DANIEL PAULO DEMITRIO MARTIN
IGNÁCIO MANAVELLA

DIRIGÍVEL HÍBRIDO AUTÔNOMO

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Graduação em Engenharia

Área de concentração: Engenharia Mecatrônica

Orientador: Professor Dr. Lucas Antonio Moscato

São Paulo
2003

DEDICATÓRIA

A luz, o sol, o ar livre
envolvem o sonho do engenheiro.
O engenheiro sonha coisas claras:
superfícies, tênis, um copo de água.

O lápis, o esquadro, o papel;
o desenho, o projeto, o número:
o engenheiro pensa o mundo justo,
mundo que nenhum véu encobre.

O Engenheiro; João Cabrão de Melo Neto

AGRADECIMENTOS

Agradecimentos aos colegas de classe e, principalmente, de grupo, que se ajudaram mutuamente; aos mestres, que assumiram a responsabilidade na lapidação de pedras brutas e aos pais, que sempre acreditaram na possibilidade dessa lapidação.

Um agradecimento especial aos ponteiros do relógio que se movimentaram numa velocidade compatível à urgência da entrega dos trabalhos.

Daniel

AGRADECIMENTOS

À meus pais e namorada pela paciência durante todo tempo de elaboração deste trabalho.

Ignácio

RESUMO

O objetivo deste trabalho é o desenvolvimento preliminar de um dirigível híbrido autônomo utilizando equipamentos e tecnologia acessíveis a um custo não muito elevado. O texto apresenta um estudo sobre dirigíveis e sobre o estado da arte de alguns equipamentos necessários à construção do protótipo final. Por fim são apresentados os testes e os resultados obtidos com o protótipo.

ABSTRACT

The objective of this research is the initial development of an autonomous hybrid airship using inexpensive equipment and technology. This text presents a study on airships and the state of the art of the equipment for the final prototype construction. Finally the tests and results with the prototype are presented.

SUMÁRIO

LISTA DE FIGURAS IX

LISTA DE TABELAS X

1. INTRODUÇÃO 1

1.1. Motivação 1

1.2. Objetivos 1

2. INTRODUÇÃO AOS DIRIGÍVEIS 3

2.1. Restrições 4

3. HIBRIDIZAÇÃO DO DIRIGÍVEL 6

4. SISTEMA DE POSICIONAMENTO 7

4.1. Dead-reckoning 8

4.2. Navegação inercial (INS – inertial navigation system) 9

4.3. Landmark (indicações terrestres) 9

4.4. Triangulação 10

4.5. Sistemas avançados 11

4.6. Considerações a respeito de sistemas de posicionamento 12

5. PROJETO DO HARDWARE DA ELETRÔNICA EMBARCADA 14

5.1. Sensores 14

5.2. Microcontrolador 18

5.3. Alimentação 20

5.4. Comunicação 23

6. SISTEMA DE CONTROLE 25

7. SISTEMA DE CONTROLE NO PROTÓTIPO 29

8. PROTÓTIPO 32

8.1. Válvula 32

8.2. Nacele 33

8.3. Eletrônica embarcada 34

8.4. Sistema de propulsão 37

8.5. Eletrônica em terra 38

8.6. Acionamento por computador 40

9. MATERIAIS E MÉTODOS 41

10. TESTES E RESULTADOS 43

10.1. Rádio 43

10.2. Transmissão de dados 44

10.3. Sensoriamento 44

10.4. Envelope 45

10.5. Empuxo insuficiente 45

10.6. Software de controle 46

10.7. Firmware 46

11. CONCLUSÕES 48

ANEXO A – CÓDIGO FONTE DO SOFTWARE DO MICROCONTROLADOR 50

ANEXO B – CÓDIGO FONTE DO SOFTWARE DE LEITURA DOS SENSORES 60

ANEXO C – CONSIDERAÇÕES SOBRE A CALIBRAÇÃO 70

ANEXO D – CÓDIGO FONTE DO SOFTWARE DE CONTROLE.....74

BIBLIOGRAFIA RECOMENDADA..... 103

APÊNDICE I – ESQUEMA DA LIGAÇÃO COMPUTADOR-RÁDIO..... I

APÊNDICE II – ESQUEMA FINAL DA ELETRÔNICA EMBARCADA..... II

APÊNDICE III – VÔO DE TESTE.....III

LISTA DE FIGURAS

FIG. 1: AUMENTO DO ERRO NO DEAD-RECKONING 8
FIG. 3: SINAL DE SAÍDA DO ADXL202 15
FIG. 4: CONFIGURAÇÃO DOS PINOS DO ADXL202 15
FIG. 5: ESTRUTURA INTERNA DO ADXL202 16
FIG. 6: PINOS DO MICROCONTROLADOR 19
FIG. 7: BATERIA DE CHUMBO-ÁCIDO 21
FIG. 8: PACK DE BATERIAS DE NI-CD 21
FIG. 9: BATERIA DE NI-MH 22
FIG. 11: FLUXOGRAMA DO ALGORITMO DE CONTROLE 31
FIG. 12: PROTÓTIPO COMPLETO (VISTA LATERAL) 32
FIG. 13: NACELE (VISTA FRONTAL) 33
FIG. 14: NACELE (VISTA INFERIOR) 34
FIG. 15: NACELE (VISTA TRASEIRA) 34
FIG. 16: PONTE H 35
FIG. 17: CIRCUITO ELETRÕNICO DO RECEPTOR DE RÁDIO 36
FIG. 18: PLACA FINAL DA ELETRÔNICA EMBARCADA 37
FIG. 19: MOTOR E HÉLICE UTILIZADOS 38
FIG. 21: ESQUEMA DE LIGAÇÃO OPEN DRAIN 40
FIG. 22: ESQUEMA DA LIGAÇÃO COMPUTADOR-RÁDIO I
FIG. 23: ESQUEMA FINAL DA ELETRÔNICA EMBARCADA II
FIG. 24: VÔO DE TESTE REALIZADO DIA 15 DE DEZEMBRO DE 2003 III

LISTA DE TABELAS

TABELA 1: COMPARAÇÃO ENTRE HIDROGÊNIO, HÉLIO E AR.....4
TABELA 2: VALORES DE CAPACITÂNCIA PARA FILTRAR FREQÜÊNCIAS....17
TABELA 3: TABELA VERDADE DO ACIONAMENTO DO CONTROLE.....39

1. INTRODUÇÃO

1.1. Motivação

Monitoramento aéreo. Cada vez mais se faz necessária a observação à distância de certos eventos e regiões como, por exemplo, áreas florestais, rebeliões em presídios, vigilância em lugares perigosos e inspeção de instalações; associando-se isso a coberturas esportivas, transporte de passageiros e carga e publicidade, tem-se um quadro bastante favorável à utilização de dirigíveis, tripulados ou não.

O custo de manutenção bem mais baixo do que o de helicópteros e sua segurança frente a falhas durante o vôo completam o quadro do potencial quase inexplorado dos dirigíveis.

1.2. Objetivos

O objetivo deste projeto é dotar um dirigível de um controle por computador que possibilite sua locomoção até um ponto no espaço sem a intervenção humana.

O aparelho será controlado remotamente por um computador que enviará sinais de controle aos motores, fazendo com que ele se mova de um ponto inicial – considerado a origem do sistema de coordenadas cartesianas – até o ponto especificado.

O sistema de controle receberá como retroalimentação a posição do balão fornecida por sensores embarcados e atuará corrigindo sua posição.

Ainda fazem parte do escopo deste projeto a construção da eletrônica necessária à operação do dirigível e o aprimoramento da parte mecânica de um modelo *indoor* adquirido.

Os diversos objetivos deste trabalho podem ser, então, enumerados:

- Acionamento remoto dos atuadores do dirigível;
- Tomada de decisão por software;
- Controle em tempo real;
- Posicionamento em três graus de liberdade: coordenadas cartesianas;
- Leitura e interpretação de sensores de posicionamento.

Objetivos secundários também podem ser enumerados:

- Possibilidade de utilizar computadores não muito rápidos;
- Independência entre dirigível e controle em terra;
- Operação sem riscos a operadores e equipamentos.

Este projeto é um passo inicial para um sistema mais complexo de um dirigível *outdoor* com automação completa que permita a sua aplicação em vôos de reconhecimento e monitoração. Sistemas redundantes e totalmente embarcados deverão, ainda, ser implementados e assim suas utilizações passam a ser aplicáveis a regiões povoadas.

2. INTRODUÇÃO AOS DIRIGÍVEIS

É chamado dirigível qualquer veículo que se possa dirigir. Neste trabalho, entretanto, chamar-se-á dirigível um veículo mais leve que o ar e dotado de um sistema de propulsão.

Este veículo é subdividido em duas partes: o envelope e a nacele. O envelope é o balão, o invólucro inflável que contem o gás menos denso do que o ar e que, por empuxo, sustenta o aparelho acima do solo. A nacele é a gôndola, a estrutura que suporta o sistema de propulsão, o combustível (fonte de energia) e o operador (quando necessário) ou o controle. Também podem ser incluídos um sistema de comunicação e as ferramentas necessárias à execução da tarefa à qual o aparelho se propõe.

Quanto à classificação, os dirigíveis podem ser rígidos, semi-rígidos e não rígidos, segundo a estrutura adotada do envelope. O tipo rígido é feito de uma armação leve, com cobertura externa, contendo vários reservatórios de gás mantidos por uma rede. Os famosos zepelins alemães e a maioria dos dirigíveis das décadas de 20 e 30 eram desse tipo. A estrutura era uma liga de alumínio e sua cobertura em tecido de algodão. Os tipos semi-rígido e não-rígido tornaram-se conhecidos como dirigíveis de pressão, porque sua forma depende da pressão interna. O semi-rígido possui apenas uma quilha de metal ao longo do depósito de gás e o não rígido não apresenta nenhuma estrutura.

É importante dizer que o gás contido no envelope não está sob pressão, isto é, encontra-se à pressão atmosférica local. São dois os motivos para isso, em primeiro lugar, se um gás está sujeito a um a pressão mais elevada, sua massa ocupa um volume menor e, portanto, sua densidade aumenta. Perdemos, assim, sustentação.

O segundo motivo é que, se por um acidente o envelope é furado, o gás à pressão atmosférica escapa muito lentamente e levaria horas, dias e até mesmo semanas para que o dirigível tivesse que voltar ao solo por perda de sustentação. Isto confere à aeronave maior segurança.

Os dirigíveis do tipo não-rígido são também conhecidos como BLIMPs e serão chamados desta maneira ao longo do trabalho.

Quanto à forma de sustentação, podem ser ainda puros e híbridos. Os dirigíveis puros são os que seguem à risca a definição, ou seja, são mais leves que o ar e sua sustentação está totalmente no envelope. Os híbridos relaxam a definição, sendo mais pesados do que o ar. A maior parte da sustentação está no envelope e a restante numa asa ou rotor (asa rotativa).

Podem ser usados tanto hélio (He) quanto hidrogênio (H_2) como gás de sustentação. A Tabela 1 lista as densidades dos gases em questão, à pressão de 1 atm.

	Hidrogênio gasoso (H_2)	Hélio (He)	Ar (mistura de gases)
Densidade (kg/m^3)	0,09	0,16	1,2

Tabela 1: comparação entre hidrogênio, hélio e ar

Ao analisar as densidades, pode ser notado que o H_2 é mais vantajoso com relação à carga útil (menor densidade); porém devido à inflamabilidade desse gás, optou-se pela segurança do He (gás inerte). Comparando, agora, com a densidade do ar, pode-se perceber que a carga útil será de aproximadamente 1 kg/m^3.

2.1. Restrições

As restrições do projeto de um dirigível estão em sua capacidade de carga e na potência dos motores. A capacidade de carga limita a quantidade de equipamento que pode ser embarcada e é ditada basicamente pelo volume de gás (hélio) que o envelope comporta, com isto deve-se escolher o tamanho ideal do balão com uma estimativa do peso dos equipamentos a serem carregados.

Os motores são responsáveis por fazer o balão locomover-se; sua potência fará a diferença entre vencer uma corrente de ar ou ficar à deriva no céu. Quando o dirigível possui motores de baixa capacidade e não tem força para vencer o vento ele é considerado *indoor*, e *outdoor* quando contrário. No dirigível *outdoor* o envelope desempenhará um papel importante funcionando como estrutura da aeronave para sustentar além da nacele superfícies de controle adicionais (aletas, profundores,...), assim o envelope é fabricado com materiais rígidos e também resistentes às intempéries ausentes ao *indoor*.

3. HIBRIDIZAÇÃO DO DIRIGÍVEL

Suponha que, durante uma operação, o receptor de rádio montado na nacele pare de funcionar adequadamente e seja perdido o controle do dirigível. Se o dirigível for puro, ele continuará flutuando e não teremos como recuperá-lo a não ser que o hélio do envelope escape e o aparelho perca sustentação.

Para evitar que isso ocorra, optou-se por construir um dirigível híbrido. Desta forma, se algo de errado acontecer, os motores pararão, a sustentação diminuirá e o aparelho retornará ao solo suavemente.

Há dois tipos de dirigíveis híbridos. O primeiro tipo é o que tem asas e apresenta características semelhantes aos aviões: a sustentação que falta é dada pela velocidade relativa da asa com relação ao vento (ar).

O segundo tipo possui um rotor e possui semelhanças com os helicópteros: a sustentação restante está relacionada com a rotação do rotor: quanto mais rápido, mais sustentação e, portanto, mais alto voa. Assim, quando o aparelho está abaixo da cota desejada, o motor é acionado de forma a impulsionar o ar para baixo, criando uma força vertical para cima que sustenta o balão. Quando acima da cota, o motor é acionado no sentido oposto, gerando uma força vertical para baixo.

Neste trabalho, a hibridização foi simplificada para apenas uma forma de acionamento: quando o dirigível estiver abaixo da cota, o funcionamento é idêntico ao explicado acima. Quando acima da cota o motor é desligado e a força da gravidade se encarrega de levar o conjunto para a altura pretendida.

4. SISTEMA DE POSICIONAMENTO

O sistema de posicionamento deve fornecer a localização da aeronave para que o controle tome as decisões para o deslocamento do robô.

As características a se destacar de um sistema de posicionamento são:

- Precisão;
- Alcance;
- Tempo de resposta.

Procura-se sempre um sistema preciso para que o controle possa trabalhar eficientemente; um grande alcance permite um espaço de locomoção amplo e tempo de resposta rápido evita que o processamento de controle não seja retardado.

Os sistemas de posicionamento conhecidos podem ser divididos em dois grandes grupos. Aqueles baseados em informações internas ao robô – relativo ao seu deslocamento – e aqueles baseados na informação externa – absoluta – de um ponto fixo.

Assim nos sistemas relativos a posição atual do robô é conhecida através da informação anterior de posição mais o conhecimento da trajetória realizada. Nos absolutos a posição atual é conhecida pela informação do exterior do robô, como pontos geográficos.

A seguir são discutidos alguns modos de posicionamento bem como vantagens e desvantagens.

4.1. Dead-reckoning

Neste sistema a posição é determinada com base na direção do movimento e velocidade onde uma contagem do tempo irá definir a distância percorrida e, juntamente com a direção, o ponto em que se está em relação ao início da contagem.

Um exemplo é o método utilizado pelos piratas para esconder tesouros, seus mapas transcreviam o posicionamento definido por número de passos a serem dados numa certa direção.

A vantagem deste método reside em sua simplicidade e custo de sensores utilizando uma bússola e um odômetro, geralmente. O tempo de resposta é rápido e sua precisão está ligada aos sensores.

Uma desvantagem é o fato dos erros acumularem-se proporcionalmente ao tempo de utilização e os erros serem somados. Com isto o alcance pode ser considerado proporcional ao deslocamento pois quanto mais se desloca maior o erro de localização.

A Fig. 1 mostra o aumento do erro num exemplo de deslocamento em linha reta com pausas. As linhas indicam o erro no ângulo de direção e as circunferências o erro da distância.

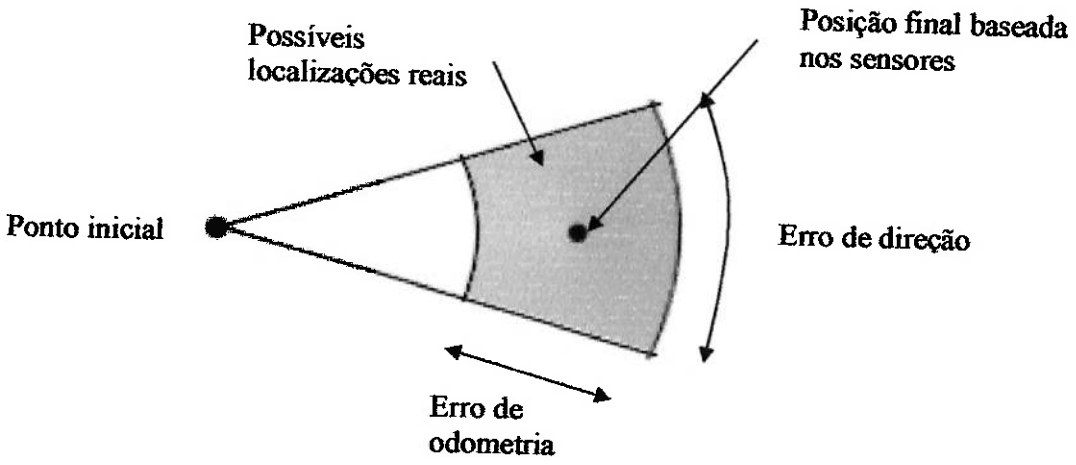

Fig. 1: Aumento do erro no dead-reckoning

4.2. Navegação inercial (INS – inertial navigation system)

Este sistema é derivado do dead-reckoning utilizando como informação as acelerações do objeto e as velocidades angulares. Com estas informações determina-se a distância percorrida e os desvios na direção de deslocamento computando a posição atual com relação ao ponto de início.

Os sensores utilizados são acelerômetros e giroscópios fornecendo aceleração e deslocamento angular. A evolução tecnológica destes sensores vem permitindo o seu uso em pequenas aplicações e seu tempo de resposta está na ordem de milisegundos.

Uma desvantagem deste sistema, assim como no dead-reckoning, é a acumulação de erros com o uso prolongado, pois os erros de cada medida são somados.

Sobre o alcance, supondo uma precisão não grosseira, pode-se dizer que será limitado pela velocidade do objeto em estudo. Para objetos rápidos o alcance será maior já que cobrirá uma grande distância em pouco tempo e assim um espaço maior com precisão aceitável.

4.3. Landmark (indicações terrestres)

Neste sistema de posicionamento o objeto tem sua localização determinada cada vez que se aproxima de um ponto de referência com posição conhecida.

Um exemplo cotidiano é quando alguém, perdido numa cidade, pede informações de como chegar a um destino e é orientado a dobrar a direita em uma loja e prosseguir em frente até a padaria. A loja e a padaria são as indicações de posicionamento que orientam a pessoa.

Este sistema é interessante em aplicações onde robôs executem percursos cíclicos. Linhas ferroviárias, ônibus ou o percurso de um pallet numa linha de montagem industrial são exemplos.

A fim de evitar erro no posicionamento deve se assegurar que o objeto sempre encontrará um landmark ou, caso se perca uma marcação, seja possível distinguir a marca perdida da próxima a ser lida; utilizando-se mais uma vez o exemplo dado, no caso de dizer-se a uma pessoa para virar na terceira rua à esquerda e ela esquecer de contar uma rua ela virará na quarta à esquerda. Portanto para evitar este tipo de problema as indicações de lojas são mais seguras por distinguir entre a terceira e a quarta rua.

O alcance dependerá do espaço coberto por landmarks e seu tempo de resposta é rápido e preciso já que ao se confirmar a localização de um landmark assume-se as coordenadas dele. Por depender de pontos únicos no espaço este é um sistema de posicionamento discreto.

4.4. Triangulação

Para a localização de um objeto a triangulação necessita de três pontos de posição conhecida e a distância destes três pontos com relação ao objeto fornecendo a localização num plano pelo cálculo da intersecção de três circunferências.

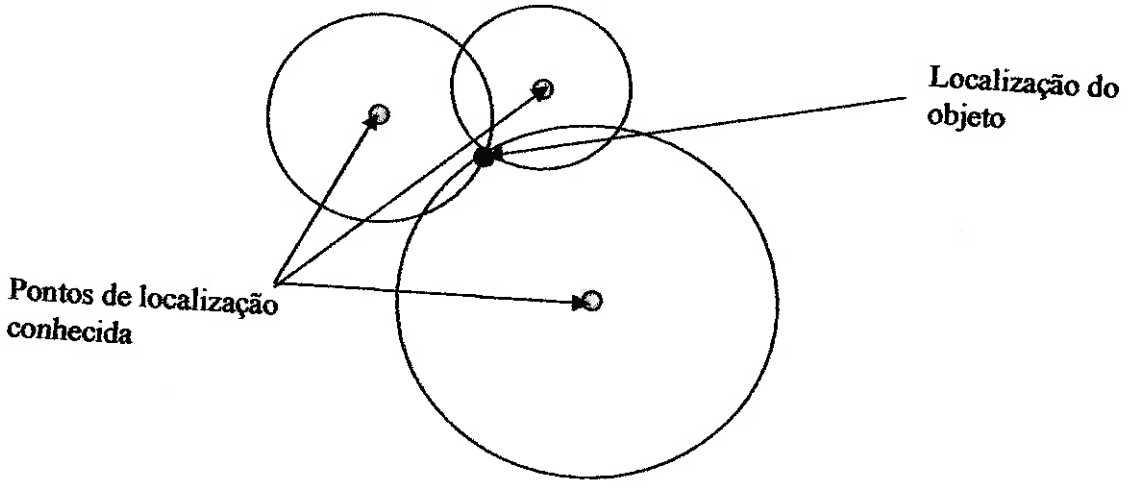

Fig. 2: Sistema de posicionamento por triangulação

Para o cálculo da distância do objeto com cada um dos vértices do triângulo é feita uma emissão de uma onda do objeto para cada um dos pontos, sabendo-se o tempo que a onda levou para percorrer a distância que os separa e a velocidade da onda a distância estará determinada.

As ondas utilizadas são as eletromagnéticas, como rádio e infravermelho ou as sonoras, como o ultra-som. O rádio é utilizado quando as distâncias a serem percorridas são grandes, como no caso do GPS (global positioning system), e o ultra-som quando as distâncias são pequenas. A onda de som perde intensidade conforme se desloca no meio físico e assim não pode ser utilizada para medir grandes distâncias além de não se propagar no vácuo enquanto que a onda de rádio é rápida demais e a contagem do tempo de deslocamento torna-se imprecisa para pequenas distâncias.

É importante salientar que neste sistema tem-se um posicionamento contínuo.

4.5. Sistemas avançados

Outros sistemas de posicionamento estão sendo desenvolvidos, e entre eles, a visão computacional. Este método tenta reproduzir nos robôs a capacidade que os animais têm de se localizar a partir do sentido da visão. Assim o robô ganha uma autonomia notável já que agora ele pode implementar um algoritmo de busca além de um sistema de segurança que faça desvios de obstáculos. A complexidade deste método reside no fato de se necessitar grande poder computacional. Outro sistema baseia-se na reprodução de superfícies apenas conhecendo a intensidade de ondas refletidas. Aqui entram estudos com ultra-som onde a percepção da intensidade com que uma onda retorna a um sensor aliada à estudos do comportamento do som gera uma imagem do ambiente próximo.

4.6. Considerações a respeito de sistemas de posicionamento

As pesquisas para a construção de um sistema de focalizaram-se em combinar um sistema relativo com um absoluto. Este fato deve-se aos sistemas de informação relativa terem uma resposta rápida, porém erro crescente com o tempo de medição e os sistemas de informação absoluta com tempo de resposta lento em relação ao outro mas com precisão fixa podendo ser utilizado para corrigir o erro do primeiro a cada atualização de resposta.

Associações comuns são Dead-reckoning com Landmark e navegação inercial com triangulação. A primeira associação foi utilizada por equipes de futebol de robô onde a marca dava a posição com precisão fixa quando o robô estava em cima de uma delas e o Dead-reckoning quando o robô não estava. A navegação inercial vem sendo estudada para uso como recurso auxiliar aos sistemas de posicionamento global que possuem tempos altos de resposta (da ordem de um minuto) em ambientes de baixa intensidade do sinal, com isto em momentos em que o sistema global não esta disponível o uso da navegação inercial fornece uma estimativa da posição.

Os requisitos para o sistema de posicionamento deste trabalho são impostos pelo tamanho da aeronave e carga útil. Assim o sistema não poderá ser embarcado havendo a necessidade da transmissão das informações dos sensores internos.

O sistema dead-reckoning traria uma complexidade de sensores necessitando de um anemômetro embarcado e outro em terra para calcular a velocidade do objeto descontando a velocidade do vento, não sendo prático seu uso aqui.

A navegação inercial é atraente visto que há disponibilidade comercial de acelerômetros. É grande o interesse no desenvolvimento deste sistema para aumentar a precisão do GPS, assim nos grandes centros onde o sinal do GPS não é captado a navegação inercial garante a localização.

O landmark é uma solução de baixo custo, mas inevitavelmente deve ser utilizada com suporte de um sistema de dead-reckoning ou navegação inercial. Porém as

aplicações em que os dirigíveis são aplicados e a imprevisibilidade do comportamento da aeronave são fatores que descartam o uso deste sistema.

A triangulação é um sistema que trará uma certa flexibilidade com relação ao anterior por ter uma precisão contínua. O uso do GPS é interessante pelo seu alcance global e sua facilidade de integração com equipamentos eletrônicos, porém sua precisão pode resultar muito instável variando de 2 metros à mais de 10 e isto seria prejudicial para o controle.

Após o exposto, a solução para este trabalho segue a tendência tecnológica de elaborar um sistema de navegação inercial que auxiliaria um sistema GPS.

5. PROJETO DO HARDWARE DA ELETRÔNICA EMBARCADA

Os componentes físicos do sistema de controle devem principalmente permitir que todos os cálculos necessários sejam executados na velocidade certa. Outras funções a serem consideradas são telemática e redundância de sistemas.

A execução dos cálculos em tempo real torna-se necessária à medida que uma demora no envio de comandos aos motores pode causar uma colisão, ou um atraso no recebimento das informações referentes à posição pode gerar comandos equivocados. Portanto, um processador que realize o controle em tempo real e uma linguagem computacional que permita tanto o acesso a informações de baixo nível quanto o uso de funções de alto nível é imprescindível para este projeto. Deste modo, todos os cálculos são efetuados antes do início do próximo ciclo de controle e o balão executa o movimento no momento certo.

A telemática é a transmissão de informação da aeronave com uma unidade não embarcada, assim algumas informações podem ser processadas mais rapidamente em computadores em terra permitindo ao processador embarcado focar no acionamento dos motores e na aquisição dos dados dos sensres.

5.1. Sensores

O sistema de localização da aeronave utiliza acelerômetros, no caso o modelo ADXL202 da Analog Devices. Este acelerômetro é produzido com tecnologia MEMS e suas principais características são seu pequeno tamanho e facilidade de tratar sua saída e filtrar ruídos indesejáveis.

O sinal de saída deste acelerômetro é do tipo pulsada (PWM) onde a aceleração corresponde à razão do tempo do sinal em alta pelo período do sinal. O período do sinal é ajustável de 0,5 a 10ms simplesmente variando o valor de um resistor. Este tipo de

saída traz vantagens no uso com microcontrolador, por não ser um sinal analógico não é necessário que o processador tenha um conversor analógico digital mas sim um temporizador preciso, Este componente fornece a mesma saída nos níveis de tensão de 3 a 6 volts a um consumo de 10mA o que permite o uso de pilhas comuns.

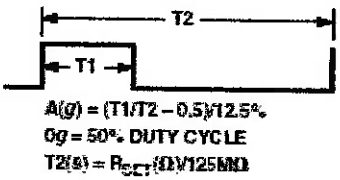

Fig. 3: Sinal de saída do ADXL202

A aceleração, então, é calculada pela seguinte equação:

$$A(g) = \frac{\left(\frac{T_1}{T_2} - 0{,}5\right)}{12{,}5\%} \qquad (1)$$

PIN CONFIGURATION

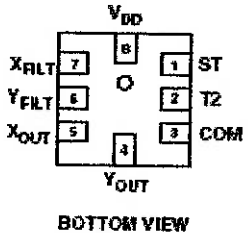

Fig. 4: Configuração dos pinos do ADXL202

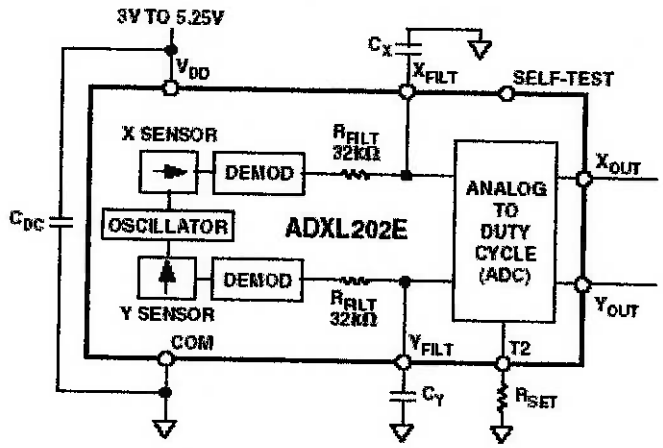

Fig. 5: Estrutura interna do ADXL202

Pelo esquema elétrico nota-se os principais elementos a serem considerados na montagem do circuito dos acelerômetros.

- V_{dd} – entrada de alimentação positiva
- COM – aterramento ou comum do componente
- X_{out} Y_{out} – saída do sinal dos acelerômetros.
- C_{cd} – capacitor de desacoplamento, sua função é evitar que falhas e flutuações no fornecimento de energia elétrica prejudiquem o funcionamento do componente, geralmente um valor de 0,1μF é suficiente.
- C_x C_y – capacitores para filtrar as freqüências que estimulam o acelerômetro. No caso estes capacitores formam um filtro passa baixa. Na tabela a seguir estão os valores de freqüência.

Banda (Hz)	Capacitância (μF)
10	0,47
50	0,1
100	0,05
200	0,027
500	0,01
5000	0,001

Tabela 2: Valores de capacitância para filtrar freqüências

- R_{set} – resistor que define o tamanho do período de pulso, a eq.(2) define o período em milisegundos sendo este limitado entre 0,5 a 10 milisegundos.

$$T2 = \frac{R_{set}(\Omega)}{125M\Omega} \tag{2}$$

O valor da aceleração corresponde à razão entre o tempo em alta do sinal pelo período. O acelerômetro foi projetado para que quando nenhuma aceleração esteja atuando sobre ele o valor da razão anterior seja a metade (50%) e para uma variação da razão em 12.5% corresponda a 1g (gravidade), sendo o fundo de escala 2g a razão pode variar de 12.5% a 75% do período.

A incerteza de leitura é devida a variações no período do pulso que é afetado pela temperatura e pela precisão do resistor de ajuste. Outro problema está no fato deste ser um acelerômetro estático e sofrer a influência da aceleração da gravidade, assim pequeno desalinhamento faz com que à leitura seja adicionada uma parcela da gravidade. Deve se anular este efeito por calibração.

O posicionamento dos sensores na aeronave visa obter um sistema de navegação inercial simplificado. O módulo teórico é constituído por três acelerômetros e um

giroscópio enquanto que o simplificado apresenta quatro acelerômetros cuja disposição na aeronave fornece as translações e a rotação em torno da vertical.

Para isso, dois acelerômetros encontram-se na frente do aparelho medindo os deslocamentos frontal e lateral, os outros dois, na parte traseira e medem os deslocamentos lateral e vertical. Pela relação entre os deslocamentos mensurados pelo par de acelerômetros laterais e sabendo-se a distância entre eles, descobre-se se o movimento é de translação, rotação ou uma associação dos dois.

5.2. Microcontrolador

O microcontrolador tem como função a correta amostragem dos sensores embarcados, acionamento dos motores em resposta a um sinal de rádio e envio dos dados dos sensores encapsulados em mensagens digitais.

Assim contabiliza-se a necessidade dos seguintes pinos:

- 4 pinos para a leitura dos acelerômetros;
- 5 pinos para acionamento dos motores;
- 1 pino para transmissão serial.

Para tornar mais simples o desenvolvimento do software, exige-se também:

- Temporizadores;
- Hardware para transmissão serial;
- Interrupções de mudança de estado.

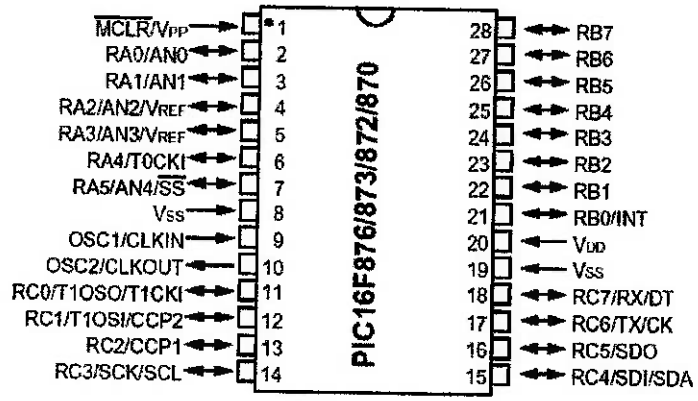

Fig. 6: Pinos do microcontrolador

O PIC16F876 atende as necessidades tanto de pinos de entrada e saída como de temporizadores.

Ele dispõe de:

- Dois pinos para comunicação serial: RC7 e RC6;
- Quatro pinos de entrada com interrupção na mudança de estado: RB4, RB5, RB6 e RB7;
- 3 temporizadores: TMR0, TMR1 e TMR2.

Sua programação é simples por ter um ciclo de instruções pequeno bem como ter um hardware que facilita o desenvolvimento de firmware.

Assim os acelerômetros são conectados a quatro pinos de aquisição quaisquer para que possa ser feita a leitura. Os sinais provenientes do rádio controle são ligados aos pinos com interrupção de mudança de estado, fazendo com que com qualquer sinal de controle o microcontrolador acione os motores e depois retorne a leitura dos acelerômetros. O pino de transmissão serial será utilizado para acionar o drive de infravermelho e transmitir dados.

5.3. Alimentação

O fornecimento de energia deve permitir que todas as operações previstas para a eletrônica embarcada possam ser executadas; assim faz-se necessário prever o consumo de energia.

Na aeronave toda a energia utilizada é de natureza elétrica; destacamos como fontes de consumo embarcadas:

- Acelerômetros: menos de 10mA de consumo e operando de 3V a 6V;
- Microcontrolador: 50mA de consumo e 3V a 6V;
- Led de transmissão: 250mA acima de 3V;
- Eletrônica adicional: 20mA operando em 5V;
- Motores: 300mA cada (acima de 3V).

Totalizando uma previsão de 1230mA.

Para suprir essa demanda de energia devemos avaliar as possibilidades de alimentação focando portabilidade.

Os principais fatores a serem observados são:

- Capacidade de carga: quanto mais carga, mais tempo durará o vôo;
- Tensão por célula: o sistema deve trabalhar em 5 volts assim e necessário baterias com tensões maiores;
- Descarga em vazio: sem uso, a bateria se descarrega com o tempo;
- Peso.

5.3.1. Chumbo ácido

Bateria utilizada em automóveis e atualmente encontrada em diversas formas e tensões. Possui grande capacidade de carga, porém apresenta grande peso (devido ao chumbo) se comparada com as demais tecnologias.

Outra desvantagem esta no uso de metal pesado não podendo ser descartada em qualquer lugar.

Fig. 7: Bateria de chumbo-ácido

5.3.2. Níquel-Cádmio (Ni-Cd)

Tecnologia recente. Muito mais leve que as de chumbo-ácido, são atraentes por serem fáceis de carregar e mais leves. Aqui também estão presentes metais pesados sendo seu descarte regulamentado por lei. A descarga em vazio também é uma desvantagem, por conseguinte deve-se ficar atento quanto ao uso destas baterias.

Fig. 8: Pack de baterias de Ni-Cd

5.3.3. Níquel-metal-hidreto (Ni-MH)

Tecnologia também recente como a Ni-Cd esta bateria consegue ser ainda mais leve e com maior capacidade de carga, considerada uma bateria ecológica por não conter elementos que agridam a natureza como os metais pesados das demais. Esta bateria é comumente utilizada em telefones celulares.

Suas desvantagens são os cuidados com a recarga, que deve ser controlada para evitar sua explosão, e a descarga em vazio – ainda maior que as Ni-Cd.

Fig. 9: Bateria de Ni-MH

5.3.4. Alcalinas

Mais comuns no mercado, as pilhas alcalinas não demonstram a menor utilidade para uma aplicação remota de alto consumo por não ser possível recarregá-las.

5.3.5. Escolha

Por serem comuns e não exigirem uma recarga cuidadosa, as baterias de Ni-Cd são as indicadas para esta aplicação.

Cada célula fornece 1.2 volts, 6 células em série nos darão 7.2 volts atendendo a necessidade de uma tensão elevada para um regulador reduzi-la a 5 volts para o sistema. Cada célula tem uma capacidade de carga da ordem de 300mAh à 2500mAh. Uma célula que forneça 600mAh será suficiente para uma autonomia de vôo de cerca de meia hora.

5.4. Comunicação

Os dados da aeronave devem ser transmitidos para o computador em terra sendo conveniente o uso de sistemas sem fios e que trabalhem com informação digital. Em teoria qualquer forma de onda pode ser utilizada para transmitir informação, dentre as conhecidas podem ser citados o rádio e o infravermelho sendo possível uma transmissão utilizando ultra-som.

A transmissão por infravermelho é composta de um led emissor de infravermelho e um foto transistor.

Qualquer fonte de infravermelho pode saturar o foto transistor sendo necessário modular a luz. Com um filtro acoplado ao foto transistor é possível filtrar uma freqüência, no caso a emitida pelo led, e distinguir um sinal de transmissão válida de uma fonte qualquer de emissão de infravermelho (o sol, lâmpadas incandescente, seres de sangue quente).

A forma em que os dados serão enviados é assíncrona serial. Esta transmissão é caracterizada por sinais em dois níveis de tensão que digitalmente são 0 e 1 e bits que são enviados em função do tempo sendo vital o uso de um relógio preciso para não

perder sincronismo na transmissão. A transmissão começa com o envio de um bit de inicio (Start Bit) que tem como função avisar ao receptor que um byte será transmitido e que a partir daquele instante começa a contagem de tempo para a obtenção do byte.

A construção do sistema de comunicação sem fio utilizará para o emissor:

- CI 555: Oscilador responsável por gerar a freqüência de modulação do infravermelho;
- LED emissor de infravermelho.

Para o receptor:

- Foto receptor de infravermelho: este componente já vem com filtro e sistema de ganho para acionar uma saída sempre que receber uma luminosidade em infravermelho dentro de freqüência filtrada que no caso depende do modelo e varia de 20kHz a 145kHz;
- MAX232 (ou SP232): conversor de sinal TTL para RS-232 e vice-versa.

Este sistema basicamente converterá sinais TTL em infravermelho que será transmitido até o receptor onde será convertidos em sinais RS-232 e interpretados pelo computador em terra. Ele apresenta certas falhas como perda de dados pela falta de visibilidade do receptor e erro nos dados como a inversão de um bit numa mensagem. Este tipo de falha pode ser contornada por software com uma repetição das informações de forma a garantir que a mensagem chegará certa.

6. SISTEMA DE CONTROLE

De modo simplificado, o sistema de controle de um dirigível autônomo, em terra ou embarcado, deve abranger os seguintes pontos:

- Aquisição dos dados referentes ao posicionamento e orientação;
- Tratamento dos dados adquiridos para interpretação das informações;
- Comparação da posição e orientação da aeronave com o destino e
- Acionamento dos atuadores para movimentação do aparelho.

Os dados dos sensores precisam ser adquiridos e armazenados em registradores de onde podem ser consultados quando necessário.

Quando cada dado é adquirido diretamente por uma porta de entrada de dados, o armazenamento torna-se mais fácil pois cada endereço consultado indica um único valor a ser tratado. No entanto, quando mais de um dado é adquirido por uma mesma porta – como no caso de uma comunicação serial ou o envio de informações pré-processadas por um outro microprocessador – deve-se tomar cuidado para se realizar uma correta separação dos dados e, primeiramente, a identificação inicial dos dados a receber.

Pode-se tomar como exemplo o mouse do computador. Ele envia informações ao processador de modo serial. Estas informações são agrupadas, num mouse simples, em três palavras de oito bits cada, ou seja, três bytes. O primeiro indica qual botão está pressionado e qual a direção de rotação dos dois encoders das direções x e y, o segundo e o terceiro dizem quantos pulsos os encoders x e y (respectivamente) leram. Para saber se estes pulsos devem ser somados ou subtraídos do valor antigo – os pulsos são sempre incrementais – o processados verifica os bits referentes à direção contidos no primeiro byte. Esta transferência de informações entre mouse e computador é, normalmente, realizada a 1200 bits por segundo (bps).

A identificação de qual byte está-se lendo é feita por um conjunto de dados que caracteriza o primeiro dos bytes, a partir daí o computador reconhece os outros dois pela

ordem na qual aparecem. Portanto a existência de uma palavra de referência para a determinação das demais é imprescindível no caso de dados enviados agrupados.

Tratamento de dados refere-se ao processamento das informações adquiridas antes de serem realmente interpretadas. Como exemplo podem ser citados:

Um potenciômetro no qual a tensão lida é proporcional ao valor da grandeza medida e não pode ser utilizada diretamente sem tratamento.

Um acelerômetro que fornece uma onda quadrada tem o valor da aceleração no *duty cycle* desta onda.

Um tubo de Pitot posicionado no nariz de um avião fornece a velocidade relativa da aeronave com relação ao vento, e não com relação ao solo, portanto deve ser tratada.

Um sistema de GPS indica a posição de um dado dispositivo nas coordenadas do planeta, mas deve ser tratado para indicar a posição em relação ao sistema de coordenadas de interesse.

A maior parte dos sensores não fornece dados absolutos, prontos para uso, portanto o sistema de controle deve estar preparado para transformá-los em valores adequados.

A comparação da posição e orientação atual com a de destino e o acionamento dos atuadores trabalham juntos para a execução do movimento. O controle deve ter um algoritmo que planeje a rota sabendo os valores das posições atual e de destino e a orientação atual. Note que não é necessário um complexo sistema de decisão baseado em algoritmo genético ou similares.

Em muitos casos, a direção é corrigida apenas quando o erro ultrapassa um certo valor de tolerância – comumente alguns graus para mais ou para menos. No restante do tempo, somente a posição é levada em conta.

Para dirigíveis outdoor (explicado nos Capítulo 2), a direção do balão não é diretamente importante devido ao forte efeito que o vento pode causar. Por exemplo, se

um dirigível deve ir de P1 para P2 e um forte vento soprar numa direção aproximadamente perpendicular à reta que une P1 e P2, o balão voará praticamente de lado para compensar este efeito (veja Fig. 10: Compensação da força do vento).

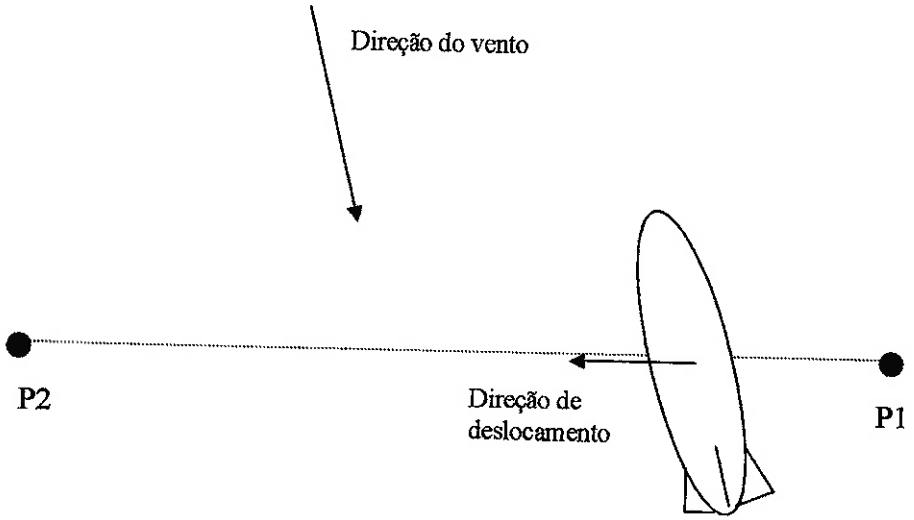

Fig. 10: Compensação da força do vento

Neste caso a direção de interesse é a de deslocamento real do aparelho e não a orientação do mesmo.

Um sistema mais avançado deve ser capaz de planejar uma trajetória entre a origem e o destino e não apenas corrigir alinhamento e posição e de seguir esta trajetória corrigindo desvios durante o percurso.

Este sistema também deve incluir uma modelagem do dirigível abrangendo as relações entre potência desenvolvida pelos atuadores e ação resultante, bem como efeitos de resistência do ar, força do vento, inércia e variação da pressão do ar – que interfere diretamente na capacidade de sustentação.

Para dirigíveis que sobrevoem regiões povoadas é exigido por lei que apresentem um sistema de controle redundante e embarcado.

Um sistema redundante é composto por dois ou mais sistemas de controle no qual um deles executa as operações enquanto o(s) outro(s) monitora(m) suas ações, caso ocorra uma falha no primeiro, o segundo assume o comando e mantém o aparelho no ar, sem riscos para as populações do local, chamado também de sistema em paralelo. Deste modo a confiabilidade do todo aumento pois é dada pelo menor dos valores de confiabilidade dos sistemas em paralelo e mesmo que um deles falhe, a aeronave permanece funcionando devido à atuação dos demais.

Um sistema é embarcado quando todo o seu processamento é realizado por um ou mais controladores que voam solidários ao dirigível. O monitoramento pode ser realizado em terra, mas nenhuma ação deve depender de comandos advindos de fora do balão.

7. SISTEMA DE CONTROLE NO PROTÓTIPO

Como o objetivo deste trabalho não é construir um dirigível de grande porte nem redundante, o sistema de controle contém apenas os princípios básicos explicados anteriormente mais uma rotina de calibração.

O software de controle pode ser resumido às suas funções básicas e demonstrado no fluxograma abaixo.

I. O início do programa fornece informações gerais sobre o trabalho e instruções para a realização do próximo passo, a saber, a calibração.

II. Como não há garantia de que nem os acelerômetros foram montados com alinhamento desejado nem de que a nacele foi corretamente montada no envelope, existe a necessidade de calibrar a leitura dos valores de aceleração. Caso ocorra um erro de alinhamento, uma translação de 1m (por exemplo) na direção x será lida como uma translação menor em x e pequenos deslocamentos em y e z serão captados. Esta calibração pede deslocamentos conhecidos e entrega uma matriz de transformação de coordenadas (rotação) para ser aplicada continuamente às medições.

III. Adquirir a posição de destino significa perguntar ao usuário a posição que ele deseja que o dirigível alcance. Os dados são fornecidos ao programa através do teclado do computador e representam as coordenadas cartesianas de destino, com relação ao zero do sistema – posição inicial em repouso do balão.

IV. A aquisição da posição e orientação atual do balão é feita pela porta serial na forma de bytes codificados que representam os valores dos *duty cycles* dos acelerômetros. Os dados são integrados duas vezes para se descobrir o deslocamento relativo à etapa anterior. A orientação é calculada a partir dos deslocamentos dos acelerômetros Y1 e Y2.

V. Sabendo-se a posição atual, o algoritmo compara-a com a de destino. Caso a distância que as separa for pequena o suficiente, o dirigível chegou ao local desejado e o programa termina, caso contrário, deve-se locomover.

VI. Se há a necessidade de locomoção, o algoritmo agora compara a orientação do balão. Se o erro de direção for menor do que 20°, isto é, a linha reta que une a posição atual com o destino formar um ângulo menor do que 20° com a orientação da aeronave, não é necessário corrigir a orientação e vice-versa.

VII. Se a direção não está satisfatória, o balão executa uma rotação em torno de seu próprio eixo para endireitá-lo. A translação em x e y é interrompida nesta ocasião, entretanto a translação em z segue independente.

VIII. Se o local desejado ainda não foi alcançado e a orientação encontra-se dentro da precisão, é executada a translação em linha reta tentando alcançar o destino.

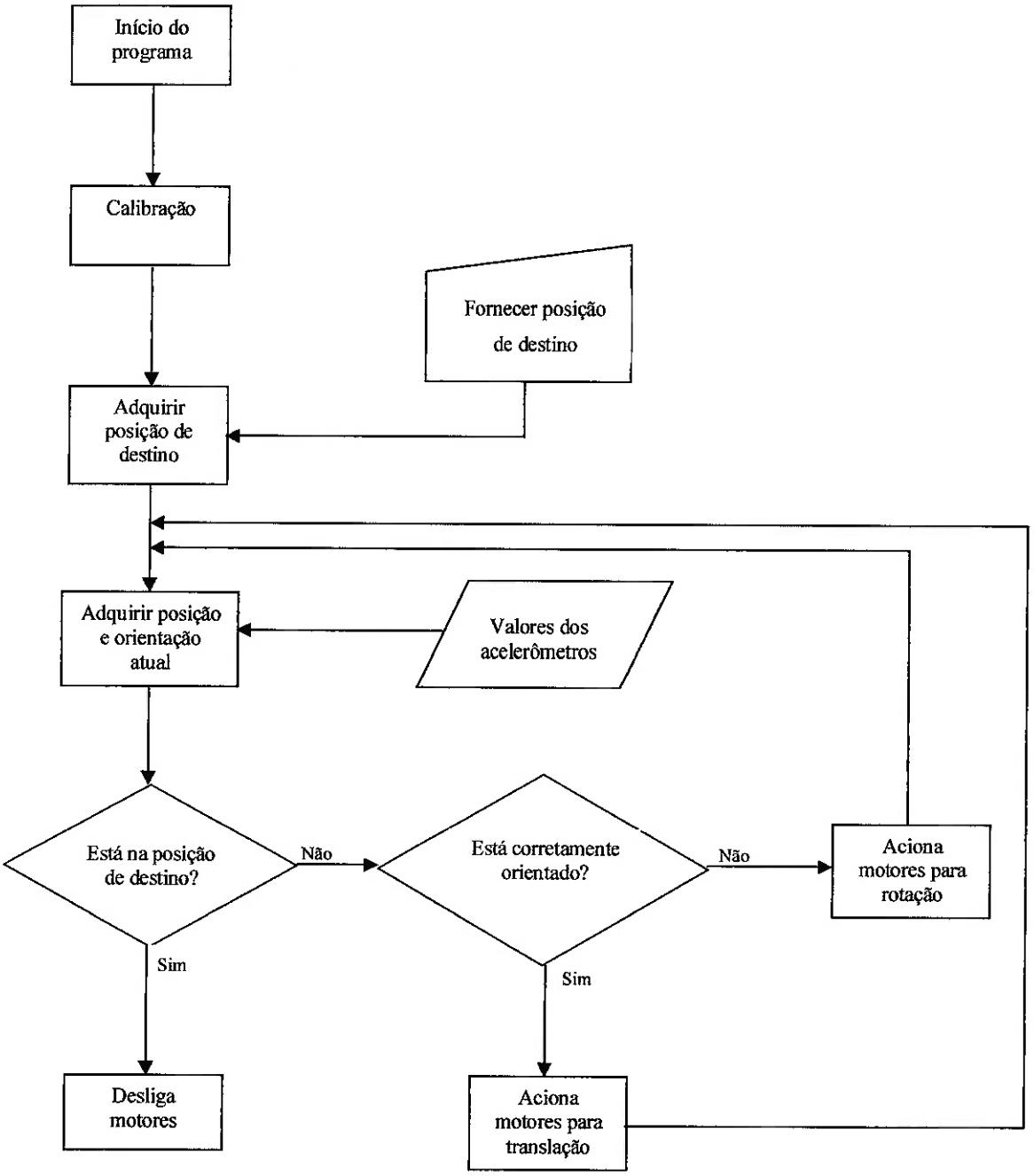

Fig. 11: Fluxograma do algoritmo de controle em terra

8. PROTÓTIPO

Utilizou-se para a construção do protótipo um modelo comercial para hobby. O envelope em formato charuto (zeppelim) mede aproximadamente 1,9m de comprimento e diâmetro máximo de aproximadamente 0,9m. Em um CAD comercial aproximou-se o formato do envelope por um elipsóide de revolução que indicou como volume interno aproximado de 0,93m³, portanto o empuxo máximo bruto seria de 930g.

Esta imprecisão nos valores é o resultado da dificuldade de se medir as dimensões reais pois o envelope se deforma, alterando as demais dimensões.

Fig. 12: Protótipo completo (vista lateral)

8.1. Válvula

A válvula adotada para vedar o orifício de entrada do gás é constituída de um bocal chato do qual parte para dentro do próprio balão uma lingüeta. Inserindo o bico injetor neste bocal, o hélio é guiado para dentro da lingüeta, abrindo-a e permitindo a passagem do gás. Ao interromper-se este fluxo, a pressão interna, mesmo que reduzida,

fecha a lingüeta, impedindo o refluxo. Para esvaziar o dirigível, é preciso inserir um tubo oco de pequeno diâmetro externo no bocal até que ultrapasse o fim da lingüeta, totalizando 300 mm.

8.2. Nacele

A nacele foi construída com madeira balsa e cola para madeira. Suas dimensões foram planejadas para que os componentes embarcados ficassem firmemente encaixados sem a necessidade de elementos de fixação, assim as baterias, os motores e a placa com o circuito foram acomodados por encaixe forçado, contudo sem forçar demais as paredes da nacele.

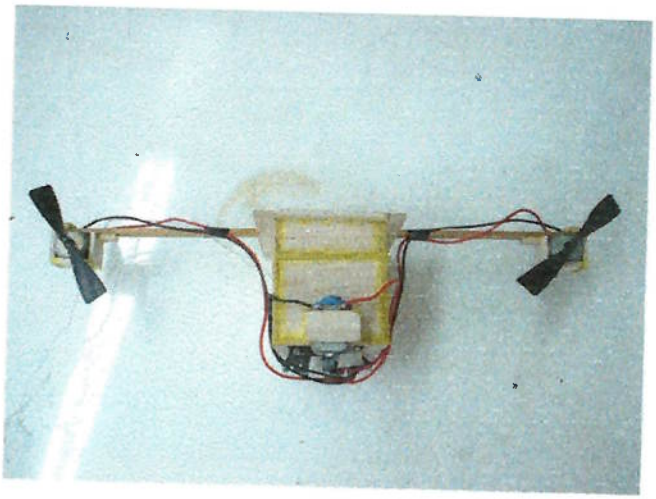

Fig. 13: Nacele (vista frontal)

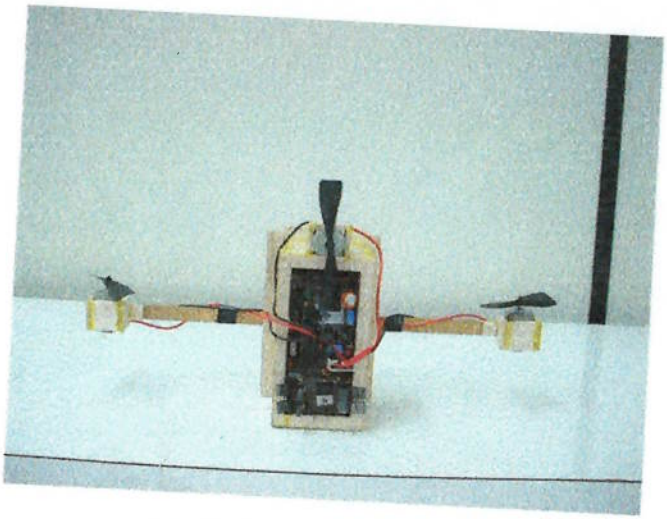

Fig. 14: Nacele (vista inferior)

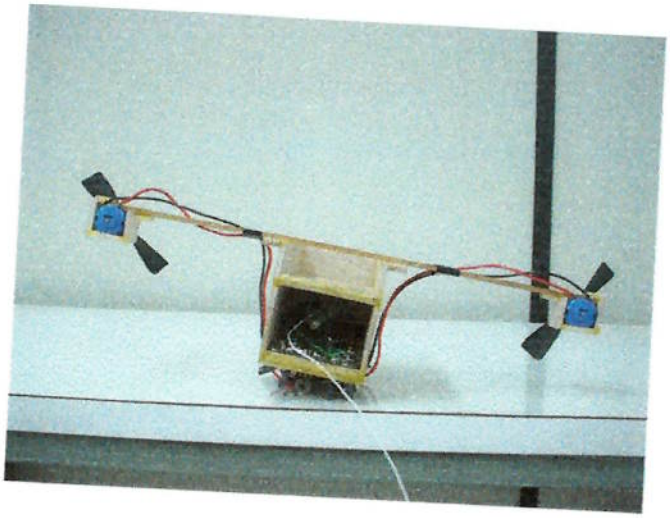

Fig. 15: Nacele (vista traseira)

8.3. Eletrônica embarcada

A eletrônica embarcada se resume a três itens:

- Acionamento dos motores;
- Microcontrolador;
- Tratamento dos sinais do rádio.

O acionamento dos motores utiliza o modelo de ponte H na qual quatro transistores são acionados (saturação) dois a dois e opostos. Quando os transistores Q1 e Q4 estão acionados, a corrente circula o motor em um sentido, fazendo-o girar, quando os transistores acionados são os Q2 e Q3, a corrente circula pelo motor no sentido contrário ao anterior, fazendo-o girar também, mas em sentido contrário. Para deixar o motor parado, basta não acionar transistor algum e, deste modo, cortar a corrente elétrica para o motor. Os resistores R2, R3, R4 e R5 são apenas limitadores de corrente evitando a queima do microcontrolador (veja Fig. 16: Ponte H).

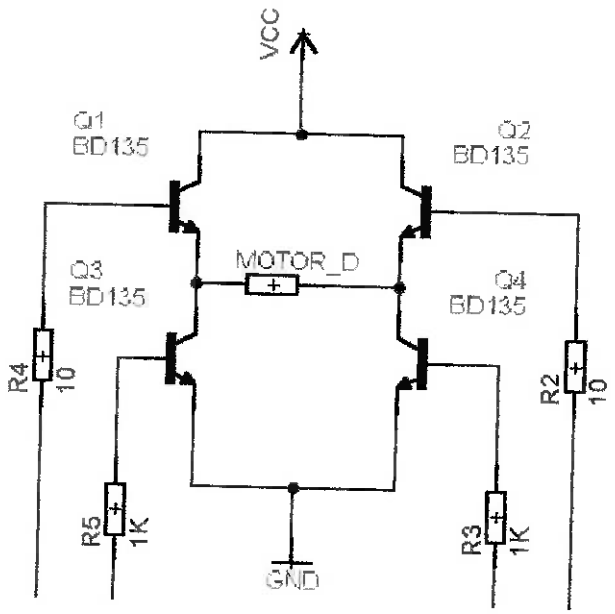

Fig. 16: Ponte H

Por serem motores de baixa tensão e com corrente média de 300mA, o transistor utilizado foi o BD135, do tipo npn. Estes transistores seguram tensões de até 50 V e

suportam correntes de até 1A, enquadrando-se ao projeto. O acionamento dos transistores é realizado por uma corrente de 10mA o que é fornecido diretamente pelos pinos do microcontrolador não necessitando de nenhum amplificador de corrente.

O rádio utilizado incialmente em um carro de controle remoto dispõe de quatro saídas: duas saídas do tipo coletor aberto (open collector) que acionavam bobinas para a direção das rodas do carro, e duas saídas semelhantes a *toten pole* ligados aos terminais do motor.

As saídas *toten pole* têm três níveis de tensão: quando não acionado, ambos os pinos estão em torno de 0,8 V não movendo o motor; acionando o comando para frente do controle remoto, um dos terminais se torna coletor com nível terra (ground) e o outro se eleva a uma tensão de 1,25 V fazendo com que o motor gire num sentido, para o outro sentido os valores se invertem.

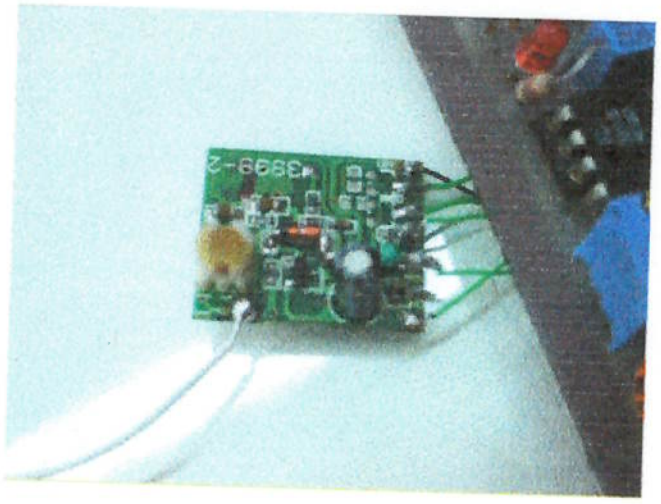

Fig. 17: Circuito eletrônico do receptor de rádio

O condicionamento do sinal de coletor aberto foi simplesmente ligado a entrada do microcontrolador com a adição de um resistor de elevação de tensão (pull-up). Já os sinais de acionamento dos motores do carro foram ligados a blocos comparadores que foram conectados ao microcontrolador e assim o sinal ficam em lógico 1 quando os pinos estão com tensão maior que 0,6 V e lógico 0 quando a tensão é menor que 0,6 V.

Estes sinais codificam comandos de acionamento dos motores que são interpretador pelo microcontrolador.

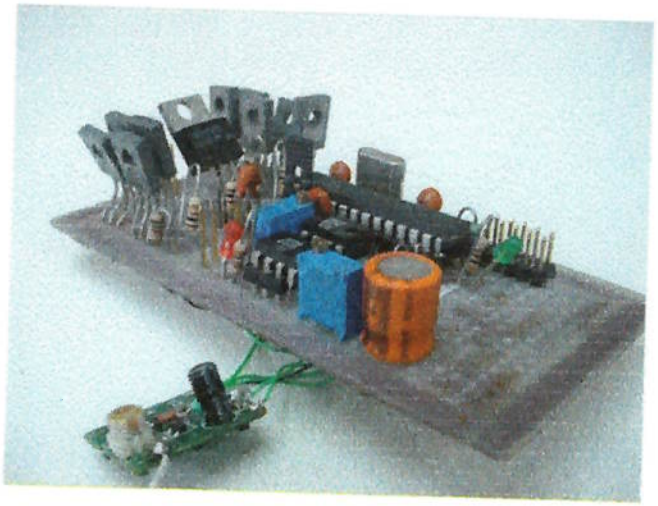

Fig. 18: Placa final da eletrônica embarcada

8.4. Sistema de propulsão

O sistema de propulsão é composto por dois motores que trabalham em conjunto para a movimentação no plano xy (rotação e translação à frente), um motor para movimentação vertical e hélices para o deslocamento do ar.

Cada motor opera com tensões entre 3 e 9V e não possui sistema de redução. Os hélices, feitos de plástico, são acoplados diretamente no eixo dos motores e foram cortados até ficarem com um diâmetro de 70 mm. Tal atitude fez-se necessária para diminuir o momento de inércia das mesmas e facilitar a atuação dos motores, que apresentam baixo torque.

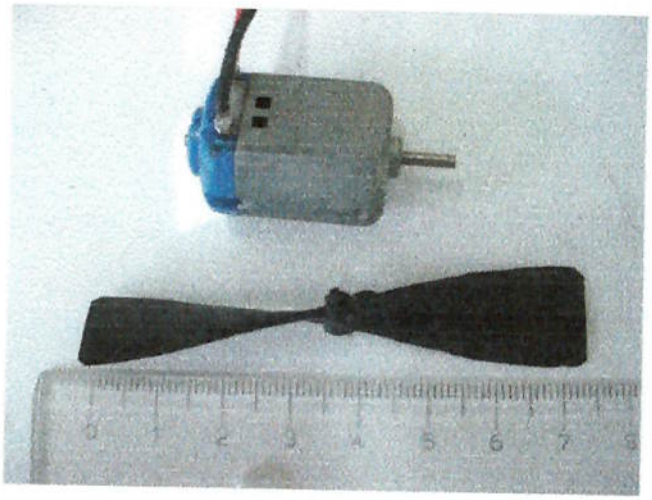

Fig. 19: Motor e hélice utilizados

8.5. Eletrônica em terra

A eletrônica em terra baseia-se no emissor de rádio original do carro. O controle possui quatro botões: frente-trás e direita-esquerda, e cada um destes dois pares apresenta um comando predominante. Outra restrição é o fato de os botões direita-esquerda só funcionarem corretamente quando um dos outros dois estiver acionado.

Assim as possíveis combinações de acionamento dos botões são:

Botão				
1	2	3	4	Movimento
0	0	0	0	desce (motores desligados)
X	0	0	0	sobe
0	X	0	0	sobe e vai para frente
X	0	X	0	sobe e gira no sentido anti-horário
X	0	0	X	sobe e gira no sentido horário
0	X	X	0	gira no sentido anti-horário
0	X	0	X	gira no sentido horário

Tabela 3: Tabela Verdade do acionamento do controle

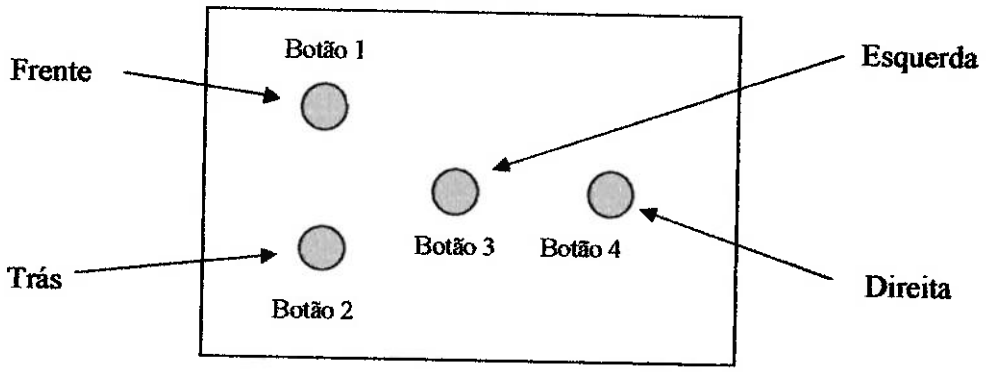

Fig. 20: Esquema de botões do emissor de rádio

Em que X simboliza o botão pressionado enquanto que o 0, não pressionado. As outras associações de botões não geram sinais válidos.

Estes botões funcionam como *open drain*:

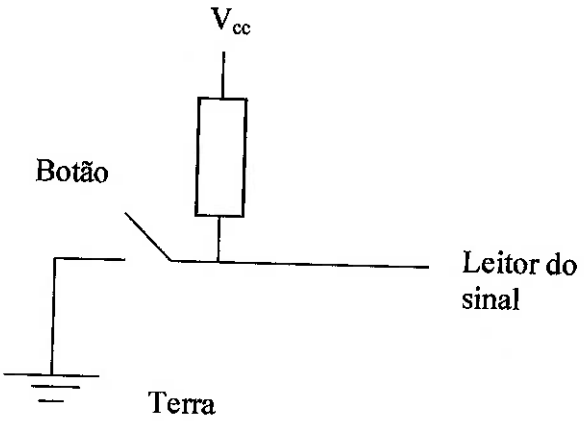

Fig. 21: Esquema de ligação open drain

Para adicionar um acionamento externo, foi feito um outro circuito *open drain* que foi ligado ao primeiro; deste modo o leitor de sinal interpreta como existindo um sinal toda vez que um dos dois circuitos for acionado. Como os acionamentos são independentes mas o sinal é único, aconselha-se a utilizar somente um circuito por vez.

8.6. Acionamento por computador

O acionamento por computador é realizado pela porta paralela. Para proteger a porta de eventuais picos de tensão e corrente, foi utilizado um *buffer* para isolá-la que , para simplificar a montagem, tem a saída do tipo *open collector* e pode ser ligada ao restante da eletrônica em terra (concordando com o circuito do rádio emissor).

9. MATERIAIS E MÉTODOS

Para o acionamento por computador, foram utilizados:

- Um cabo *flat* completo para porta paralela;
- Um circuito integrado (CI) 74LS07 para isolar o computador do resto do circuito;
- Um soquete para o CI;
- O emissor de rádio.

Este acionamento foi montado juntamente com o receptor de infravermelho que utiliza:

- Receptor de infravermelho: TSOP 7000 da Vishay
- Regulador: LM2940
- Conversor TTL / RS232: MAX232

Para este conjunto funcionar, é necessário fornecer dois níveis de tensão, um de 3V para o emissor de rádio e 9V para o resto do circuito. Esta tensão é reduzida para 5V (TTL) com o auxílio do regulador. Estes níveis de tensão são fornecidos através de uma fonte regulável.

A eletrônica embarcada contou com:

- 1 capacitor de 100 μF;
- 2 capacitores de 33 μF;
- 3 *jumpers*;
- 9 transistores BD135;
- 6 resistores de 1k Ω;

- 4 resistores de 10 Ω;
- 3 resistores de 4k7 Ω;
- 1 resistor de 5k Ω;
- 1 cristal oscilador de 20 MHz;
- 1 PIC 16F876P;
- 1 LM317;
- 2 CIs TCA 520B;
- o receptor de rádio.

A fonte de energia é composta por um pack de baterias de Ni-Cd.

O emissor de infravermelho é composto por.

- Emissor de infravermelho TSFH 5200

Todos os circuitos foram montados em placas de fenolite com furos padrão. Os esquemas dos circuitos podem ser vistos nos Apêndices I e II.

Ainda foram necessários ferro de solda, sugador para estanho, estanho e fios.

O software de controle foi desenvolvido na linguagem C, utilizando o compilador TurboC. O computador de desenvolvimento foi um Pentium MMX, 166 MHz com 64 megabytes de memória RAM, rodando o sistema operacional DOS.

O programa do microcontrolador foi desenvolvido com o auxílio do software MPLAB.

Primeiramente, todas as funções do software de controle foram testadas isoladamente utilizando o método de fornecer à função entradas para saídas conhecidas. Ao se cobrir todas as possibilidades de casos e exceções, tendo recebido as respostas esperadas, a função era considerada correta e pronta para a próxima etapa de validação.

Depois, cada função era associada às outras com as quais se relaciona no programa e era aplica o método de testes citado acima.

10. TESTES E RESULTADOS

Foram os problemas encontrados durante a fase de testes, desde falha de funcionamento de componentes a danos físicos na estrutura. Neste capítulo são descritos tais problemas e como foram tratados.

O modelo utilizado não correspondeu às expectativas pois sua capacidade de carga se viu alterada por algum dano ao envelope. Assim fez-se necessário uma série de adaptações entre elas alteração do rádio controle utilizado e simplificações na eletrônica embarcada.

10.1.Rádio

Devido a contratempos, o rádio-controle original foi substituído por um modelo simples e popular com reduzidas combinações de movimentos.

O primeiro possuía quatro canais independentes e movimentação completa no plano horizontal com os comandos frente, trás, direita (girar no sentido horário) e esquerda (girar no sentido anti-horário) e no vertical com os comandos sobe e desce, sem contar o neutro (motores não acionados), totalizando 15 diferentes modos de acionamento dos motores.

Já o controle utilizado, por possuir apenas dois canais não totalmente independentes, apresentou sérias restrições à quantidade de movimentos diferentes. Como as combinações possíveis não superavam 7, optou-se apenas pelos comandos frente-sobe, sobe, direita-sobe, esquerda-sobe, direita, esquerda e repouso.

O esquema de ligações entre o computador e o rádio pode ser visto no Apêndice I.

10.2. Transmissão de dados

O hardware de transmissão infra-vermelho demonstrou falhas severas na transmissão de dados no seu formato final. Sabe-se, entretanto, que a troca de informações via infra-vermelho é utilizada atualmente em diversos segmentos; portanto conclui-se que a transmissão é possível mas deve-se utilizar equipamentos qualificados e os componentes mais comumente encontrados não apresentam esta característica.

Mesmo com um sistema de informações redundante, não foi possível estabelecer um elo seguro de comunicação via infra-vermelho e, frente a isso, a conversação entre o microcontrolador e o computador foi realizada através de um par de cabos e padrão RS-485 de linhas balanceadas de transmissão.

Após esta alteração, a redundância de informações tornou-se desnecessária e não houve mais casos de perda de informação ou recebimento de informação incorreta.

10.3. Sensoriamento

Os acelerômetros apresentavam uma grande variação de valores mesmo quando estavam em repouso. Foi instalado um filtro analógico para eliminar este ruído bem como a aplicação de filtros atenuadores e compensação do efeito da gravidade por software. Ainda assim os valores de aceleração obtidos, depois da dupla integração, representavam um erro que inviabiliza sua aplicação.

Este erro de posicionamento variava de apenas alguns centímetros a mais de 100 metros ao se executar testes de 60s de duração. Devido a isso, os acelerômetros foram removidos do protótipo, o que colaborou com a redução de carga e simplificação da eletrônica embarcada.

10.4. Envelope

O envelope apresentou alguns defeitos físicos como furos e falhas na união entre os gomos. Estas falhas causavam uma perda de sustentação e as seguintes ações foram tomadas para minimizar esta perda:

- Cargas rápidas de hélio antes dos testes a fim de suprir a constante fuga do gás.
- Uso de balões auxiliares de menor volume como forma de sustentação. Estes balões colaboraram com um empuxo conjunto de aproximadamente 20g.
- Alteração do lastro. O lastro responsável pelo peso aparente maior do que zero era constantemente alterado até o ponto de ser completamente removido devido à perda de sustentação. Esse fato era indício de que uma nova carga de hélio era necessária.

10.5. Empuxo insuficiente

Para aumentar a capacidade de movimentação – que era muito limitada no modelo original – e comportar toda a nova eletrônica embarcada, foram utilizados motores maiores e mais potentes, uma nacele maior e mais resistente e a fonte de alimentação foi alterada. Estas alterações conferiram ao dirigível uma carga maior do que a sua capacidade de sustentação. Para resolver esta situação, utilizou-se balões menores como auxilio para a sustentação e a redução da carga.

As baterias contribuíam com a maior parcela da massa da aeronave, sendo formadas por 6 células de 1,6 V totalizando 7,2 V. Reduziu-se para 4 células (4,8 V) tornando o regulador de tensão para o microcontrolador desnecessário, permitindo a elevação do dirigível.

A queda da tensão fornecida ao sistema não representou perda de eficiência por parte dos motores e não prejudicou o desempenho global do sistema.

Para estabilizar a alimentação durante o acionamento dos motores foi necessária a inserção de um capacitor.

10.6. Software de teste do controle

Sem a possibilidade de utilização confiável dos acelerômetros para posicionamento, a execução do software completo que controla o dirigível tornou-se impraticável; portanto foram criados mais dois programas para os testes finais.

O primeiro deles envia comandos de acionamento dos motores de acordo com a posição de destino (objetivo que foi fornecido no início do programa) e a posição atual fornecida pelo operador, a qual inclui as coordenadas do dirigível e a orientação naquele momento.

O segundo apenas envia comandos de acionamento dos motores de acordo com teclas pressionadas pelo usuário para uma rotina de teleoperação de testes.

10.7. Firmware

Os teste de firmware foram realizados em uma placa protótipo, cujo esquema final encontra-se no Apêndice II.

Os testes revelaram que o rádio adaptado apresenta muita susceptibilidade a ruídos. Os efeitos destes ruídos foram minimizados utilizando-se um *debouncing*, no qual os sinais são lidos novamente alguns instantes depois da mudança de estado, certificando que a alteração era realmente um comando provindo do emissor de rádio e

não de alguma fonte estranha. Este procedimento diminui a chance de os motores serem acionados involuntariamente.

A versão final utilizada do firmware encontra-se no Anexo A e acabou sendo um programa simples devido à redução das funções do hardware pela eliminação dos sensores e da transmissão de dados.

11. CONCLUSÕES

O objetivo de se construir um dirigível comandado por computador foi alcançado, porém com restrições não previstas inicialmente provenientes de problemas encontrados durante os testes.

Tanto o desenvolvimento quanto a operação do protótipo foram realizados num computador sem grandes recursos, possibilitando sua implementação em máquinas de menor custo, aumentando as vantagens deste aparelho.

A partir dos resultados deste trabalho, fica comprovado que a navegação inercial é um sistema interessante em teoria, todavia desaconselhado para sistemas de dinâmica lenta, no qual recomenda-se um sistema de posicionamento absoluto.

Como comentado no corpo do texto, a navegação inercial deve trabalhar em conjunto com outra forma de posicionamento, mas deve dispor de equipamento mais sofisticado do que o utilizado neste trabalho e com filtros mais eficientes.

A transmissão de dados via infravermelho demonstrou-se inapropriada para esta aplicação devido à necessidade de constante contato visual e a real possibilidade de mutação de dados (inversão de bits), mesmo com a utilização de filtros e envio redundante de informações. Deve-se, portanto, utilizar a transmissão por rádio ou equivalente. É importante lembrar que o infravermelho ainda é bastante indicado para aplicações menos vitais.

Os cuidados com a carga máxima mostraram-se mais importantes do que era esperado pois a capacidade de elevação do atuador vertical é bastante limitada, portanto o dirigível deve estar quase em equilíbrio (peso aparente próximo de zero).

Os próximos passos para um trabalho deste segmento são a implementação de um balão *outdoor* com um sistema de posicionamento mais apurado e com um alcance mais abrangente, como o GPS. Com um sistema de posicionamento confiável, será possível aumentar a complexidade do sistema de controle deixando a aeronave completamente autônoma.

A inclusão de eletrônica embarcada com o propósito de monitoração mostrará a grande versatilidade deste aparelho se comparado a outros disponíveis.

ANEXO A – CÓDIGO FONTE DO SOFTWARE DO MICROCONTROLADOR

```
;*************************************************
;
;       Trabalho de Formatura
;       2003
;       Dirigivel hibrido autonomo
;
;       Alunos: Daniel Paulo Demitrio Martin
;               Ignácio Manavella
;
;*************************************************
;
;       Programa de acionamento de motores
;
;Descrição
;       Este programa faz uma leitura dos canais de rádio nos pinos RB7, RB6, RB5 e RB4 e
;       decodifica o comando para o acionamento correto dos motores
;


;***ARQUIVOS COM AS DEFINIÇÕES DA RAM
#INCLUDE <P16F876.INC>


;PALAVRA DE CONFIGURAÇÃO INTERNA DO MICROCONTROLADOR

        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_OFF &
_HS_OSC & _WRT_ENABLE_OFF & _LVP_OFF & _DEBUG_OFF & _CPD_OFF

;*********
;*MEMÓRIA*
;*********

;BANK 0
        CBLOCK 0X20


                T1
                T2
                AUX

        ENDC

;BANK 1
```

```
        CBLOCK 0X70

                W_TEMP          ;registradores para interrupção
                STATUS_TEMP

        ENDC

;*******
;*FLAGS*
;*******



;***  ENTRADAS  ***
;RADIO
#DEFINE RADIO1  PORTB,4
#DEFINE RADIO2  PORTB,5
#DEFINE RADIO3  PORTB,6
#DEFINE RADIO4  PORTB,7

;***  SAIDAS  ***
#DEFINE MOTOR1_D        PORTB,0
#DEFINE MOTOR1_E        PORTB,1
#DEFINE MOTOR2_D        PORTB,2
#DEFINE MOTOR2_E        PORTB,3

#DEFINE MOTOR_UP        PORTC,0


;***  MACROS  ***

BANK0   MACRO
        BCF     STATUS,RP1
        BCF     STATUS,RP0
        ENDM

BANK1   MACRO
        BSF     STATUS,RP0
        BCF     STATUS,RP1
        ENDM

BANK2   MACRO
        BSF     STATUS,RP1
        BCF     STATUS,RP0
        ENDM
```

```
BANK3  MACRO
       BSF     STATUS,RP1
       BSF     STATUS,RP0
       ENDM


;************
;* PROGRAMA *
;************

       ORG     0x00     ;ENDEREÇO INICIAL DE PROCESSAMENTO
       GOTO    INICIO


;***  INTERRUPÇÃO  ***

       ORG     0x04
       MOVWF W_TEMP          ;SALVA W EM W_TEMP
       SWAPF   STATUS,W
       MOVWF STATUS_TEMP     ;SALVA STATUS EM STATUS_TEMP

       GOTO    INT_EXIT


;***FINALIZA INTERRUPÇÃO

INT_EXIT
       SWAPF   STATUS_TEMP,W
       MOVWF STATUS          ;RECUPERA STATUS
       SWAPF   W_TEMP,F
       SWAPF   W_TEMP,W      ;RECUPERA W
       RETFIE                ;RETORNA DA INTERRUPÇÃO

       ;****FIM DA INTERRUPÇÃO****


;CONFIGURAÇÕES
INICIO
       ;*** CONFIGURANDO PORTAS ***

       BANK1

;PORT A
       MOVLW B'00001100'
       MOVWF TRISA

;PORT B
       MOVLW B'11110000'     ;ENTRADAS DOS ACELEROMETROS
```

```
        MOVWF TRISB

;PORT C
        MOVLW B'00000000'       ;
        MOVWF TRISC

        BANK0
        MOVLW B'00000000'
        MOVWF ADCON0
        BANK1
        MOVLW B'000000110'      ;TODOS OS PINOS DIGITAIS
        MOVWF ADCON1


        ;*** CONFIGURANDO TIMERS ***
;TIMER0          ;NÃO UTILIZADO
        BANK0
        CLRF    TMR0


;TIMER1          ;PARA CONTAR TEMPO DOS ACELEROMETROS
        MOVLW B'00000001'       ;INCREMENTO A CADA CICLO DE MÁQUINA
        MOVWF T1CON

;TIMER2          ;NÃO UTILIZADO


        ;*** CONFIGURANDO INTERRUPÇÕES ***
        BANK1
        MOVLW B'00000000'
        MOVWF OPTION_REG    ;


        MOVLW B'01100000'       ;INTERRUPÇÕES DESABILITADAS
        MOVWF INTCON            ;

        MOVLW B'00000000'
        MOVWF PIE1

        MOVLW B'00000000'
        MOVWF PIE2

        BANK0


        ;***  INICIALIZAÇÃO DAS VARIAVEIS  ***
```

```
        CLRF    PORTA
        CLRF    PORTB
        CLRF    PORTC
        CLRF    TMR0

        ;PROGRAMA DE TESTES

        BSF             MOTOR1_D
        BSF             MOTOR2_D
        CALL    DELAY
        NOP
        CALL    DELAY
        NOP
        CALL    DELAY

        BCF             MOTOR1_D
        BCF             MOTOR2_D
        BSF             MOTOR1_E
        BSF             MOTOR2_E
        CALL    DELAY
        NOP
        CALL    DELAY
        NOP
        CALL    DELAY
        CALL    DELAY
        CLRF    PORTB

        GOTO    MAIN    ;COMEÇA PROGRAMA

;***  ROTINAS  ***

DELAY           ;DELAY DE PROPÓSITO GERAL PARA EVITAR LEITURA
CONSTANTE DO RÁDIO
        MOVLW .253
        MOVWF T1

LOOP
        MOVLW .252
        MOVWF T2
        DECFSZ T2,F
        GOTO    $-1

        DECFSZ T1,F
        GOTO    LOOP
```

```
        RETURN

DELAY_DEBOUNCE      ;DELAY DE DEBOUNCE DO RÁDIO PARA EVITAR QUE
RUIDOS INDESEJÁVEIS ACIONEM OS MOTORES
        MOVLW .250
        MOVWF T1
        DECFSZ T1,F
        RETURN


COMANDOS_BALAO      ;FAZ A LEITURA DO RÁDIO E O ACIONAMENTO
CORRETO DA PONTE H

        MOVF   PORTB,W              ;ARMAZENA O COMANDO ENVIADO
        ANDLW  0XF0                 ;MASCÁRA OS BITS DE INTERESSE
        MOVWF AUX

;DESCOBRE QUAL FOI O COMANDO

;1
        MOVLW B'10100000'           ;TESTA QUAL O CÓDIGO ENVIADO
        XORWF  AUX
        BTFSC  STATUS,Z
        GOTO   SOBE_FRENTE          ;ACIONA COMANDO

;2
        MOVLW B'10010000'
        XORWF  AUX
        BTFSC  STATUS,Z
        GOTO   SOBE_H
;3
        MOVLW B'01010000'
        XORWF  AUX
        BTFSC  STATUS,Z
        GOTO   SOBE_AH
;4
        MOVLW B'00100000'
        XORWF  AUX
        BTFSC  STATUS,Z
        GOTO   SOBE
;5
        MOVLW B'10000000'
        XORWF  AUX
        BTFSC  STATUS,Z
        GOTO   H
```

```
;6
        MOVLW B'01000000'
        XORWF AUX
        BTFSC STATUS,Z
        GOTO  A_H


        CLRF  PORTB
        BCF           MOTOR_UP

        RETURN

;ACIONAMENTO

SOBE_FRENTE   ;ACIONA DE FORMA A FAZER A AERONAVE SUBIR E IR PARA
FRENTE

        CALL  DELAY_DEBOUNCE      ;AGUARDA UM TEMPO PARA FAZER
NOVA LEITURA

        MOVF  PORTB,W             ;FAZ NOVA LEITURA
        ANDLW B'11110000'
        XORLW B'10100000'

        BTFSS STATUS,Z            ;CONFERE SE É MESMO ESTE COMANDO
        RETURN                    ;NÃO É ESTE COMANDO, RETORNA

        ;SIM, PROCEDE COM O ACIONAMENTO DOS MOTORES

        BCF           MOTOR1_E
        BSF           MOTOR1_D
        BCF           MOTOR2_E
        BSF           MOTOR2_D

        BSF           MOTOR_UP

        RETURN



SOBE_H        ;ACIONA DE FORMA QUE A AERONAVE SUBA E ROTACIONE
HORÁRIO SOBRE SEU EIXO

        CALL  DELAY_DEBOUNCE

        MOVF  PORTB,W
        ANDLW B'11110000'
        XORLW B'10010000'
```

```
        BTFSS   STATUS,Z
        RETURN

        ;SIM, PROCEDE COM O ACIONAMENTO DOS MOTORES

        BSF             MOTOR_UP

        BCF             MOTOR1_D
        BSF             MOTOR1_E
        BCF             MOTOR2_E
        BSF             MOTOR2_D

        RETURN

SOBE_AH                 ;PARA QUE A AERONAVE SUBA E ROTACIONE NO
SENTIDO ANTI-HORÁRIO

        CALL    DELAY_DEBOUNCE      ;AGUARDA UM TEMPO PARA FAZER
NOVA LEITURA

        MOVF    PORTB,W             ;FAZ NOVA LEITURA
        ANDLW   B'11110000'
        XORLW   B'01010000'

        BTFSS   STATUS,Z        ;CONFERE SE É MESMO ESTE COMANDO
        RETURN                  ;NÃO É ESTE COMANDO, RETORNA

        ;SIM, PROCEDE COM O ACIONAMENTO DOS MOTORES
        BCF             MOTOR1_E
        BSF             MOTOR1_D
        BCF             MOTOR2_D
        BSF             MOTOR2_E

        BSF             MOTOR_UP

        RETURN

SOBE            ;PARA QUE A AERONAVE SUBA

        CALL    DELAY_DEBOUNCE      ;AGUARDA UM TEMPO PARA FAZER
NOVA LEITURA

        MOVF    PORTB,W             ;FAZ NOVA LEITURA
        ANDLW   B'11110000'
        XORLW   B'00100000'
```

```
        BTFSS   STATUS,Z        ;CONFERE SE É MESMO ESTE COMANDO
        RETURN                  ;NÃO É ESTE COMANDO, RETORNA

        ;SIM, PROCEDE COM O ACIONAMENTO DOS MOTORES

        CLRF    PORTB

        BSF             MOTOR_UP

        RETURN

H               ;PARA QUE A AERONAVE ROTACIONE NO SENTIDO HORÁRIO
APENAS

        CALL    DELAY_DEBOUNCE          ;AGUARDA UM TEMPO PARA FAZER
NOVA LEITURA

        MOVF    PORTB,W                 ;FAZ NOVA LEITURA
        ANDLW   B'11110000'
        XORLW   B'10000000'

        BTFSS   STATUS,Z        ;CONFERE SE É MESMO ESTE COMANDO
        RETURN                  ;NÃO É ESTE COMANDO, RETORNA

        ;SIM, PROCEDE COM O ACIONAMENTO DOS MOTORES
        BCF             MOTOR1_D
        BSF             MOTOR1_E
        BCF             MOTOR2_E
        BSF             MOTOR2_D

        BCF             MOTOR_UP

        RETURN

A_H             ;PARA A AERONAVE ROTACIONAR NO SENTIDO ANTI-HORÁRIO
APENAS

        CALL    DELAY_DEBOUNCE          ;AGUARDA UM TEMPO PARA FAZER
NOVA LEITURA

        MOVF    PORTB,W                 ;FAZ NOVA LEITURA
        ANDLW   B'11110000'
        XORLW   B'01000000'

        BTFSS   STATUS,Z        ;CONFERE SE É MESMO ESTE COMANDO
        RETURN                  ;NÃO É ESTE COMANDO, RETORNA
```

```
        ;SIM, PROCEDE COM O ACIONAMENTO DOS MOTORES
        BCF             MOTOR1_E
        BSF             MOTOR1_D
        BCF             MOTOR2_D
        BSF             MOTOR2_E

        BCF             MOTOR_UP

        RETURN



        ;***  PROGRAMA PRINCIPAL  ***

MAIN
        CALL    COMANDOS_BALAO      ;ACIONA MOTORES
        CALL    DELAY               ;ESPERA UM TEMPO ENTRE AS LEITURAS DO
SINAL DE RÁDIO
        GOTO    MAIN
        END
;FIM
```

ANEXO B – CÓDIGO FONTE DO SOFTWARE DE LEITURA DOS SENSORES

```
;*************************************************
;
;       Trabalho de Formatura
;       2003
;       Dirigivel hibrido autonomo
;
;       Alunos: Daniel Paulo Demitrio Martin
;               Ignácio Manavella
;
;*************************************************
;
;       Programa de leitura dos acelerometros
;
;Descrição
;       Este programa mede os tempos dos acelerometros e os envia em forma serial.
;       Para isto um timer é utilizado como a base de tempo e o microcontrolador
;       permanece verificando as mudanças de sinal para registrar o tempo de periodo
;       e do tempo em alta do sinal necessários para o cálculo da aceleração.


;***ARQUIVOS PARA PIC 16F877
#INCLUDE <P16F877.INC>

;CONFIGURAÇÃO DE FLAGS
        __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_OFF &
_XT_OSC & _WRT_ENABLE_OFF & _LVP_OFF & _DEBUG_OFF & _CPD_OFF

;*********
;*REGISTRADORES DA MEMÓRIA*
;*********


;BANK 0 (0X20 -> 0X7F)
        CBLOCK 0X20

                A1_H    ;registradores para guardar aceleração
                A1_L
                A2_H
                A2_L
                A3_H
                A3_L
                A4_H
                A4_L
```

```
                P1_H
                P1_L

                P2_H
                P2_L

        ENDC

;VISIVEL EM TODOS OS BANCOS

        CBLOCK 0X70

                W_TEMP          ;registradores para interrupção
                STATUS_TEMP

        ENDC


;***  ENTRADAS  ***
;acelerometros - > PORTB COM INT DE MUDANÇA DE ESTADO
#DEFINE ACEL1   PORTB,4
#DEFINE ACEL2   PORTB,5
#DEFINE ACEL3   PORTB,6
#DEFINE ACEL4   PORTB,7


;***  MACROS  ***

;ACESSO DE BANCOS
BANK0   MACRO
        BCF     STATUS,RP1
        BCF     STATUS,RP0
        ENDM

BANK1   MACRO
        BSF     STATUS,RP0
        BCF     STATUS,RP1
        ENDM

BANK2   MACRO
        BSF     STATUS,RP1
        BCF     STATUS,RP0
        ENDM

BANK3   MACRO
        BSF     STATUS,RP1
        BSF     STATUS,RP0
        ENDM
```

```
;************
;* PROGRAMA *
;************

ORG     0x00    ;ENDEREÇO INICIAL DE PROCESSAMENTO
NOP
GOTO    INICIO



;***********************
;***  INTERRUPÇÃO  ***
;***********************

ORG     0x04
MOVWF W_TEMP            ;SALVA W EM W_TEMP
SWAPF   STATUS,W
MOVWF STATUS_TEMP       ;SALVA STATUS EM STATUS_TEMP

GOTO    INT_EXIT


;*** FINALIZA INTERRUPÇÃO ***

INT_EXIT
        SWAPF   STATUS_TEMP,W
        MOVWF STATUS            ;RECUPERA STATUS
        SWAPF   W_TEMP,F
        SWAPF   W_TEMP,W        ;RECUPERA W
        RETFIE                  ;RETORNA DA INTERRUPÇÃO

        ;****FIM DA INTERRUPÇÃO****


        ;*******************
        ;***CONFIGURAÇÕES***
        ;*******************

INICIO
        ;*** CONFIGURANDO PORTAS ***

        BCF     STATUS,RP1
        BSF     STATUS,RP0      ;ALTERA PARA O BANCO 1

;PORT A
        MOVLW B'00001100'
```

```
        MOVWF TRISA

;PORT B
        MOVLW B'11110000'       ;ENTRADAS DOS ACELEROMETROS
        MOVWF TRISB

;PORT C
        MOVLW B'00000000'       ;
        MOVWF TRISC

;PORT D
        MOVLW B'00000000'       ;
        MOVWF TRISD

;PORT E
        MOVLW B'00000000'       ;
        MOVWF TRISE


        ;*** CONFIGURAÇÃO DOS PINOS PARA NIVEIS TTL ***

        BANK0
        MOVLW B'00000000'
        MOVWF ADCON0
        BANK1
        MOVLW B'000000110'      ;TODOS OS PINOS DIGITAIS
        MOVWF ADCON1


        ;*** CONFIGURANDO COMUNICAÇÃO SERIAL ***

        BANK1
        MOVLW B'00100100' ;brgh high
        MOVWF TXSTA
        MOVLW .103      ;2400bit/s -> 240 byte /s -> 1byte/0,00416s
        MOVWF SPBRG
        BANK0
        MOVLW B'10010000'
        MOVWF RCSTA



        ;*** CONFIGURANDO TIMERS ***

;TIMER0          ;NÃO UTILIZADO
        BANK0
        CLRF    TMR0
```

```
;TIMER1          ;PARA CONTAR TEMPO DOS ACELEROMETROS
        MOVLW B'00000001'
        MOVWF T1CON


;TIMER2          ;NÃO UTILIZADO


        ;*** CONFIGURANDO INTERRUPÇÕES ***
        BANK1

        MOVLW B'00000000'
        MOVWF OPTION_REG    ;


        MOVLW B'01100000'   ;INTERRUPÇÕES DESABILITADAS
        MOVWF INTCON        ;

        MOVLW B'00000000'
        MOVWF PIE1

        MOVLW B'00000000'
        MOVWF PIE2

        BANK0


        ;***  INICIALIZAÇÃO DAS VARIAVEIS  ***

        CLRF    PORTA
        CLRF    PORTB
        CLRF    PORTC
        CLRF    PORTD

MAIN


;ACELEROMETRO 1

        BTFSC   ACEL1   ;AGUARDA A BORDA DE DESCIDA DO SINAL DO
ACELEROMETRO
        GOTO    $-1

        BTFSS   ACEL1   ;AGUARDA A BORDA DE SUBIDA DO ACELEROMETRO
        GOTO    $-1
```

```
        CLRF    TMR1L   ;MARCA INICIO DO PERIODO E DO SINAL EM ALTA
        CLRF    TMR1H

        BTFSC   ACEL1   ;AGUARDA DESCIDA DO SINAL
        GOTO    $-1

        MOVF    TMR1L,W         ;REGISTRA TEMPO DO SINAL EM ALTA DO
ACELEROMETRO
        MOVWF A1_L
        MOVF    TMR1H,W
        MOVWF A1_H

        BTFSS   ACEL1   ;AGUARDA O FINAL DO PERIODO
        GOTO    $-1

        MOVF    TMR1L,W
        MOVWF P1_L
        MOVF    TMR1H,W
        MOVWF P1_H


;ACELEROMETRO 2

        BTFSC   ACEL2   ;AGUARDA A BORDA DE DESCIDA DO SINAL DO
ACELEROMETRO
        GOTO    $-1

        BTFSS   ACEL2   ;AGUARDA A BORDA DE SUBIDA DO ACELEROMETRO
        GOTO    $-1

        CLRF    TMR1L   ;MARCA INICIO DO PERIODO E DO SINAL EM ALTA
        CLRF    TMR1H

        BTFSC   ACEL2   ;AGUARDA DESCIDA DO SINAL
        GOTO    $-1

        MOVF    TMR1L,W         ;REGISTRA TEMPO DO SINAL EM ALTA DO
ACELEROMETRO
        MOVWF A2_L
        MOVF    TMR1H,W
        MOVWF A2_H

;ACELEROMETRO 3

        BTFSC   ACEL3   ;AGUARDA A BORDA DE DESCIDA DO SINAL DO
ACELEROMETRO
        GOTO    $-1

        BTFSS   ACEL3   ;AGUARDA A BORDA DE SUBIDA DO ACELEROMETRO
```

```
        GOTO    $-1

        CLRF    TMR1L  ;MARCA INICIO DO PERIODO E DO SINAL EM ALTA
        CLRF    TMR1H

        BTFSC   ACEL3  ;AGUARDA DESCIDA DO SINAL
        GOTO    $-1

        MOVF    TMR1L,W          ;REGISTRA TEMPO DO SINAL EM ALTA DO
ACELEROMETRO
        MOVWF A3_L
        MOVF    TMR1H,W
        MOVWF A3_H

        BTFSS   ACEL3  ;AGUARDA O FINAL DO PERIODO
        GOTO    $-1

        MOVF    TMR1L,W
        MOVWF P2_L
        MOVF    TMR1H,W
        MOVWF P2_H

;ACELEROMETRO 4

        BTFSC   ACEL4  ;AGUARDA A BORDA DE DESCIDA DO SINAL DO
ACELEROMETRO
        GOTO    $-1

        BTFSS   ACEL4  ;AGUARDA A BORDA DE SUBIDA DO ACELEROMETRO
        GOTO    $-1

        CLRF    TMR1L  ;MARCA INICIO DO PERIODO E DO SINAL EM ALTA
        CLRF    TMR1H

        BTFSC   ACEL4  ;AGUARDA DESCIDA DO SINAL
        GOTO    $-1

        MOVF    TMR1L,W          ;REGISTRA TEMPO DO SINAL EM ALTA DO
ACELEROMETRO
        MOVWF A4_L
        MOVF    TMR1H,W
        MOVWF A4_H


;ENVIA ACELERAÇÕES

        BTFSS   PIR1,TXIF        ;BUFFER DE ENVIO ESTÁ VAZIO?
        GOTO    $-1              ;NÃO,CONTINUA ESPERANDO
```

```
;OPCODE ACEL1
        MOVLW 0XFF              ;SIM, ENTÃO MANDA PRIMEIRO OPCODE
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;A1_L
        MOVF    A1_L,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;A1_H
        MOVF    A1_H,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;OPCODE          PERIODO
        MOVLW 0XF5
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;P1_L
        MOVF    P1_L,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;P1_H
        MOVF    P1_H,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;OPCODE ACEL2
        MOVLW 0X00
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1
```

```
;A2_L
        MOVF    A2_L,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;A2_H
        MOVF    A2_H,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;OPCODE ACEL 3
        MOVLW 0XFA
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1
;A3_L
        MOVF    A3_L,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

        BTFSS   PIR1,TXIF
        GOTO    $-1

;A3_H
        MOVF    A3_H,W
        MOVWF TXREG


        BTFSS   PIR1,TXIF
        GOTO    $-1

;OPCODE PERIODO 2

        MOVLW 0X05
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;P2_L
        MOVF    P2_L,W
        MOVWF TXREG
```

```
        BTFSS   PIR1,TXIF
        GOTO    $-1

;P2_H
        MOVF    P2_H,W
        MOVWF   TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1


;OPCODE ACEL 4
        MOVLW 0X0A
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1
;A4_L
        MOVF    A4_L,W
        MOVWF TXREG

        BTFSS   PIR1,TXIF
        GOTO    $-1

;A4_H
        MOVF    A4_H,W
        MOVWF TXREG

        GOTO    MAIN    ;RETORNA PARA MAIS UMA LEITURA
END     ;***FIM***
```

ANEXO C – CONSIDERAÇÕES SOBRE A CALIBRAÇÃO

Para evitar distorções de interpretação dos sinais dos sensores é necessário levar em conta um possível desalinhamento dos acelerômetros e aplicar uma transformação de coordenadas para obter-se os deslocamentos reais.

É assumido que os acelerômetros de cada circuito integrado têm sou eixo perpendicular e ambos são paralelos às superfícies superior e inferior do encapsulamento.

Para a calibração, o operador deve pressionar a tecla *ENTER* do teclado e mover, sem movimentos bruscos ,o dirigível 1m na direção X – aquela para a qual o balão aponta – e, quando tiver alcançado a distância, deve pressionar novamente a tecla *ENTER*. Deve-se tomar cuidado para não rotacionar o balão em hipótese alguma durante este procedimento pois a calibração considera somente a translação.

Os sinais são integrados duas vezes para obter-se o deslocamento e assim chega-se a um vetor coluna $\overline{X_{3x1}}$, cujos elementos representam os deslocamentos na direção X, Y e Z, respectivamente:

$$\overline{X} = \begin{bmatrix} X_x \\ X_y \\ X_z \end{bmatrix}$$

A seguir deve-se repetir o procedimento para X, mas movendo a aeronave 1m na direção Y – aquela para a esquerda do balão – e depois 1m da direção Z – aquela para cima do balão. Note que, deste modo, é formada uma base ortonormal positiva. São formados mais dois vetores colunas: $\overline{Y_{3x1}}$ e $\overline{Z_{3x1}}$.

$$\overline{Y} = \begin{bmatrix} Y_x \\ Y_y \\ Y_z \end{bmatrix} \text{ e } \overline{Z} = \begin{bmatrix} Z_x \\ Z_y \\ Z_z \end{bmatrix}$$

Pela teoria de transformação de coordenadas homogêneas e do explicado acima, deduz-se que:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \overline{M} * \begin{bmatrix} X_x \\ X_y \\ X_z \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \overline{M} * \begin{bmatrix} Y_x \\ Y_y \\ Y_z \end{bmatrix}, \quad \text{e} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \overline{M} * \begin{bmatrix} Z_x \\ Z_y \\ Z_z \end{bmatrix}$$

Sendo $\overline{M_{3x3}}$ a matriz de transformação homogênea.

$$\overline{M} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Estas relações podem ser rearranjadas em três sistemas lineares:

$$\begin{cases} 1 = a_{11}X_x + a_{12}X_y + a_{13}X_z \\ 0 = a_{21}X_x + a_{22}X_y + a_{23}X_z \\ 0 = a_{31}X_x + a_{32}X_y + a_{33}X_z \end{cases};$$

$$\begin{cases} 0 = a_{11}Y_x + a_{12}Y_y + a_{13}Y_z \\ 1 = a_{21}Y_x + a_{22}Y_y + a_{23}Y_z \\ 0 = a_{31}Y_x + a_{32}Y_y + a_{33}Y_z \end{cases} \text{ e}$$

$$\begin{cases} 0 = a_{11}Z_x + a_{12}Z_y + a_{13}Z_z \\ 0 = a_{21}Z_x + a_{22}Z_y + a_{23}Z_z \\ 1 = a_{31}Z_x + a_{32}Z_y + a_{33}Z_z \end{cases}.$$

Destes sistemas, compõe-se uma relação entre os vetores $\overline{X_{3x1}}$, $\overline{Y_{3x1}}$ e $\overline{Z_{3x1}}$, os elementos da matriz $\overline{M_{3x3}}$ e os deslocamentos absolutos nas direções X, Y e Z:

$$\begin{bmatrix} X_x & X_y & X_z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_x & X_y & X_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & X_x & X_y & X_z \\ Y_x & Y_y & Y_z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Y_x & Y_y & Y_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Y_x & Y_y & Y_z \\ Z_x & Z_y & Z_z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Z_x & Z_y & Z_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Z_x & Z_y & Z_z \end{bmatrix} * \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

cujas incógnitas são os elementos a_{ij} da matriz $\overline{M_{3x3}}$.

Esta relação, então, pode ser decomposta em novos três sistemas lineares:

$$\begin{bmatrix} X_x & X_y & X_z \\ Y_x & Y_s & Y_z \\ Z_x & Z_y & Z_z \end{bmatrix} * \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix};$$

$$\begin{bmatrix} X_x & X_y & X_z \\ Y_x & Y_s & Y_z \\ Z_x & Z_y & Z_z \end{bmatrix} * \begin{bmatrix} a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ e}$$

$$\begin{bmatrix} X_x & X_y & X_z \\ Y_x & Y_s & Y_z \\ Z_x & Z_y & Z_z \end{bmatrix} * \begin{bmatrix} a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

Resolvendo-se estes sistemas, encontram-se todos os elementos da matriz $\overline{M_{3x3}}$.

Durante a execução do programa, toda vez que é realizada a leitura dos acelerômetros, deve-se aplicar esta transformação do seguinte modo:

$$[valorlido]_{3x1} * \overline{M_{3x3}} = [valorcalibrado]_{3x1}$$

E este *valorcalibrado* é o que será utilizado nos demais cálculos do algoritmo.

Para a resolução dos sistemas lineares, para facilitar os cálculos computacionais e reduzir a necessidade de tratamento de exceções, utiliza-se o método de Cramer (dos determinantes).

ANEXO D – CÓDIGO FONTE DO SOFTWARE DE CONTROLE

```
/*DHA 001          Programa que gerencia todo o sistema terrestre
                   (computador em terra), chamando as funcoes
                   necessarias e monitorando se o balao executa a
                   operacao desejada.*/

/*-----------------------bibliotecas ----------------------------*/


#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <dos.h>


/*------------------------ definicoes ----------------------------*/


#define PORT1 0x3f8 /*endereco da porta serial COM1*/
#define d 0.05 /*distancia entre os acelerometros frontal e traseiro*/
#define PARALELA 0x0378 /*endereco da porta paralela*/
#define pi 3.14159265 /*constante pi*/
#define DT 0.125 /*intervalo de tempo para entre duas leituras
                          consecutivas*/
#define gravidade 9.8 /*aceleracao da gravidade*/
#define medidas 10 /*numero de medidas para filtragem*/

#define toleranciaangular 0.349 /*20 graus*/
/*faixa angular na qual e aceitavel o deslocamento em linha reta do balao.
```

```
Fora desta faixa o dirigivel deve corrigir seu curso*/

/*---------------------- chamadas das funcoes----------------------*/

double Calculaacel(double T1, double T2);

double Calculadistancia (double posicao[3], double posicaof[3]);

double Determinante(double A[3], double B[3], double C[3]);

int Calibracao(double X[3], double Y[3], double Z[3], double M[3][3]);

double Errodirecao (double posicao[3], double posicaof[3], double theta);

double Integra(double valor, double inicial, float deltat);

void Leitura(int acel[4][2]);

void Iniciaserial(void);

int Leia(void);

void Valormedio (double media[4]);

void Media(double valor[4], double aceleracao[4][10]);

void Procedecalib(double X[3], double Y[3], double Z[3]);

double Calculadistanciaplano (double posicao[3], double posicaof[3]);
```

```
/*---------------------------- main ----------------------------*/


int main(){
    double posicao[3]; /*guarda as coordenadas cartesianas do dirigível*/
    double comparacao[3]; /*guarda a posicao de destino a ser alcancada
                            nomomento*/
    double destino[3][10]; /*guarda as posicoes de destino (até 10 trios
                            coordenados)*/
    double calibX[3], calibY[3], calibZ[3]; /*gurdam os vetores para gerar a
                                            matriz de calibracao (matriz de
                                            transformacao de coordenadas
                                            homogeneas)*/
    double Matriz[3][3]; /*matriz de calibracao (matriz de transformacao de
                        coordenadas homogeneas)*/

    double distanciaplano; /*guarda a distancia do balao ao ponto desejado
                        no plano emque esta*/

    double alfa; /*direcao do dirigivel*/
    double alfap; /*velocidade angular do balao*/
    double alfapp; /*aceleracao angular do balao*/
    double deltaalfa; /*erro de alinhamento do balao*/

    int T[4][2]; /*recebe os dados dos dutty cicles*/

    double aceleracoesA[4]; /*guarda as aceleracoes em X, Y1, Y2 e Z dos
                            acelerometros*/
```

```
double media[4];
double ultimas10acel[4][10];

double aceleracoesG[3]; /*guarda as aceleracoes em X, Y e Z globais*/
double velocidadesG[3]; /*guarda as velocidades em X, Y e Z globais*/
double acelY;                /*guarda a aceleracao no eixo Y dos acelerometros*/
double acelcalib[3];   /*guarda as aceleracoes depois de aplicar a
                             calibracao*/

int matrizOK; /*vale 1 se foi possivel encontrar Matriz e 0 se nao foi*/
int passos; /*quantos pontos de destino serao usados*/
int i, j; /*contadores*/

outportb(PARALELA , 0x1f); /*desligandomotores*/

/*cabecalho*/
printf("\n\n\n\n");
printf("*********************************************************\n\n");
printf("\tPMR 2550 - Projeto de Conclusao do Curso II\n\n");
printf("\tDaniel Paulo Demitrio Martin\n");
printf("\tIgnacio Manavella\n\n");
printf("\tDirigivel Hibrido Autonomo\n");
printf("aplicativo DHA 001.exe \n\n");
printf("*********************************************************\n\n\n");

Iniciaserial(); /*iniciacao da porta serial*/
Valormedio(media);

for(i=0; i<4; i++){
    for(j=0; j<10; j++){
        ultimas10acel[i][j]=0;
```

```
            }
        aceleracoesA[i]=0;
        aceleracoesG[i]=0;
        velocidadesG[i]=0;
        posicao[i]=0;
        }
    alfa = alfap = alfapp = deltaalfa = 0;
    /*zerando valores iniciais*/

    /*instrucoes*/
    printf("Para, enquanto o dirigivel estiver em movimento, verificar sua posicao, aperte a tecla ENTER.\n");
    printf("Nota: este procedimento pode gerar atrasos\n\n");

    printf("Entre com o numero de pontos a serem alcancados (ate 10).\n");
    scanf("%d", &passos);

    /*leitura das posicoes de destino*/
    for(i=0; i<passos; i++){
        printf("\nEntre com as coordenadas do ponto %d.\n", i+1);
        scanf("%lf", &destino[0][i]);
        scanf("%lf", &destino[1][i]);
        scanf("%lf", &destino[2][i]);
        }

    Procedecalib(calibX, calibY, calibZ); /*capta dados para calibracao*/
    matrizOK = Calibracao(calibX, calibY, calibZ, Matriz); /*calibra*/

    if(matrizOK){
        /*Se foi possivel encontrar a matriz de calibracao...*/
        for (i=0; i<passos; i++){
            /*passando para o proximo passo...*/
```

```
comparacao[0] = destino[0][i];
comparacao[1] = destino[1][i];
comparacao[2] = destino[2][i];

for( ; Calculadistancia(posicao, comparacao)>1; ){
    /*se o dirigivel ainda nao chegou ao ponto desejado (dentro da
    tolerancia de posicao)...*/

    Leitura(T); /*le acelerometros*/

    aceleracoesA[0]=Calculaacel(T[0][0], T[0][1]);
    aceleracoesA[1]=Calculaacel(T[1][0], T[1][1]);
    aceleracoesA[2]=Calculaacel(T[2][0], T[2][1]);
    aceleracoesA[3]=Calculaacel(T[3][0], T[3][1]);
    /*calcula as aceleracoes X, Y1, Y2 e Z*/

    Media(aceleracoesA, ultimas10acel);
    aceleracoesA[0]=aceleracoesA[0]-media[0];
    aceleracoesA[1]=aceleracoesA[1]-media[1];
    aceleracoesA[2]=aceleracoesA[2]-media[2];
    aceleracoesA[3]=aceleracoesA[3]-media[3];
    /*aplicacao de correcoes*/

    acelY=(aceleracoesA[1]+aceleracoesA[2])/2;
    /*calcula a aceleracao resultante em Y*/

    /*Calibracao das aceleracoes*/
    acelcalib[0]=Matriz[0][0]*aceleracoesA[0] +
        Matriz[0][1]*acelY + Matriz[0][2]*aceleracoesA[2];

   acelcalib[1]=Matriz[1][0]*aceleracoesA[0] +
```

```
        Matriz[1][1]*acelY + Matriz[1][2]*aceleracoesA[2];

    acelcalib[2]=Matriz[2][0]*aceleracoesA[0] +
        Matriz[2][1]*acelY + Matriz[2][2]*aceleracoesA[2];

    /*aplicacao do efeito da rotacao*/
    aceleracoesG[0] = acelcalib[0]*cos(alfa) - acelcalib[1]*sin(alfa);
    aceleracoesG[1] = acelcalib[0]*sin(alfa) + acelcalib[1]*cos(alfa);
    aceleracoesG[2] = acelcalib[2];

    /*calculo das velocidades*/
    velocidadesG[0]=Integra(aceleracoesG[0], velocidadesG[0], DT);
    velocidadesG[1]=Integra(aceleracoesG[1], velocidadesG[1], DT);
    velocidadesG[2]=Integra(aceleracoesG[2], velocidadesG[2], DT);

    /*calculo da posicao*/
    posicao[0]=Integra(velocidadesG[0], posicao[0], DT);
    posicao[1]=Integra(velocidadesG[1], posicao[1], DT);
    posicao[2]=Integra(velocidadesG[2], posicao[2], DT);

    /*calculo da aceleracao angular*/
    alfapp=(aceleracoesA[1]-aceleracoesA[2])/d;
    /*calculo da velocidade angular*/
    alfap=Integra(alfapp, alfap, DT);
    /*calculo da posicao angular (direcao)*/
    alfa=Integra(alfap, alfa, DT);

    if (alfa > pi) alfa = alfa -2*pi;
    if (alfa < -pi) alfa = alfa +2*pi;
    /*alfa deve estar entre -pi e pi*/
```

```
deltaalfa = Errodirecao(posicao, comparacao, alfa);
distanciaplano = Calculadistanciaplano(posicao, comparacao);

if((deltaalfa >= -toleranciaangular) && (deltaalfa <=
    toleranciaangular)){
/*se o dirigivel estivar dentro da tolerancia angular...*/

    if (distanciaplano > 1){
        /*ainda longe -> vai para frente e sobe*/
        outportb(PARALELA , 0x0d);
        }

    if ((posicao[2] > comparacao[2]) && (distanciaplano <= 1)){
        /* z > cota e perto -> desce*/
        outportb(PARALELA , 0x1f);
        }

    if ((posicao[2] <= comparacao[2]) && (distanciaplano <= 1)){
        /* z <= cota e perto -> sobe*/
        outportb(PARALELA , 0x1e);
        }
    }/*if((deltaalfa >= -0.349) && (deltaalfa <= 0.349))*/

else{
    /*se o dirigivel nao estiver alinhado, devera rotacionar*/
    if ((posicao[2] > comparacao[2]) && (deltaalfa <
            -toleranciaangular)){
        /* z > cota e muito para a esquerda -> H*/
        outportb(PARALELA , 0x09);
        }
```

```
        if ((posicao[2] <= comparacao[2]) && (deltaalfa <
                -toleranciaangular)){
            /* z <= cota e muito para a esquerda -> sobe e H*/
            outportb(PARALELA , 0x1a);
            }

        if ((posicao[2] > comparacao[2]) && (deltaalfa >
                toleranciaangular)){
            /* z > cota e muito para a direita -> AH*/
            outportb(PARALELA , 0x05);
            }

        if ((posicao[2] <= comparacao[2]) && (deltaalfa >
                toleranciaangular)){
            /* z <= cota e muito para a direita -> sobe e AH*/
            outportb(PARALELA , 0x16);
            }
        }/*else*/

    if (kbhit()){
        /*se uma tecla for apertada...*/
        if(getch() == 13){
            /*... e se essa tecla for o ENTER...*/
            printf("X = %.3lf m\n", posicao[0]);
            printf("Y = %.3lf m\n", posicao[1]);
            printf("Z = %.3lf m\n", posicao[2]);
            printf("Angulo = %.3lf graus\n\n", alfa*180/pi);
            }/*if(getch() == 13)*/
        }/*if (kbhit())*/

}/*for( ; Calculadistancia(posicao, comparacao)>1; )*/
```

```
            }/*for (i=0; i<passos; i++){ */

            outportb(PARALELA , 0x1f); /*desligandomotores*/
            printf("\nDesligando motores.\n");
            printf("Multiplos objetivos atingidos.\n");
            printf("Finalizando programa.\n");
            return 0;
            }
        else{
            printf("Nao foi possivel executar acalibracao.\n");
            printf("Multiplos objetivos abortados.\n");
            printf("Finalizando programa.\n");
            return 0;
            }
        return 0;
        }


/*------------------------------ funcoes ---------------------------*/


double Calculaacel(double T1, double T2){
    /* Dados T1 e T2 obtidos pelo acelerometro (duty cycle), calcula a
    aceleracao referente*/
    double g;
    g=((T1/T2-0.5)/0.125)*9.8;
    return (g);
    }

double Calculadistancia (double posicao[3], double posicaof[3]){
    /* Calcula a distancia entre a posicao atual do balao e a posicao de
```

```
    destino
      posicao indica a posicao atual do balao, enquanto que posicaof
    indica a posicao de destino.*/

    double deltax, deltay, deltaz;
    /*diferenca entre o atual e o desejado para cada coordenda*/
    double distancia;
    /*distancia entre o atual e o desejado*/

    deltax = posicaof[0]-posicao[0];
    deltay = posicaof[1]-posicao[1];
    deltaz = posicaof[2]-posicao[2];

    distancia = pow((pow(deltax, 2)+pow(deltay, 2)+pow(deltaz, 2)), 0.5);
    return (distancia);
    }

double Determinante(double A[3], double B[3], double C[3]){
    /*Calcula o determinante da matriz 3x3 formada pelas coluas A, B e C*/
    double det;
    det = (A[0]*B[1]*C[2]) + (A[1]*B[2]*C[0]) + (A[2]*B[0]*C[1]);
    det = det - (A[2]*B[1]*C[0]) - (A[1]*B[0]*C[2]) - (A[0]*B[2]*C[1]);
    return (det);
    }


int Calibracao(double X[3], double Y[3], double Z[3], double M[3][3]){
    /* E feita a leitura dos acelerometros e calculado os deslocamentos
    referentes a 1m em x, 1m em y e 1m em z. A partir destes valores e
    montada uma matriz que relaciona os valores dos acelerometros com os
    reais deslocamentos (matriz de transformacao (rotacao) de coordenadas).
```

```
As incognitas sao os elementos desta matriz (M)
  Neste caso e usada a regra de Cramer.
  X indica o deslocamento mostrado pelos acelerometros referente a
um deslocamento de 1m em x. O mesmo para Y e Z.
  M e a matriz de relacao entre valores mostrado e real*/

double colunaX[3], colunaY[3], colunaZ[3];
/*colunas da matriz que multiplica as incognitas*/

double resultadoX[3], resultadoY[3], resultadoZ[3];
/* guardam o resultado da multiplicacao da matriz formada por colunaX,
Y e Z pelas incognitas do sistema. Sao referentes as movimentacoes de
1m em x, y e z.*/

double det, detX, detY, detZ;
/*determinantes para a solucao do sistema por Cramer*/

resultadoX[0] = resultadoY[1] = resultadoZ[2] = 1;
resultadoX[1] = resultadoX[2] = resultadoY[0] =
     resultadoY[2] = resultadoZ[0] = resultadoZ[1]=0;
/*Montagem do resultado*/

colunaX[0] = X[0];
colunaX[1] = Y[0];
colunaX[2] = Z[0];
/*Montagem da colunaX*/

colunaY[0] = X[1];
colunaY[1] = Y[1];
colunaY[2] = Z[1];
/*Montagem da colunaY*/
```

```
colunaZ[0] = X[2];
colunaZ[1] = Y[2];
colunaZ[2] = Z[2];
/*Montagem da colunaZ*/

det = Determinante(colunaX, colunaY, colunaZ);

if (det==0){
    printf("Erro! Solucao inpossivel.\n");
    return (0);
    }

detX = Determinante(resultadoX, colunaY, colunaZ);
detY = Determinante(colunaX, resultadoX, colunaZ);
detZ = Determinante(colunaX, colunaY, resultadoX);

M[0][0] = detX/det;
M[0][1] = detY/det;
M[0][2] = detZ/det;
/*resolucao da primeira linha de M*/

detX = Determinante(resultadoY, colunaY, colunaZ);
detY = Determinante(colunaX, resultadoY, colunaZ);
detZ = Determinante(colunaX, colunaY, resultadoY);

M[1][0] = detX/det;
M[1][1] = detY/det;
M[1][2] = detZ/det;
/*resolucao da segunda linha de M*/
```

```
    detX = Determinante(resultadoZ, colunaY, colunaZ);
    detY = Determinante(colunaX, resultadoZ, colunaZ);
    detZ = Determinante(colunaX, colunaY, resultadoZ);

    M[2][0] = detX/det;
    M[2][1] = detY/det;
    M[2][2] = detZ/det;
    /*resolucao da terceira linha de M*/

    return (1);
    }

double Errodirecao (double posicao[3], double posicaof[3], double theta){
    /* Calcula a diferenca entre a direcao atual do balao e aquela que
    deveria ter para atingir a posicao de destino seguindo uma linha reta
     (x, y, z): posicao atual
     (xf, yf, zf): posicao de destino
     theta: direcao atual do balao*/

    double phi, delta[2];

    delta[0]=posicaof[0] - posicao[0];
    delta[1]=posicaof[1] - posicao[1];

    if (delta[0]==0 && delta[1]==0){
        phi=0;
        return (phi);
        }
    else{
        phi=atan2(delta[1], delta[0]);
        phi=phi-theta;
```

```
        }
    if (phi > 3.14159265359) phi = phi -6.28318530718;
    if (phi < -3.14159265359) phi = phi +6.28318530718;
    return (phi);
    }

double Integra(double valor, double inicial, float deltat){
    /* Recebe o valor da aceleracao/velocidade e integra no tempo,
    fornecendo a velocidade/posicao. Calcula somente para uma direcao
    (x, y ou z).
     valor: magnitude da grandeza a ser integrada
     inicial: antiga magnitude da integral da grandeza analizada
     deltat: periodo duranteo qual "valor" e considerada constante.*/

    double resultado;
    resultado = inicial + valor * deltat;
    /*E utilizado um integrador de ordem zero, isto e, a integral e
    calculada como sendo a soma de todos os retangulos (do grafico)
    valor X tempo.*/

    return (resultado);
    }

void Leitura(int acel[4][2]){
    /*Le a porta serial, identifica qual informacao esta recebendo
    (espera ate receber o identificador correto), aplica um filtro
    nesta leitura e disponibiliza-a para o resto do programa*/

    int byteM;/*guarda o byte mais significativo*/
    int bytem;/*guarda o byte menos significativo*/
    int flag;/*indica se a leitura continua ou nao*/
```

```
int codigo; /* verificador da variavel a ser lida */

flag=1; /*continue lendo*/
/*-------------------------- X --------------------------*/
byteM = bytem = 0;
/*prepara buffer para nova leitura*/

while(flag){
    codigo = Leia();
    if (codigo == 0xff) flag=0;
    }
/*Espera a palavra de identificacao do acelerometro X*/

bytem=Leia(); /*byte menos significativo*/
byteM=Leia(); /*byte mais significativo*/

acel[0][0] = bytem + byteM*256; /*monta numero*/

/*-------------------------- Y1 --------------------------*/
byteM = bytem = 0;
/*prepara buffer para nova leitura*/

flag=1; /*setup da flag*/

while(flag){
    codigo = Leia();
    if (codigo == 0x00) flag=0;
    }
/*Espera a palavra de identificacao do acelerometro Y1*/

bytem=Leia(); /*byte menos significativo*/
```

```
byteM=Leia(); /*byte mais significativo*/

acel[1][0] = bytem + byteM*256; /*monta numero*/

/*-------------------------- Txy1 --------------------------*/
byteM = bytem = 0;
/*prepara buffer para nova leitura*/

flag=1; /*setup da flag*/

while(flag){
    codigo = Leia();
    if (codigo == 0xf5) flag=0;
    }
/*Espera a palavra de identificacao do periodo dos acelerometros
 X e Y1*/

bytem=Leia(); /*byte menos significativo*/
byteM=Leia(); /*byte mais significativo*/

acel[0][1]= acel[1][1] = bytem + byteM*256; /*monta numero*/

/*-------------------------- Y2 --------------------------*/
byteM = bytem = 0;
/*prepara buffer para nova leitura*/

flag=1; /*setup da flag*/

while(flag){
    codigo = Leia();
    if (codigo == 0xfa) flag=0;
```

```
        }
    /*Espera a palavra de identificacao do acelerometro Y2*/

    bytem=Leia(); /*byte menos significativo*/
    byteM=Leia(); /*byte mais significativo*/

    acel[2][0] = bytem + byteM*256; /*monta numero*/

    /*------------------------- Z -------------------------*/
    byteM = bytem = 0;
    /*prepara buffer para nova leitura*/

    flag=1; /*setup da flag*/

    while(flag){
        codigo = Leia();
        if (codigo == 0x0a) flag=0;
        }
    /*Espera a palavra de identificacao do acelerometro Z*/

    bytem=Leia(); /*byte menos significativo*/
    byteM=Leia(); /*byte mais significativo*/

    acel[3][0] = bytem + byteM*256; /*monta numero*/

/*------------------------- Ty2z -------------------------*/
    byteM = bytem = 0;
    /*prepara buffer para nova leitura*/

    flag=1; /*setup da flag*/
```

```
    while(flag){
        codigo = Leia();
        if (codigo == 0x05) flag=0;
        }
    /*Espera a palavra de identificacao do periodo dos acelerometros
     Y2 e Z*/

    bytem=Leia(); /*byte menos significativo*/
    byteM=Leia(); /*byte mais significativo*/

    acel[2][1]= acel[3][1]= bytem + byteM*256; /*monta numero*/

    return;
    /*fim da funcao*/
    }

void Iniciaserial(void){
    printf("Iniciando porta serial COM1.....\n\n");
    printf("Desligando interrupcoes.....");
    outportb(PORT1 + 1 , 0);   /* Desliga as Interrupcoes - Port1 */
    printf(" pronto.\n");

    /*      PORT 1 - Especificacoes da Comunicacao       */

    outportb(PORT1 + 3 , 0x80);  /* SET DLAB ON */
    printf("Ajustando baud rate.....");
    outportb(PORT1 + 0 , 0x30);
    /* Set Baud rate - Divisor Latch Low Byte */
                /* Default 0x03 =  38,400 BPS */
                /*       0x01 = 115,200 BPS */
                /*       0x02 =  57,600 BPS */
```

```
                    /*      0x06 =  19,200 BPS */
                    /*      0x0C =   9,600 BPS */
                    /*      0x18 =   4,800 BPS */
                    /*      0x30 =   2,400 BPS */
                    /*      0x60 =   1,200 BPS */
        outportb(PORT1 + 1 , 0x00);
        /* Set Baud rate - Divisor Latch High Byte */

        printf(" pronto.\n");
        printf("Outras configuracoes.....");
        outportb(PORT1 + 3 , 0x03);  /* 8 Bits, No Parity, 1 Stop Bit */
        outportb(PORT1 + 2 , 0xC7);  /* FIFO Control Register */
        outportb(PORT1 + 4 , 0x0B);  /* Turn on DTR, RTS, and OUT2 */
        printf("pronto.\n\n");
        }

int Leia(void){
    /*Adquire dado da porta serial*/
    int status;
    do{
        status=inportb(PORT1+3) & 0x01;
        }while (status != 0x01);
        /*espere ate ter uma palavra pronta*/
    return ((int)inportb(PORT1));
    /*leia palavra*/
    }

void Valormedio (double media[4]){
    /* Realiza medidas leituras dos acelerometros, estando o balao parado,
    e calcula a aceleracao residual que deve ser desconsiderada. Esta
    aceleracao e a media das leituras */
```

```
double soma[4]; /*somatoria*/

int acel[4][2]; /*guarda a leitura dos acelerometros*/
double aceleracao[4][medidas];
/*guarda a aceleracao de cada aceleromatro*/

int i; /*contador*/

printf("\nIniciando leitura para filtragem.\n");
soma[0]=soma[1]=soma[2]=soma[3]=0;

for(i=0; i<medidas; i++){
    /*leitura dos acelerometros*/
    Leitura(acel);
    aceleracao[0][i]=Calculaacel(acel[0][0], acel[0][1]);
    aceleracao[1][i]=Calculaacel(acel[1][0], acel[0][1]);
    aceleracao[2][i]=Calculaacel(acel[2][0], acel[2][1]);
    aceleracao[3][i]=Calculaacel(acel[3][0], acel[2][1]);
    }

printf("Leitura realizada.\n");

printf("\nIniciando calculos da filtragem.\n");

for(i=0; i<medidas; i++){
    /*somatoria para calculo damedia*/
    soma[0] = soma[0] + aceleracao[0][i];
    soma[1] = soma[1] + aceleracao[1][i];
    soma[2] = soma[2] + aceleracao[2][i];
    soma[3] = soma[3] + aceleracao[3][i];
```

```
        }

    /*calculo da media*/
    media[0] = soma[0]/medidas;
    media[1] = soma[1]/medidas;
    media[2] = soma[2]/medidas;
    media[3] = soma[3]/medidas;

    printf("Calculos finalizados.\n\n");
    retum;
    }

void Media(double valor[4], double aceleracao[4][10]){
    /*Adiciona um valor de aceleracao na lista das ultimas
    10 leituras, elimina a mais antiga e tira a media destes
    ultimos 10 valores de aceleracao */

    double soma[4]; /*guarda a somatoria d*/
    int i; /*contador*/

    soma[0]=soma[1]=soma[2]=soma[3]=0;

    for (i=0; i<9; i++){
        /*elimina o valormais antigo*/
        aceleracao[0][i]=aceleracao[0][i+1];
        aceleracao[1][i]=aceleracao[1][i+1];
        aceleracao[2][i]=aceleracao[2][i+1];
        aceleracao[3][i]=aceleracao[3][i+1];
        }

   aceleracao[0][9]=valor[0];
```

```
    aceleracao[1][9]=valor[1];
    aceleracao[2][9]=valor[2];
    aceleracao[3][9]=valor[3];
    /*adiciona o ultimo valor lido*/

    for(i=0; i<10; i++){
        /*realiza a somatoria*/
        soma[0]=soma[0]+aceleracao[0][i];
        soma[1]=soma[1]+aceleracao[1][i];
        soma[2]=soma[2]+aceleracao[2][i];
        soma[3]=soma[3]+aceleracao[3][i];
        }

    valor[0]=soma[0]/10;
    valor[1]=soma[1]/10;
    valor[2]=soma[2]/10;
    valor[3]=soma[3]/10;
    }

void Procedecalib(double X[3], double Y[3], double Z[3]){
    int T[4][2]; /*recebeos dados dos dutty cicles*/
    int aux; /*auxilio na calibracao*/
    double aceleracoes[4]; /*guarda as aceleracoes em X, Y1, Y2 e Z*/
    double deslocamentos[4]; /*guarda os deslocamentos em X, Y1, Y2 e Z*/
    double velocidades[4]; /*guarda as velocidades em X, Y1, Y2 e Z*/
    double deslocY;

    aux=0;
    T[0][0]=T[0][1]=T[1][0]=T[1][1]=0;
    T[2][0]=T[2][1]=T[3][0]=T[3][1]=0;
```

```
printf("\nProceda a calibracao!\n");
printf("Para cada movimento a seguir, aperte a tecla ENTER ao inicia-lo e ao termina-lo.\n");

/*---------------------------------- X ----------------------*/
printf("\nMova o balao 1m em linha reta na direcao X.\n");
printf("NAO gire o balao!\n");

while(aux==0){
    aux=kbhit();
    }
getch();
aux=0;

while (aux==0){

    Leitura(T); /*le acelerometros*/

    aceleracoes[0]=Calculaacel(T[0][0], T[0][1]);
    aceleracoes[1]=Calculaacel(T[1][0], T[1][1]);
    aceleracoes[2]=Calculaacel(T[2][0], T[2][1]);
    aceleracoes[3]=Calculaacel(T[3][0], T[3][1]);
    /*calcula aceleracoes*/

    velocidades[0]=Integra(aceleracoes[0], velocidades[0], 0.5);
    velocidades[1]=Integra(aceleracoes[1], velocidades[1], 0.5);
    velocidades[2]=Integra(aceleracoes[2], velocidades[2], 0.5);
    velocidades[3]=Integra(aceleracoes[3], velocidades[3], 0.5);
    /*calcula velocidades*/

    deslocamentos[0]=Integra(velocidades[0], deslocamentos[0], 0.5);
    deslocamentos[1]=Integra(velocidades[1], deslocamentos[1], 0.5);
```

```
        deslocamentos[2]=Integra(velocidades[2], deslocamentos[2], 0.5);
        deslocamentos[3]=Integra(velocidades[3], deslocamentos[3], 0.5);
        /*calcula deslocamentos*/
        /*E admitido que nao ha rotacao*/

        aux=kbhit();
        }

 getch();
 aux=0;

 deslocY=(deslocamentos[1]+deslocamentos[2])/2;

 X[0]=deslocamentos[0];
 X[1]=deslocY;
 X[2]=deslocamentos[2];

 printf("Calibracao de X concluida.\n");

 /*---------------------------------- Y ----------------------*/
 printf("\nMova o balao 1m em linha reta na direcao Y.\n");
 printf("NAO gire o balao!\n");

 while(aux==0){
      aux=kbhit();
      }
 getch();
 aux=0;

 while (aux==0){
```

```
        Leitura(T); /*le acelerometros*/

        aceleracoes[0]=Calculaacel(T[0][0], T[0][1]);
        aceleracoes[1]=Calculaacel(T[1][0], T[1][1]);
        aceleracoes[2]=Calculaacel(T[2][0], T[2][1]);
        aceleracoes[3]=Calculaacel(T[3][0], T[3][1]);
        /*calcula aceleracoes*/

        velocidades[0]=Integra(aceleracoes[0], velocidades[0], 0.5);
        velocidades[1]=Integra(aceleracoes[1], velocidades[1], 0.5);
        velocidades[2]=Integra(aceleracoes[2], velocidades[2], 0.5);
        velocidades[3]=Integra(aceleracoes[3], velocidades[3], 0.5);
        /*calcula velocidades*/

        deslocamentos[0]=Integra(velocidades[0], deslocamentos[0], 0.5);
        deslocamentos[1]=Integra(velocidades[1], deslocamentos[1], 0.5);
        deslocamentos[2]=Integra(velocidades[2], deslocamentos[2], 0.5);
        deslocamentos[3]=Integra(velocidades[3], deslocamentos[3], 0.5);
        /*calcula deslocamentos*/
        /*E admitido que nao ha rotacao*/

        aux=kbhit();
        }

getch();
aux=0;

deslocY=(deslocamentos[1]+deslocamentos[2])/2;

Y[0]=deslocamentos[0];
Y[1]=deslocY;
```

```
Y[2]=deslocamentos[2];

printf("Calibracao de Y concluida.\n");

/*---------------------------------- Z ----------------------*/
printf("\nMova o balao 1m em linha reta na direcao Z.\n");
printf("NAO gire o balao!\n");

while(aux==0){
     aux=kbhit();
     }
getch();
aux=0;

while (aux==0){

     Leitura(T); /*le acelerometros*/

     aceleracoes[0]=Calculaacel(T[0][0], T[0][1]);
     aceleracoes[1]=Calculaacel(T[1][0], T[1][1]);
     aceleracoes[2]=Calculaacel(T[2][0], T[2][1]);
     aceleracoes[3]=Calculaacel(T[3][0], T[3][1]);
     /*calcula aceleracoes*/

     velocidades[0]=Integra(aceleracoes[0], velocidades[0], 0.5);
     velocidades[1]=Integra(aceleracoes[1], velocidades[1], 0.5);
     velocidades[2]=Integra(aceleracoes[2], velocidades[2], 0.5);
     velocidades[3]=Integra(aceleracoes[3], velocidades[3], 0.5);
     /*calcula velocidades*/

     deslocamentos[0]=Integra(velocidades[0], deslocamentos[0], 0.5);
```

```
        deslocamentos[1]=Integra(velocidades[1], deslocamentos[1], 0.5);
        deslocamentos[2]=Integra(velocidades[2], deslocamentos[2], 0.5);
        deslocamentos[3]=Integra(velocidades[3], deslocamentos[3], 0.5);
        /*calcula deslocamentos*/
        /*E admitido que nao ha rotacao*/

        aux=kbhit();
        }

    getch();
    aux=0;

    deslocY=(deslocamentos[1]+deslocamentos[2])/2;

    Z[0]=deslocamentos[0];
    Z[1]=deslocY;
    Z[2]=deslocamentos[2];

    printf("Calibracao de Z concluida.\n");
    }

double Calculadistanciaplano (double posicao[3], double posicaof[3]){
    /* Calcula a distancia no plano XY entre a posicao atual do balao
    e a posicao de destino
      posicao indica a posicao atual do balao, enquanto que posicaof
    indica a posicao de destino.*/

    double deltax, deltay;
    /*diferenca entre o atual e o desejado para cada coordenda*/
    double distancia;
    /*distancia entre o atual e o desejado*/
```

```
deltax = posicaof[0]-posicao[0];
deltay = posicaof[1]-posicao[1];

distancia = pow((pow(deltax, 2)+pow(deltay, 2)), 0.5);
retum (distancia);
}
```

BIBLIOGRAFIA

LAAS. França. Projeto francês de dirigível autônomo por INS, GPS e visão. Disponível em http://www.laas.fr/~simon/eden/robots/blimp.php . Acesso em Abril de 2003.

ROBOAT. Itália. Projeto de um robô barco orientado por GPS. Disponível em http://dec1.wi-inf.uni-essen.de/~astephan/FIRC2000/s_5.html . Acesso em Abril de 2003.

CAN A PEN REMEMBER WHAT IT HAS WRITTEN USING INERTIAL NAVIGATION?: AN EVALUATION OF CURRENT ACCELEROMETER TECHNOLOGY. Projeto teórico de uma caneta que guarda o que escreve com o uso de acelerômetros. Disponível em http://xenia.media.mit.edu/~verp/projects/smartpen/node15.html . Acesso em Abril de 2003.

PIC16F87X DATASHEET. USA. Documentação referente ao microcontrolador PIC16F876. Disponível em www.microchip.com . Acesso em Novembro de 2003.

ADXL 202 DATASHEET. USA. Documentação referente aos acelerômetros. Disponível em www.analog.com. Acesso em Agosto de 2003

SCHILDT, H. **Turbo C:** Guia do Usuário. São Paulo: McGraw-Hill, 1988, 415p.

BUCHANAN, W. **Applied PC Intefacing, Graphics and Interrupts**: Harlow, Addison-Wesley, 1996, 383p

ELFES, A. et al. **A Semi-Autonomous Robotic Airship for Environmental Monitoring Missions**

US-LTA. **The Model 138S Airship Specifications**

BARONE, M. **Álgebra Linear.** São Paulo, IME-USP, 1988, 312p.

Boulos, P.; Camargo, I.. **Geometria Analítica:** um tratamento vetorial. São Paulo, McGraw-Hill, 1987, 385p.

Humes, A. et al. **Noções de Cálculo Numérico.** São Paulo, IME-USP, 1984, 201p.

APÊNDICE I – ESQUEMA DA LIGAÇÃO COMPUTADOR-RÁDIO

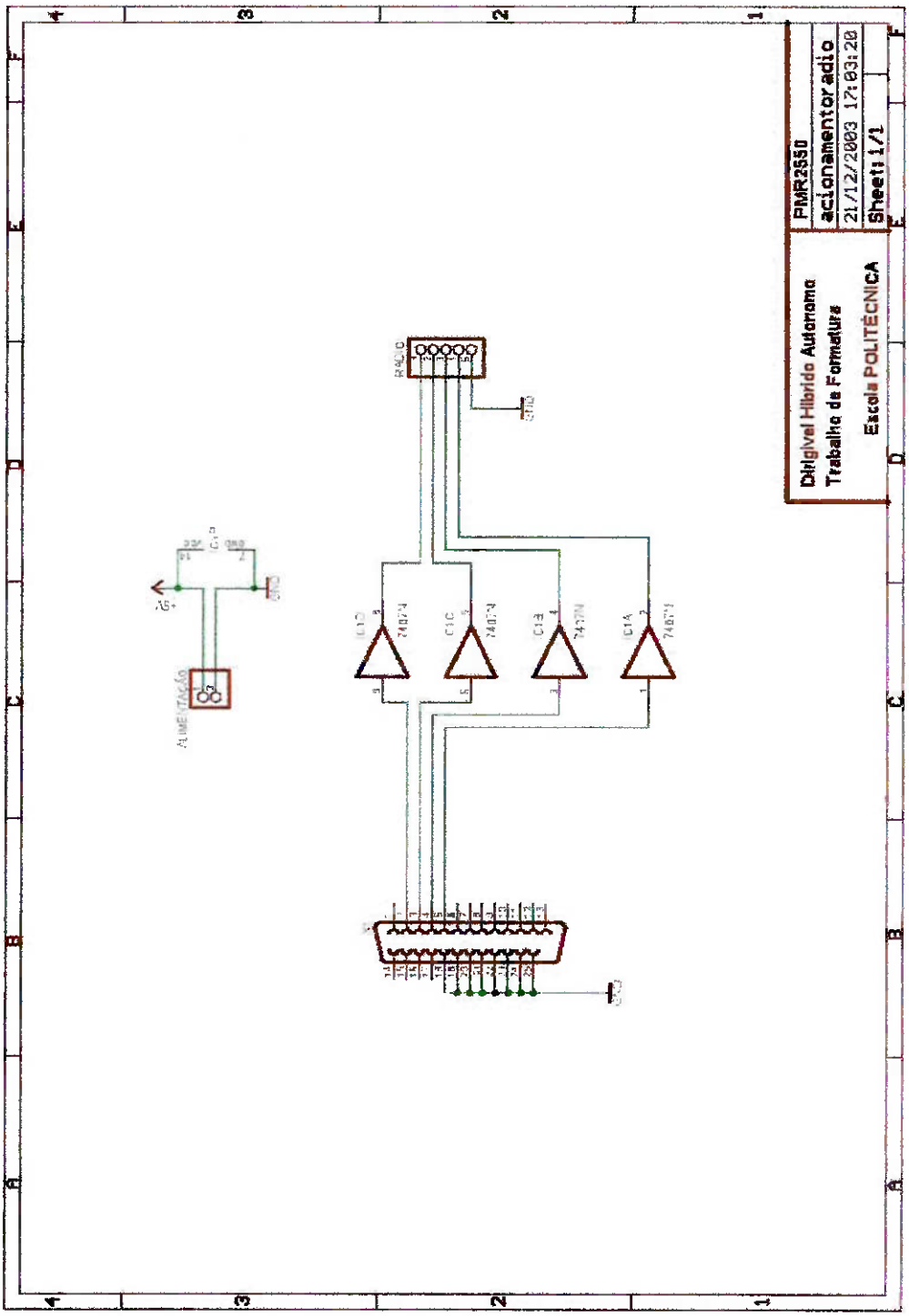

Fig. 22: Esquema da ligação computador-rádio

APÊNDICE II – ESQUEMA FINAL DA ELETRÔNICA EMBARCADA

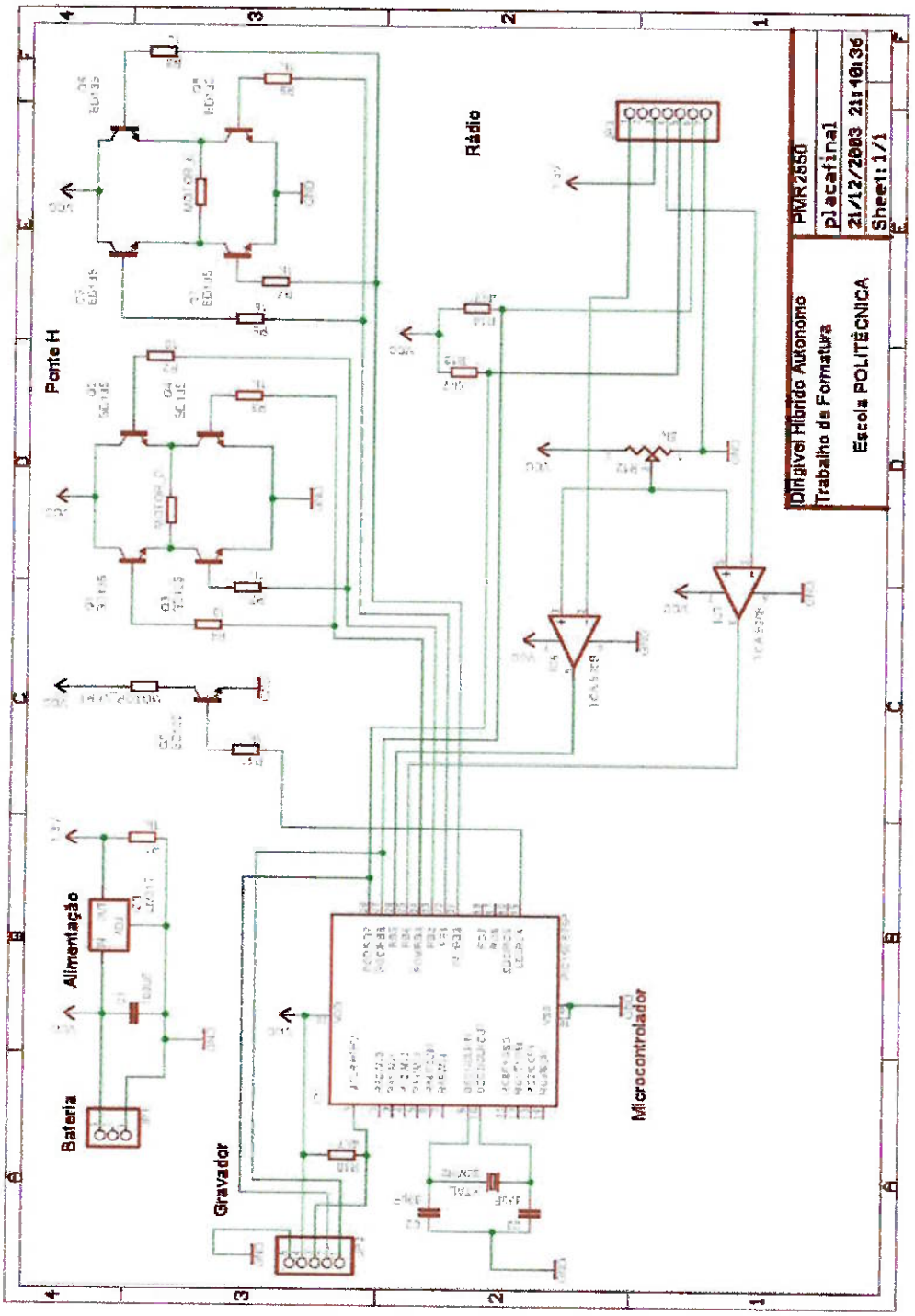

Fig. 23: Esquema final da eletrônica embarcada

APÊNDICE III – VÔO DE TESTE

Fig. 24: Vôo de teste realizado dia 15 de dezembro de 2003