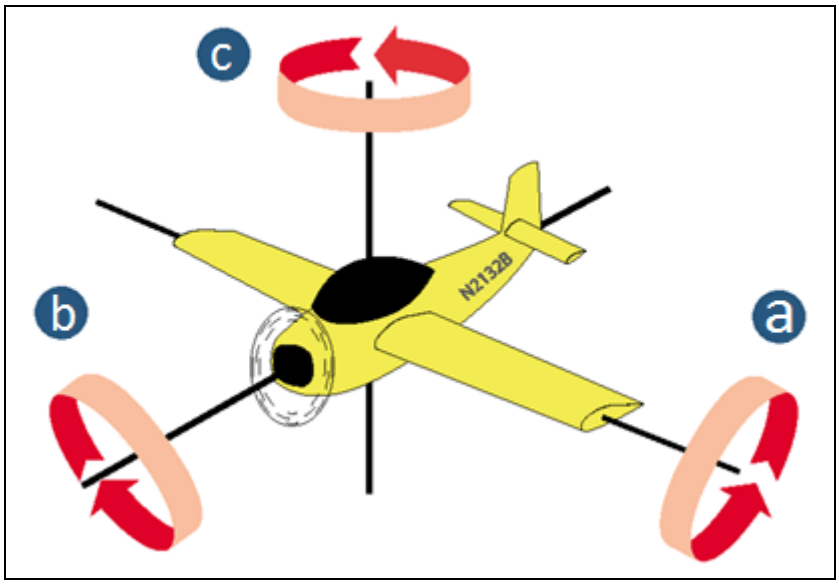
2.1 AEROMODELO

> O modelo de um aeroplano é uma versão reduzida de um aeroplano em seu tamanho normal. Os modelos e os objetos em seu tamanho normal têm os mesmos tipos de relação entre os comprimentos de suas diferentes partes. [...] Um modelo é apenas um meio de transferir alguma relação de sua forma real para outra forma. (BOLTON, 1993, p. 4).

Quando se fala em aeromodelos pode-se pensar em apenas um *hobby*, porém seu uso pode e é abrangido em outras áreas. Um exemplo do uso do aeromodelo é sua utilização como forma de estudo de aerodinâmica e há pouco tempo serve como instrumento de patrulha aérea. O aeromodelo atua também na área de testes de novas tecnologias, servindo como protótipo de desenvolvimento e testes.

2.1.1 Eixos do aeromodelo

Conforme apresentado por Beires (1964, p. 22), o avião pode movimentar-se no espaço em todas as direções. Para entender estes movimentos é necessário traçar três eixos pelo seu centro de gravidade, formando assim um triedro (Figura 1).



Fonte: adaptado de Caluta (2004).

Figura 1 – Triedro – Eixos de movimentação

As nomenclaturas que representam os movimentos do aeromodelo no espaço são descritas a seguir:

a) picar ou cabrar: este movimento traduz-se pelo ângulo de descida ou de subida respectivamente. Ocorre quando o mesmo gira em torno do eixo transversal;
b) bancar ou rolar: manifesta-se pelo soerguimento de uma ou outra asa. Corresponde a inclinação em torno do eixo longitudinal;
c) guinar: movimento em que é efetuada uma rotação em torno do eixo vertical.

As superfícies responsáveis por estes movimentos são descritas no item 2.2.1 e ilustradas na Figura 2 na página 17.

2.2 NAVEGAÇÃO

"Ao processo de determinação da posição e da velocidade ao longo do tempo de um dado veículo em relação a um sistema de coordenadas previamente fixado, dá-se o nome de navegação" (OLIVEIRA, 2000, p. 31).

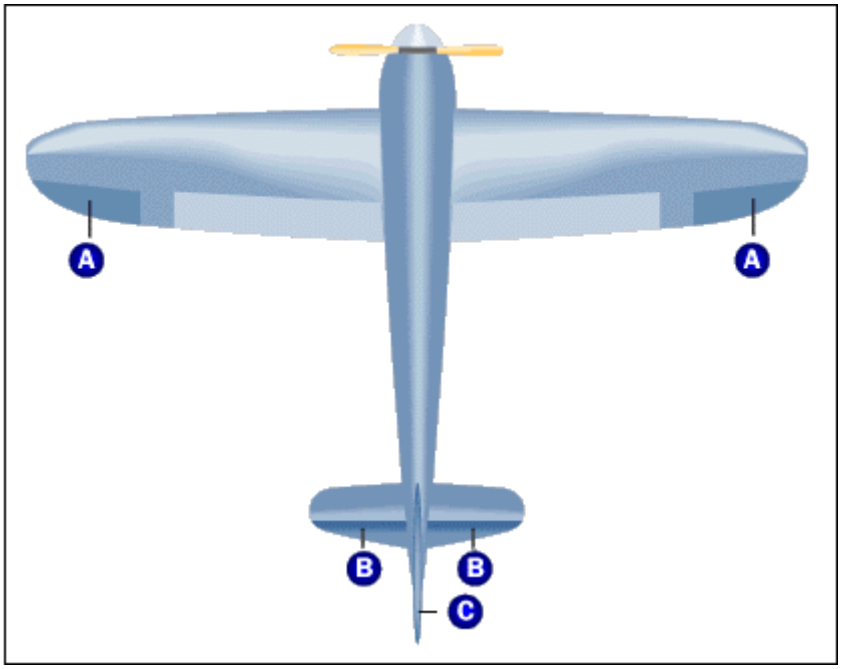
Ainda, Oliveira (2000, p. 31) coloca que uma das tarefas do sistema de navegação é a determinação da orientação de um triedro fixo ao corpo em relação a um triedro externo ao corpo (normalmente considerado como inercial).

2.2.1 Superfícies de controle

Para movimentar-se no espaço aéreo o aeromodelo faz uso de suas superfícies de controle ativadas através de servomecanismos, abordados na seção 2.3.2.

Suas superfícies de controle estão localizadas nos estabilizadores horizontais e verticais (ilustrados na Figura 2):

a) aileron: identificado pela letra “A” está localizado nas extremidades da asa à frente do centro da fuselagem do aeromodelo e é responsável por efetuar a rolagem do mesmo, ou seja, proporciona um movimento de rotação em relação ao seu eixo longitudinal;
b) profundor: identificado pela letra “B” está localizado na asa horizontal fixada no final da fuselagem e serve para controlar a altitude do aeromodelo;
c) leme: identificado pela letra “C” está localizado também no final da fuselagem do aeromodelo na forma de uma quilha.



Fonte: adaptado de Sensores (2000).

Figura 2 – Superfícies de controle

2.2.2 Navegação inercial

Sakajiri (2008, p. 19) define os sistemas de navegação inercial como os únicos meio de

navegação que não necessitam de informações externas para esta tarefa. Um sistema de navegação inercial possui sensores inerciais, girômetros e acelerômetros, que medem as velocidades angulares e acelerações lineares e por meio de algoritmos que calculam a posição atual da plataforma, atuam de maneira a mudar seu posicionamento para a posição desejada.

2.3 RECURSOS

Nesta sessão são tratados os itens de *hardware* e *software* utilizados para o desenvolvimento do mecanismo de controle e do sistema de testes.

2.3.1 Micro controlador

Em geral os micro controladores são processadores de pequeno porte e baixo custo, possuindo em um único *chip* o processador, os registradores, a memória, as interfaces e os barramentos necessários para o seu funcionamento. Na história dos semicondutores, os micro processadores foram os que mais rápido se difundiram, devido as suas características como *design* simples, alta velocidade, grande diversidade de modelos e fabricantes, entre outras (PEREIRA, 2007, p. 17).

Os micro controladores funcionam de maneira dedicada e são utilizados para controlar funções específicas. São encontrados geralmente dentro de outros dispositivos como: celulares, computadores de mão, MP3 *players*, entre outros. Possuem uma memória *Read Only Memory* (ROM) onde é armazenado o *software*, uma memória *Random Access Memory* (RAM) dedicada para o armazenamento de informações temporárias e uma *Central Processing Unit* (CPU) responsável por processar os dados do sistema. Os micro controladores são dotados também de conexões paralelas, permitindo a interação com outros dispositivos (BRAIN, 2009, p. 2).

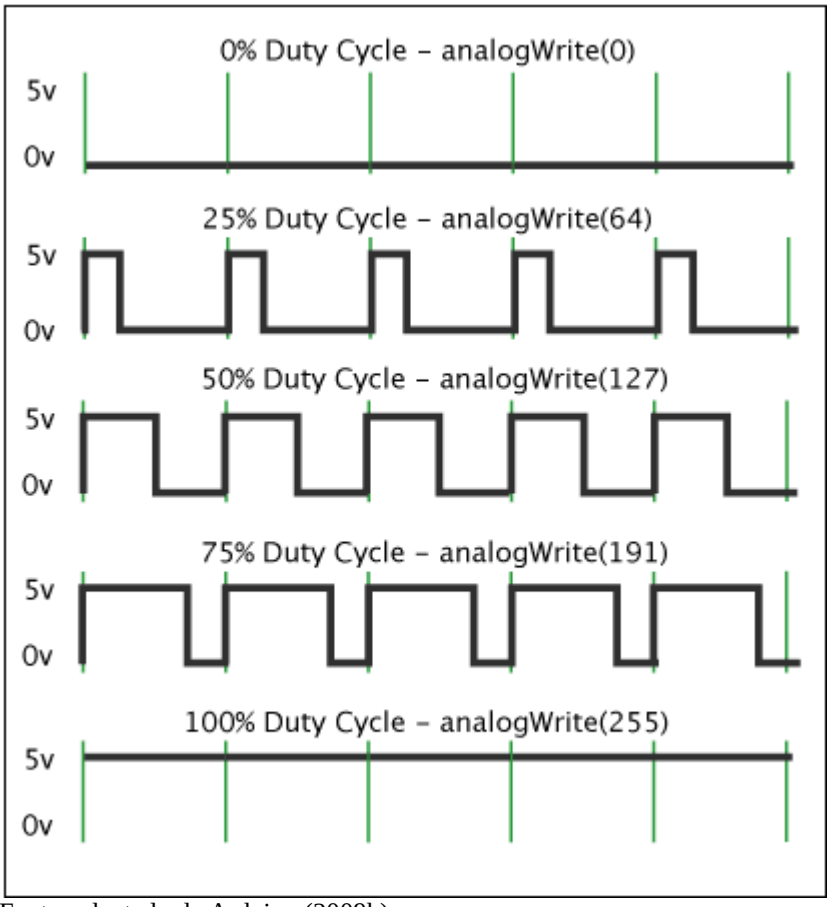
2.3.1.1 ATMEGA328

O micro controlador ATMEGA é fabricado pela empresa Atmel Corporation fundada em 1984. A empresa tem em seu escopo de produção memórias EEPROM e Flash, micro controladores com a arquitetura ARM e também com arquitetura própria Atmel AVR, chips para *smartcard*, circuitos integrados para automóveis e outros componentes eletrônicos.

Segundo Atmel (2010), o micro controlador ATMega 328 é um micro controlador de 8 bits, com Arquitetura Harvard e pertence a família AVR da Atmel. Todos os modelos desta família compartilham uma arquitetura e conjunto de instruções básicas. Possui três portas I/O digitais (PORT B, C e D), conversor Analógico/Digital (A/D), comparador analógico e interfaces seriais.

O micro processador possui 32 pinos, desses, 23 podem ser utilizados como entrada ou saída digital, inclusive os pinos de entrada analógica. Os pinos de entrada e saída estão divididos em três portas PB, PC e PD, mas cada pino pode ser configurado independentemente como entrada ou saída. Os pinos digitais também podem transmitir sinais do tipo *Pulse-Width Modulation* (PWM) (ATMEL, 2010).

PWM é uma técnica utilizada para obter resultados analógicos através de meios digitas. O controle funciona como uma onda quadrada que alterna entre ligada e desligada. O tempo de duração em que o sinal fica ligado é chamado de largura do pulso. Na Figura 3 é possível ver o funcionamento deste sinal com quatro diferentes tempos de duração (ARDUINO, 2009b).



Fonte: adaptado de Arduino (2009b).

Figura 3 – Sinal PWM

2.3.2 Sensores / atuadores

Um sensor é definido como um dispositivo que recebe e responde a um estímulo ou sinal. Sensores são largamente usados na indústria e robótica. Entre eles, cita-se:

a) acelerômetro: é um dispositivo eletromecânico com tamanho e custos reduzidos. Composto basicamente por duas superfícies de cerâmica polarizadas que quando submetidas a pressão mecânica geram uma carga elétrica, a qual é proporcional á força aplicada, sendo esta precisamente calculada utilizando a segunda lei de Newton, que diz que força é igual a massa vezes a aceleração (COELHO, 2007);

b) girômetro: conforme descrito por Oliveira (2000, p. 91) em “Estudo Estatístico dos Processos Envolvidos em uma Plataforma de Atitude Solidária”, funciona como um rotor animado de uma alta velocidade de rotação, apoiado em uma montagem que permite o eixo de rotação inclinar-se livremente em relação à base na qual está montado. Esta propriedade permite descobrir qual a variação angular em relação à

posição anterior;

c) servomecanismo: é um exemplo de mecanismo atuador, responsável por fazer a associação entre as partes eletrônicas e mecânicas. Este dispositivo é controlado eletronicamente através do sinal PWM. Este por sua vez dispõe de uma superfície giratória a qual realiza uma rotação de até 180º de acordo com o sinal PWM recebido. O servo mecanismo é apresentado na Figura 4.

Fonte: Futaba (2008).
Figura 4 – Servomecanismo

2.3.3 ArduPilot

A placa ArduPilot foi criada com o objetivo de ser utilizada no desenvolvimento de mecanismos de piloto automático para Veículos Não Tripulados (do inglês *Unmanned Aerial Vehicle* - UAV). Esta placa não possui sensores, entretanto é possível utilizar as entradas A/D para conectar os sensores ou, como no caso deste projeto, utilizar a comunicação serial como forma de interação com a placa que possui os sensores.

A mesma pode ser utilizada para controlar carros, barcos ou aviões autônomos. Possui um controle contra falhas em caso de perda do sinal do rádio controle, utilizado para controlar o modelo, transferindo o controle do modelo para placa e depois novamente para o piloto.

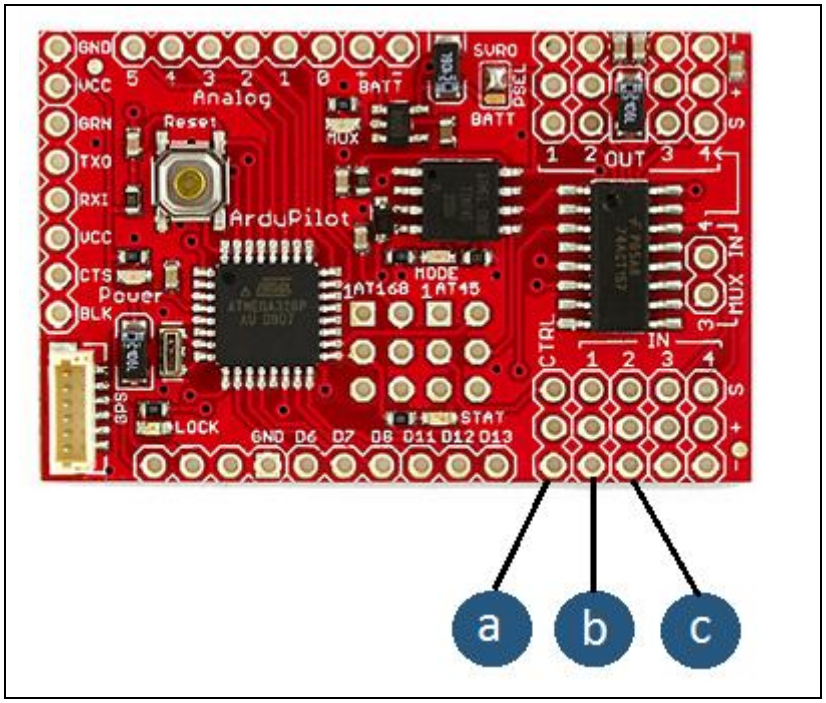
Esta placa (Figura 5) possui cinco entradas para canais PWM, dos quais cada um é formado pela junção de três pinos, um positivo, um negativo e um responsável por transmitir o sinal PWM. Para a realização do protótipo apenas três canais são utilizados, são eles:

a) canal ctrl: é responsável por informar quem comanda os servo mecanismos conectados na placa, quando este canal recebe um sinal PWM com a característica ligado, por mais de cinquenta por cento do tempo, ele delega o controle dos servos para a placa, caso este sinal esteja com a característica desligado o controle dos

servos passa a ser do piloto;

b) canal 1: é um canal PWM que recebe o sinal no aileron;

c) canal 2: é um canal PWM que recebe o sinal do profundor.



Fonte: adaptado de SparkFun (2009).

Figura 5 – ArduPilot

2.3.4 Atomic IMU 6 Degrees Of Freedom (DOF)

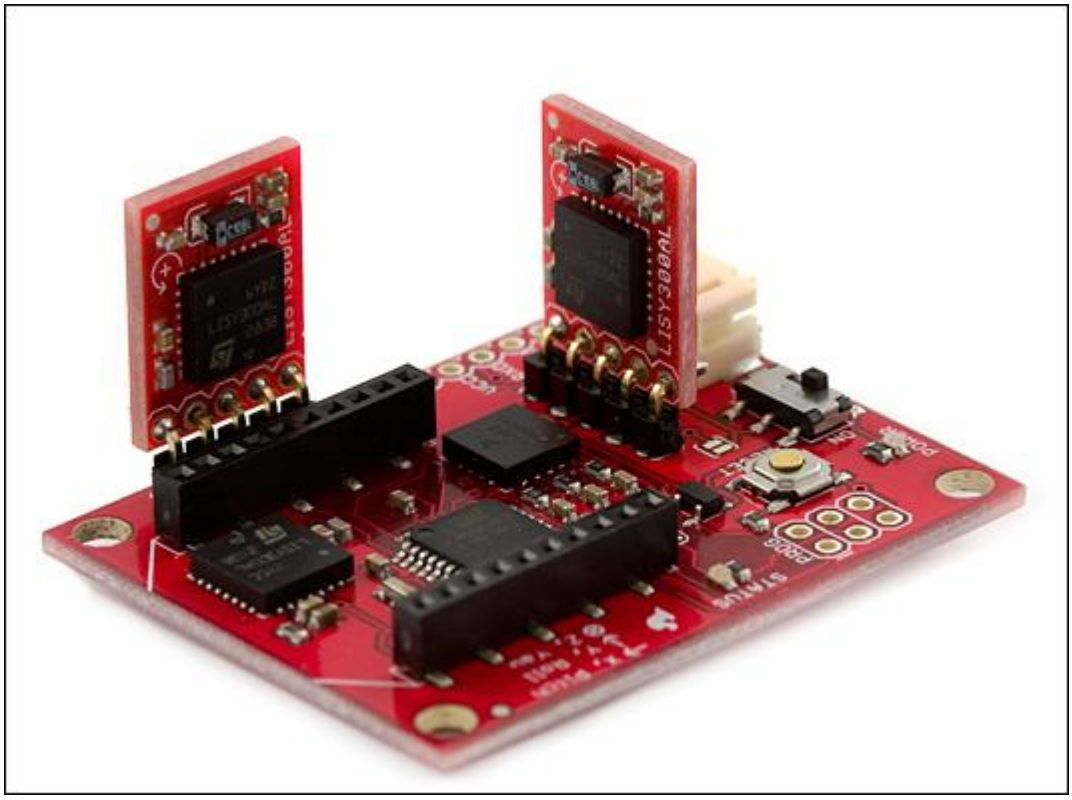
A placa Atomic IMU é tida como uma unidade de medição inercial, desenvolvida para capturar as movimentações que ocorrem nos eixos X, Y e Z e entregar estas medições com uma boa performance. Esta unidade pode ser conectada a outro dispositivo através de um módulo XBee[1] ou por uma conexão serial via cabo e é alimentada por uma fonte de tensão de Corrente Continua (CC) que pode variar de 3.4V até 10V.

A unidade possui um código de fábrica que é responsável por fazer a medição dos sensores por meio da conversão analógico/digital e entregar todas as medições em uma cadeia de caracteres através de sua porta serial. O código fonte desta placa é gratuito e pode ser baixado no mesmo *site* onde a mesma é adquirida.

A unidade é composta pelos seguintes componentes:

a) um processador ATMega328 que contém o código embutido e faz as conversões analógico/digital das medições dos sensores e envia através de comunicação serial;
b) um acelerômetro de três eixos MMA7260Q;
c) três girômetros LISY300AL dispostos de maneira que cada um possa medir um eixo de movimento.

Na Figura 6 pode-se ver a imagem da placa.



Fonte: SparkFun (2009).

Figura 6 – Atomic IMU

2.3.5 Processing IDE

Processing é uma linguagem de programação, ambiente de desenvolvimento e uma comunidade *on-line* desde 2001. Foi criada por dois estudantes do *Massachusetts Institute of Technology* (MIT) para inicialmente servir como uma ferramenta voltada para o ensino da computação gráfica, porém rapidamente evoluiu para uma ferramenta capaz de criar trabalhos profissionais.

---

[1] XBee é um dispositivo sem fio de 2.4GHz utilizado para comunicação serial (SPARK FUN, 2009).

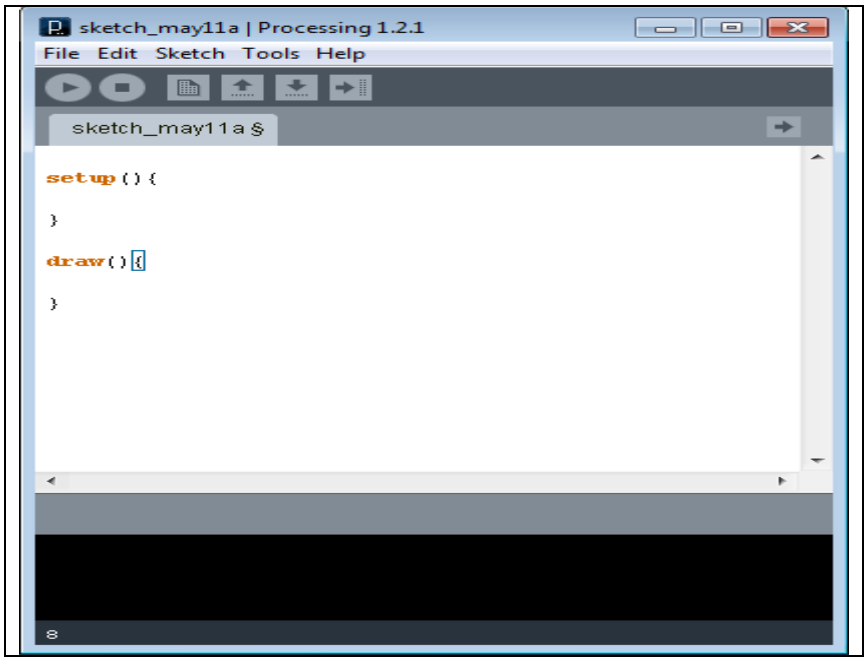


Figura 7 – Processing IDE

O ambiente Processing IDE é muito simples e possui uma *interface* amigável e de fácil entendimento. Processing possui dois métodos principais que não podem ser chamados em seu programa, estes métodos são: `setup()` e `draw()`. Estes métodos são reservados do ambiente Processing. O método `setup()` é chamado imediatamente ao iniciar o programa, onde são definidos alguns parâmetros do sistema, como tamanho da tela e comunicação serial. O método `draw()` é chamado a cada atualização de *frame* (esta controlada pelo ambiente Processing IDE), por padrão chamado 60 vezes por segundo, que é a taxa padrão definida pelo ambiente para atualização de tela. Este método contém as chamadas para as rotinas responsáveis por desenhar a tela.

2.3.6 Arduino IDE

O Arduino IDE é uma aplicação multiplataforma escrita em Java na qual é derivada dos projetos Processing e Wiring[2]. Foi desenvolvido a fim de introduzir a programação de mecanismo de hardware a pessoas menos familiarizadas com o desenvolvimento de *software*. Possui uma interface amigável, permitindo enviar o programa para o dispositivo de *hardware*

[2] Wiring é uma plataforma de prototipagem eletrônica composta por uma linguagem de programação e um ambiente de desenvolvimento.

com apenas um clique.

Para programar em Arduino é necessário definir apenas duas funções, `setup()` e `loop()`. Como mencionado em 2.3.5 o método `setup()` serve para definir parametrizações de início do sistema e é o primeiro método a ser chamado quando o programa é iniciado. O método `loop()` é chamado logo após o método `setup()` finalizar sua execução e como o nome do método sugere, ele é o laço principal do programa, sendo chamado novamente ao final de sua execução, até que o circuito seja desligado.

2.4 CONTROLE REMOTO

Controle remoto é o nome que se dá ao dispositivo que controla um aparelho através de ondas infravermelho ou rádio frequência, ou qualquer dispositivo que controle um objeto sem a necessidade de contato físico com o mesmo.

O controle remoto é possui duas partes, transmissor e receptor, que serão tratadas nos itens 2.4.1 e 2.4.2, respectivamente.

2.4.1 Transmissor

O transmissor é o dispositivo que fica em poder da pessoa que está controlando o objeto proposto. Através deste dispositivo o controlador informa quais ações o objeto controlado deve tomar.

Sua comunicação com o dispositivo receptor é feita através de ondas infravermelho ou sinais de rádio frequência.

O modelo de transmissor utilizado para controlar o aeromodelo é o modelo SkySport4 da empresa Futaba.

2.4.2 Receptor

O receptor como o próprio nome já diz é um módulo responsável por receber os sinais

enviados pelo transmissor e deve ser instalado no objeto o qual se está controlando.

O receptor neste caso converte o sinal recebido para um sinal PWM que é enviado para os servomecanismos que estão conectados por um meio físico a ele e que por sua vez estão ligados mecanicamente as superfícies de controle. A ligação entre servomecanismo e receptor é vista na Figura 8.

O modelo de receptor utilizado para controlar o aeromodelo é o modelo R168DF da empresa Futaba.

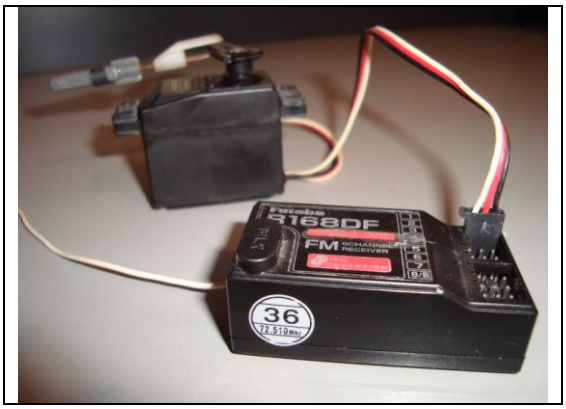


Figura 8 - Servomecanismo ligado ao receptor

2.5 AUTOMAÇÃO

Automação é um sistema automático de controle pelo qual os mecanismos verificam seu próprio funcionamento, efetuando medições e introduzindo correções, sem a necessidade da interferência do homem (AUTOMAÇÃO, 2012). Segundo Lanari (2004 apud AUTOMAÇÃO, 2012) considera-se como automação a aplicação de técnicas computadorizadas ou mecânicas para diminuir o uso de mão de obra em qualquer processo, especialmente o uso de robôs na linha de produção.

Abaixo são abordados quatro itens considerados para o desenvolvimento do protótipo embarcado.

2.5.1 Sistemas embarcados

Segundo Embedded (2012), um sistema embarcado, ou *firmware*, é um sistema micro

processado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Suas primeiras aplicações são datadas de 1960, porém sua utilização foi amplamente difundida em 1980 quando componentes externos foram integrados ao mesmo chip do processador, permitindo assim uma redução no tamanho do circuito e aumento de suas funcionalidades.

Diferente de computadores pessoais, um sistema embarcado realiza um conjunto de tarefas pré-definidas, geralmente com requisitos específicos, desta maneira através da engenharia pode-se aperfeiçoar o projeto reduzindo seu tamanho, recursos computacionais e custo do produto (EMBEDDED, 2012).

Sistemas embarcados são encontrados em: MP3 Players, computadores de bordo em veículos, semáforos, etc.

2.5.2 Sistemas de controle

Um Sistema de Controle de Voo (SCV) é um conjunto de sistemas computacionais embarcados ou não na aeronave, que permite controlar subidas, descidas, nivelamentos, entre outros. O principal objetivo de um SCV de uma aeronave é determinar e controlar a atitude e posição durante o voo. A atitude é a orientação de uma aeronave no espaço de acordo com um sistema de referência fixo na própria aeronave (SAMPAIO, 2006, p. 72).

2.5.3 Sistema de malha fechada

O sistema adotado para o desenvolvimento do protótipo consiste em um sistema de controle de malha fechada, descrito por Bolton (1993) como um sistema que é realimentado da saída para a entrada e usado para modificar a entrada, de tal maneira que a saída seja mantida constante mesmo com modificações nas condições de operação.

Um sistema de malha fechada é constituído por subsistemas básicos, sendo eles:

a) elemento de comparação: compara o valor desejado, ou de referência, da variável controlada com o valor medido e determina o sinal de erro que indica quanto o valor da saída está desviado do valor desejado;
b) elemento de controle: decide qual ação tomar quando recebe um sinal de erro;

c) elemento atuador: é usado para provocar uma mudança no processo de forma a corrigir o erro;
d) processo: o processo ou planta é o sistema no qual uma variável está sendo controlada;
e) elemento de medida: gera um sinal relacionado com a condição da variável que está sendo controlada e fornece um sinal de realimentação para o elemento de comparação, para que ele determine se existe um erro.

2.5.4 Filtro de Kalman

O filtro de Kalman teve sua origem na década de sessenta, quando o então matemático Rudolph E. Kalman publicou seu famoso artigo descrevendo um processo recursivo capaz de solucionar problemas relacionados a filtragem de dados em sistemas de controle elétricos (AIUBE, 2005).

O filtro de Kalman consiste em um conjunto de equações matemáticas resultando em um eficiente processo recursivo de estimação de estados, podendo ser estimados os estados passados, o estado presente e os estados futuros. Este tem por objetivo utilizar medições de grandezas realizadas ao longo do tempo, prejudicadas pelo ruído e gerar resultados que se aproximam dos valores reais das grandezas medidas (KALMAN, 2012).

Atualmente o filtro pode ser encontrado em diversas aplicações e é uma parte essencial no desenvolvimento de tecnologias espaciais e militares, podendo ser encontrado também em aplicações mais simples como em rádios FM (KALMAN, 2012).

2.6 TRABALHOS CORRELATOS

Em relação aos trabalhos correlatos têm-se *"Co-Pilot Flight Stabilization System"* (FMA DIRECT, 2007) e o "Sistema de Controle de Atitude Embarcado para Vôo Autônomo de Aviões em Escala" (SAMPAIO, 2006).

O FMA Direct (2007) é um sistema comercial que serve como um sistema de estabilização de voo. O funcionamento deste sistema consiste em medir através de sensores

infravermelho a diferença entre a terra e o dióxido de carbono na atmosfera, fornecendo em tempo real durante o dia ou durante a noite a posição em que se encontra. A sua ação pode ser visualizada como o equivalente a uma corda esticada no horizonte, entre a terra e o espaço e o modelo se mantém sempre com esta corda posicionada em seu centro, de modo que ele somente assume outra posição se comandado pelo piloto. O sistema entra em funcionamento a partir do momento em que os controles são colocados na posição central, entendendo desta maneira que o modelo deva ser estabilizado. Porém, este sistema não funciona perfeitamente quando o tempo encontra-se nublado, devido a interferência causada pelas nuvens sobre os sensores. A Figura 9 apresenta a unidade de controle e a unidade dos sensores do FMA Direct.

Fonte: FMA Direct (2007).

Figura 9 – FMA Direct

Sampaio (2006) desenvolveu um sistema que possui uma arquitetura embarcada dotada de sensores inerciais, tendo seu funcionamento de forma interativa com uma base de controle a qual envia informações de trajetória e o sistema embarcado, através do *Global Positioning System* (GPS), determina as direções a serem seguidas. O projeto tem propósito para ser utilizado em sistemas de monitoramento ambiental e fins militares, objetivando fazer fotografias aéreas.

3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo apresenta o desenvolvimento do protótipo do mecanismo, bem como sua utilização.

São abordadas as ferramentas de desenvolvimento utilizadas e as dificuldades encontradas durante o desenvolvimento do protótipo.

O protótipo terá seu funcionamento demonstrado com um aeromodelo suspenso por elásticos.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O protótipo deverá possuir os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF):

a) o protótipo deverá utilizar servomecanismos para a interação com as superfícies de controle do aeromodelo (RF);
b) o sistema deverá ser ativado e desativado pelo piloto (RF);
c) a comunicação entre as placas será feita de forma digital (RNF);
d) a placa Atomic IMU deverá somente realizar a leitura e transmissão dos valores obtidos pelos sensores (RF);
e) a placa ArduPilot deverá realizar os cálculos de correção do posicionamento do aeromodelo (RF);
f) a placa ArduPilot deverá realizar o acionamento dos servomecanismos (RF);
g) o sistema deverá interromper a queda do aeromodelo (RF);
h) o sistema deverá utilizar sensores para efetuar as leituras do posicionamento do aeromodelo (RNF);
i) o sistema de tomada de decisão e leitura de informações deverá ser implementado em Arduino (RNF).

3.2 ESPECIFICAÇÃO

A ferramenta utilizada para o desenvolvimento dos diagramas de casos de uso, diagrama de sequência e diagrama de classes foi o Enterprise Architect e para a confecção do fluxograma foi utilizada a ferramenta Microsoft Visio.

3.2.1 Casos de uso

A Figura 10 mostra os casos de uso relacionados ao protótipo desenvolvido.

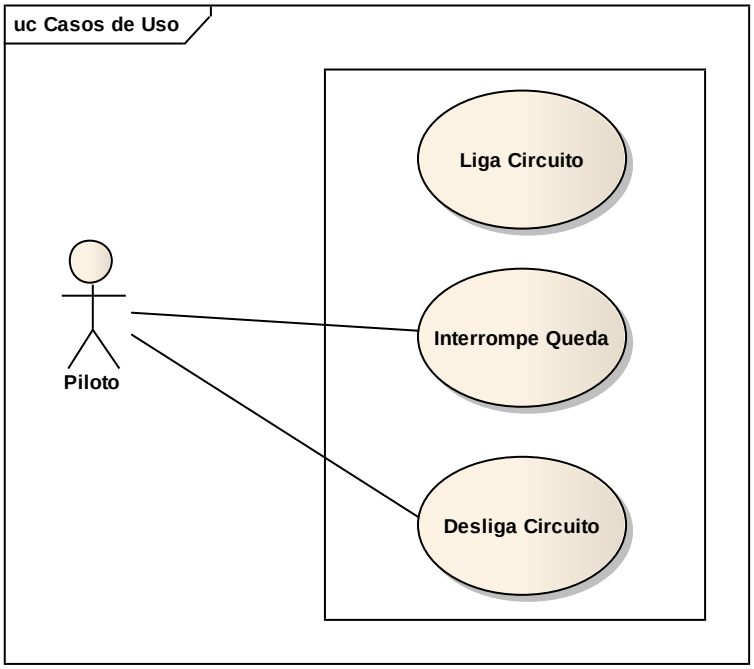


Figura 10 – Casos de uso

3.2.2 Diagrama de sequência

O diagrama de sequência da Figura 11 apresenta a seqüência de acontecimentos executados durante o período em que o mecanismo embarcado encontra-se ligado. No diagrama percebe-se o momento em que é realizada a captura das leituras dos sensores, calculada a posição do aeromodelo e aplicada a solução de nivelamento.

No momento em que o circuito é ligado pelo piloto o *software* embarcado envia para a

placa Atomic IMU uma sequência de três caracteres para proceder a inicialização do circuito responsável por fazer a leitura dos sensores.

Ao iniciar o circuito da placa Atomic IMU a mesma passa a fazer as medições dos sensores, montando uma cadeia de bytes e transmitindo a mesma pela porta serial para a interpretação pelo programa embarcado da placa ArduPilot. Este envio das medições é realizado de forma ininterrupta, parando somente no momento em que o a placa é desligada.

Ao receber a cadeia de bytes enviada pela placa Atomic IMU, o sistema embarcado faz a decodificação dos dados recebidos, atribuindo os valores as suas variáveis correspondentes e a cada nova cadeia de bytes recebida o mecanismo executa a função `kalman(a,b,c)` que procede com os cálculos necessários para encontrar o posicionamento do aeromodelo no espaço. Este cálculo é executado durante todo o tempo em que o circuito está ligado.

A partir do momento em que o piloto habilita a função `interrompe_queda(x,y)`, o sistema passa a enviar aos servomecanismos comandos informando a posição o qual ele deve estar para corrigir o nivelamento do aeromodelo. Esta rotina se repete até o momento em que o piloto desabilite a função.

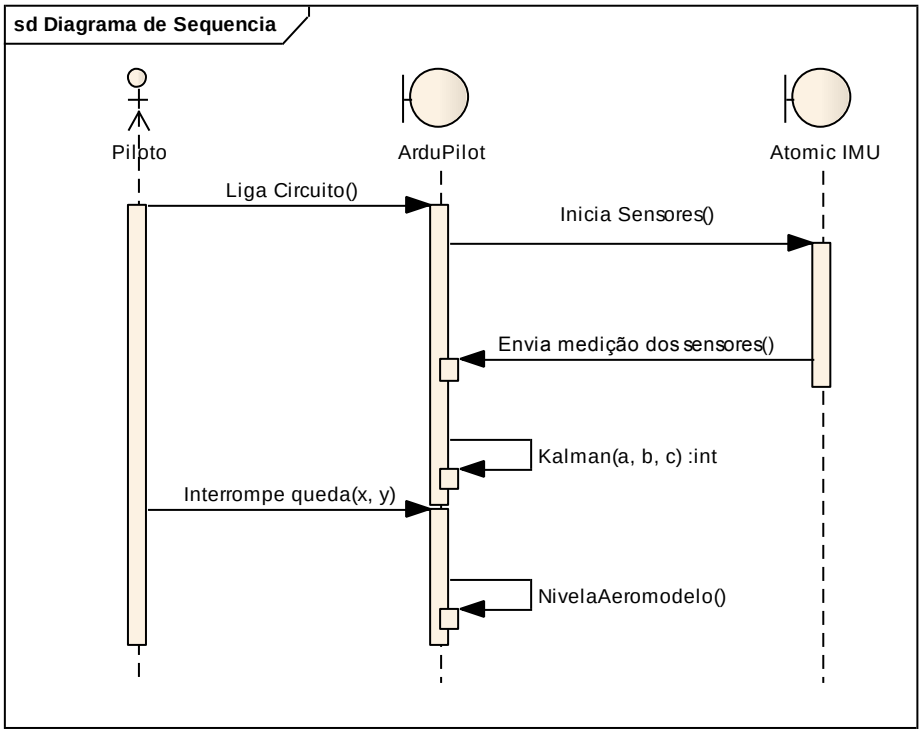


Figura 11 – Diagrama de seqüência

3.2.3 Fluxograma

O fluxograma oferece uma visão geral do funcionamento do mecanismo embarcado. Este é ilustrado na Figura 12.

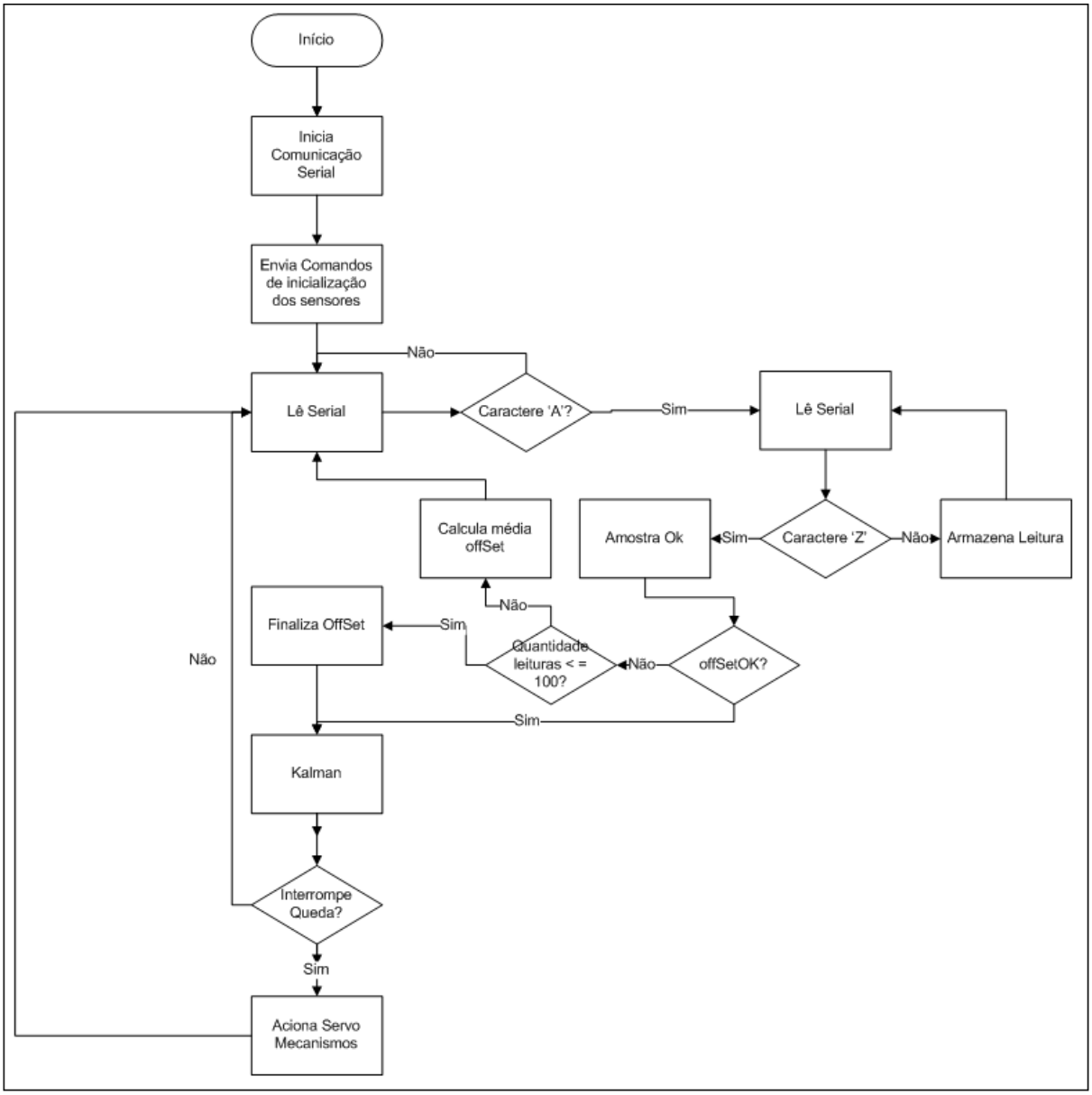


Figura 12 – Fluxograma do mecanismo embarcado

3.2.3.1 Diagrama de classes

Os diagramas de classes apresentados em Figura 13 e Figura 14 exibem as principais classes utilizadas no desenvolvimento do sistema de teste e do mecanismo embarcado.

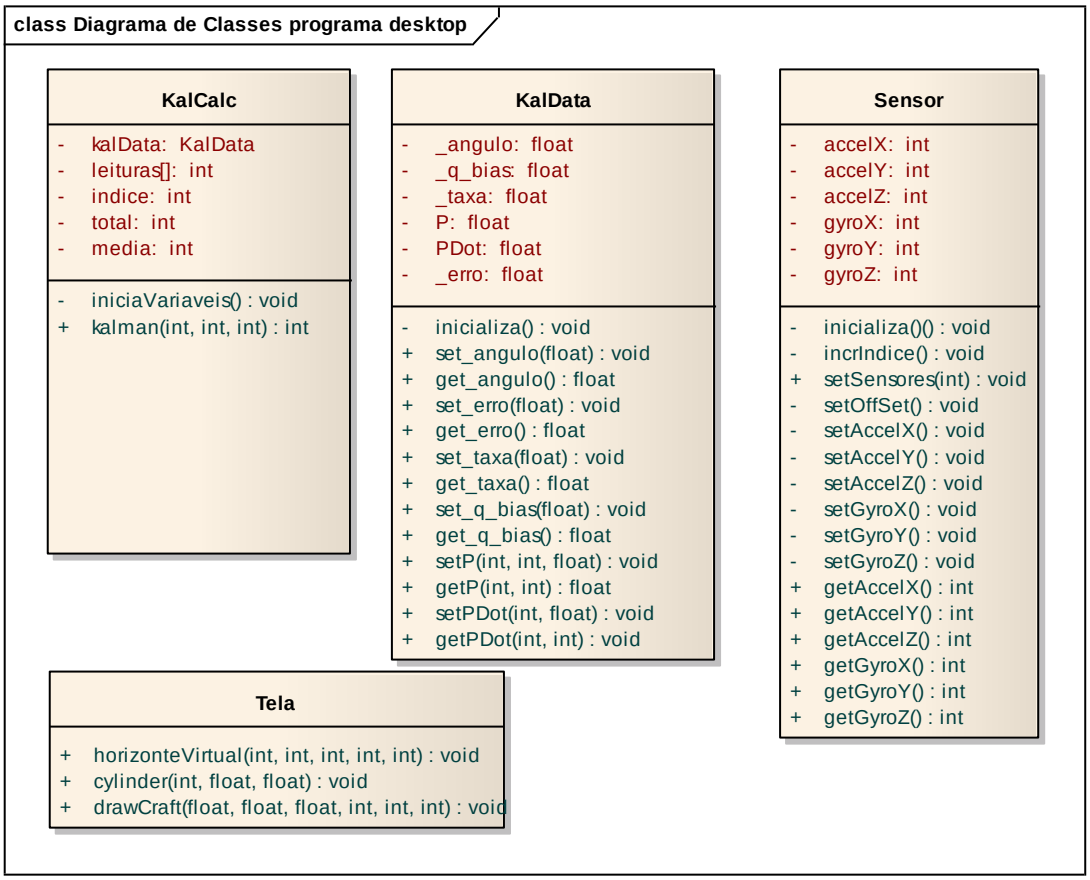


Figura 13 – Diagrama de classes programa de simulação

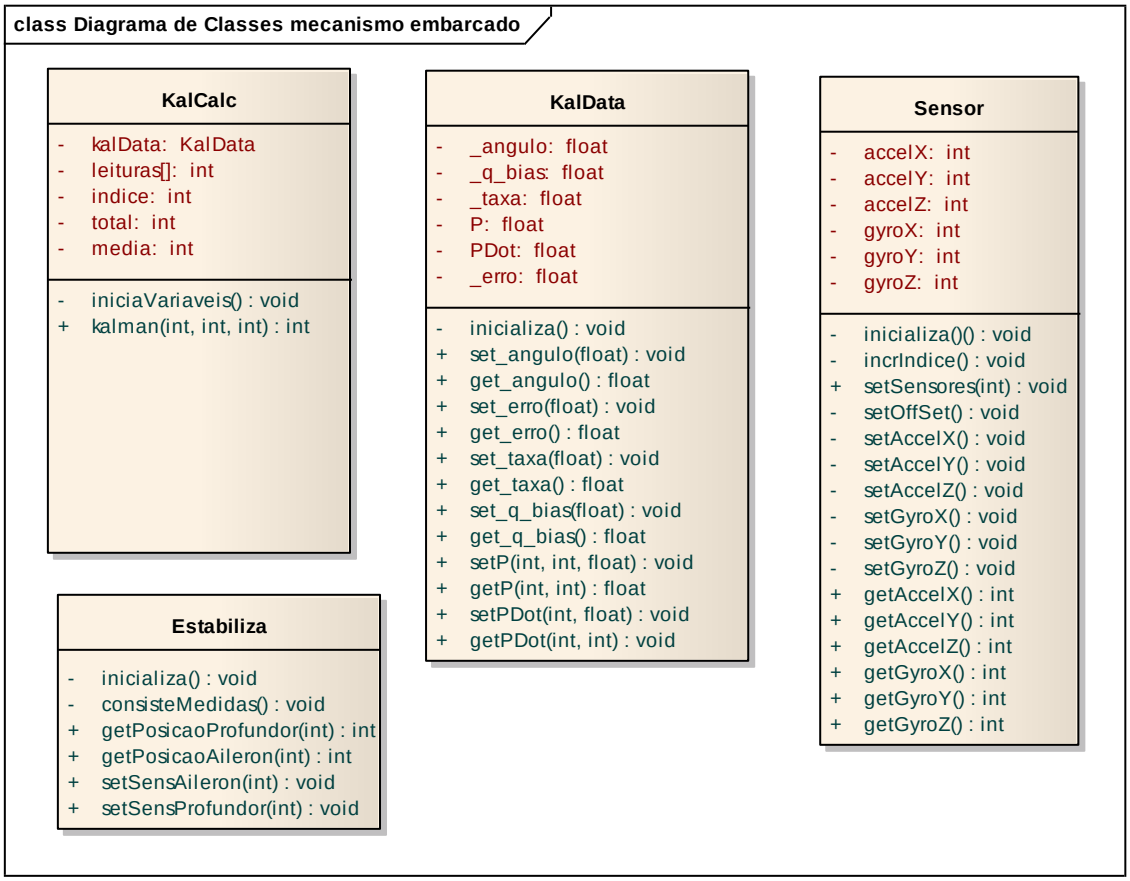


Figura 14 – Diagrama de classes mecanismo embarcado

As principais classes utilizadas para o desenvolvimento do programa de simulação e do mecanismo embarcado são:

a) `KalCalc`: classe que implementa o cálculo do filtro de Kalman;
b) `KalData`: classe responsável por armazenar as variáveis do filtro de Kalman;
c) `Sensor`: esta classe armazena e fornece a leitura dos sensores;
d) `Estabiliza`: classe responsável por calcular a posição dos servomecanismos.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas para o desenvolvimento do protótipo e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Os sistemas foram desenvolvidos utilizando duas linguagens de programação distintas, Processing e Arduino.

A linguagem Processing foi utilizada para o desenvolvimento do sistema de simulação, que tem como objetivo fazer a representação gráfica do aeromodelo no computador e servir como ferramenta de depuração do algoritmo do filtro de Kalman por se tratar da mesma implementação feita no programa embarcado.

A linguagem de programação Arduino foi utilizada para o desenvolvimento do mecanismo embarcado, o qual é programado para a placa ArduPilot e tem como objetivo interromper a queda do aeromodelo.

Nas seções seguintes são apresentados os componentes de *hardware* e técnicas utilizadas para o desenvolvimento do sistema de simulação e do sistema embarcado.

3.3.1.1 Hardware utilizado

Os *hardwares* utilizados para o desenvolvimento do protótipo conforme mencionados

nas sessões 2.3.3 e 2.3.4 foram as placas ArduPilot e Atomic IMU, por fornecer toda a estrutura de hardware necessária para implementar o protótipo.

Ao ligar a placa Atomic IMU, há uma interface de comunicação serial que quando conectada ao computador permite o desenvolvedor fazer uma série de parametrizações através de um terminal *telnet* conforme apresentado na Figura 15.

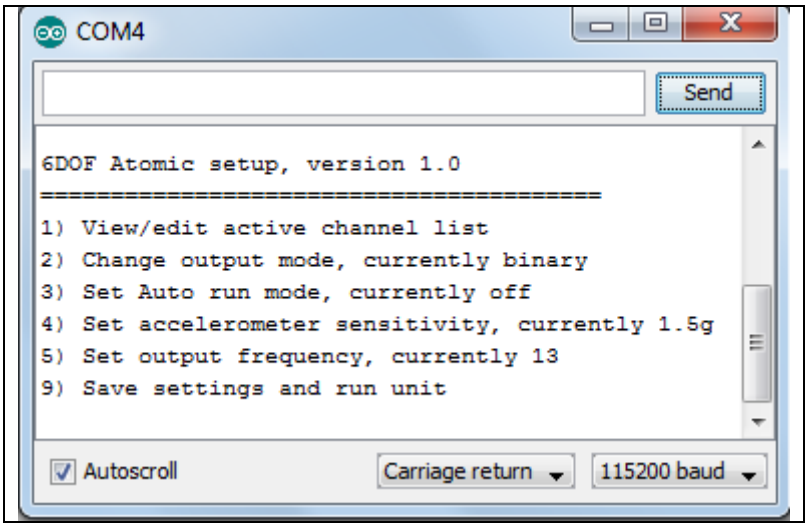


Figura 15 – Tela de configuração da placa Atomic IMU

No menu de configurações é possível selecionar quais sensores estão ativos para efeitos de medição e se os mesmos vão estar contidos na cadeia de bytes enviada pela porta serial com a medição dos mesmos. Pode ainda ser definido se o modo de transmissão serial será por meio binário ou ASCII. Pode ser configurado um modo de início automático, não sendo necessário enviar parâmetros de inicialização para a placa começar a enviar as medições.

Desta maneira é possível também selecionar a frequência com que a amostra é enviada pela porta serial e por fim configurar a sensibilidade dos acelerômetros medida em Gravidade (G) para as seguintes: 1.5G, 2G, 4G, 6G.

No Quadro 1 é possível observar os comandos que podem ser enviados à placa juntamente com seu significado e na Figura 16 é ilustrado o fluxograma de funcionamento do envio da cadeia de bytes contendo a medição dos sensores.

Byte	**Significado**
“%”, ASCII 37	Ajusta a sensibilidade do acelerômetro em 1.5G
“&”, ASCII 38	Ajusta a sensibilidade do acelerômetro em 2G
“ ´ ”, (apostrofe), ASCII 39	Ajusta a sensibilidade do acelerômetro em 4G
“(”, ASCII 40	Ajusta a sensibilidade do acelerômetro em 6G
“)”, ASCII 41	Ajusta a frequência de amostra para 50Hz
“*”, ASCII 42	Ajusta a frequência de amostra para 100Hz
“+”, ASCII 43	Ajusta a frequência de amostra para 150Hz
“,”, ASCII 44	Ajusta a frequência de amostra para 200Hz
“-”, ASCII 45	Ajusta a frequência de amostra para 250Hz
“#”, ASCII 35	Inicia a transmissão da sequência de amostra em modo binário
“ ”, (espaço), ASCII 32	Interrompe a transmissão das amostras, caso não esteja enviando amostras, é apresentado o menu de configuração

Fonte: adaptado de Monge (2011, p. 44).

Quadro 1 – Comandos da placa Atomic IMU

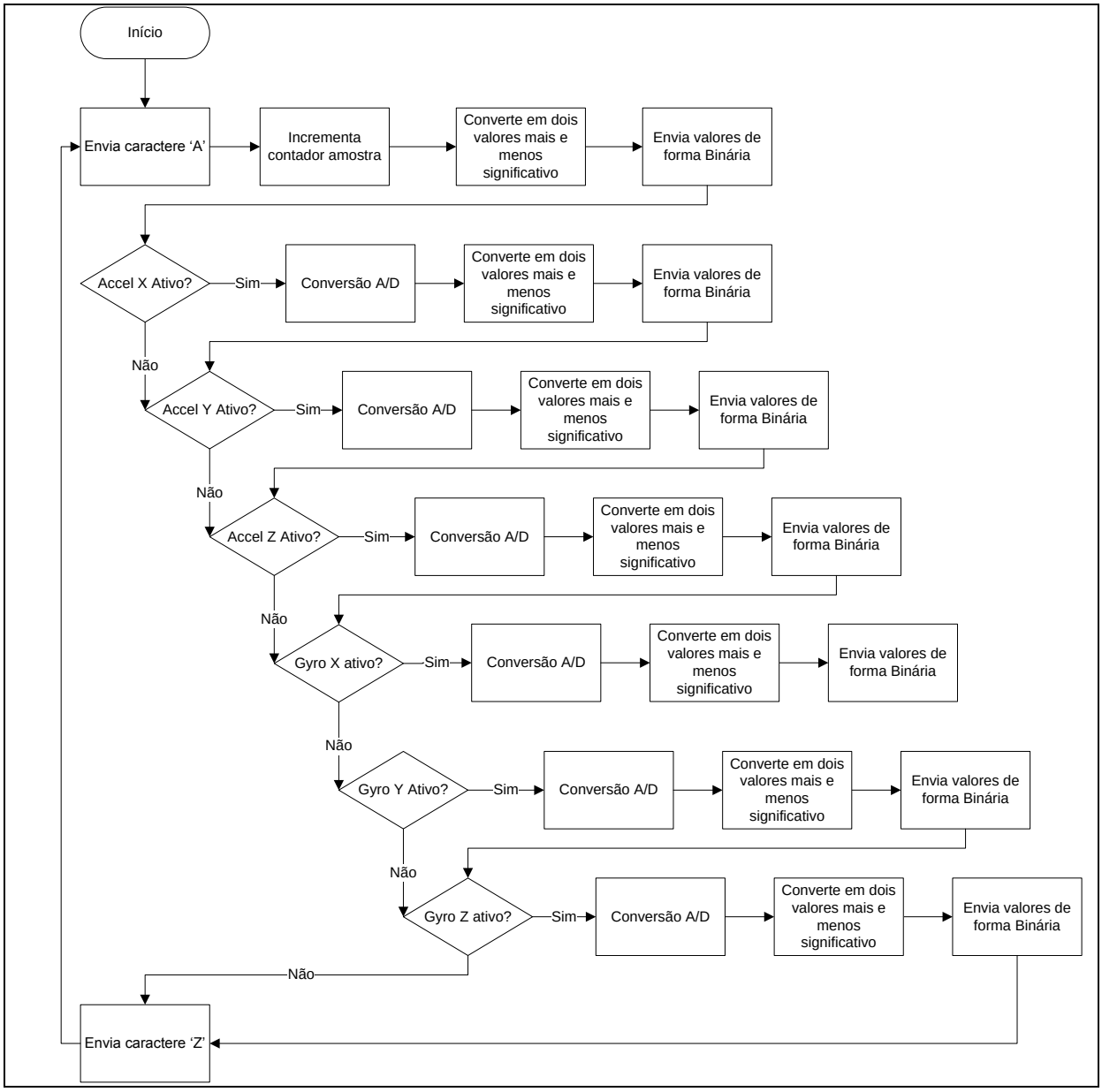


Figura 16 – Fluxograma Atomic IMU

Na Figura 16 é possível observar que há um procedimento denominado “Converte em dois valores, mais significativo e menos significativo”. Esta conversão dá-se devido ao fato de ter sido escolhida a forma de envio binário pela serial. A conversão A/D dos sensores retorna um número que vai de 0 (Zero) até (1023) o que ultrapassa o armazenamento de um byte, desta maneira o valor recebido pela conversão A/D é dividido por 256 para obter o valor mais significativo e o valor menos significativo. Sendo assim cada sensor envia dois bytes que compõem o valor de sua medição.

A Figura 17 ilustra a sequência de bytes enviada pela porta serial. Cada quadrado verde representa a leitura de um sensor, na Figura 17 cada byte está separado por uma | (barra) o que não ocorre na comunicação serial. Os valores encontrados a esquerda de cada retângulo verde são os valores mais significativos da medição, devem ser multiplicados por 256 e depois somados com os valores encontrados a direita do retângulo verde para obter a medição original do sensor.

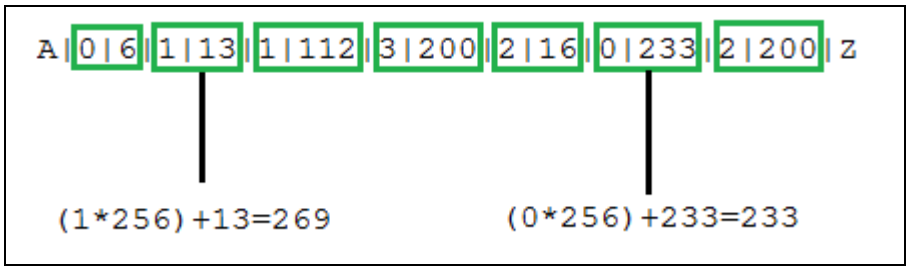


Figura 17 – Cadeia de caracteres

O caractere “A” indica o início de uma amostra de leituras dos sensores, enquanto o caractere “Z” indica o término desta amostra. Caso após receber a sequência de bytes contendo as medições dos sensores não venha o caractere “Z” em seguida, toda a amostra é descartada, pois não é possível confiar na mesma.

3.3.1.2 `Atan2(x,y)`

A função `atan2(x,y)` é utilizada para retornar um valor em radianos relativo as medidas cartesianas informadas em seus parâmetros `x` e `y`.

Esta função foi primeiramente utilizada na programação de sistemas implementados na

linguagem FORTRAN[3] e logo passou a ser utilizada em outros ramos como engenharia e ciência (ATAN2, 2012). A função retorna um valor medido em radianos, sendo que este valor pertence ao intervalo que vai de $-\pi$ até π caracterizando assim uma volta completa no circulo trigonométrico (Figura 18).

A função é muito útil, quando a aplicação envolve vetores em um espaço cartesiano, pois encontra o posicionamento angular de um ponto em direção a outro. É amplamente utilizada na computação gráfica, cuja função é converter as matrizes de rotação dos ângulos de Euler[4].

A função, como abordado acima, retorna um valor que vai de -180º até 180º, desta maneira quando o ângulo obtido for positivo o ponto descoberto é encontrado no quadrante 1 ou 2 e quando o valor retornado é negativo, o ponto encontrado está no quadrante 3 ou 4, conforme o circulo trigonométrico representado no plano cartesiano ilustrado na Figura 18.

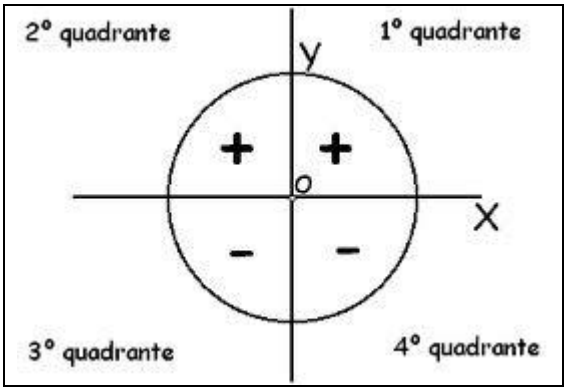


Figura 18 – Circulo trigonométrico no plano cartesiano

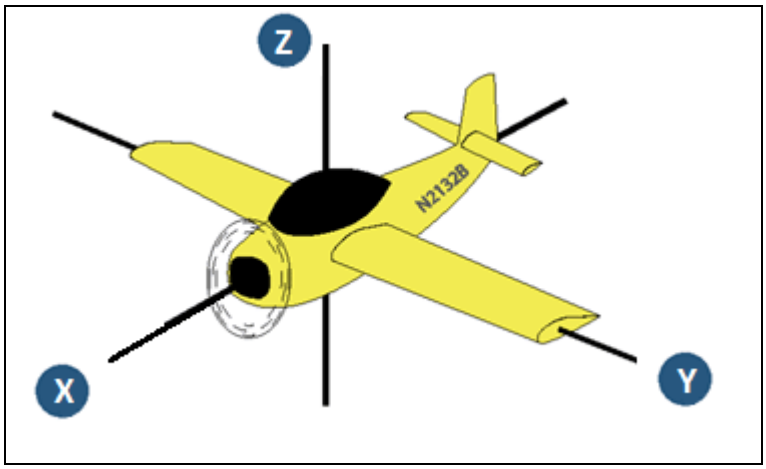
A figura abaixo exemplifica como funciona o posicionamento mostrado como referência. Neste exemplo são utilizados valores em uma escala de 0 até 1, porém no funcionamento do sistema, os valores informados com parâmetro para a função variam de -500 até +500. Os valores são referentes às medições dos acelerômetros que medem os eixos X, Y e Z. Os dados do acelerômetro são obtidos através da função `getAccel`**`N`**`()` da classe `Sensor` onde **`N`** deve ser substituído X, Y ou Z.

Para o funcionamento da função `atan2(x,y)` é necessário informar dois valores de posição cartesiana. O valor `x`, primeiro parâmetro da função, é o eixo o qual se quer saber a inclinação, é informada então a medição do acelerômetro do eixo X através da função

[3] FORTRAN é uma linguagem de programação desenvolvida pela empresa IBM em 1950. O significado em inglês do acrônimo é IBM Mathematical FORmula TRANslation System.
[4] Ângulos de Euler são os ângulos resultantes da rotação feitas nos eixos X,Y,Z de um plano cartesiano tridimensional em relação a seu ponto estável encontrado quando X=0, Y=0 e Z=0.

`getAccelX()` da classe `Sensor`. O segundo parâmetro, `y`, é o eixo utilizado como referência para obter o local do ponto no plano cartesiano, neste caso é utilizado como referência a medição do acelerômetro do eixo Z retornado pela função `getAccelZ()`.

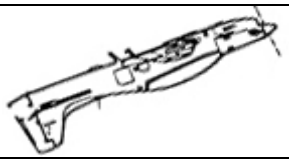
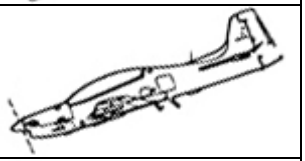
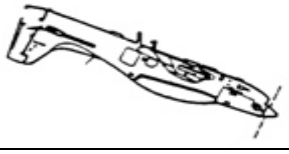
A Figura 19 representa no aeromodelo os eixos que estão sendo mensurados pelos acelerômetros e girômetros.



Fonte: adaptado de Caluta (2004).

Figura 19 – Representação dos eixos X, Y e Z no aeromodelo

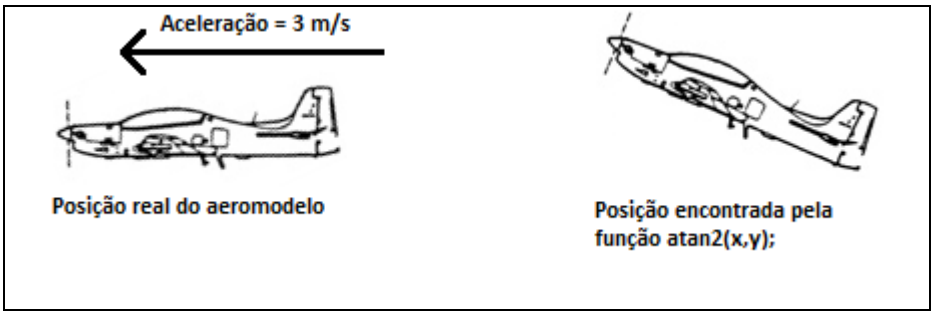
Desta forma ao informar as medições dos acelerômetros e dos eixos X e Z respectivamente, é obtida a inclinação do eixo transversal, sabendo assim se o avião está em um ângulo de subida ou descida. O quadro abaixo apresenta a relação entre os ângulos encontrados pela função `atan2(x,y)` e o posicionamento do aeromodelo ilustrados também na Figura 20.

Ângulo (`atan2(x,y)`)	Posição Aeromodelo	
0 < atan2(x,y) < 90	O aeromodelo apresenta um ângulo de subida.	
90 < atan2(x,y) < 180	O aeromodelo apresenta um ângulo de subida, porém o mesmo esta posicionado de cabeça para baixo.	
-90 < atan2(x,y) < 0	O Aeromodelo apresenta um ângulo de descida.	
-180 < atan2(x,y) < -90	O Aeromodelo apresenta um ângulo de descida, porém de cabeça para baixo.	

Fonte: adaptado de Superfícies de Controle (2010).

Figura 20 – Posicionamento do aeromodelo conforme resposta da função da primeira coluna

Entretanto, a utilização isolada da função `atan2(x,y)` para obter o posicionamento do aeromodelo no espaço é falha quando o mesmo está em movimento, pois os acelerômetros sofrem com o ruído causado pelo motor e pela inércia de seus movimentos. No momento em que o avião está em aceleração, a função retorna um posicionamento falho, informando desta maneira que o avião apresenta um ângulo de subida quando na verdade o mesmo está nivelado. Isto ocorre devido ao aumento do valor da medição do acelerômetro que mede o eixo X, pois este é influenciado pela aceleração do aeromodelo (Figura 21).



Fonte: adaptado de Superfícies de Controle (2010).

Figura 21 – Erro da função `Atan2(x,y)`

Foi proposto então o algoritmo do filtro de Kalman para sanar as interferências causadas pelas forças inerciais de aceleração de locomoção, utilizando assim os girômetros em conjunto com os acelerômetros para integrar o cálculo da estimativa de posicionamento.

3.3.1.3 Algoritmo de Kalman

Para o desenvolvimento do protótipo, foi utilizado o algoritmo do filtro de Kalman desenvolvido por Muñoz (2008). Este algoritmo foi escrito para executar em um *hardware* Arduino utilizando as medições de sensores, acelerômetro e girômetro contidos no Nintendo Wii.

O código utilizado como referência para o desenvolvimento do projeto é programado em uma estrutura de funcionamento sequencial, no qual foi realizada uma mudança do algoritmo para o padrão de orientação a objetos, que por fim teve duas versões desenvolvidas.

Ambas versões, sendo uma versão programada em Java, a qual é utilizada no programa de simulação e outra versão programada em C++, utilizada pela plataforma Arduino para desenvolvimento do mecanismo embarcado.

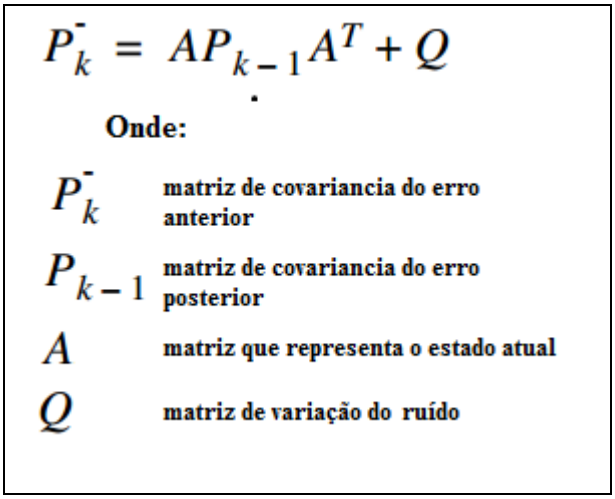
O algoritmo do filtro de Kalman é fundamental para o funcionamento do protótipo, pois sem o mesmo não seria possível obter os dados para calcular o posicionamento do

aeromodelo no espaço.

Devido a limitação do processamento do micro-controlador, Muñoz (2008) propõe uma otimização deste cálculo passando a computar somente os termos que explicitamente não seriam zerados pelo algoritmo, da mesma maneira algumas matrizes foram convertidas em simples variáveis devido um ou mais termos apresentarem o valor zero durante todo o cálculo. Com estas alterações o código fica menos legível quando existe a necessidade de efetuar uma depuração, entretanto estas alterações fazem com que a CPU utilize menos tempo de processamento para finalizar o cálculo.

Segundo Aiube (2005, p. 77) as equações do filtro de Kalman podem ser agrupadas em dois tipos distintos: equações de atualização do tempo e equações de atualização da medição. Estes dois grupos de equações funcionam em conjunto como um sistema de retroalimentação[5].

A equação de atualização do tempo, também denominada de equação de previsão, representada pela Figura 22, é responsável pelo avanço das variáveis de estado e das covariâncias no tempo para se obter as estimativas para o próximo instante (AIUBE, 2005, p. 78). Sua implementação é ilustrada no Quadro 2.



Fonte: Correa (2005, p. 20).

Figura 22 – Fórmula do estado do sistema

```
kalData.setPDot(0, Q_angle - kalData.getP(0,1) - kalData.getP(1,0));
kalData.setPDot(1,- kalData.getP(1,1));
kalData.setPDot(2,- kalData.getP(1,1));
kalData.setPDot(3, Q_gyro);
```

Quadro 2 – Implementação da fórmula da Figura 22

[5] Retroalimentação é o nome dado ao procedimento através do qual parte do sinal de saída de um sistema é transferida para a entrada deste mesmo sistema (RETROALIMENTAÇÃO, 2011).

As equações de atualização das medições são responsáveis pela retroalimentação do filtro, incorporando desta maneira uma nova informação à variável observada nas estimativas anteriores para obter um ganho na estimação posterior (AIUBE, 2005, p. 78).

As equações responsáveis pela atualização das medições são representadas pelas fórmulas das Figura 23, Figura 24 e Figura 25.

O ganho do sistema é um peso atribuído as estimativas e medições calculadas na fórmula da Figura 25. A equação responsável por calcular o ganho do sistema é:

$$K_k = AP_k C^T (CP_k C^T + S_Z)^{-1}$$

onde:

K_k **matriz de ganho do sistema**

C **matriz que relaciona o estado medido com o estado estimado**

S_Z **estimativa de variação do ruído**

Fonte: Ruziska (2009).

Figura 23 – Fórmula do calculo do ganho

No Quadro 3 é possível ver a implementação da fórmula ilustrada na Figura 23.

```
Float       C_0   = 1;
float       PCt_0 = C_0 * kalData.getP(0,0);
float       PCt_1 = C_0 * kalData.getP(1,0);
float       E     = R_angle + C_0 * PCt_0;
float       K_0   = PCt_0 / E;
float       K_1   = PCt_1 / E;
```

Quadro 3– Trecho do código de atualização do ganho do sistema

Atualização da matriz de covariância do sistema é apresentada na Figura 24 e sua implementação é ilustrada no Quadro 4. Esta matriz contém as informações referentes as variações entre o estado atual e o próximo estado do sistema.

$$P_{k+1} = AP_k A^T + S_w - AP_k C^T S_z^{-1} CP_k A^T$$

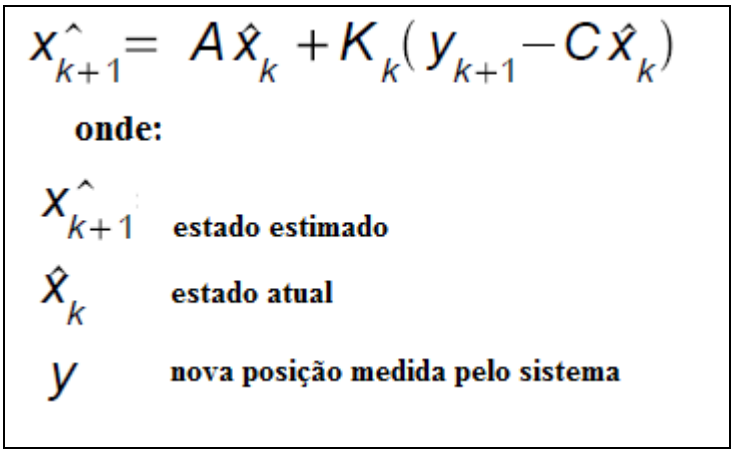
Fonte: Rusika (2005).

Figura 24 – Fórmula da atualização da matriz de covariância

```
kalData.setP(0,0, kalData.getP(0,0) - K_0 * t_0);
kalData.setP(0,1, kalData.getP(0,1) - K_0 * t_1);
kalData.setP(1,0, kalData.getP(1,0) - K_1 * t_0);
kalData.setP(1,1, kalData.getP(1,1) - K_1 * t_1);
```

Quadro 4 – Atualização da matriz de covariância

A fórmula que estima o estado futuro, apresentada na Figura 25, utiliza em seu contexto o ganho calculado pela fórmula da Figura 23, em que caso o sistema não tenha sofrido alteração, **K** tende a 0, resulta em uma não variação da previsão do estado futuro. A implementação é ilustrada no Quadro 5.



Fonte: Ruziska (2009).

Figura 25 – Fórmula de estimativa do estado

```
angle_m = atan2( accelEixoMedido, accelEixoReferencia);
float angle_err = angle_m - kalData.get_angulo();
kalData.set_angulo(kalData.get_angulo()+ (K_0 * angle_err));
```

Quadro 5 – Implementação da fórmula da Figura 25

Todo o cálculo do filtro de Kalman foi embutido em uma única função chamada `kalman(a,b,c)` na classe `KalCalc`, tornando mais sua utilização mais amigável, sendo necessário informar apenas três parâmetros para obter o resultado esperado. No Quadro 6 pode-se observar a chamada da função `kalman(a,b,c)`.

```
eixoX = KEixoX.kalman(sensor.getAccelX(), sensor.getAccelZ(),
sensor.getGyroX());
eixoY = KEixoY.kalman(sensor.getAccelY(), sensor.getAccelZ(),
sensor.getGyroY());
```

Quadro 6 – Chamada da função do filtro de Kalman

A função `kalman(a,b,c)` recebe respectivamente as medições dos sensores dos acelerômetros X, Z e do girômetro X, retornando em seguida o valor em graus da inclinação do eixo X, informado no primeiro parâmetro, este responsável por indicar se o aeromodelo está com o nariz voltado para o céu ou para a terra.

A segunda chamada da função `kalman(a,b,c)`, que recebe as medições dos sensores Y, Z e do girômetro Y, retorna o valor em graus referente a inclinação lateral do aeromodelo, informando se este está inclinado para a esquerda ou para a direita.

Dentro da função `kalman(a,b,c)` é possível observar os cálculos feitos para a obtenção de uma medição livre de ruídos.

3.3.1.4 Desenvolvimento do programa de simulação

O programa de simulação foi desenvolvido com o objetivo de implementar o mesmo algoritmo utilizado no mecanismo embarcado possibilitando assim uma melhor depuração a fim de verificar o correto funcionamento do algoritmo do filtro de Kalman.

Para melhor visualização do posicionamento da placa, foi desenvolvido um horizonte virtual, utilizado em aeronaves, o qual revela de forma virtual o posicionamento real do aeromodelo em relação a linha do horizonte. Para o desenvolvimento do horizonte virtual foi utilizado a biblioteca OPENGL, responsável por conter os métodos de rotação utilizados para simular o ângulo de rolagem do aeromodelo. O horizonte virtual é apresentado na Figura 26, este é movimentado pelos valores retornados pelo filtro de Kalman.

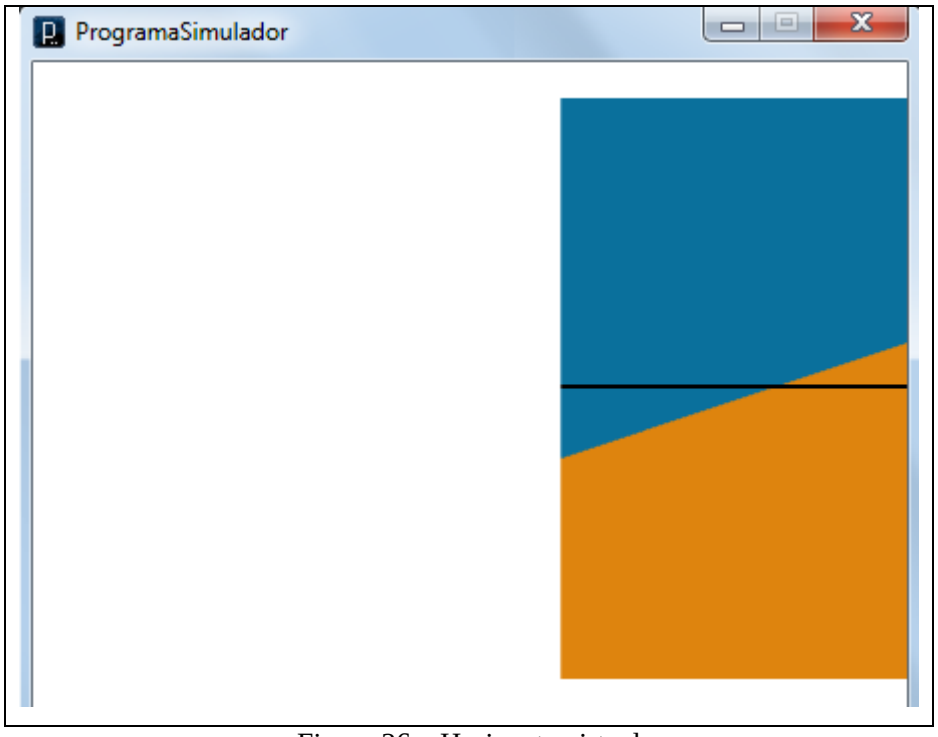


Figura 26 – Horizonte virtual

O desenho tridimensional do aeromodelo é utilizado para demonstrar a posição real na

qual o aeromodelo está no momento. Para a representação tridimensional do aeromodelo, é utilizada a função desenvolvida por Matthews (2009). O modelo tridimensional é apresentado na Figura 27, na qual contém dois aeromodelos desenhados, o primeiro representa a movimentação do aeromodelo somente com as medições dos acelerômetros, já o segundo é movimentado pelos valores resultantes do filtro de Kalman, reduzindo o ruído e aprimorando seu posicionamento.

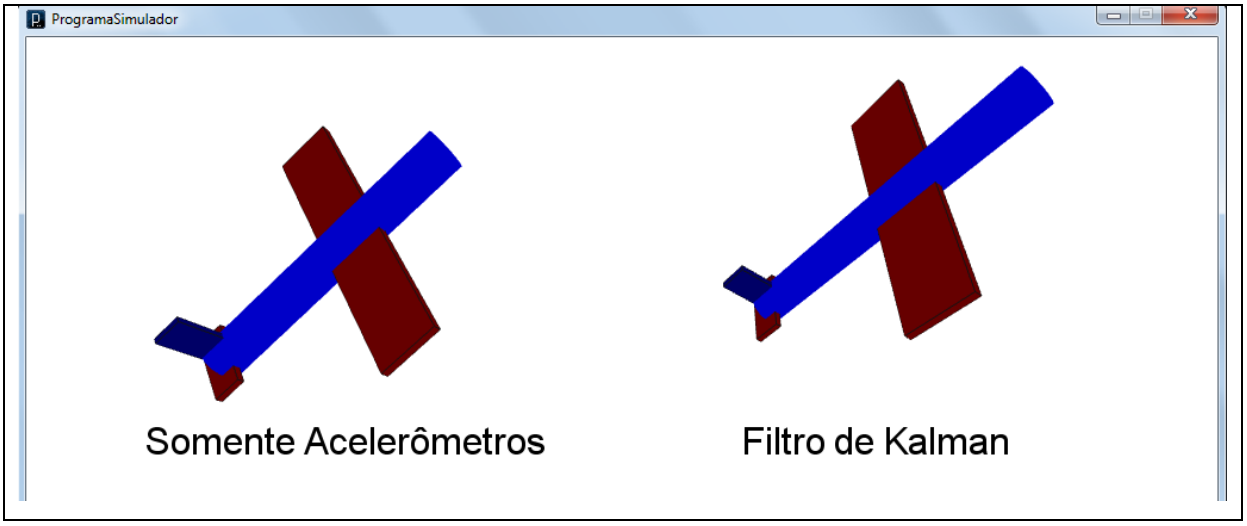


Figura 27 – Modelos tridimensionais

O programa apresenta também gráficos comparativos relacionando as medições puras dos acelerômetros e as medições após a aplicação do filtro de Kalman. Este gráfico é ilustrado na Figura 28.

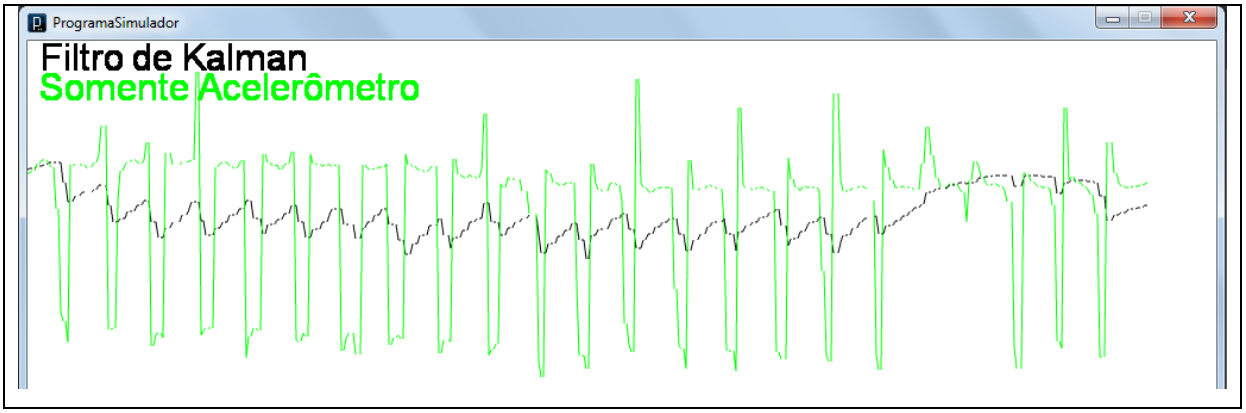


Figura 28 – Gráfico de variação das medições

Antes do sistema iniciar a apresentação das medições, é necessário que este defina determinados parâmetros de inicialização. Estes parâmetros são informados no método `setup()`, que é executado uma única vez, na inicialização do programa. Nele é realizada a inicialização da classe responsável pela comunicação serial com a placa Atomic IMU, definição do tamanho da tela, inicialização das classes `KalCalc`, `Grafico`, `Desenho` e `Sensor`, conforme apresentado do Quadro 7.

```
void setup(){
  size(1000,600, OPENGL);
  testex = new KalTeste();
  testey = new KalTeste();

  myPort = new Serial(this, Serial.list()[0],115200);
  myPort.write(char(39));
  myPort.write(char(42));
  myPort.write(char(35));
}
```

Quadro 7 – Código do método `setup()`

O método `setup()`, após iniciar a comunicação pela porta serial, envia três caracteres para a placa Atomic IMU, estes caracteres são responsáveis por ajustar a sensibilidade do acelerômetro para 4G, ajustar a frequência da amostra para 100Hz e iniciar o envio das medições.

Após a inicialização dos sensores o programa passa a receber a sequência de bytes que contém as medições dos sensores, a mesma é interpretada pelo programa através do método `serialEvent()` que é chamado a cada evento serial (interrupção) disparado pelo microprocessador. A leitura das informações provenientes da serial deve ser feita no momento em que a interrupção ocorre caso contrário o sistema armazena as leituras com atraso, desvirtuando desta modo o cálculo da posição do aeromodelo.

É possível observar no Quadro 8 o código responsável por ler as informações recebidas pela porta serial. Este código é implementado dentro da rotina `serialEvent()` que armazena de forma temporária a informação recebida na variável `dados[]` e ao final da amostra chama o método que decodifica e armazena as medições recebidas.

```
void serialEvent (Serial myPortx) {
   if (indice <0) {
    if(myPort.read() == 65) {
      indice = 0;
    }
  }else {
    if (-1 < indice  && indice < 14) {
      dados[indice] = myPort.read();
      indice++;
    }
    else {
      if (indice > 13) {
        if(myPort.read() == 90) {
          indice = -1;
          sensor.setSensores(dados);
          loop_virtual();
        }}}}}
```

Quadro 8 – Processo de leitura da amostra dos sensores

Após a inicialização da transmissão, é preciso fazer a calibração das medições dos sensores. A classe responsável por fazer a calibração, implementada no método `setOffSet()` o objetivo desta calibração é encontrar os valores capturados no momento em que o sistema está estável, obtendo desta maneira o zero de referencial a fim de subtrair este das amostras retornadas ao longo do funcionamento do sistema. A classe `Sensor` é responsável por fazer a calibração, armazenar as leituras, calcular as médias das últimas leituras e retornar a medida do sensor solicitado pelo programa.

No momento em que o sistema finaliza os procedimentos apresentados anteriormente, é realizado o cálculo do filtro de Kalman, sendo responsável por atualizar as variáveis `eixoX` e `eixoY` com os valores resultantes do filtro. As variáveis são utilizadas como parâmetro para o desenho dos gráficos, do modelo tridimensional e do horizonte virtual.

Os desenhos são impressos na tela através da função `draw()` que é chamada 60 vezes por segundo pelo próprio ambiente Processing.

3.3.1.5 Desenvolvimento do software embarcado

O mecanismo embarcado foi programado em linguagem C/C++ utilizando o ambiente de desenvolvimento Arduino IDE.

Este é o software responsável por aplicar as medições resultantes do filtro de Kalman para encontrar o posicionamento do aeromodelo e aplicar os movimentos às superfícies de controle através dos servomecanismos a fim de estabilizar o aeromodelo.

O funcionamento do mecanismo embarcado dá-se de forma similar ao funcionamento do *software* de simulação. As primeiras rotinas como inicialização da comunicação serial, inicialização da placa Atomic IMU, armazenamento das leituras, são realizadas exatamente da mesma maneira.

Após o *software* estar iniciado e rodando, o programa passa a calcular a posição na qual os servomecanismos devem se movimentar para estabilizar o aeromodelo. A classe responsável pelo cálculo do posicionamento destes é a classe `Estabiliza`.

A classe `Estabiliza` armazena valores necessários para o correto funcionamento dos servomecanismos. Nesta classe é possível informar a reversão de comandos do aeromodelo, seus limites de movimentação e a informação de sua posição central.

A reversão de comandos do aeromodelo é responsável por controlar a orientação do

movimento das superfícies de controle. Esta orientação pode variar a cada tipo de aeromodelo, pois depende do local da montagem do servomecanismo.

A limitação dos comandos é necessária para impedir que o sistema envie ao servomecanismo uma posição na qual a superfície de controle não consiga alcançar. No momento em que está situação ocorre, devido a limitação física das superfícies de controle, o servomecanismo permanece ativo tentando alcançar seu posicionamento, o que pode ocasionar um excesso de temperatura e resultar na queima do circuito do mesmo.

O Quadro 9 apresenta o trecho do código responsável por retornar a posição na qual o servomecanismo deve ser posicionado e a implementação da limitação do movimento.

```
public float getPosicaoProfundor(int posicao){ //posição em graus

    float anguloServo;

    if(posicao > 90 || posicao < - 90)
       anguloServo = 0;
    else{
      anguloServo = ((map(posicao,-90/sensProfundor, 90/sensProfundor,
centroServoProfundor-maxProfundor, maxProfundor+centroServoProfundor )
)*reversoProfundor);
    }

    if(anguloServo-centroServoProfundor > maxProfundor)
      anguloServo = maxProfundor+centroServoProfundor;
    else
      if(anguloServo < centroServoProfundor-maxProfundor)
        anguloServo = centroServoProfundor-maxProfundor;

    return anguloServo;
  }
```

Quadro 9 – Função `getPosiçãoProfundor()`

É possível observar no Quadro 9 a utilização da função `map()`, esta função é responsável por fazer o mapeamento do valor informado no parâmetro da função `getPosiçãoProfundor()` para um valor que pertença ao espectro de movimento do servomecanismo.

A classe ainda implementa a sensibilidade a qual o aeromodelo vai responder a estabilização proposta pelo algoritmo de Kalman. Esta sensibilidade influencia no tempo em que o aeromodelo é estabilizado. Quanto mais sensível o sistema, maior é o tempo levado para o mesmo atingir o estado estabilizado.

O cálculo de Kalman, implementado na classe `KalCalc`, esta classe efetua o cálculo de apenas um eixo do aeromodelo, desta maneira é necessário instanciar duas variáveis do tipo `KalCalc`, uma responsável por calcular o posicionamento ângulo do eixo longitudinal, representado no código pela variável `kalcEixoX` e a outra responsável por calcular o ângulo

do eixo transversal, representado no código pela variável `kalcEixoY`.

As funções `kalman()` de cada objeto são chamadas somente após o sistema receber a cadeia de bytes com sucesso. O Quadro 10 apresenta o momento em que as funções são chamadas.

```
anguloEixoX = kalcEixoX.kalman(sensor.getAccelX(), sensor.getAccelZ(),
sensor.getGyroX());
anguloEixoY = kalcEixoY.kalman(sensor.getAccelY(), sensor.getAccelZ(),
sensor.getGyroY());
```

Quadro 10 – Chamada do método `kalman()`

A implementação do método `kalman()` é representada no Quadro 11.

```
public int kalman (int accelEixoMedido, int accelEixoReferencia, int
gyroEixoMedido){
 float q;
  if((millis()-lastread) >= mydt){
    lastread=millis();
    q_m=((gyroEixoMedido*0.00392156862)/(180/PI));

    q = q_m - kalData.get_q_bias();

    kalData.setPDot(0, Q_angle - kalData.getP(0,1) - kalData.getP(1,0));
    kalData.setPDot(1,- kalData.getP(1,1));
    kalData.setPDot(2,- kalData.getP(1,1));
    kalData.setPDot(3, Q_gyro);

    kalData.set_taxa(q);
    kalData.set_angulo(kalData.get_angulo() + (q*dt));

    kalData.setP(0,0, kalData.getP(0,0) + kalData.getPDot(0)* dt);
    kalData.setP(0,1, kalData.getP(0,1) + kalData.getPDot(1)* dt);
    kalData.setP(1,0, kalData.getP(1,0) + kalData.getPDot(2)* dt);
    kalData.setP(1,1, kalData.getP(1,1) + kalData.getPDot(3)* dt);
    R_angle= 0.0098039;
    angle_m = atan2( accelEixoMedido, accelEixoReferencia);
    float         angle_err = angle_m - kalData.get_angulo();
    float         C_0 = 1;
    float         PCt_0 = C_0 * kalData.getP(0,0);
    float         PCt_1 = C_0 * kalData.getP(1,0);
    float         E =R_angle+ C_0 * PCt_0;
    float         K_0 = PCt_0 / E;
    float         K_1 = PCt_1 / E;
    float         t_0 = PCt_0;
    float         t_1 = C_0 * kalData.getP(0,1);

    kalData.setP(0,0, kalData.getP(0,0) - K_0 * t_0);
    kalData.setP(0,1, kalData.getP(0,1) - K_0 * t_1);
    kalData.setP(1,0, kalData.getP(1,0) - K_1 * t_0);
    kalData.setP(1,1, kalData.getP(1,1) - K_1 * t_1);
    kalData.set_angulo(kalData.get_angulo()+ (K_0 * angle_err));
    kalData.set_q_bias(kalData.get_q_bias() + K_1 * angle_err);
  }
  return int(kalData.get_angulo()*57.295779513082);
}
```

Quadro 11 – Método `kalman()`

3.3.2 Operacionalidade da implementação

A operacionalidade do mecanismo inicia-se com as conexões das placas ArduPilot e Atomic IMU. O Segundo passo é proceder com a fixação da placa Atomic IMU no centro gravitacional do aeromodelo. O terceiro passo indica como as conexões entre a placa ArduPilot, o receptor do aeromodelo e os servomecanismos. O quarto consiste em ligar o circuito e verificar a orientação das superfícies de controle, a fim de verificar se estas estão atuando na direção correta.

3.3.2.1 Conexões da placa ArduPilot e Atomic IMU

As conexões existentes entre as placas ArduPilot e Atomic IMU proveem a transferência de informações e a alimentação.

A Figura 29 apresenta as conexões entre as placas.

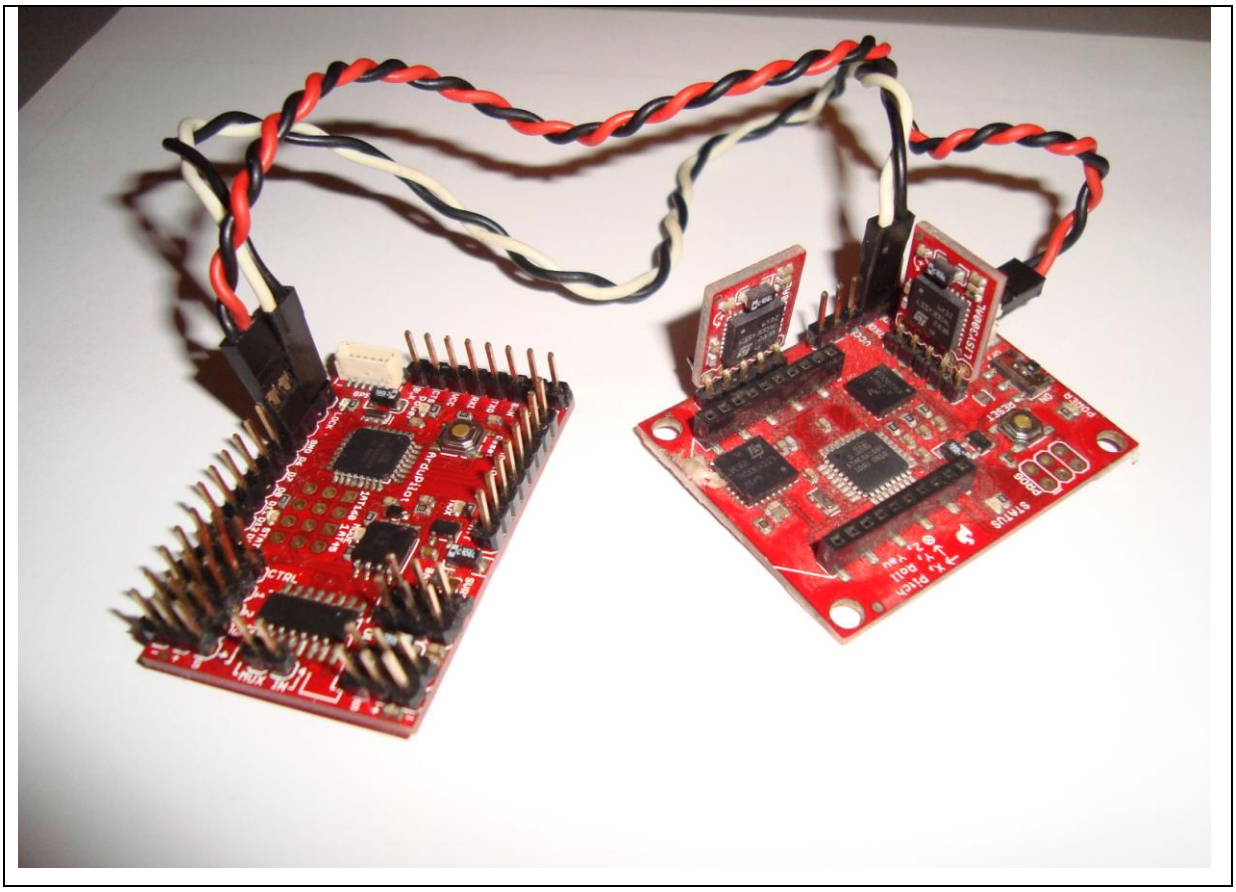


Figura 29 – Conexões entre as placas Atomic IMU e ArduPilot

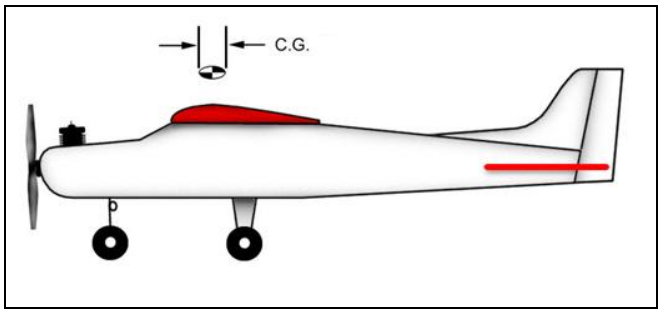
Os fios preto e branco são responsáveis pela comunicação serial entre as placas, em

que a placa ArduPilot envia os comandos de inicialização para a placa Atomic IMU e esta por sua vez envia a cadeia de bytes contendo as medições dos sensores. O fio branco deve ser conectado ao pino TX da placa ArduPilot e conectado no pino RX da placa Atomic IMU, já o fio preto deve estar conectado no pino RX da placa ArduPilot e no pino TX da placa Atomic IMU, caracterizando desta forma a comunicação serial.

A fonte de alimentação deve ser conectada na placa ArduPilot, que por sua vez, através dos fios preto e vermelho alimenta a placa Atomic IMU.

3.3.2.2 Montagem da placa Atomic IMU no aeromodelo

A placa Atomic IMU deve obrigatoriamente estar no centro de gravidade do aeromodelo. A informação do centro de gravidade do aeromodelo vem acompanhada das instruções do próprio aeromodelo. A representação do posicionamento do centro de gravidade do aeromodelo é ilustrado na Figura 30. A placa Atomic IMU foi fixada em um bloco de espuma, a fim de remover o excesso vibração resultante do funcionamento do motor do aeromodelo e este fixado no corpo do aeromodelo (Figura 31).



Fonte: Como Balancear seu Aeromodelo (2010).

Figura 30 – Centro de gravidade

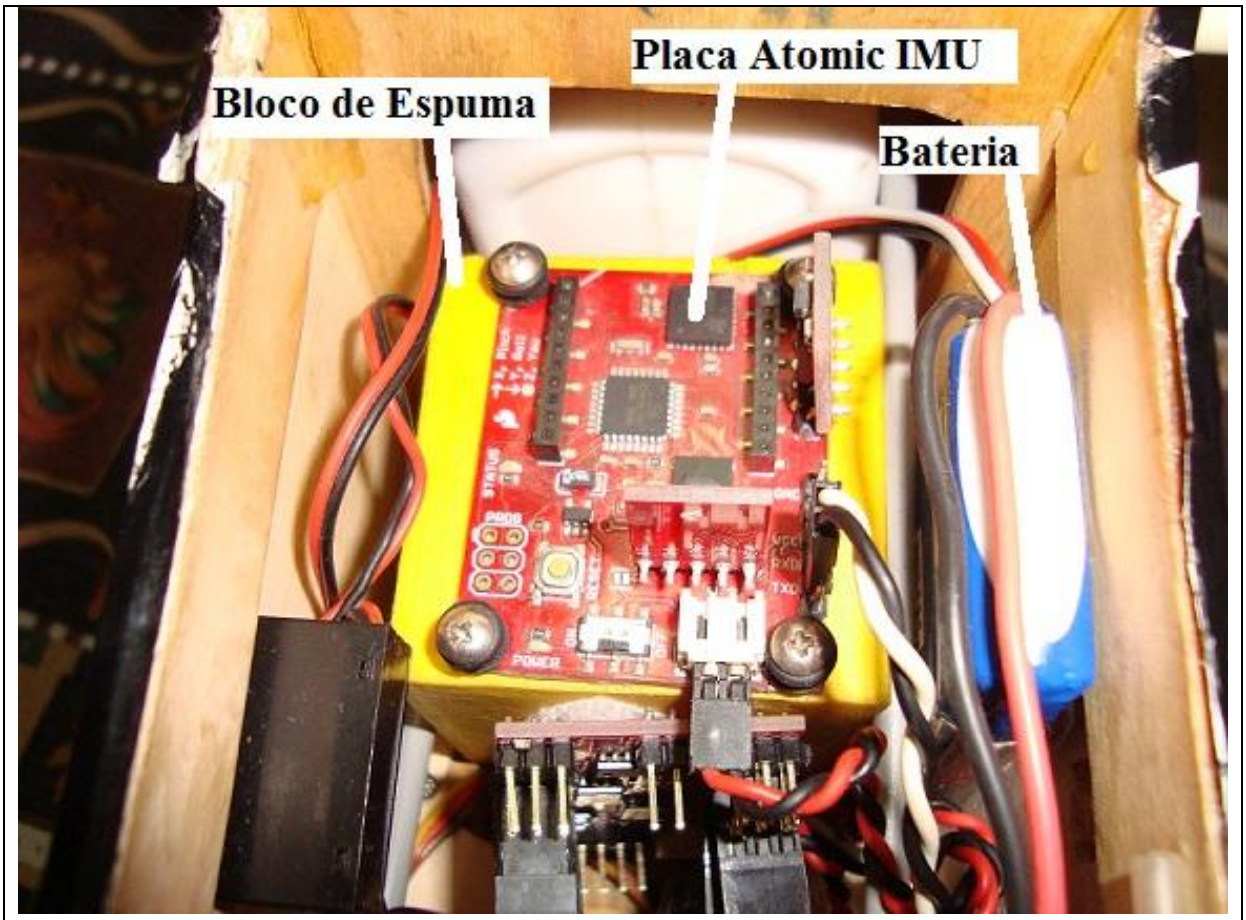


Figura 31 – Montagem da placa Atomic IMU no aeromodelo

3.3.2.3 Conexões do mecanismo embarcado x receptor FM

Para o funcionamento do mecanismo embarcado, a placa ArduPilot é montada como parte intermediária entre o receptor FM e os servomecanismos, a Figura 32 representa de maneira simplificada as conexões originais entre os servomecanismos e o receptor FM, enquanto a Figura 33 representa as conexões resultantes com a introdução da placa ArduiPilot no circuito.

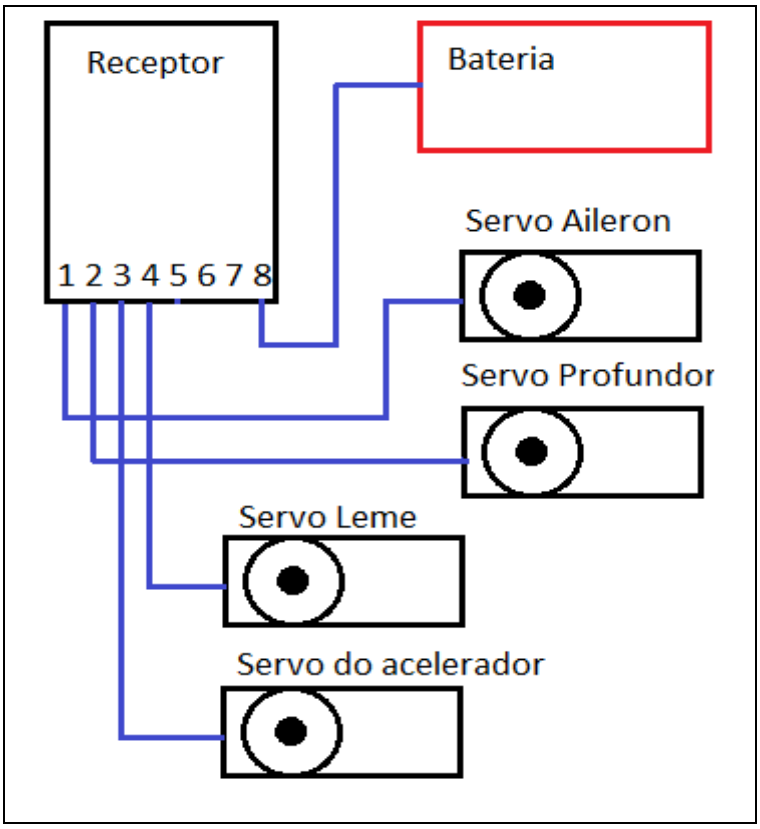


Figura 32 – Esquema de ligação do circuito original

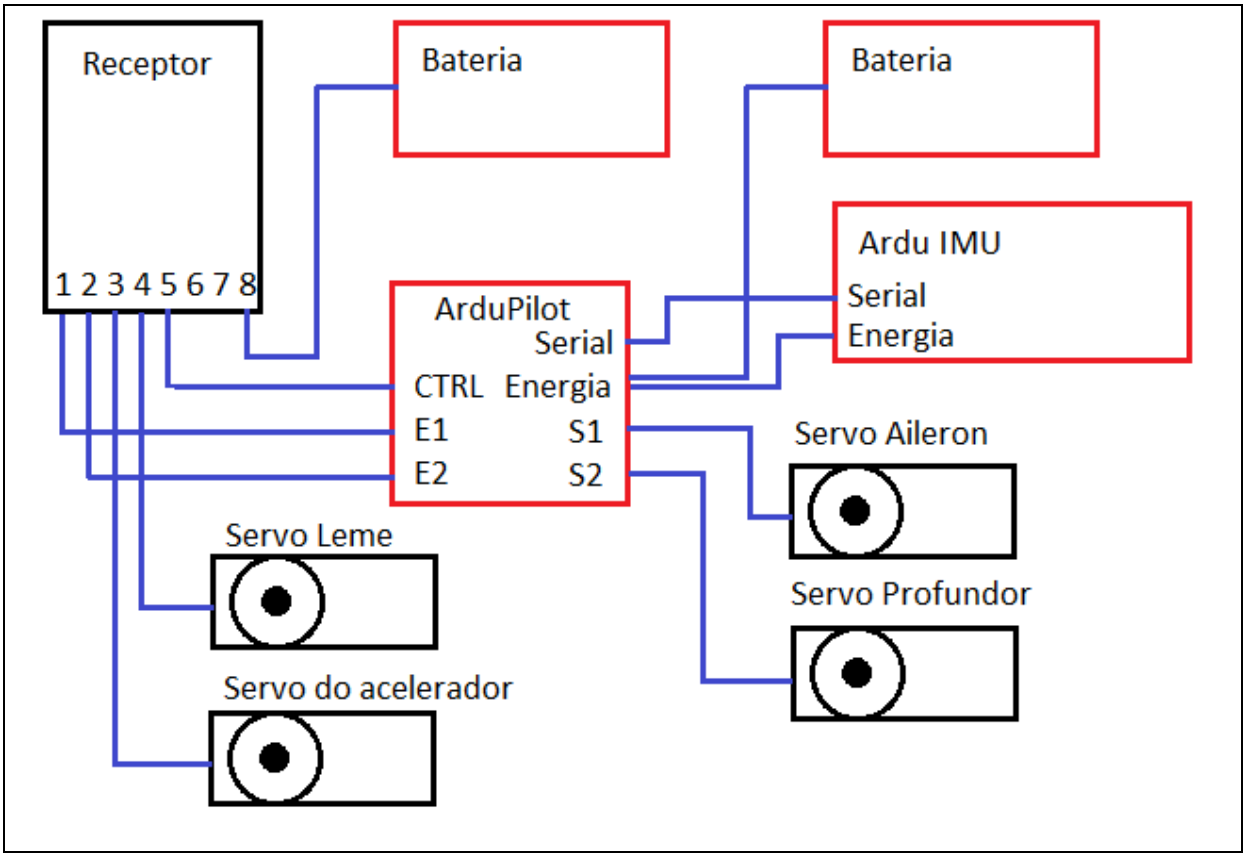


Figura 33 – Esquema de ligação do circuito com a placa Atomic IMU

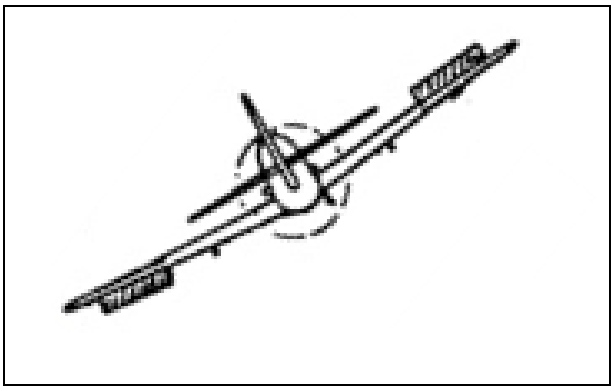
É possível verificar que os servomecanismos do leme e acelerador não estão

interligados com a placa ArduPilot, desta forma no momento em que a placa ArduPilot é acionada para interromper a queda, o piloto ainda possui o controle destes dois comandos.

3.3.2.4 Verificar a orientação das superfícies de controle

Após ligado o circuito, é necessário ativar o mesmo em solo, antes da decolagem a fim de verificar se as superfícies de comando estão respondendo as instruções enviadas pela placa ArduPilot. Esta verificação é necessária devido a posição de montagem dos servomecanismos variar de acordo com cada tipo de aeromodelo.

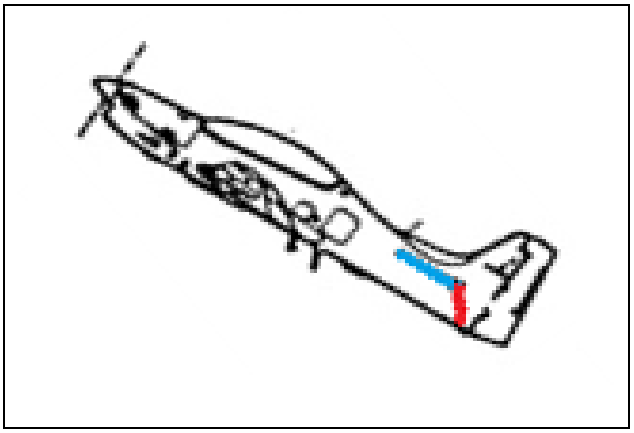
Posicionando-se na parte traseira do aeromodelo, ao inclinar este para a esquerda, a superfície de comando do lado direito deve-se posicionar para cima e a superfície de comando do lado esquerdo deve-se posicionar para baixo, conforme ilustrado na Figura 34.



Fonte: adaptado de Superfícies de Controle (2010).

Figura 34 – Posição comando aileron

Para verificar a superfície de comando do profundor é necessário levantar a frente do aeromodelo, obtendo desta forma a movimentação da superfície de comando para baixo, destacado em vermelho, conforme Figura 35.



Fonte: adaptado de Superfícies de Controle (2010).

Figura 35 – Posição do comando profundor

Durante a verificação da orientação dos comandos, caso as superfícies de controle não apresentem o resultado descrito acima é necessário fazer uma alteração nas variáveis do *software* embarcado.

3.3.2.5 Ativação do mecanismo durante o voo

A ativação do mecanismo que interrompe a queda durante o voo, é realizada mediante o acionamento de uma chave localizada no radio controle (Figura 36) onde no momento em que é acionada o *software* embarcado assume o controle do aeromodelo interrompendo a queda e posicionando o aeromodelo em uma trajetória de voo em linha reta.

Figura 36 – Chave responsável por acionar o mecanismo

3.4 RESULTADOS E DISCUSSÃO

Como testes efetuados para garantir o correto funcionamento do mecanismo, o aeromodelo foi submetido as mais variadas situações de risco as quais o mesmo pode enfrentar durante o vôo de um piloto principiante e até mesmo de um piloto profissional.

A Figura 37 apresenta as situações em que o aeromodelo foi exposto e a ilustração do posicionamento do mesmo.

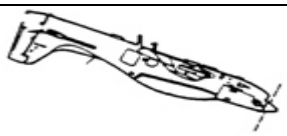
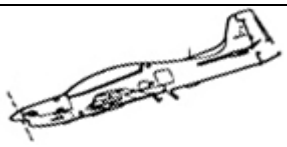
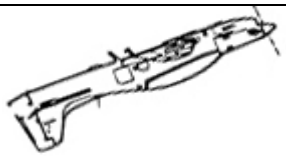
	Queda com avião de cabeça para baixo: Ação tomada: acionar os ailerons até estar de cabeça para cima e então fazer o uso do profundor para finalizar a estabilização.
	Queda com o avião de cabeça para cima: Ação tomada: utilizar o profundor para proceder com a estabilização.
	Subida com o avião de cabeça para cima: Ação tomada: utilizar o profundor para proceder com a estabilização.
	Subida com o avião de cabeça para baixo: Ação tomada: acionar os ailerons até estar de cabeça para cima e então fazer o uso do profundor para finalizar a estabilização.
	Vôo em linha reta: Ação tomada, utilizar os ailerons e o profundor para manter o avião nivelado.

Figura 37 – Posicionamento do aeromodelo

O *software* simulador foi um ferramenta fundamental para a implementação do algoritmo embarcado. Este *software* possui em sua tela dois componentes gráficos que apresentavam a exata posição do aeromodelo calculada pelo filtro de Kalman. Desta maneira foi possível verificar o correto funcionamento do algoritmo e o tempo de resposta do mesmo.

Durante os primeiros testes da implementação do filtro, o sistema apresentava resultados totalmente aleatórios, não correspondendo a posição real da placa Atomic IMU. Após várias operações de depuração utilizando o *software* de simulação foi percebido que o problema ocorria já na transmissão da cadeia de bytes da placa Atomic IMU, pois a mesma não estava enviando a leitura dos girômetros X e Z, o que ocasionou em uma redução da cadeia de bytes enviada, resultando em um armazenamento incorreto dos valores dos sensores. Após a ativação dos dois sensores pelo menu de configuração da placa Atomic IMU, o algoritmo passou retornar os resultados esperados.

Após a correta transmissão das medições dos sensores, o cálculo do algoritmo de Kalman se mostrou lento, atrasando em até três segundos a posição real do aeromodelo, onde em um sistema de trabalho em tempo real três segundos comprometem o funcionamento de todo o projeto. Após realizar aperfeiçoamentos em valores de variáveis de referência do algoritmo, o mesmo passou a calcular o posicionamento em um tempo satisfatório que não comprometia o funcionamento do mecanismo.

Na Figura 38 é apresentado um gráfico comparando as medições obtidas em bancada dos sensores sem a aplicação do filtro e após a aplicação do filtro.

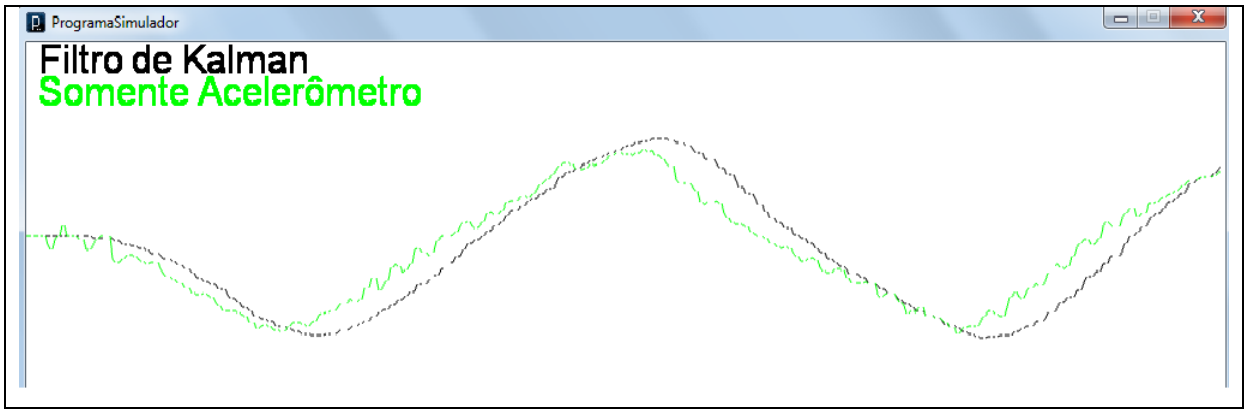


Figura 38 – Gráfico eixo x com pouco ruído

A Figura 39 apresenta um gráfico com as medições do eixo x contendo uma maior quantidade de ruído.

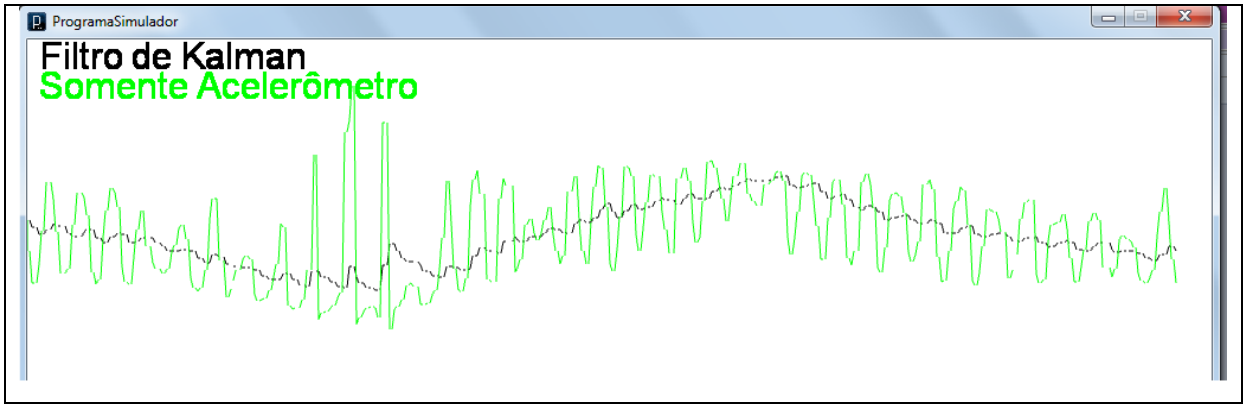


Figura 39 – Gráfico eixo x com muito ruído

Para o desenvolvimento deste projeto foi necessária a aquisição de dois *hardwares*, já abordados nos itens 2.3.3 e 2.3.4, cujo custo de aquisição foi superior a R$ 800,00, somados o frete e impostos. Não foi necessária a aquisição do aeromodelo para realização dos testes.

Nos dois primeiros testes práticos realizados, o aeromodelo foi danificado, devido a duas quedas, a primeira queda ocorreu devido a um mau funcionamento do *hardware*, pois o mesmo foi montado de forma irregular no aeromodelo.

O mau funcionamento do mecanismo ocorreu devido as placas ArduPilot e Atomic IMU receberem a alimentação diretamente pelo receptor, o qual alimentava os servo mecanismos. Ocorria que no momento em que o servo mecanismo fazia uma alteração no seu ângulo ocasionava uma queda de tensão no circuito ocasionando um desligamento da placa ArduPilot. Como solução para o problema foi introduzida uma bateria extra, a qual alimentava somente o mecanismo embarcado, desta maneira eliminando a queda de tensão.

A segunda queda ocorreu por falha do piloto por não estar voando alto suficiente para

o mecanismo proceder com o nivelamento do aeromodelo

O Quadro 12 apresenta o comparativo das características do protótipo desenvolvido com os trabalhos correlatos.

	FMA Direct	Sampaio (2006)	Este Projeto
Sofre interferência de nuvens	X		
Não necessita GPS para funcionar	X		X
Dispositivo é acionado Automaticamente	X	X	
Calcula a posição exata do aeromodelo sem interferência de meios externos		X	X
Interrompe a queda do aeromodelo	X		X

Quadro 12 – Características do protótipo desenvolvido e trabalhos correlatos

O fato do sistema não sofrer interferência das nuvens proporciona ao usuário do sistema uma maior confiabilidade, pois não depende dos fatores climáticos para apresentar um bom funcionamento.

Por não existir a necessidade de um GPS, o custo do projeto é reduzido, bem como sua implementação é feita de forma mais amigável, resultando em um algoritmo que pode ser compreendido no momento em que o piloto optar por fazer alguma modificação no sistema.

A não ativação automática do mecanismo permite um voo sem interferência do mecanismo, não corrigindo o posicionamento do aeromodelo a todo instante, permitindo ao piloto familiarizar-se com a instabilidade de voo resultando em um aprendizado do controle do aeromodelo.

Não é necessária a interação com variáveis externas como GPS e gases atmosféricos para executar os cálculos de nivelamento, pois o circuito está completamente integrado ao aeromodelo.

E por fim quando acionado o mecanismo por parte do piloto o mesmo procede com a interrupção da queda do aeromodelo.

4 CONCLUSÕES

Após as fases de testes foi possível constatar que os objetivos propostos foram alcançados. A utilização do filtro de Kalman para a descoberta do posicionamento do aeromodelo no plano tridimensional e para a eliminação do ruído foi parte fundamental para o funcionamento do projeto. A placa Atomic IMU utilizada como ponto de referência do aeromodelo mostrou-se muito eficiente por possuir todos os sensores necessários para encontrar o posicionamento do aeromodelo e por possuir uma *interface* de configuração amigável. Da mesma maneira a placa ArduPilot por conseguir armazenar todo o código do sistema embarcado e apresenta uma performance condizente com a necessária para suportar o funcionamento de todo o sistema.

A utilização de sensores inerciais e angulares mostrou ser mais confiável por não sofrerem interferência de meios externos ao sistema, como ocorre em *FMA Direct*() bem como a não necessidade de utilização de um componente GPS, Sampaio (2006), o que aumentou a desempenho do processador.

Para realização do trabalho fez-se necessário o estudo dos ambientes de programação Arduino e Processing, bem como o estudo do funcionamento das placas Atomic IMU e ArduPilot e o estudo do filtro matemático de Kalman juntamente com o algoritmo escrito por Muños (2009).

Há no entanto uma limitação no projeto desenvolvido, pois apesar do filtro de Kalman obter resultados eficientes quando utilizado em um ambiente com ruídos. Seu funcionamento não é garantido quando utilizado em aeromodelos com motores a combustão, pois estes produzem uma vibração excessiva, salvo quando o motor do aeromodelo estiver desligado. Neste caso o filtro retorna valores aleatórios, que ao serem aplicados aos servomecanismos do aeromodelo podem resultar em uma queda.

Com relação ao objetivo de armazenar as leituras dos sensores durante o voo para uma posterior análise dos resultados obtidos, a realização deste não foi possível pela pouca capacidade de memória da placa e pelo dispositivo possuir somente uma porta de comunicação serial disponível.

4.1 EXTENSÕES

Existem pontos que podem ser agregados ou melhorados no protótipo como sugestão pode-se citar:

a) utilizar um dispositivo que mede a velocidade aerodinâmica no momento em que é realizada a estabilização do aeromodelo a fim de verificar se existe a necessidade de aumentar sua velocidade para evitar a queda;
b) adicionar botões para fazer a reversão dos comandos das superfícies de controle, eliminando assim a necessidade desta modificação ser realizada somente via código;
c) adicionar um dispositivo de transmissão RF que envie as informações dos sensores em tempo real para uma estação em terra, capaz de armazenar os dados do voo a fim de analisar o desempenho do algoritmo ou armazenar em um *logger* dentro do avião;
d) desenvolver um interpretador do algoritmo de Kalman a fim de fazer uma verificação inicial antes de iniciar o voo, informando ao piloto a real possibilidade de estabilização do aeromodelo, caso a vibração seja excessiva o *software* deve avisar o piloto.

REFERÊNCIAS BIBLIOGRÁFICAS

AIUBE, Fernando A. L. **Modelagem dos preços futuros de commodities:** Abordagem do filtro de partículas. 2005. 183 f. Tese (Doutorado em Engenharia de Produção) – PUC Rio, Rio de Janeiro. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=7604@1>. Acesso em: 22 maio 2012.

ARDUINO. [S.l.], 2009a. Disponível em: <http://arduino.cc/en/Tutorial/HomePage>. Acesso em: 01 fev. 2012.

_____. **PWM**. [S.l.], 2009b. Disponível em: <http://arduino.cc/it/Tutorial/PWM>. Acesso em: 04 abr. 2012.

ATAN2. In: WIKIPEDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2012. Disponível em: <http://en.wikipedia.org/wiki/Atan2>. Acesso em: 12 maio 2012.

ATMEL Corporation. **MegaAVR**. San Jose, [2010?]. Disponível em: <http://www.atmel.com/products/microcontrollers/avr/megaavr.aspx?tab=overview>. Acesso em: 01 fev. 2012.

AUTOMAÇÃO. In: WIKIPEDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2012. Disponível em: <http://pt.wikipedia.org/wiki/Automa%C3%A7%C3%A3o>. Acesso em: 16 maio 2012.

BEIRES, José S. de. **Aviação elementar**: noções práticas para uso dos alunos-pilotos nas escolas de aviação civil. São Paulo: LEP, 1964.

BOLTON, William. **Engenharia de controle**. Tradução Valceres Vieira Rocha e Silva. São Paulo: Makron Books, 1993.

BRAIN, Marshall. **Como funcionam os micro-controladores**. [S.l.], 2009. Disponível em: <http://eletronicos.hsw.uol.com.br/microcontroladores.htm>. Acesso em: 05 fev. 2012.

CALUTA. **Lição 1:** vôo direto e nivelado. [S.l.], 2004. Disponível em: <http://caluta.atspace.com/>. Acesso em: 02 abr. 2012.

COELHO, Guilherme. **Acelerômetros**: o truque da detecção dos movimentos na nova geração de celulares. [S.l.], 2007. Disponível em: <http://webinsider.uol.com.br/index.php/2007/11/29/acelerometros/>. Acesso em: 10 fev. 2012.

COMO BALANCEAR SEU AEROMODELO. In: DICAS DE CONTRUÇÃO. [S.l.], 2010. Disponível em: <http://radiocontrolado.com.br/cg.asp>. Acesso em: 11 mar. 2012.

CORREA, Leonardo G. C. **Inferência da qualidade de produtos de destilação utilizando redes neurais e filtro de Kalman estendifo**. 2005. 57 f. Dissertação (Mestrado em Engenharia Elétrica) – PUC Rio, Rio de Janeiro. Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=7588@1>. Acesso em: 23 maio 2012.

EMBEDDED ARCHITECTS. **O que é um sistema embarcado**. Brasília, [2012?]. Disponível em : <http://www.embarc.com.br/p1586.aspx>. Acesso em: 16 maio 2012.

FMA DIRECT. **Co-pilot flight stabilization system**. Maryland, [2007?]. Disponível em: <http://www.revolectrix.com/cpd4_description_tab.htm>. Acesso em: 28 fev. 2012.

FUTABA. **Standards servo**. Tóquio, 2008. Disponível em: <http://www.futaba-rc.com/servos/analog.html>. Acesso em: 24 maio 2012.

KALMAN. In: WIKIPEDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2012. Disponível em: <http://pt.wikipedia.org/wiki/Filtro_de_Kalman>. Acesso em: 3 fev. 2012.

LANARI, Antônio P.; BORGES, Geovany A. **Desenvolvimento de um sistema de localização 3D para aplicação em robôs aéreos**. 2007. Dissertação (Mestrado em Engenharia Elétrica) - Universidade de Brasília, Brasília. Disponível em: <http://repositorio.bce.unb.br/handle/10482/3025>. Acesso em: 11 fev. 2012.

MATTHEWS, Anderson G. **ArduIMU ground station**. Melbourne, 2009. Disponível em: <http://diydrones.com/profiles/blogs/arduimu-groundstation-written>. Acesso em: 20 abr. 2012.

MONGE, Julian G. **Diseño e implementacion de um sistema de control remoto com sensores de movimiento y rotacion utilizando el protocolo de comunicacion inalámbrica ZIgBee**. 2011. 106 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica) – Universidade de Costa Rica, Cidade universitária Rodrigo Facio. Disponível em: <http://eie.ucr.ac.cr/uploads/file/proybach/pb2011/pb2011_005.pdf>. Acesso em: 14 apr. 2012.

MUÑOZ, Jordi. **First arduino IMU test**. San Diego, 2008. Disponível em: <http://diydrones.com/profiles/blogs/705844:BlogPost:19829>. Acesso em: 23 fev. 2012.

OLIVEIRA, Adjame A. G. **Estudo estatístico dos processos envolvidos em uma plataforma de atitude solidária**. 2000. 193 f. Dissertação (Mestrado em Engenharia e Tecnologia Espaciais / Mecânica Espacial e Controle) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos. Disponível em: <http://mtc-m16.sid.inpe.br/col/sid.inpe.br/iris@1913/2005/07.29.17.17/doc/publicacao.pdf >. Acesso em: 15 fev. 2012.

PEREIRA, Fábio. **Tecnologia ARM**: micro-controladores de 32 bits. São Paulo: Érica, 2007.

RETROALIMENTAÇÃO. In: WIKIPEDIA, a enciclopédia livre. [S.l.]: Wikimedia Foundation, 2011. Disponível em: <http://pt.wikipedia.org/wiki/Retroalimenta%C3%A7%C3%A3o>. Acesso em: 22 maio 2012.

ROCHA, Luciano V. **Algoritmo de alinhamento e nivelamento de um sistema de navegação inercial do tipo solidário ("STRAPDOWN")**. 2006. 165 f. Dissertação (Mestrado em Engenharia Mecânica) - Instituto Militar de Engenharia, Rio de Janeiro. Disponível em: <http://www.ime.eb.br/arquivos/teses/se4/mec2006/2006Luciano.pdf>. Acesso em: 10 mar. 2012.

RUZISKA, Marco A.; MICHELAN, Roberto. **Fusão sensorial aplicada a robótica móvel**. 2009. 10 f. Artigo para conclusão de curso (Pós-Graduação em Engenharia de Sistemas) – Uniasselvi, Blumenau. Obtido através de mensagem recebida por <aksizur@gmail.com> em: 18 abr. 2012.

SAKAJIRI, Alberto H. **Processamento digital de sinais de um giroscópio a fibra óptica modulado por onda quadrada**. 2008. 79 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica) – Instituto Tecnológico de Aeronáutica, São José dos Campos. Disponível em: <http://www.bd.bibl.ita.br/TGsDigitais/lista_resumo.php?num_tg=000549745>. Acesso em: 14 fev. 2012.

SAMPAIO, Ronivaldo P. **Sistema de controle de atitude embarcado para vôo autônomo de aviões em escala**. 2006. 186 f. Dissertação (Mestrado em Mecatrônica) – Escola politécnica e Instituto de Matemática, Universidade Federal da Bahia, Salvador. Disponível em: <http://wiki.dcc.ufba.br/Mecatronica/Dissertacao9>. Acesso em: 01 mar. 2012.

SENSORES DO AVIÃO. In: HOW STUFF WORKS. [S.l.], 2000. Disponível em: <http://ciencia.hsw.uol.com.br/avioes21.htm>. Acesso em: 11 mar. 2012.

SPARK FUN. Boulder, 2009. Disponível em: <http://www.sparkfun.com>. Acesso em: 01 mar. 2012.

SUPERFICIES DE CONTROLE. In: TOY WING, tudo sobre aeromodelo. [S.l.], [2010?]. Disponível em: <http://www.toywing.com.br/superficie-de-controle>. Acesso em: 15 abr. 2012.