Alexandre Secchin de Melo

Implementação de um Quadrotor como Plataforma de Desenvolvimento para Algoritmos de Controle

Vitória - ES, Brasil

30 de junho de 2010

Alexandre Secchin de Melo

Implementação de um Quadrotor como Plataforma de Desenvolvimento para Algoritmos de Controle

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Orientador:
Evandro Ottoni Teatini Salles

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória - ES, Brasil

30 de junho de 2010

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Central da Universidade Federal do Espírito Santo, ES, Brasil)

Melo, Alexandre Secchin de, 1981-
M528i Implementação de um quadrotor como plataforma de desenvolvimento para algoritmos de controle / Alexandre Secchin de Melo. – 2010.
113 f. : il.

Orientador: Evandro Ottoni Teatini Salles.
Dissertação (mestrado) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Helicópteros. 2. Acelerômetros. 3. Giroscópios. 4. Microcontroladores. 5. Rotores. I. Salles, Evandro Ottoni Teatini. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 621.3

Dissertação de Mestrado sob o título *"Implementação de um Quadrotor como Plataforma de Desenvolvimento para Algoritmos de Controle"*, a ser defendida por Alexandre Secchin de Melo em 30 de junho de 2010, em Vitória, Estado do Espírito Santo, pela banca examinadora constituída pelos professores:

Prof. Dr. Evandro Ottoni Teatini Salles
Orientador

Prof. Dr. José Geraldo das Neves Orlandi
Instituto Federal do Espírito Santo - Campus Serra

Prof. Dr. Luis Eduardo Martins de Lima
Instituto Federal do Espírito Santo - Campus Vitória

Prof. Dr. Klaus Fabian Côco
Universidade Federal do Espírito Santo

Resumo

Este trabalho visa a implementação de um objeto voador não-tripulado, em formato miniatura, com quatro rotores como plataforma de desenvolvimento, como parte de uma pesquisa mais abrangente. O objetivo final, ainda por ser alcançado, é chegar a um veículo voador miniatura com o máximo grau de autonomia de decisões baseadas no sensoreamento a bordo e poder computacional embarcados, estratégia do controle inteligente, e tarefa a cumprir. Tal implemetação, até o momento, consiste em uma máquina eletro-mecânica de baixo custo, cuja parte eletrônica a bordo, um microcontrolador de 8 *bits*, acelerômetros e giroscópios do tipo MEMS, permite a implementação de um controlador de voo genérico para automatizar a sua estabilização em torno dos eixos X, Y e Z. Tem vasta gama de aplicações como: inspeções aéreas em diversos ambientes, como linhas de transmissão elétrica, detecção de foragidos da polícia, monitoramento de plantações e rebanhos, bem como tomadas de filmagens para as indústrias cinematográfica e imobiliária. Testes realizados com o protótipo até agora sugerem a implementação bem sucedida de um controlador de estabilização de voo.

Abstract

This work aims the implementation of a unmanned, four-rotor miniature flying machine as a development platform, part of a long term research. The final goal, yet to be achieved, is the realization of a flying object with maximum practicable degree of autonomous decisions based on the on-board sensory and computational power, control strategy and assigned task. Up to this work, it consists of a low cost electro-mechanical hardware, whose electronic part allows the implementation of an 8-bit microcontroller-based, MEMS accelerometer and gyroscopes, allows the implementation of a generic fly control for attitude stabilization. The broad spectrum of applications includes: the inspection of various kinds of environments such as electric power transmission lines, police surveillance of woods and hard-to-reach places with sky sight for fugitive detection, crops and herd monitoring, as well as film takings for the cinematographic and real estate industries. Tests undertaking so far with the prototype suggest the successful implementation of a fly attitude controller.

Dedicatória

Dedico este trabalho ao professor Ailson.

Agradecimentos

Agradeço a Deus por ter chegado até aqui. Aos meus pais pelo incentivo. Agradeço ao professor Ailson Rosetti de Almeida, que já vem pesquisando nesse assunto a 10 anos, sempre confiante neste trabalho e pelo incalculável conhecimento transmitido com todo o prazer ao longo desse período de tempo. Agradeço ao professor Evandro Ottoni Teatini Salles por várias sugestões de grande valia que ajudaram a nortear o trabalho e por sua disposição. Agradeço ao Dr. Antônio de Pádua, que nos doou o Helicóptero a combustão Baron Alpha II utilizado no início do trabalho. Agradeço à CAPES por dois anos ininterruptos de bolsa. Agradeço aos meu amigos contemporâneos no PPGEE Christiano Couto Gava e Rodrigo Rosenfeld por me ajudarem em diversas tarefas. Agradeço os amigos do laboratório CISNE, LAI e Labtel, por suas contribuições e/ou pelo companherismo. Agradeço ao Klaus Fabian Côco e ao Érico Piredda por me receberem bem e pelo companheirismo quando entrei no laborotório CISNE, pois estes já eram membros. Agradeço também ao Eduardo Barcellos por me ajudar a soldar alguns componentes com encapsulamento SMD com seus equipamentos profissionais. Ao Claudiney e ao seu irmão, vulgo Guta, pelo seus serviços prestados no almoxarifado dos laboratórios da UFES. Agradeço ao professor Teodiano Freire Bastos Filho por nos ter emprestado por muito tempo o seu módulo de ultrassom para testes. Ao professor Hans-Jörg Schneebeli por me doar o livro do Boanerges sobre aerodinâmica para helicópteros. Agradeço à professora Raquel Frizera Vassalo pelo livro de Redes Neurais. Agradeço ainda aos professores Moisés Renato Nunes Ribeiro, Marcelo Eduardo Vieira Segatto e Márcio de Alemida Có, que acreditaram na termino deste trabalho. Por fim, agradeço a todos os outros que direta ou indiretamente contribuiram neste trabalho e ao PPGEE (Programa de Pós-Graduação em Engenharia Elétrica) por permitir o meu reingresso ao programa.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

1.1 Contextualização

Com o advento da tecnologia MEMS[1] (*Micro-Electro-Mechanical Systems*), muitos dispositivos mecânicos — Figura 1.1 — e instrumentos de medição tais como acelerômetros, giroscópios, barômetros, passaram a ter dimensões milimétricas (após encapsulados) e massa da ordem de miligramas nos últimos anos. Isto facilitou o instanciamento destes em pequenas placas de circuito impresso, consequentemente, a miniaturização de muitos equipamentos. Além do mais, o crescente uso de tais dispositivos em bens de consumo, contribuiu para sua produção em larga escala tornando-os acessíveis no mercado.

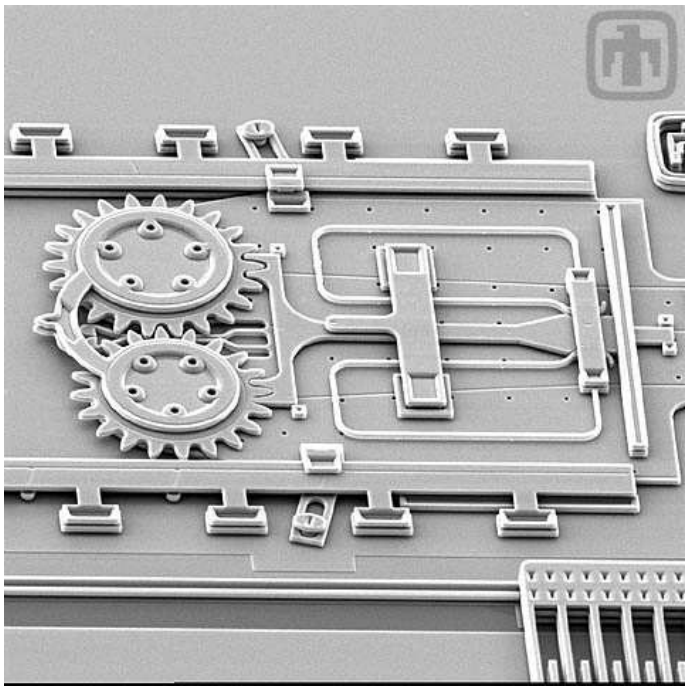

Figura 1.1: Dispositivo mecânico — embreagem do tipo MEMS [Paul Mc Whorter 2008].

[1]MEMS: tecnologia emergente que encapsula um sistema formado por sensores e/ou atuadores, elementos micro-mecânicos (pêndulos, engrenagens etc.) além de circuitos eletrônicos em um único *chip* através de técnicas e ferramentas desenvolvidas pela indústria de circuitos integrados.

Em virtude disto e de outros avanços tecnológicos, como o melhoramento na relação carga-massa[2] das baterias, o controle de voo automático de pequenas aeronaves, principalmente as não-tripuladas — conhecidas como UAVs[3] (*Unmanned Aerial Vehicles*) no âmbito militar —, veio a suceder em vista da versatilidade destas em diversas aplicações. Em especial, as elétricas, por possuírem simplicidade mecânica (o que lhes confere um grande potencial de miniaturização e redução de peso) e por não levarem combustível inflamável a bordo, experimentam forte desenvolvimento.

Quando equipada de uma câmera fotográfica/filmadora que transmita a imagem ao vivo através de um *link* de RF (rádio frequência), uma aeronave não-tripulada do tipo VTOL[4] de pequeno porte pode ser útil nas seguintes ocasiões:

- fotos e filmagens panorâmicas de baixo custo para a indústria cinematográfica, mercado imobiliário (vista superior de casas, terrenos, chácaras e sítios), reportagens de telejornais, eventos esportivos etc. [Perspectives Aerials 2010];
- localização de um bandido em recintos onde não há visada direta; assim, evita-se que um policial seja surpreendido pelo bandido durante a sua procura [Draganflyer 2010];
- inspeção de linhas de transmissão e distribuição elétrica;
- monitoramento de plantações e grandes rebanhos;

Entre as principais UAVs elétricas de pequeno porte — aviões, helicópteros e dirigíveis (*blimps*) [Bouabdallah, S. Murrieri, P. 2004] — destaca-se o helicóptero, em virtude de: possuir seis graus de liberdade (DOF[5]), que lhe conferem manobrabilidade, além de o classificar como uma aeronave do tipo VTOL; possui um bom *payload*[6]; possui capacidade de miniaturização [Prinz, F. B. 1999] e uso *indoor* (dentro de recintos) ou *outdoor* (em campo aberto); apesar de ter uma autonomia menor em relação às demais UAVs citadas. Porém, com o crescente avanço da tecnologia das baterias elétricas, é possível que tal desvantagem se torne menos expressiva no futuro.

O avião, Figura 1.2, apesar de ter um bom *payload* e ser mais estável (uma vez que tenha atingido uma velocidade mínima para obter sustentação aerodinâmica), não pode decolar e

[2]Carga-massa: relação entre a carga em [mAh] e a sua massa em [g].

[3]UAVs: aeronaves não-tripuladas de tamanho e formas variadas inicialmente utilizada em fins militares e controladas remotamente ou de forma autônoma com voos pré-programados.

[4]VTOL: *Vertical Take Off and Landing*. Aeronave com capacidade de decolar e aterrissar verticalmente.

[5]*Degrees of Freedom* — graus de liberdade. Um helicóptero possui os 6 possívies graus de liberdade: translação ao longo nos três eixos X, Y e Z e rotação em torno destes mesmos eixos: *pitch*, *roll* e *yaw, respectivamente*.

[6]Carga transporatada por uma aeronave, inclusive o que é necessário para a sua operação.

aterrissar verticalemente (além precisar de um pista de pouso e decolagem), nem se aproximar de outros objetos como um helicóptero ou dirígel, sendo utilizado na maioria das vezes em voo *outdoor*.

Figura 1.2: Avião elétrico monomotor.

O dirigível, Figura 1.3, por ser classificado com uma aeronave mais leve que o ar [Bouabdallah, S. Murr não precisa de propulsores para gerar sustentação aerodinâmica, limitando o uso destes dispositivos apenas para se locomover. Embora tem uma estabilidade maior, seja silencioso e mais fácil de se controlar, são mais volumosos e possuem um *payload* inferior ao dos helicópteros e aviões do mesmo porte. Portanto, limita-se o seu uso a publicidade em eventos, estampado alguma propaganda no seu balão (normalmente preenchido com gás hélio, por ser mais leve que o ar), principalmente em eventos realizados dentro de recintos.

Figura 1.3: Dirigivel elétrico voando *indoor*.

Nos helicópteros convencionais, as sustenção aerodinâmica é provida por um único rotor, o principal, cujo torque de reação é anulado pelo rotor de calda, ou secundário. No entanto, existe

outra topologia de helicóptero, em forma de plataforma, onde a sua força de sustenção é divida em quatro ou mais rotores de mesma potência e de forma que o torque de reação de um cancele o de outro.

Isto permite que tal paradigma de helicóptero realize todas as manobras (ainda com seis graus de liberdade) que um convencial controlando apenas a velocidade de cada rotor de forma independente. Assim, dispensa-se o uso de mecanismos complexos de ajuste de ângulo de incidência das pás de um rotor, como o passo coletivo[7] e passo cíclico[8] usados no helicóptero convencional.

Apesar do helicóptero quadrotor (plataforma voadora com quatro rotores) ter surgido por volta de 1920 — Figura 1.4 —, este ficou esquecido por décadas por conta da sua dificuldade de estabilização por parte do piloto.

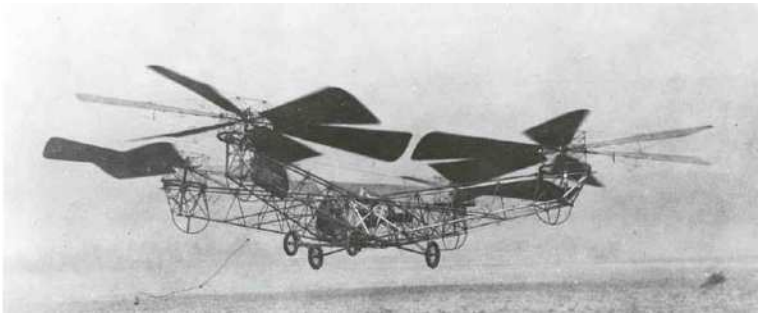

Figura 1.4: Quadrotor de De Bothezat em 1923 [Wikipedia 2010].

No entanto, nos últimos anos, com os avanços tecnológicos já citados (tecnologia MEMs e baterias leves), estudos a cerca da estabilização de voo do quadrotor, agora automática, foram retomados para utilizá-lo no âmbito das UAVs, visto que mecanicamente este é mais simples que um helicóptero convencional de mesmo porte, o que diminui custos de construção e manutenção; além de permitir uma maior proximidade e interação com ambiente, pois durante o voo armazena menos energia cinética em um rotor por este ser menor (sustenção divida em quatro rotores), e em caso de um rotor colidir com outro corpo, os estragos são menores.

1.2 Objetivo

O objetivo desse trabalho é construir uma UAV elétrica do tipo VTOL, especificamente um quadrotor com seus rotores posicionados nos extremos de uma estrutura com forma semelhante

[7]Passo coletivo: ajuste do ângulo de incidência das pás de um rotor de forma homogênea ao longo de um giro completo deste rotor.

[8]Passo cíclico: ajuste do ângulo de incidência de forma heterogênea de pás de um rotor ao logo de um giro completo deste rotor.

a de um "x", segundo a Figura 1.5 — com massa e dimensões reduzidas[9] como uma plataforma de desenvolvimento de baixo custo.

A presente abordagem permite a elaboração e teste de técnicas de controle para a estabilização de voo. Os principais componentes do quadrotor são: quatro motores elétricos, quatro ESCs[10], uma placa microcontrolada, uma placa de sensoreamento, um rádio receptor e uma bateria.

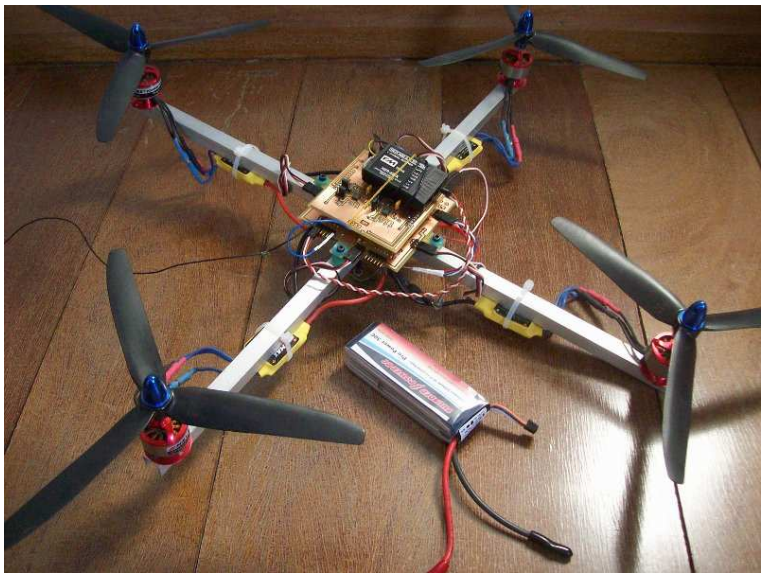

Figura 1.5: Foto do quadrotor do presente trabalho [Alexandre Secchin de Melo 2010].

1.3 Estudos Relacionados

Como pode ser visto em diversos trabalhos acadêmicos, [Quemel, P. H. R. 2009], [Bouabdallah, S. 2004], [Hoffmann, G. et all 2004], e [Bresciani, T. 2008], vários estudos já foram feitos a respeito da estabilidade de voo usando técnicas de controle clássico, principalmente utilizando um controlador PID (Proporcional, Integral e Derivativo) sintonizado a partir de um complexo modelo matemático de um quadrotor genérico linearizado sobre um ponto de operação (voo pairado).

Segundo [Waslander, S. L. Hoffmann, G. M. 2005], técnicas de controle clássico, ainda que funcionais, são insuficientes em prover estabilidade a um quadrotor devido ao complexo fluxo de ar induzido pelos seus 4 rotores. Por outro lado, existem outras técnicas de controle

[9] Massa em torno de 1kg (sem carga) e dimensões entre 50 e 60cm (para o quadrotor deste trabalho estas dimensões podem variar dentro desta faixa conforme o posicionamento dos rotores na estrutura e o tamanho destes), permitindo uma aeronave como tal voar *indoor* (ambientes interiores).

[10] ESC — controlador eletrônico de velocidade para motores do tipo *brushless*, normalmente controlado a partir de um sinal de PWM (*pulse width modulation*).

que ainda foram pouco exploradas no âmbito dos quadrotores elétricos, principalmente aquelas que se baseam em técnicas inteligentes [Hartman, E. Keeler, J. D. 1989], [Hornik, M. 1989], [Sontag, E. D. 1993] de controle como o neural e nebuloso (*fuzzy*) que são por natureza não-lineares e aproximadores universais de função (modelamento) [Newman 2003].

Um trabalho nessa linha de pesquisa já vem sendo realizado no Brasil. No entanto, até a última publicação [Quemel, P. H. R. 2009] este se limitou a simulações sobre modelagem matemática de um quadrotor utilizando técnica de controle clássico, neste caso, PID.

Das referências citadas, pouco é divulgado a respeito do *hardware* (esquemas) e *firmware* utilizados, principalmente detalhes de implementação. Assim sendo, o escopo do presente trabalho se limita ao projeto e construção de um *hardware* e *firmware* que dê subsídio para a implementação de alguma técnica de controle (bloco 5 da Figura 3.1), isto é, um quadrotor como plataforma de desenvolvimento.

1.4 Estrutura da Dissertação

Inicialmente, no capítulo 2, informações preliminares a respeito do princípio de funcionamento do quadrotor implementado e da eletrônica a bordo deste, são apresentadas a fim de facilitar a compreensão deste trabalho. Onde a eletrônica a bordo compreende uma placa microcontroladora, uma placa de sensoreamento, controladores eletrônicos de velocidade para motores *brushless* e um rádio receptor conforme a Figura 2.1.

A implementação do quadrotor, tanto a parte de *hardware* (parte elétrica e eletrônica) quanto a de *firmware*[11], encontram-se ao longo do capítulo 3. Este *firmware* tem como objetivo não só validar o *hardware*, mas como também servir de *driver*, isto é, um programa que interfaceie a eletrônica do quadrotor com o controlador a ser implementado.

O *firmware* é descrito por partes nas seções 3.2.1, 3.2.2, 3.2.3, 3.2.4, e 3.2.6, onde as principais rotinas implementadas são acompanhadas pelos seus respectivos fluxogramas, estando no anexo A o código-fonte escrito na linguagem C pertinente ao *firmware* como um todo.

Para a validação do *hardware*, no capítulo 4 apresenta-se os resultados de testes, coletando-se informações sobre todos os sinais de interesse com todos os sinais analógicos e todos os canais do rádio receptor.

No último capítulo (5), encontra-se a conclusão a respeito do trabalho além de sugestões

[11] Aplicativo gravado na memória de programa (não volátil) de um microcontrolador (neste caso memória *flash*) evitando a recarga do programa toda vez tal dispositivo for ligado.

para melhorias e implementações futuras.

2 Referencial Teórico

Neste capítulo, são apresentados alguns conceitos que elucidam o princípio de funcionamento do quadrotor, cujo diagrama de blocos encontra-se na Figura 2.1, bem como informações fundamentais a respeito da eletrônica utilizada.

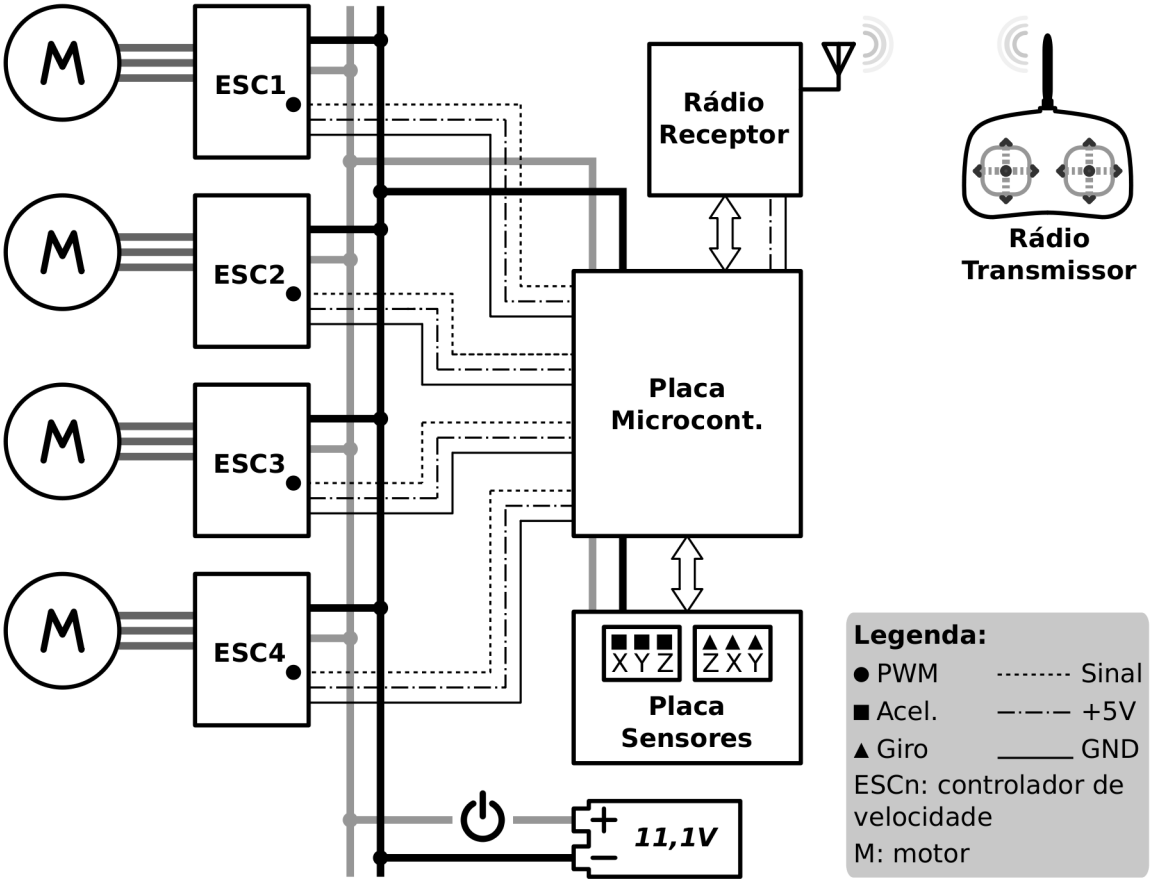

Figura 2.1: Diagrama de blocos elétrico-eletrônico do quadrotor.

2.1 Princípio de Funcionamento do Quadrotor

Um helicóptero pode assumir várias configurações no que diz respeito à disposição e complexidade dos seus rotores. Contudo, dentre várias, a configuração que conjuga simplicidade mecânica de rotores e versatilidade em manobras é a configuração em forma de plataforma. Dentre elas, a mais conhecida é a do quadrotor — Figura 1.5 — devido a sua simplicidade de construção e funcionamento.

Na configuração quadrotor, quatro rotores de mesmas dimensões estão acoplados, cada um, a um motor; e, cada um destes está fixado em uma das extremidades de uma estrutura em forma de "x". Conforme a Figura 2.2, dois rotores, 1 e 2 — de extremidades opostas em relação ao centro da estrutura — giram no sentido anti-horário e os outros dois, 3 e 4, no sentido horário a fim de anular o torque de reação[1] que cada um cria. Assim, as pás dos rotores 1 e 2 devem ter ângulo de incidência[2] invertido em relação às pás dos rotores 3 e 4 para que se crie sustentação aerodinâmica positiva [Boanerges, A. V.].

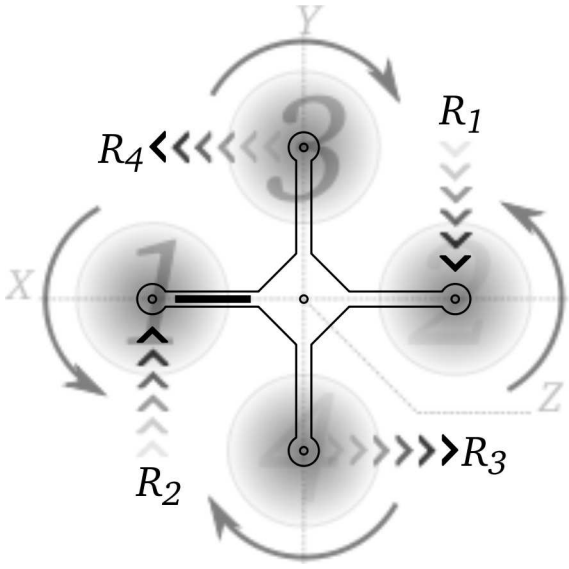

Figura 2.2: Vista superior de um quadrotor. Onde o motor 1 corresponde a sua frente e o motor 3 a sua direita.

Na Figura 2.2, *R1*, *R2*, *R3* e *R4* representam as reações criadas pelos torques de cada rotor. *R1* e *R2* se somam e tendem a girar o quadrotor em torno do seu eixo vertical no sentido horário. *R3* e *R4* também se somam, entretanto, tendem a girar o quadrotor no sentido anti-horário. Se todos os rotores permanecerem na mesma velocidade, a soma de *R1* e *R2* se cancela com a soma de *R3* e *R4*, portanto, o torque de reação resultante torna-se nulo neste caso.

[1] torque que faz a aeronave girar em torno do seu eixo imaginário vertival — eixo Z.

[2] ângulo formado entre o plano de rotação — plano ortogonal ao eixo do rotor — e uma pá de um rotor [Boanerges, A. V.].

Através do controle individual de velocidade de cada rotor, é possível realizar todas as manobras que um helicóptero convencional é capaz, sem qualquer mecanismo extra para ajustar o ângulo de incidência das pás dos rotores ao longo do seu giro, tais como os mecanismos de passo coletivo e passo cíclico dos helicópteros convencionais[3] [Boanerges, A. V.]. Isto resulta em um rotor simples, isto é, um propulsor rígido[4] como o da Figura 2.3 acoplado diretamente no eixo do motor (no caso de uma aeroname de dimensões reduzidas, geralmente um motor de corrente contínua — motor DC, cujo princípio de funcionamento se encontra no apêndice A).

Figura 2.3: Propulsor tri-pá.

Normalmente, em rádio controle (aeromodelismo), o movimento vertical se dá através do comando *throttle* (aceleração[5]). O movimento na horizontal pode ser realizado por dois outros comandos: *pitch* (arfagem) e *roll* (rolagem). Com o comando *pitch*, a aeronave é inclinada para frente ou para trás. Já com o comando *roll* a aeronave é inclida para um lado ou para o outro. Por último, o movimento em torno do seu eixo vertical (Z), isto é, uma guinada, é feito pelo comando *yaw* (guinada).

Para movimentar-se verticalmente, a velocidade dos rotores do quadrotor devem aumentar ou diminuir simultaneamente e com mesma intensidade para que o quadrotor respectivamente suba ou desça, isto é, acelerando ou desacelerando de forma simultânea os quatro motores — *throttle*.

[3]Helicópteros com um rotor principal para gerar sustentação aerodinâmica e um rotor de calda para a estabilização do torque de reação.

[4]Conjunto de pás aerodinâmicas fixadas em um ponto comum (centro do rotor) sem qualquer grau de liberdade, a não ser a flexibilidade conferida pelo material que as constituem.

[5]No contexto dos helicópteros a combustão, o comando throttle equivale a aumentar a injeção de combustível (acelerador), isto é, acelerar motor. Já no contexto dos quadrotores, consiste em aumentar a velocidade do quatro motores simultaneamente

Já para um movimento na horizontal, ou seja, ao longo do eixo X ou Y, as velocidades dos rotores devem ser controladas da seguinte forma: diminui-se a velocidade de um rotor e aumenta-se no mesmo tanto a velocidade do motor que gira no mesmo sentido (motor oposto) para que não haja desequilíbrio nos torques de reação. Os outros dois motores devem permanecer na mesma velocidade. Isto faz com que o quadrotor se encline, dando origem a um movimento horizontal, sem perder sustentação.

Por exemplo, para um movimento horizontal a frente, a velocidade do rotor 1 deve diminuir no mesmo tanto que se aumenta a velocidade do rotor 2, conforme a Figura 2.4, o que corresponde ao comando *pitch*. Isto faz com que a força de sustentação resultante tenha uma projeção também sobre o eixo X, resultando no movimento. Analogamente, diminuindo a velocidade do rotor 3 (que está do lado direito do quadrotor) e aumentando no mesmo tanto a do rotor 4, o quadrotor se movimenta para direita (ao longo do eixo Y), correspondendo ao comando *roll*.

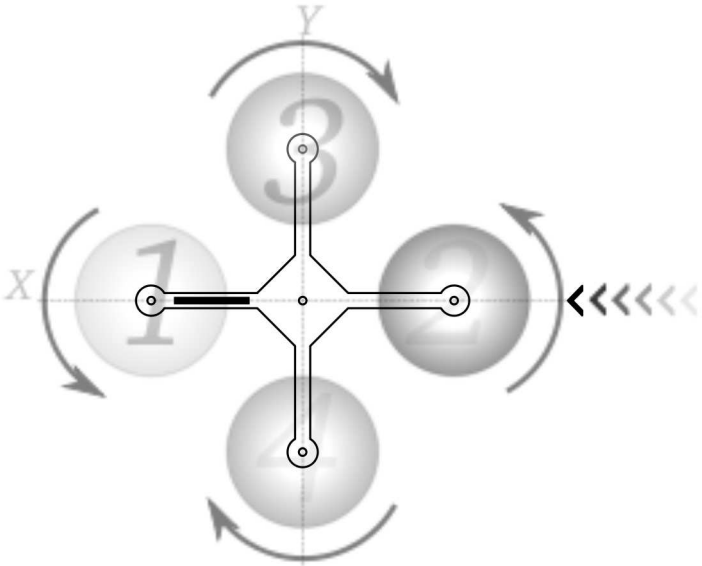

Figura 2.4: movimento horizontal a frente, onde o rotor mais escuro (2) possui maior rotação e o rotor (1) menor rotação, enquanto os rotores 3 e 4 mantêm suas rotações.

E, para fazer um giro — guinada (comando *yaw*) — em torno do eixo vertical (eixo Z), seja no sentido horário ou anti-horário, basta que se aumente igualmente a velocidade de dois rotores que giram no mesmo sentido no mesmo tanto que se diminui a velocidade dos outros dois motores, e vice-versa, conforme a Figura 2.5.

Estes movimentos elementares, reunidos na Tabela 2.1, podem ser mesclados possibili-

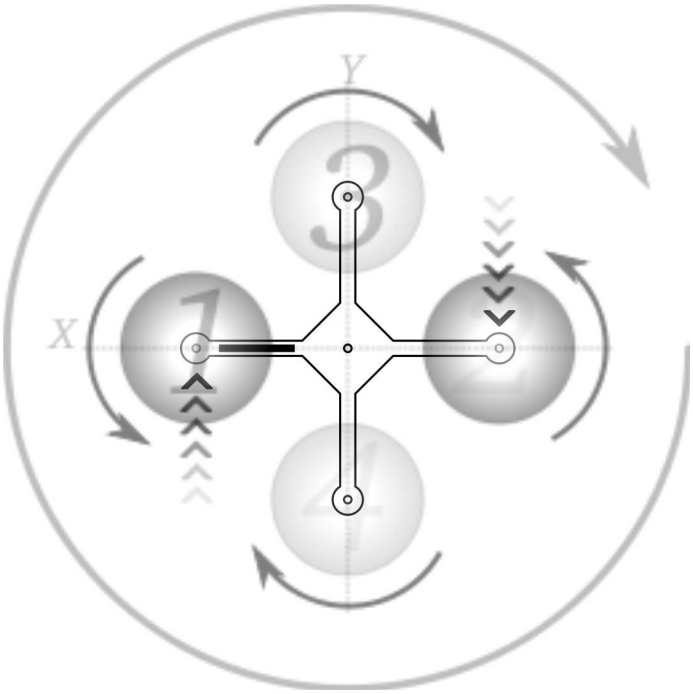

Figura 2.5: movimento em torno do eixo Z no sentido horário. Motores 1 e 2 rodando mais rápido que os motores 3 e 4.

tando o quadrotor se movimentar/orientar em qualquer direção no espaço, o que lhe confere versatilidade em manobras, prescindindo o uso de mecanismos complexos em seus rotores.

Movimento	**Motor 1**	**Motor 2**	**Motor 3**	**Motor 4**
para cima (*throttle*+)	$v_1 + \Delta v$	$v_2 + \Delta v$	$v_3 + \Delta v$	$v_4 + \Delta v$
para baixo (*throttle*−)	$v_1 - \Delta v$	$v_2 - \Delta v$	$v_3 - \Delta v$	$v_4 - \Delta v$
para frente (*pitch*+)	$v_1 - \Delta v$	$v_2 + \Delta v$	v_3	v_4
para trás (*pitch*−)	$v_1 + \Delta v$	$v_2 - \Delta v$	v_3	v_4
para direita (*roll*+)	v_1	v_2	$v_3 + \Delta v$	$v_4 - \Delta v$
para esquerda (*roll*−)	v_1	v_2	$v_3 - \Delta v$	$v_4 + \Delta v$
horário (*yaw*+)	$v_1 + \Delta v$	$v_2 + \Delta v$	$v_3 - \Delta v$	$v_4 - \Delta v$
anti-horário (*yaw*−)	$v_1 - \Delta v$	$v_2 - \Delta v$	$v_3 + \Delta v$	$v_4 + \Delta v$

Tabela 2.1: Movimentos elementares do quadrotor e a correspondente variação de velocidade em cada um de seus motores. Onde, v_i é a velocidade de um rotor i, com i igual a 1, 2, 3 ou 4.

A Tabela 2.1, pode ainda ser resumida através das relações 2.1, 2.2, 2.3 e 2.4:

$$v_1 \propto (throttle - pitch - yaw), \tag{2.1}$$

$$v_2 \propto (throttle + pitch - yaw), \tag{2.2}$$

$$v_3 \propto (throttle - roll + yaw), \tag{2.3}$$

$$v_4 \propto (throttle + roll + yaw), \tag{2.4}$$

isto é, a velocidade de cada motor é proporcional ao comando de voo *throttle* e a outros dois comandos conforme a sua posição na estrutura: *pitch* e *yaw* (para os motores 1 e 2) ou *roll* e *yaw* (para os motores 3 e 4).

2.2 Modulação por Posição de Pulso

Em aeromodelismo, para se controlar uma aeronave ou embarcação remotamente, utiliza-se um *link* de RF (Rádio Frequência) do tipo FM (*Frequence Modulation*) composto por um rádio transmissor e um rádio receptor, cuja portadora pode ser de 72MHz ou 2,4GHz, onde tais frequências são liberadas pelo orgão competente (ANATEL no Brasil) para este fim.

A Figura 2.6[6] ilustra um transmissor típico usado em RC (Rádio Controle) para controlar manualmente aeronaves remotamente, cujos comandos são dados através de dois *sticks* (alavancas) — um controlado pela mão esquerda e o outro pela mão direita, que neste trabalho, são chamados de *stick* A e B, respectivamente. Cada um deles pode ser inclinado para frente ou para trás, e para um lado ou para o outro.

Tratando-se de um helicóptero, o *stick* A é responsável pelos comandos de voo *throttle* e *yaw*. Quando empurado para frete ou para trás, aumenta ou diminui a velocidade dos rotores. Este mesmo *stick*, quando empurrado para direita ou esquerda, faz com que o helicóptero gire no sentido horário ou anti-horário — *yaw* (guinada).

Já o *stick* B é responsável pelos comandos de voo *pitch* e *roll*. Quando empurrado para frente ou para trás, a aeronave deve se inclinar para frente ou para trás. E, quando empurrado para um lado ou para o outro, a aeronave deve se inclinar (rolar) para um lado ou para o outro.

Tratando-se ainda de um típico transmissor usado em RC, os comandos de voo são enviados serialmente, de 20ms em 20ms, e modulados por posição de pulso, isto é, no formato PPM (*Pulse Position Modulation*) [Ryan, M. 2002]. Neste formato, a posição que cada *stick* do transmissor assume — logitudinalmente ou transversalmente — é representada por um pulso

[6]Os transmissores utilizados em aeromodelismo possuem mais funcionalidades — botões e chaves — agregadas. Todavia, na Figura 2.6 apenas os *sticks* estão representados por serem os únicos recursos do transmissor utilizados no quadrotor deste trabalho.

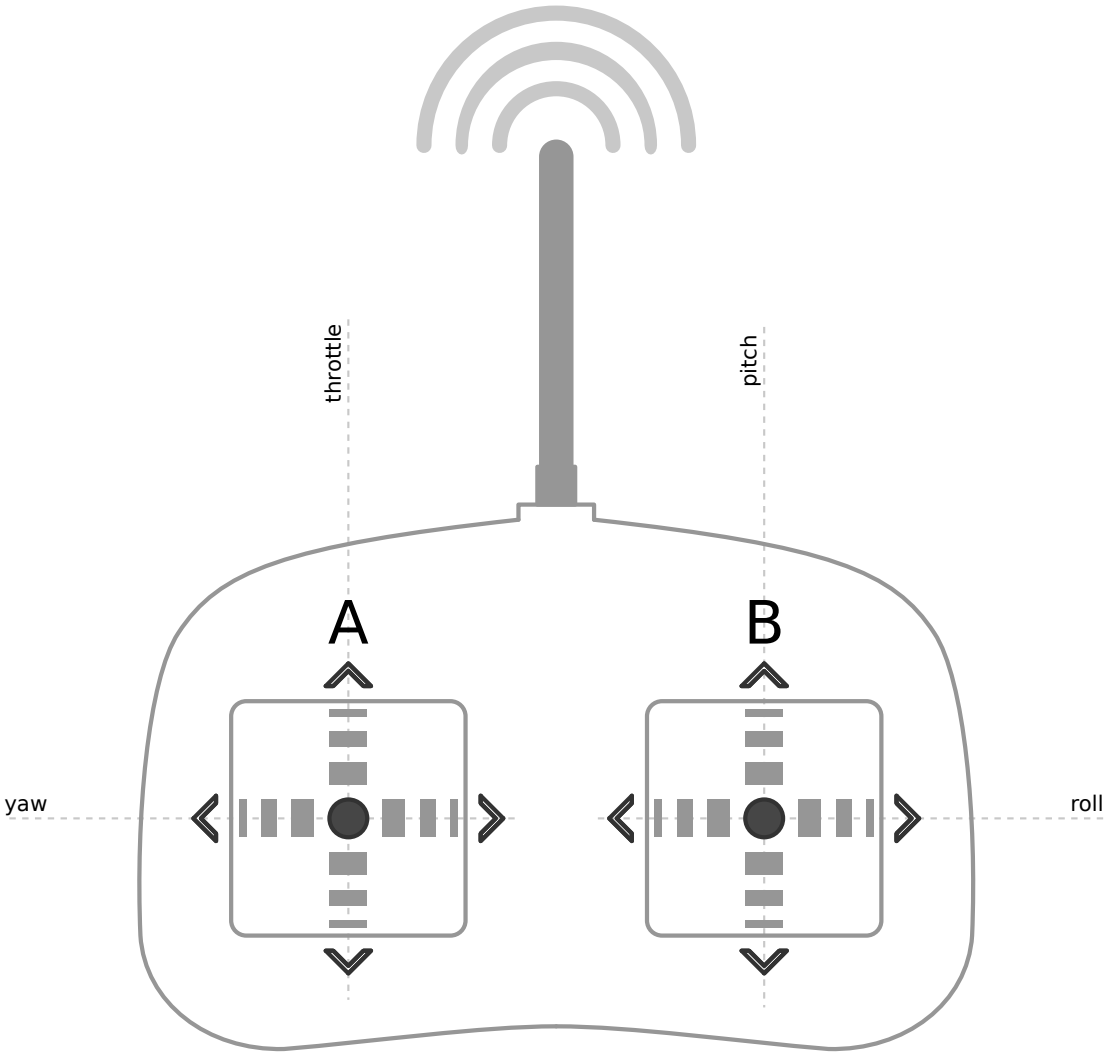

Figura 2.6: Rádio transmissor com dois *sticks* (círculos na figura) visto de cima utilizado em aeromodelismo. Ambos *sticks* com movimento horizontal e vertical.

invertido com largura de 0,3ms (normalmente), cujo término pode assumir qualquer instante entre 1 e 2ms. As Figuras 2.7 a) e c) ilustram um *stick* assumindo as posições extremas e a Figura 2.7 b) o mesmo *stick* na posição central. Logo, cada canal ocupa uma janela de tempo que pode variar de 1ms a 2ms.

Segundo a Figura 2.8, para um rádio transmissor de 6 canais, estes são enviados em forma de carrilhão, ou seja, um atrás do outro: canal 1, canal 2, canal 3, canal 4, canal 5 e canal 6. Após serem enviados, o sinal é mantido em nível lógico alto e, 0,3ms antes de terminar o quadro de 20ms, o sinal retorna para o nível lógico baixo com propósito de sincronismo — início de um novo carrilhão.

O receptor — Figura 2.9 (o mesmo da Figura 2.1) — é dividido basicamente em dois blocos: o bloco de RF[7] e o bloco de demultiplexação. O primeiro bloco simplesmente remove

[7]O circuito de RF do rádio receptor tem a frequência do seu oscilador dada pelo cristal (acessível por fora do

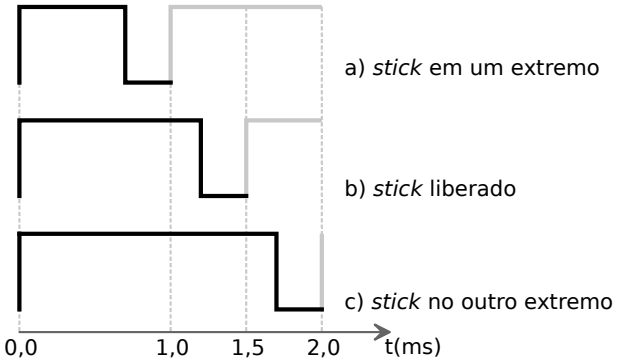

Figura 2.7: Posição do pulso detro da janela de 2ms (PPM) em função da posição da manete (*stick*)

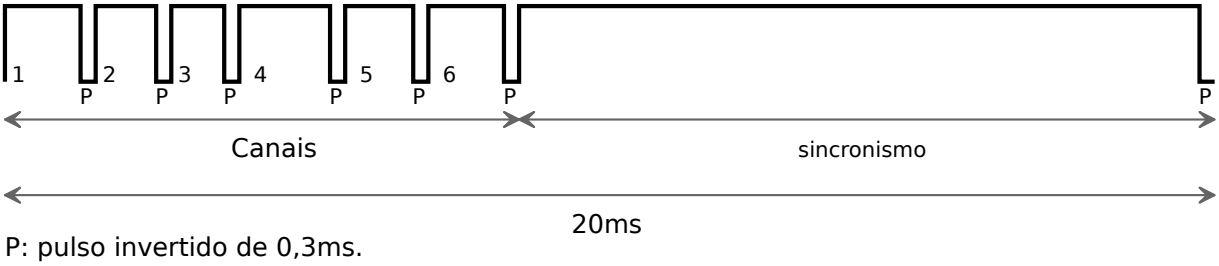

Figura 2.8: Carrilhão para um rádio de seis canais e sinal de sincronísmo.

a portadora usada no sinal de RF com objetivo de reconstituir o carrilhão de canais. O segundo bloco é responsável por demultiplexar o carrilhão de canais em seis sinais de PWM que podem ser usados como sinais de entrada em servo-motores ou controladores eletrônicos de velocidade (ESC — *Electronic Speed Control*) para motores.

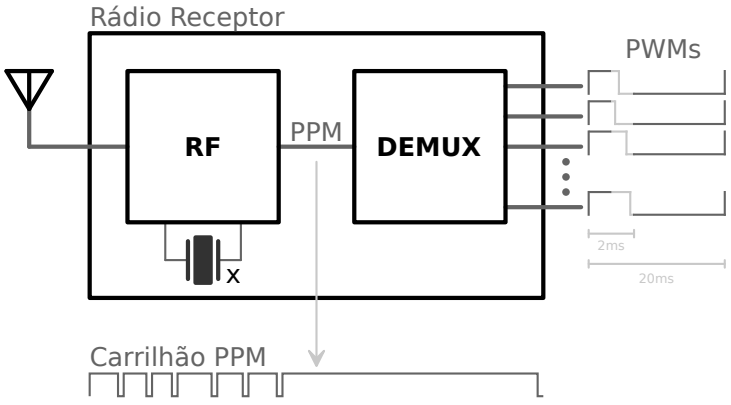

Figura 2.9: Rádio receptor.

Um sinal de PWM gerado pelo rádio receptor possui um período de 20ms, pois está atrelado ao período de envio do carrilhão pelo rádio transmissor. O tempo mínimo que um sinal de PWM

rádio) idententificado pela letra *X* na Figura 2.9. Este cristal pode ser substituído caso algum outro *link* de rádio já esteja operando na mesma frequência, o que é comum acontecer em aeromodelismo. Se for o caso, o cristal do correspondente rádio transmissor, também removível, deve ser trocado.

permanece em nível lógico alto (normalmente 5V) é de 1ms, e o tempo máximo, 2ms. O restante do tempo, o sinal permanece em nível lógico baixo até que se complete os 20ms.

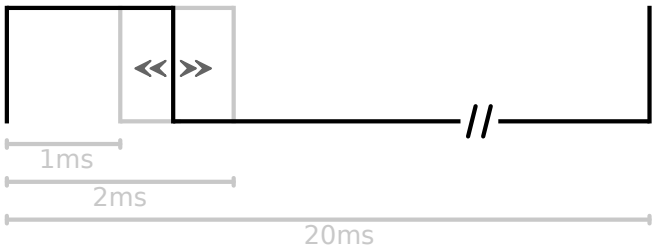

Figura 2.10: Sinal de PWM gerado pelo rádio receptor para alimentar a entrada de sinal de um servo-motor ou ESC.

Conforme a Figura 2.10, para um servo-motor que possui posicionamento entre 0 e 180°, quando o sinal transita de nível lógico alto para baixo em 1ms o servo se posiciona em um extremo e quando o sinal transita de nível lógico alto para baixo em 2ms o servo se posiciona no outro extremo. Uma transição de nível lógico alto para baixo em qualquer instante entre 1ms e 2ms, corresponde a um posicionamento entre 0 e 180°.

Já em um ESC — circuito que será apresentado na seção 3.1.1 —, uma transição nos instantes de 1 e 2 ms correspondem respectivamente ao motor parado e ao motor girando em sua velocidade máxima (em RPM); enquanto uma transição em qualquer instante entre 1ms e 2ms corresponde a uma velocidade intermediária.

2.3 Instrumentação

Normalmente, a instrumentação necessária para realizar um controlador que estabilize o giro em torno dos eixos X, Y e Z de um quadrotor, isto é, que o mantenha alinhado com o horizonte (eixo Z do quadrotor paralelo ao vetor aceleração da gravidade), é constituída por dipositivos como acelerômetros, funcionando como sensores de inclinação, e giroscópios para detectar velocidade angular em torno de um eixo [Stingu, E].

Os dispositivos utilizados neste trabalho são do tipo MEMS, portanto, de dimensões e peso reduzidos quando comparado aos mecânicos — Figura 2.11: exemplo de um giroscópio mecânico — o que favorece a implementação de pequenas aeronaves microcontroladas.

O princípio de funcionamento no qual é baseado o acelerômetro triaxial (XYZ) deste trabalho, o MMA7260Q do fabricante Freescale, consiste em um circuito com três microplacas capacitivas — por eixo —, onde uma delas está situada entre as outras duas, formando um circuito equivalente a dois capacitores em série. Na ausência de aceleração — Figura 2.12 a)

Figura 2.11: Giroscópio mecânico de massa girante desencapsulado.

— a placa do meio encontra-se equidistante das outras duas placas. Portanto, a diferença de potencial V_1 da Figura 2.12 a) é igual a V_2, na mesma figura.

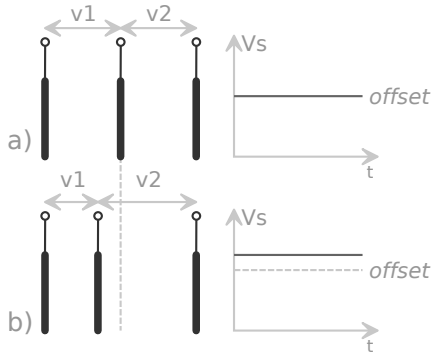

Figura 2.12: a) Acelerômetro sem aceleração; b) Acelerômetro submetido à aceleração.

Por outro lado, perante a uma variação de velocidade a e/ou da ação da gravidade[8] a_g, a placa central se desloca alterando a capacitância dos dois capacitores; consequentemente, a diferença de potencial cresce em um e diminui no outro. Logo, o a diferença entre V_1 e V_2 é proporcional à soma das acelerações a e a_g correspondendo a um desvio em torno do *offset* ilustrado na Figura 2.12 b).

Já os giroscópios utilizados, LPR510AL (biaxial — XY) e LYS510LH (Z), ambos do fabricande ST Microelectronics, têm o princípio de funcionamento baseado na força de Coriolis [Santos, I. F. 2001] e em material piezoelétrico. Tais dispositivos possuem uma massa, que quando submetida a uma diferença de potencial senoidal — gerada internamente nesse dispositivo — vibra em uma determinada direção, por ser de natureza piezoelétrica.

[8]Gravidade: propriedade atrativa que a terra exerce sobre os corpos.

Quando o giroscópio rotaciona, ou seja, adquire uma velocidade angular, a massa interna ao dispositivo passa a vibrar tanto na direção original quanto na direção ortogonal. Onde o segundo movimento vibratório é resultado da força de Coriolis. Esta nova vibração cria uma diferença de potencial, que estrategicamente coletada na estrutura piezoelétrica, corresponde à presente velocidade angular do dispositivo.

2.4 Princípios Sobre ESCs

Diferentemente dos motores de corrente contínua (DC), os motores do tipo *brushless*, cujo princípio de funcionamento se encontra no apêndice A, não são alimentados através de escovas, mas sim através de um circuito eletrônico. Este circuito, além de prover energia aos enrolamentos de um motor *brushless* a partir de uma fonte DC, também realiza um controle de velocidade em malha fechada. Por isto, tal circuito é chamado de ESC (*Electronic Speed Control*) e a sua ilustração em função das suas entradas e saídas se encontra na Figura 2.13.

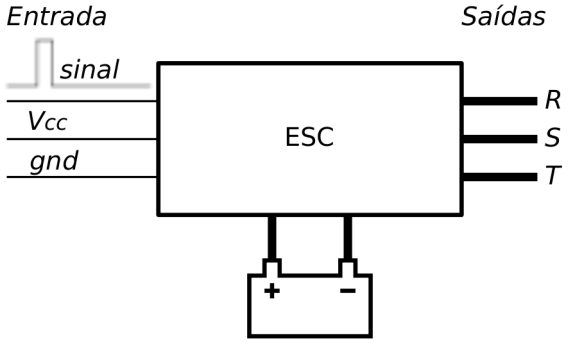

Figura 2.13: ESC genérico, onde R, S e T são os terminais que devem ser conectados às três fases de um motor *brushless*.

Segundo a Figura 2.15, um ESC é basicamente dividido em duas partes: uma de controle e outra de potência. A primeira, normalmente é constituída por um microcontrolador, que a partir de um sinal de PWM (*setpoint*) com período de 20ms (o mesmo descrito na seção 2.2), gera três outros, normalmente trapezoidais e defasados entre si de 120° conforme a figura 2.14. O período destes pode variar de fabricante para fabricante, sendo que valores típicos são 8KHz e 16kHz.

Para fazer o controle de velocidade em malha fechada, alguns modelos de ESCs lêem os sinais gerados por sensores de efeito hall — bloco "sensores"na Figura 2.15 — estrategicamente instalados no motor *brushless*. Já outros, são capazes de ler a força contra-eletromotriz (FCEM) induzida nos enrolamentos dos motores para fazer o controle de velocidade e são conhecidos

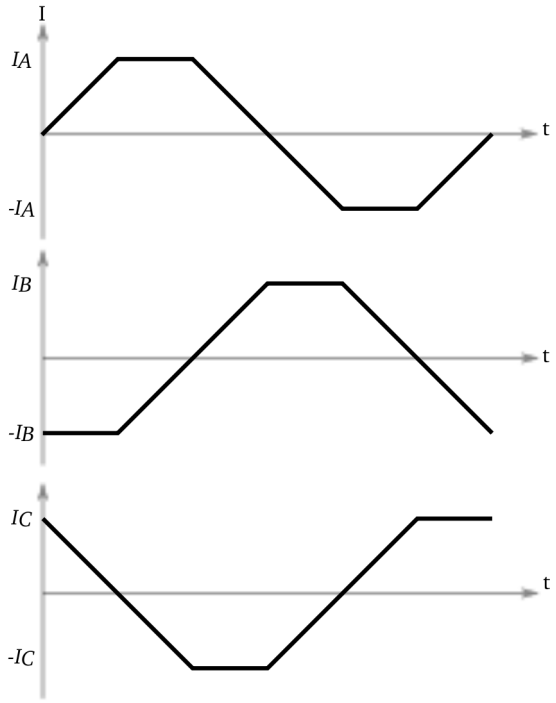

Figura 2.14: Forma de onda de corrente gerada para cada fase (enrolamento).

como ESCs *sensorless*, apesar de usarem uma rede resistiva no bloco "sensores"da Figura 2.15 para coletar a informação de FCEM e fazer a retroalimentação.

Existem ainda implementações de ESCs que ao invés de ter um sinal de entrada no formato PWM, tal sinal é entregue ao seu microcontrolador via barramento I^2C [9]. Assim, pode-se comandar até 127 ESCs com apenas dois fios (SDA e SCL) por um mesmo microcontrolador, devido o esquema de endereçamento presente neste protocolo de comunicação. No entanto, para ESCs com entrada no formato PWM, precisa-se de um pino (sinal de PWM) para cada ESC a ser comandado.

A segunda parte (estágio) do ESC tem como entradas os sinais gerados pela primeira (estágio microcontrolado) — A1, A2, B1, B2, C1 e C2 — onde amplifica-se a corrente das três ondas (fases) através de transistores do tipo MOSFET (dois por enrolamento), que além de possuírem capacidade de corrente suficiente para alimentar os enrolamentos do motor *brushless*, também funcionam como chave para ligar ou desligar o enrolamento segundo o sinal de PWM gerado pelo microcontrolador.

[9] I^2C (*Inter-Integrated Circuit*): padrão (protocolo) de comunicação serial a dois fios (linhas), cujo dado (informação) trafega por uma linha bidirecional, normalmente chamada de SDA (*Serial Data*), a uma taxa de transmissão binária imposta por um sinal de *clock* na outra linha, normalmente chamada de SCL (*Serial Clock*); onde cada dispositivo presente neste barramento, mestre ou escravo, tem um endereço na faixa de 0 a 127.

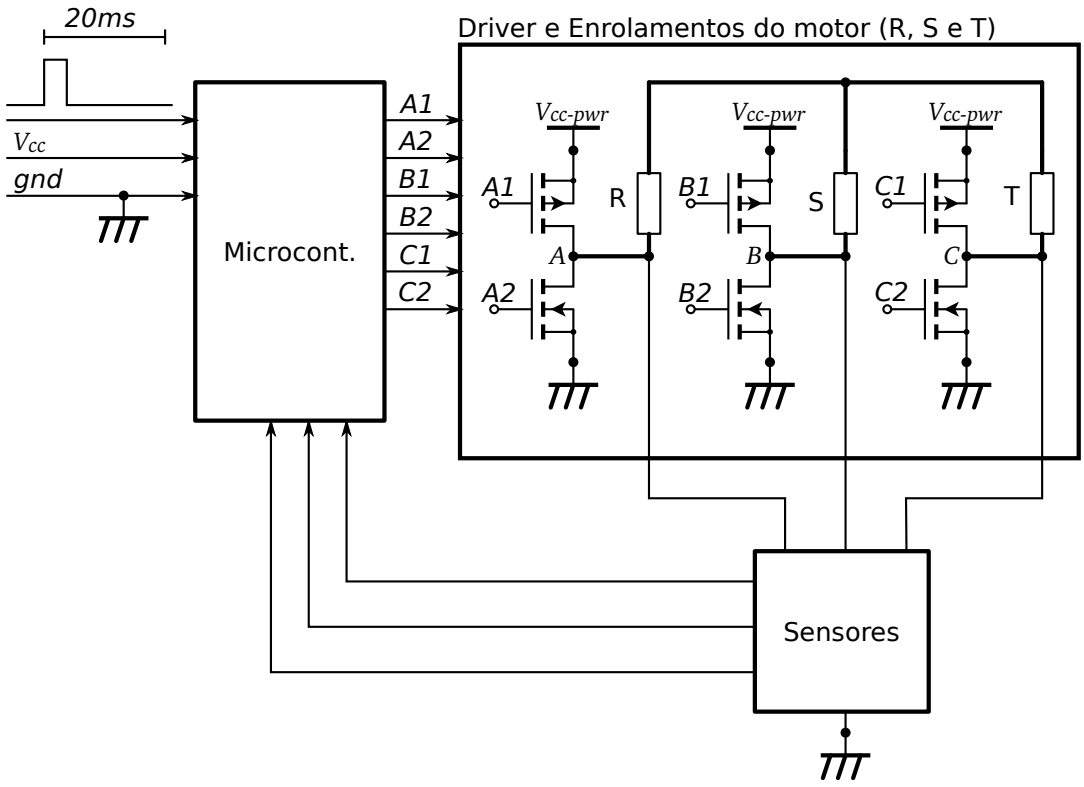

Figura 2.15: Diagrama de blocos simplificado de um ESC.

3 Implementação

A implementação deste trabalho se dividi nas seguintes etapas:

- *hardware*, representado pelo bloco 3 da Figura 3.1;
- *firmware*, compreendido pelo bloco 4 da Figura 3.1;
- e *software*, correspondendo ao bloco 8 da Figura 3.1.

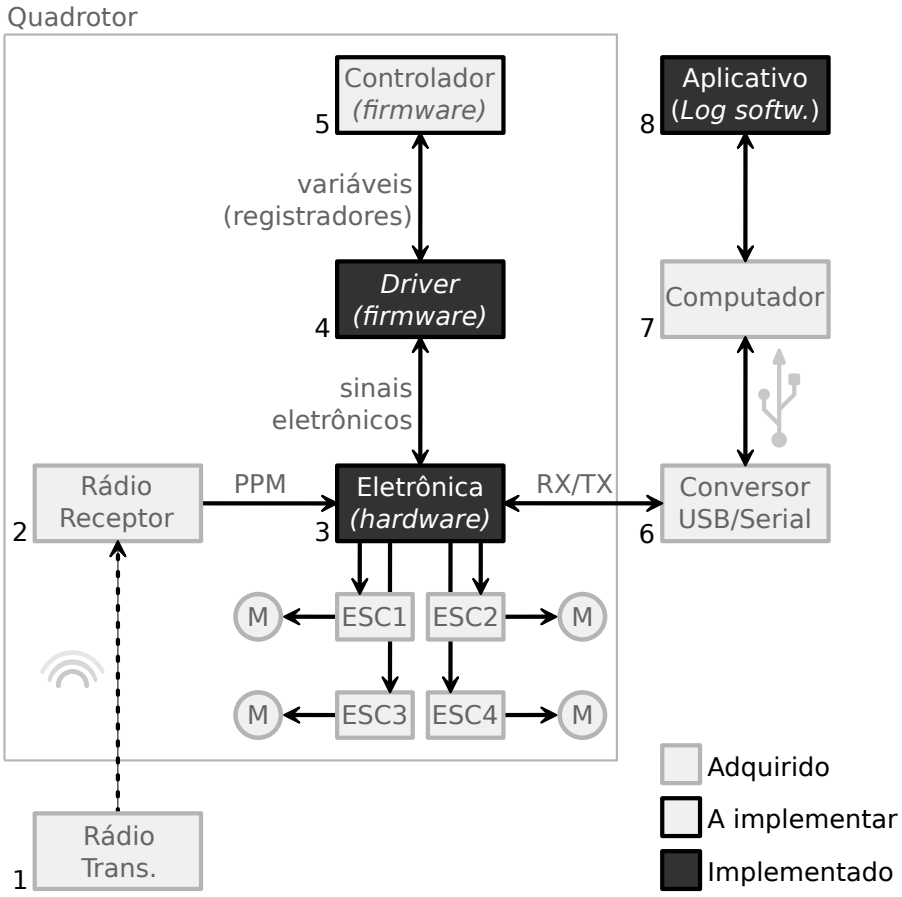

Figura 3.1: Diagrama de blocos da implementação.

O quadrotor como um todo consistem na estrutura mecânica, parte elétrica e eletrônica (*hardware*, bloco 3 da Figura 3.1). Exceto os motores *brushless* com seus respectivos ESCs

e o rádio receptor, que foram adquiridos por se encontrarem prontos e a preço acessível, as demais partes elétricas e eletrônicas foram projetadas e construídas a partir de componentes elementares[1] para atender as exigências do projeto, conforme descrito na seção 3.1.

O *firmware* do bloco 4 na Figura 3.1 foi desenvolvido como um *driver*, isto é, de forma a interfacear o *hardware* com a rotina de controle (controlador) que venha ser implementada, bloco 5 da Figura 3.1. Assim, o usuário que implementa um controlador pode abstrair-se dos detalhes de implementação das blocos abaixo do bloco 5, acessando (leitura/escrita) apenas variáveis (parâmetros como valores de aceleração e giro, já em forma numérica) comuns entre os blocos 4 e 5. Tal *firmware* (*driver*) está divido em rotinas, onde a explicação das principais delas encontram-se na seção 3.2.

Foi implementado um *software* para um computador (estação de trabalho, bloco 7 da Figura 3.1), que através de comunicação serial (por fio), permite conectá-lo ao quadrotor para realizar *log* de dados, tais como coleta dos sinais dos acelerômetros, giroscópios e canais do rádio receptor. O circuito eletrônico utilizado para realizar tal comunicação serial, representado pelo bloco 6 da Figura 3.1, consiste um um conversor de nível TTL/RS232 conectado a um segundo conversor, RS232/USB, que por sua vez, está conectado ao computador via porta USB (*Universal Serial Bus*).

O *sofware* foi escrito na linguagem Python e a sua listagem encontra-se no anexo B, bem como instruções de uso.

3.1 *Hardware*

A parte elétrica do quadrotor contém os seguintes componentes: quatro motores elétricos do tipo *brushless*; uma bateria do tipo Lipo[2] (fonte de tensão em corrente contínua para o quadrotor); e barramentos elétricos — um positivo e outro negativo, onde as cargas (ESCs e a placa microcontrolada) são conectadas conforme a Figura 2.1. Já a parte eletrônica, que também pode ser vista na Figura 2.1, divide-se em: uma placa microcontrolada; uma placa de sensoriamento; quatro controladores eletrônicos de velocidade e um rádio receptor.

[1] Elementares: neste contexto, trata-se de resistores, capacitores, circuitos integrados (reguladores de tensão, microcontrolador, sensores), placa de circuito impresso, bateria, propulsores etc.

[2] Lipo: lítio-polímero. Material usado na construção de baterias com boa relação peso-carga se comparada com as demais disponíveis no mercado.

3.1.1 Motor *brushless* e ESC utilizados

Foram empregados 4 unidades do motor *brushless* E-max 2822 — Figura 3.2 — junto com seus respectivos ESCs E-max, onde as principais características deste motor são listadas a seguir:

- funciona com uma tensão de 11,1V;
- possui potência de 170W;
- possui 12 polos (4 por fase);
- atinge a velocidade máxima de 12000RPM;
- possui um rotor *outrunner* (por fora);
- é capaz de erguer uma massa de até 700g quando encontra-se acoplado em seu eixo um propulsor de 9 polegadas;
- possui uma massa de apenas 39g.

Figura 3.2: Motor *brushless* empregado do fabricante E-max, modelo 2822.

Devido ao princípio de funcionamento deste motor — descrito no apêndice A —, isto é, sem escovas, este elimina comutações mecânicas entre um enrolamento e a fonte de tensão (como em um motor de corrente contínua comum), o que diminui o ruído elétrico gerado pelo motor, além da extinção do centelhamento.

Tais características são interessantes em um quadrotor, pois: pelo fato de o circuito eletrônico de controle estar próximo aos motores (no caso do quadrotor), é importante que estes gerem o mínimo de ruído possível. Ademais, por não haver centelhamento durante a comutação (visto que esta é feita eletronicamente por um ESC, ao invés de mecânica, como nos motores de corrente contínua), o rendimento do motor aumenta, poupando a bateria do quadrotor, conferindo-lhe assim maior autonomia de voo.

Já o controlador eletrônico de velocidade — Figura 3.3 —, possui as seguintes características:

- é do tipo *sensorless*;
- é capaz de drenar uma corrente de até 18A a 11,1V;
- tem como entrada um sinal de PWM com período de 20ms e amplitude de 5V — igual ao sinal de saída para o rádio receptor descrito na seção 2.2 (padrão em RC).

Figura 3.3: ESC E-max empregado.

Estes dois tipos de componentes, motores e ESCs, foram alimentados a partir de uma bateria apropriada de Lipo igual a da Figura 3.4 com as seguintes características:

- possui tensão nominal de 11,1V, isto é, três células de 3,7V ligadas em série, conforme o esquema da Figura 3.5;
- possui capacidade de carga igual 2250mAh;
- pode fornecer 68A de forma contínua;
- tem dimensões igual a 10,5cm x 3,4cm x 2,4cm;
- possui massa de 170g.

Esta bateria possui um terminal diferenciado, conector B segundo o esquema elétrico da Figura 3.5, que permite monitorar a tensão de cada célula durante a sua carga/descarga — feita através dos terminais indicados por C1 e C2 na Figura 3.5 — com a finalidade de fazê-la de forma balanceada, o que prolonga a sua vida útil.

Figura 3.4: Bateria Lipo do fabricante Thunder Power.

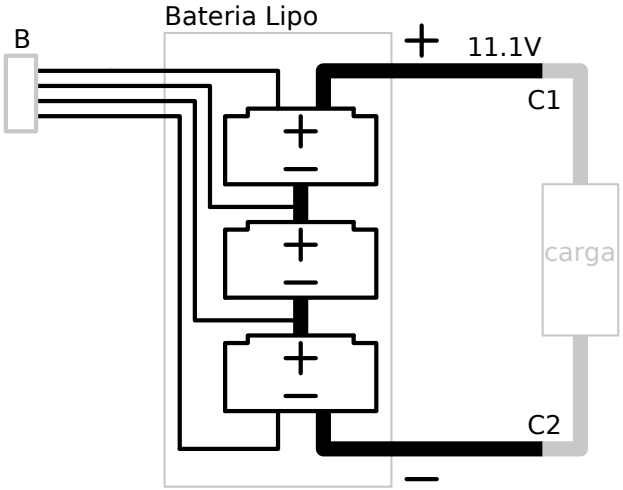

Figura 3.5: Esquema elétrico de uma bateria Lipo conectada a uma carga.

3.1.2 Rádio Receptor

Para que uma manobra seja executada por intervenção de um piloto, um (ou mais) comandos de voo devem ser enviados para o quadrotor — um novo *set point* para o controlador em operação, conforme será explicado na seção 3.2.4. Então, um rádio receptor deve estar a bordo para receber tal comando. Por isso, foi empregado um típico rádio receptor utilizado em RC do fabricante Futaba modelo NER-226X operando com uma portadora de 72,150MHz — Figura 3.6 —, cujo princípio de funcionamento já foi descrito na seção 2.2 de forma genérica.

Este rádio possui cinco canais de saída, sendo que apenas quatro deles — *throttle*, *pitch*, *roll* e *yaw* — são utilizados no quadrotor (capturados pela placa microcontrolada conforme será explicado na seção 3.1.3), além de um canal de entrada que também não é utilizado nesta aplicação. Cada canal de saída, além do sinal no formato PWM, é acompanhado de um pino com V_{cc} (+5V) e outro com *GND* para que se possa alimentar um servo-motor ou um ESC diretamente com 5V.

Como o carrilhão não está disponível originalmente por meio de um conector no rádio Fu-

Figura 3.6: Rádio Futaba NER-226X empregado.

taba, o mesmo foi desmontado, e com auxílio de um osciloscópio foi identificado e extraído — através da soldagem de um fio — tal carrilhão. O ponto de onde foi extraído o carrilhão é o ponto entre as duas principais etapas do circuito de um rádio receptor, a de RF e de demultiplexação, de acordo com a Figura 2.9. Esta adaptação foi realizada com o objetivo de ter duas opções de coleta dos canais: serialmente ou paralelamente; onde estas serão explicadas na seção a seguir (3.1.3).

3.1.3 Placa Microcontroladora

A placa principal, ou microncotrolada, deve ser responsável por:

- capturar os sinais oriundos do sensores: acelerômetros (X, Y e Z) e giroscópios (X, Y e Z);
- capturar os canais (comandos de voo *throttle*, *pitch*, *roll* e *yaw*) do rádio receptor;
- rodar algum algoritmo de controle para estabilização de voo (controlador do bloco 5 na Figura 3.1);
- gerar 4 sinais de PWM para os ESCs baseados na ação de controle tomada por um mais controladores do bloco 5 na Figura 3.1;
- transferir/receber informações para/de um computador em terra via porta de comunicação serial, isto é, realizar *log* de dados e configuração de parâmetros do quadrotor e/ou de um controlador que venha ser implementado.

A partir destas necessidades, como componente principal da placa, foi especificado o dispositivo Atmega168-20PU (microcontrolador) [Atmel 2007] por possuir além de uma CPU para rodar o algoritmo de um controlador, os seguintes módulos no mesmo encapsulamento:

- um conversor A/D de 10 *bits*, utilizado para coletar os sinais analógicos provenientes dos sensores;
- um temporizador de 16 *bits* (*timer1*) usado na geração do período de 20ms para o sinais de PWM utilizados como entrada para os ESCs;
- dois temporizadores de 8 *bits* (*timer0* e *timer2*). Cada um deles, junto com seus respectivos registradores de comparação, A e B, são responsáveis por determinar por quanto tempo dois dos quatro sinais de PWM[3] permanecerão em nível lógico alto;
- pino de I/O sensível à transição (borda de subida ou descida) quando configurado como entrada digital para capturar o carrilhão de canais, via interrupção de *hardware*;
- uma USART para fazer o *log* de dados, além de outros dois módulos de comunicação serial, um no padrão I2C e outro no padrão SPI (*Serial Periferal Interface*), que podem ser utilizados para conectar mais dispositivos à placa principal.

Além das característica listadas, este microcontrolador possui uma CPU RISC de 8 *bits* e arquitetura Harvard[4]; 16kB de memória de programa (*flash*[5]), 1kB de memória SRAM[6]; 512B de memória EEPROM[7] e pode operar com uma frequência de *clock* de até 20MHz.

Então, para este microcontrolador, foi confeccionada uma placa de circuito impresso (PCI), cuja fotografia encontra-se na Figura 3.7, segundo o esquema[8] ilustrado na Figura 3.8 com o seu respectivo *layout* no anexo C.

O projeto da PCI microcontroladora — placa principal — foi feito de forma que outra placa, neste caso a de sensoreamento, possa ser conectada por cima. Assim, se for necessário substituir o sensoreamento do quadrotor por algum outro ou trocar algum sensor, a placa principal permanece sem qualquer alteração e/ou adaptação.

Para isto, dois conectores — *pinheaders* com 10 pinos do tipo fêmea (vide *layout* da PCI na Figura C.1 do anexo C) —, CN9 e CN10, além da tensão da bateira (11,1V, desprezando a queda

[3] A forma como os temporizadores trabalham em conjunto para gerar um dado sinal de PWM será explicada na seção 3.2.5.

[4] Possui barramentos de dados e instruções diferenciados, diferentemente da arquitetura de Von Neumann que possui somente um barramento para as duas finalidades.

[5] *Flash*: memória que não perde os seus dados quando é desligada e normalmente é utilizada para armazenar código de instruções de um programa podendo ser apagada em blocos.

[6] SRAM: memória RAM do tipo estática (*static*), ou seja, não precisa de *refresh* para manter os seus dados assim como as dinâmicas — baseadas em capacitores que podem ser descarragados ao longo do tempo. Normalmente este tipo de memória é utilizada para armazenar variáveis de um programa em plena execução.

[7] EEPROM: memória utilizada para salvar permanentemente parâmetros de configuração de um programa/dispositivo, isto é, se esta for desligada, os dados contidos nesta não serão perdidos.

[8] Os valores dos componentes contidos no esquema da placa principal encontram-se na Tabela C.1 do anexo C.

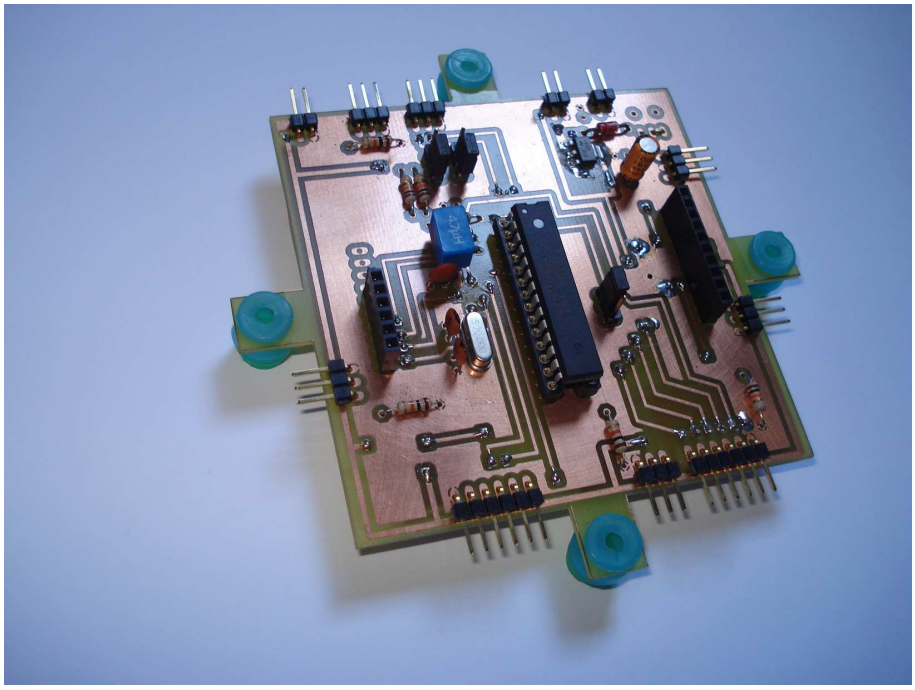

Figura 3.7: Placa dupla face confeccionada para o microcontrolador Atmega168.

do diodo de proteção D1[9]), disponibilizam 6 canais analógicos e dois barramentos para comunicação serial, um I2C e outro SPI, visto que já existem sensores que possuem internamente um conversor A/D e entregam a o resultado da conversão em um destes padrões de comunicação serial.

No caso de utilizar o barramento I2C, dois dos canais analógicos — AD4 e AD5 na Figura 3.8 — não poderão ser utilizados, pois os pinos do microcontrolador onde estes são conectados, são compartilhados, respectivamente, com os sinais SDA (*serial data* — bidirecional) e SCL (*serial clock*) do barramento I2C. No entanto, uma vez optado pelo barramento I2C através dos conectores CN3 e CN4 (colocando um *jumper* nos pinos 1-2 destes conectores), até 128 dispositivos (entre eles acelerômetros, giroscópios, altímetro, bússola e GPS) poderão estar conectados ao microcontrolador, visto que neste padrão de comunicação serial cada dispositivo conectado no barramento possui um endereço dentro da faixa de 0 a 127.

Quanto à captura dos canais do rádio receptor, através do conector CN12 da Figura 3.8, quatro pinos de I/O do microcontrolador estão disponíveis, por onde pode ser feita a captura de cada canal do rádio receptor — *throttle*, *pitch*, *roll* e *yaw* — separadamente por *polling*[10], também é possível fazê-lo serialmente, através do pino INT0 (PD2) ou INT1 (PD3). Estes dois

[9]D1: diodo usado para proteger o circuito da placa principal caso a bateria do quadrotor for conectada com polaridade invertida.

[10]Técnica utilizada para monitorar uma pino de entrada de sinal digital ou um *bit* de um registrador onde a CPU de tempo em tempo verifica o estado (nível) deste pino.

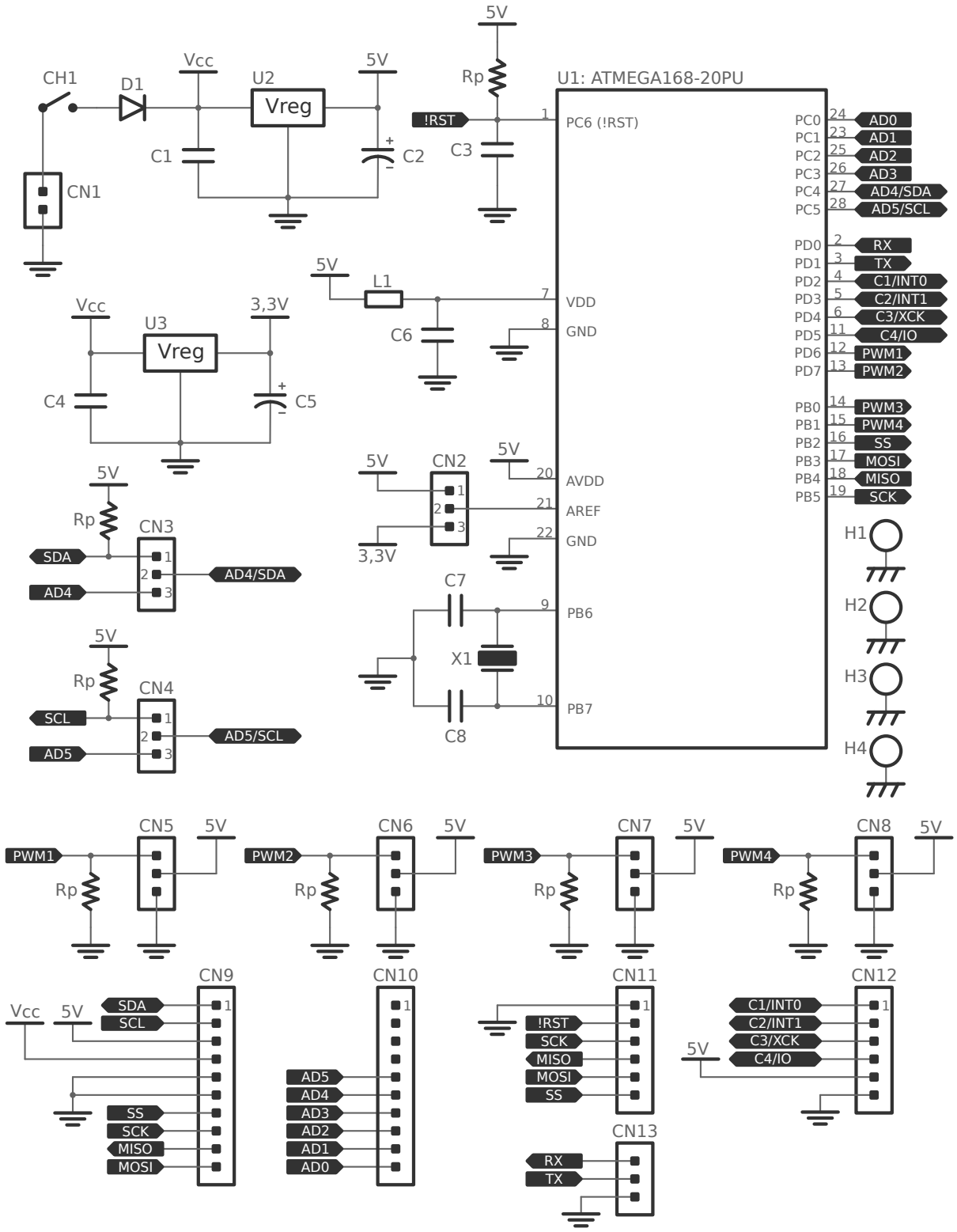

Figura 3.8: Esquema da placa microcontrolada.

pinos são sensíveis à transição de estado (borda de subida ou descida), permitindo que os canais sejam capturados serialmente se em um deles for injetado o sinal oriundo da saída da etapa de RF do rádio receptor da seção 3.1.2, isto é, o carrilhão.

Para o microcontrolador funcionar na maior frequência de *clock* definida pelo fabricante (20MHz — frequência utilizada neste trabalho), ele deve ser alimentado com uma tensão entre 4,5 e 5,5V. Por isto, e por conta dos ESCs[11], foi utilizado um regulador de tensão 7805 como fonte de tensão na placa microcontrolada. Todavia, antes da tensão de 5V chegar no pino do microcontrolador, esta passa por um filtro LC — componentes L1 e C6 no esquema da Figura 3.8 — com a finalidade de filtrar eventuais ruídos que possam chegar pela linha de alimentação do microcontrolador.

Através dos conectores CN5, CN6, CN7 e CN8 da Figura 3.8, os quatro sinais de PWM são entregue aos ESCs junto com a alimentação de 5V para a etapa microcontrolada destes últimos. Cada um destes conectores possui um resistor de *pull-down* para evitar que a entrada de ESC flutue e dispare o seu respectivo motor. Isto pode ocorrer caso o pino do microcontrolador utilizado para esta finalidade seja configurado como entrada equivocadamente durante o desenvolvimento de um *firmware* ou durante a gravação[12] do microcontrolador.

Como as saídas analógicas dos sensores utilizados neste trabalho excursionam dentro da faixa 0 e 3.3V, a tensão de referência do conversor A/D do microcontrolador pode ser ligada a um regulador de tensão de 3.3V (U3) através do conector CN2 da Figura 3.8, quando um *jumper* é colocado na posição 2-3 deste conector.

Através do conector CN2 da Figura 3.8, é possível escolher 3,3V ou 5V como tensão de referência para o conversor A/D do microcontrolador. Para os sensores da seção 3.1.4 (com faixa dinâmica menor que 3,3V), é interessante que o *jumper* permaneça na posição 2-3, assim, o passo de conversão do conversor A/D diminui (se comparado com a segunda opção), aumentando a precisão da conversão (seja ela feita em 8 ou 10 *bits*).

Para a realização de *logs* de dados, foram empregados os pinos RX, TX da USART0 do microcontrolador através do conector CN13 da Figura 3.8 a fim de realizar comunicação serial assíncrona. Assim, os dados podem ser transferidos para uma estação de trabalho para análise em tempo real ou posteriormente. Através da USART0, também é possível configurar parâmetros do quadrotor (constantes gravadas na memória EEPROM), tais como os utilizados na sintonia de um controlador automático, sem ter que regravar todo *firmware* na memória de programa do microcontrolador.

[11] Os ESCs precisam ter duas fontes de alimentação: uma de 5V para alimentar sua etapa microcontrolada e outra para etapa de potência (para acionar os motores), que é conectada diretamente à bateria.

[12] Durante a gravação do microcontrolador, todos os pinos deste que não são utilizados na comunicação com o computador que o grava, por precaução são configurados como entrada para evitar curto-circuito.

Visto que a UART0 pode operar no modo SPI (usando os pinos RX como MISO, TX como MOSI e XCK[13] como SCK), existe ainda a possibilidade de conectar um módulo conversor SPI/USB[14] junto com um *pendriver* para fazer *log* de dados sem utilizar fios entre o quadrotor e a estação de trabalho, contudo, sem a possibilidade de analisá-lo em tempo real.

Por fim, um conector CN11 da Figura 3.8 diponibiliza os pinos da porta serial SPI para que o microcontrolador possa ser programado sem precisar ser removido da placa princiapal.

3.1.4 Placa de Sensoriamento

A PCI de sensoriamento, Figura 3.9, com esquema ilustrado[15] na Figura 3.10, é constituída basicamente por:

- um dispositivo acelerômetro triaxial — X, Y e Z — do tipo MEMS analógicos, com sensibilidade[16] de até 800mV/g, todos contidos em um único enacapsulamento — dispositivo MEMS MMA7260Q (X-Y-Z) — para detecção de inclinação.
- três giroscópios MEMS com sensibilidade de 100 °/s. Dois deles, X e Y, contidos em um encapsulamento — dispositivo biaxial LPR510AL (X-Y) —, para detecção de arfagem (giro em torno do eixo X) e rolagem (giro em torno do eixo Y); e outro, dispositivo LY510LH (Z), para detecção de guinada (giro em torno do eixo Z).

Os acelerômetros — U3 na Figura 3.10 — podem trabalhar com acelerações máximas (dinâmica mais estática) segundo a configuração feita através dos pinos de entrada digital *g-sel1* (*g-select 1*) e *g-sel2* (*g-select 2*), onde as suas possíveis combinações com suas respectivas sensibilidades estão reunidas na Tabela 3.1.

g-sel1	g-sel2	Possíveis aceleraçõs máximas	Sensibilidade
0	0	1,5g	800mV/g
0	1	2,0g	600mV/g
1	0	4,0g	300mV/g
1	1	6,0g	200mV/g

Tabela 3.1: Possíveis acelerações máximas dos dispositivo MMA7260Q.

[13] O pino C3/XCK só pode ser usado como XCK — USART0 operando no modo SPI — caso este pino não esteja sendo usado para monitorar um canal do rádio.

[14] Módulo normalmente baseado no *chip* FTDI2232 que permite a criação de diretórios e arquivos em um *pendriver*, podendo neste escrever ou ler através de comando enviados pela interface SPI.

[15] Os valores dos componentes encontram-se na Tabela C.2 do anexo C.

[16] A sensibilidade do dispositivo MMA7260Q pode ser escolhida conforme a Tabela 3.1.

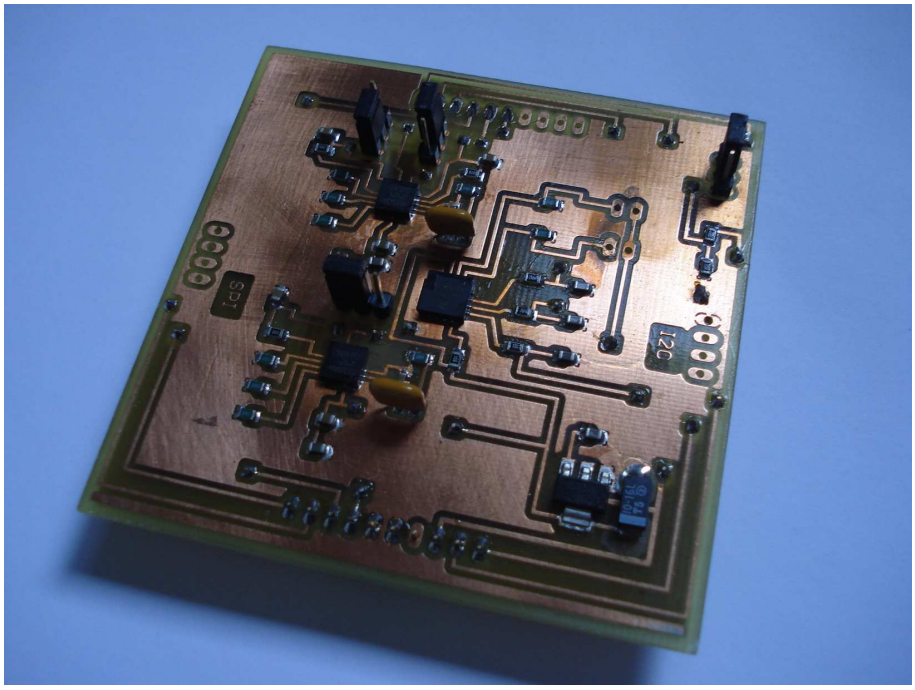

Figura 3.9: Placa dupla face confeccionada para os sensores.

A seleção da aceleração máxima (fundo de escala) pode ser feita através dos conectores CN1 e CN2 na placa de sensoriamento conectando (1 — 3,3V) ou não (0 — GND, via resistor de *pull-down*) um *jumper*. Como o intuito, neste trabalho, é detectar a projeção da gravidade sobre os eixos X e Y, nunca haverá acelerações maiores que 1g (despresando a aceleração dinâmica), portanto a configuração correspondente à primeira linha da Tabela 3.1 é o suficiente.

Em especial, os dispositivos LPR510AL e LY510LH — identificados por U1 e U2 na Figura 3.10 —, possuem um amplificador embutido para cada eixo com um ganho de 4 vezes, cuja entrada pode ter como sinal a saída filtrada (rotulada por 1Gn, onde *n* pode ser X, Y ou Z na Figura 3.10) do seu respectivo giroscópio. Conforme a aplicação, resolução da conversão A/D (número de *bits*) e controlador utilizado para estabilizar o quadrotor, tal amplicaficação pode ser conveniente.

Cada acelerômetro e giroscópio possuem conectados em suas saídas um filtro RC passa-baixas, recomentado pelo fabricante, para atenuar ruídos, implementado com os componentes da Tabela 3.2, junto com sua respectiva frequência de corte. No caso dos acelerômetros, filtra-se não só ruídos como também variações bruscas de aceleração dinâmica, o que é desejável na presente aplicação, pois o uso de tal dispositivo tem a finalidade de detectar inclinação (*tilt*), isto é, aceleração estática (aceleração da gravidade).

Como normalmente o acelerômetro Z não é utilizado na estabalização de um quadrotor, sendo a estabilização no eixo Z realizada a partir de um altímetro (que pode ser implementado

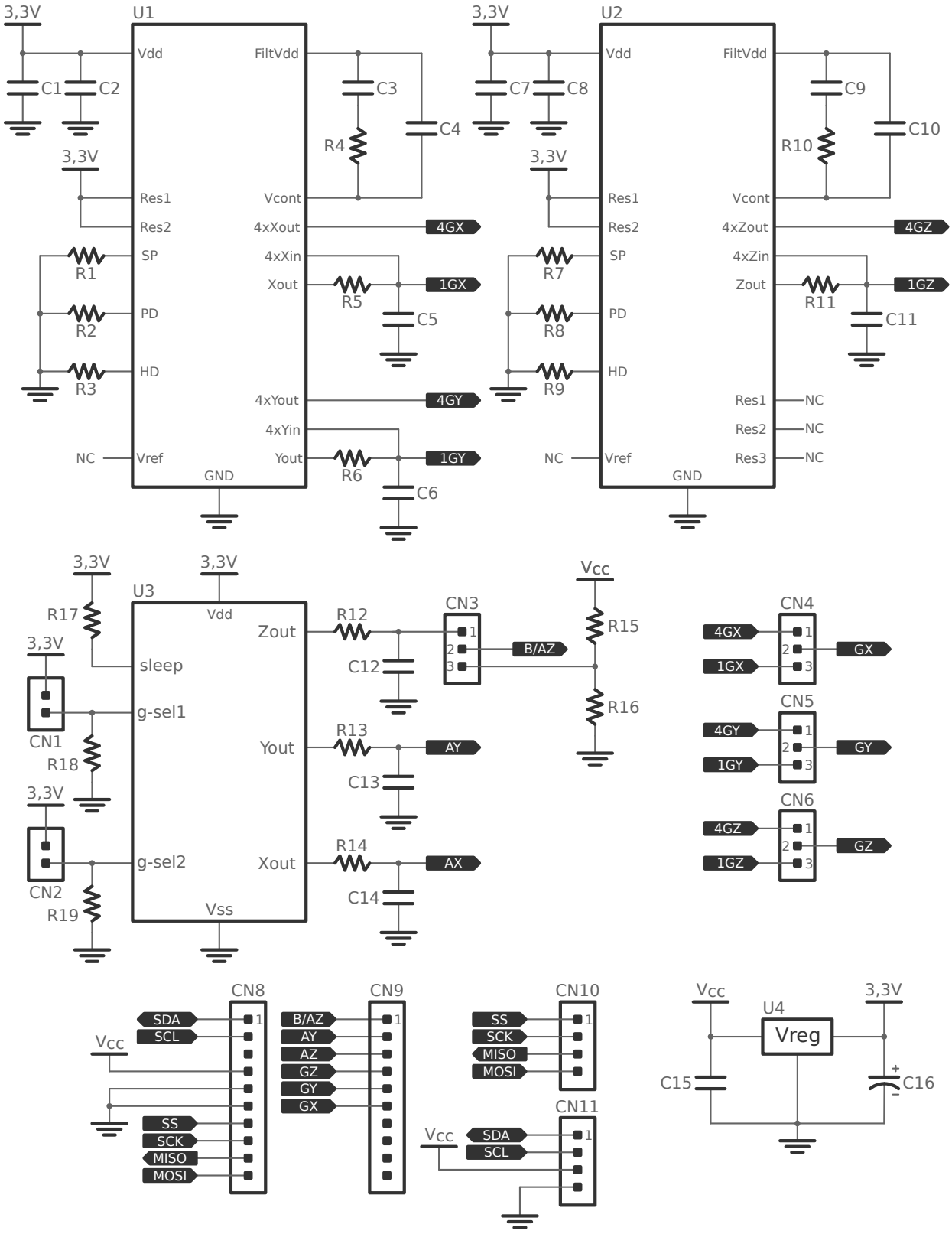

Figura 3.10: Esquema da placa de sensoreamento. U1 e U2 são os giroscópios LPR510AL e LY510LH, respectivamente U3 o acelerômtro triaxial MMA7260Q.

com um medidor de pressão atmosférica), a entrada analógica do microcontrolador disponibilizada para tal acelerômetro pode ser comutada no conector CN3 da Figura 3.10 por meio de um

Sensor	**R [kΩ]**	**C [μF]**	f_c **[Hz]**
Acel. X	1	0.1	16
Acel. Y	1	0.1	16
Acele Z	1	0.1	16
Giro. X	33	2.2	140
Giro. Y	33	2.2	140
Giro. Z	33	2.2	140

Tabela 3.2: Configuração dos filtros passa-baixas dos sensores.

jumper colocado na posição 2-3, a fim de monitorar a tensão da bateria através de um divisor de tensão.

Assim, se a tensão da bateria se aproximar de um limite de segurança (que depende do número de células da bateria, ESC e motor utilizado), uma rotina específica pode ser implementada no microcontrolador para sinalizar tal condição através de um sinal luminoso (por meio de LED conectado a um pino de I/O do conector CN12 da placa principal — Figura 3.8 —, caso não esteja sendo usado para ler os canais do rádio por *polling*), sonoro (por meio de *buzzer*) ou até mesmo realizar um pouso imediato automático como medida de segurança.

Ademais, esta placa possui um regulador de tensão para 3,3V (tensão utilizada para alimentação dos sensores), U4 na Figura 3.10, e dois conectores: CN11 na Figura 3.10, que dá acesso ao barramento I2C, Vcc (tensão da bateria) e GND; e CN10 na Figura 3.10, com o barramento SPI. Isto permite que uma terceira placa possa ser conectada por cima da placa de sensoriamento adicionando funcionalidades através destes barramentos.

3.2 *Firmware*

Para implementar o bloco 4 da Figura 3.1, foi desenvolvido um *firmware* onde as suas principais rotinas são responsáveis por:

- gerenciar a conversão A/D dos sinais dos sensores, via interrupção de *hardware*;
- capturar os canais de rádio receptor via interrupção de *hardware*;
- gerar os sinais de PWM para os ESCs baseado na resposta de um ou mais controladores e nos temporizadores — *timer0*, *timer1* e *timer2*, que também geram interrupção de *hardware*;
- e, realizar *log* via comunicação serial caso seja requisitada;

onde o nome de cada rotina (função) que compõe o *firmware* estão reunidas na Tabela 3.3, bem com os seu parâmetros de entrada e o que retorna.

Nome da função	**Parâmetros**	**Sinopse**	**Retorna**
Config	—	configura o microcontrolador (periféricos)	—
GetTime	—	concatena os dois registradores de 8 *bits* do *timer1* para gerar um valor de 16 *bits*	valor do *Timer1* em 16 *bits* para um dado instante
CalcT	instante t_1, instante t_2	calcula a diferença de tempo entre dois instantes t_1 e t_2	$T = t_2 - t_1$
ISR	vetor de interrupção	rotina de serviço de int. de *hardware*	—
StartADConversions	—	inicia uma nova conversão A/D	—
EndOfTransmission	—	para transmissão da caracteres no módulo UART	—
ISR(USART_TX_vect)	vetor de interrupção	rotina de serviço de int. para o evento de fim de transmissão de um caractere	—
StartOfTransmission	—	incia a transmissão de caracteres via UART	—
Filter	ponteiro para o somatório, ponteiro para o *buffer* do amostras, amostra atual	filtro de média	valor filtrado
main	—	rotina principal	—

Tabela 3.3: Rotinas que compõe o *firmware*.

3.2.1 Captura dos Sinais Analógicos

Para realizar a captura dos sinais analógicos — sinais dos acelerômetros e giroscópios — foram implementadas duas funções: uma que inicia o conversor A/D configurando-o para gerar interrupções de *hardware* no fim de cada conversão e selecionar o primeiro canal (AD0), sendo esta executada de forma síncrona na rotina principal; e outra, que por se tratar de uma rotina de interrupção de *hardware* (depende do tempo de conversão utilizado), acontece de forma assíncrona em relação ao programa principal. Esta rotina é responsável por salvar o valor da última conversão na devida variável e reconfigurar/reiniciar o conversor para uma nova conversão em um novo canal analógico, conforme o fluxograma da Figura 3.11, que corresponde à função ISR(ADC_vect) da Tabela 3.3.

Considerando o acelerômetro da seção 3.1.4 com uma sensibilidade de 800mV/g, ao inclinar o quadrotor com um ângulo α, Figura 3.12, a aceleração da gravidade será projetada sobre o eixo X conforme a equação 3.1, portanto a saída, em tensão, do acelerômetro terá um valor correspondente à equação 3.2. A Tabela 3.4 ilustra os valores de saída do acelerômetro para diversos valores de α, desconsiderando o deslocamento (*offset*[17]) de 1,65V pertinente a este dispositivo.

$$g_x = g.sen(\alpha); \tag{3.1}$$

$$v(\alpha) = 0,8.sen(\alpha); \tag{3.2}$$

α	**sen(α)**	**v(α) [V]**
1°	0,0174	0,0139
5°	0,0871	0,0697
10°	0,1736	0,1398
15°	0,2588	0,2070
30°	0,5000	0,4000
45°	0,7071	0,5656

Tabela 3.4: Relação entre o ângulo de inclinação e tensão de saída do acelerômetro.

Uma vez que o passo *P* do conversor A/D é dado pela equação 3.3 [Ailson Rosetti de Almeida 2004]:

$$P = \frac{V_{ref}}{2^n - 1}; \tag{3.3}$$

[17] Mediante variações de aceleração positiva e negativa, este *offset* permite que o sinal excursione dentro de uma faixa positiva de valores.

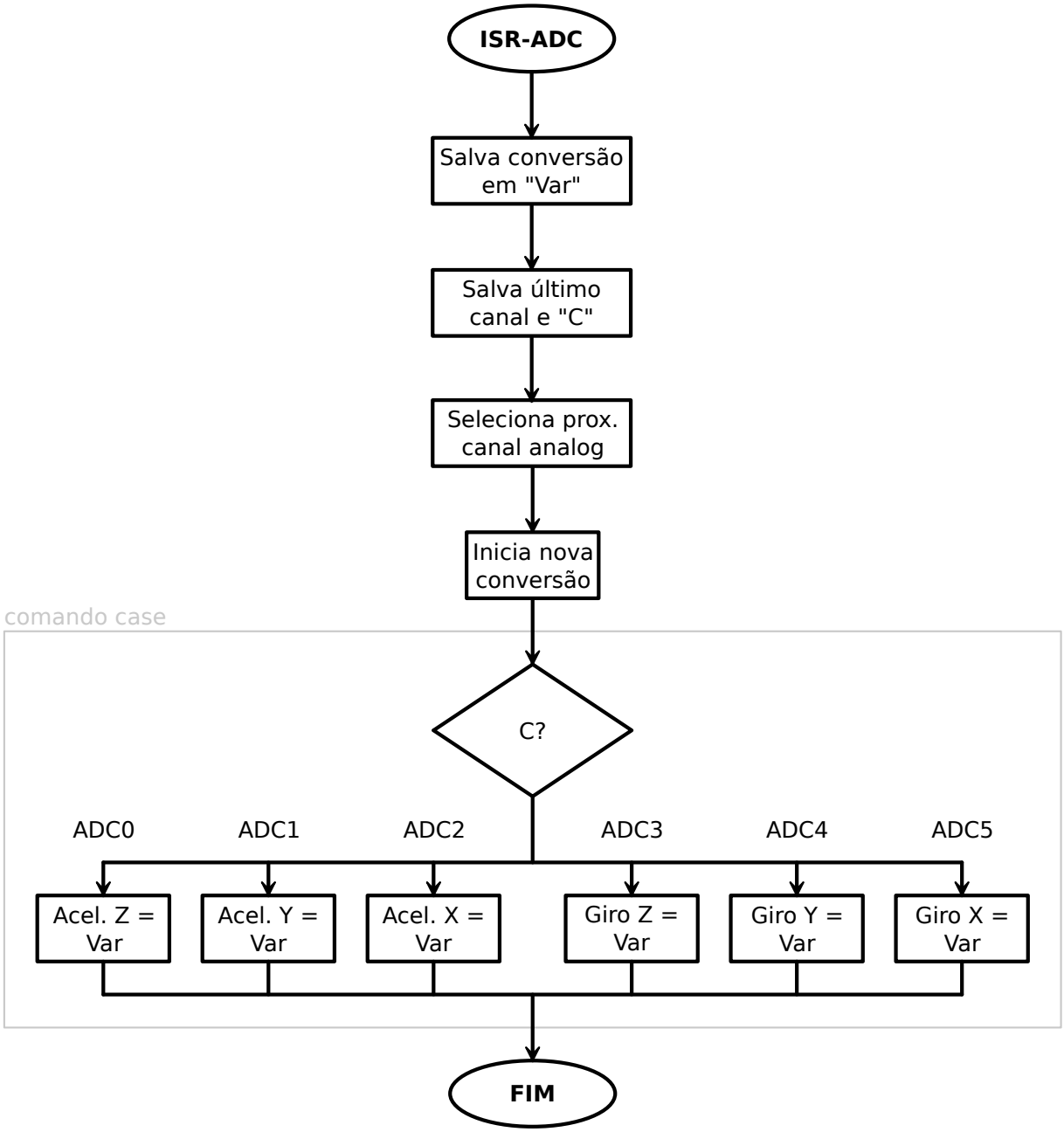

Figura 3.11: Fluxograma para o rotina de captura dos sinais analógicos.

Onde V_{ref} é igual a 3,3V, este conversor terá uma resolução de 0,0129V (12,9mV), que é menor que a tensão correspondente a 1° para o conversor trabalhando com tal sensibilidade (800mV/g). Caso o conversor opere com conversão em 10 *bits*, este passo será fracionado por 4, aumentando a resolução.

Quanto à conversão A/D de um sinal procedente de um giroscópio da seção 3.1.4, se esta for feita em 8 *bits* (com um passo de 12,9mV), é possível detectar uma velocidade angular de 5,6°/s[18], dado que tal dispositivo analógico fornece uma tensão de 2,5mV/°/s com um *offset*

[18]A título de ilustração, essa velocidade é um pouco menor que a de um ponteiro ($360°/60s = 6°/s$) de um

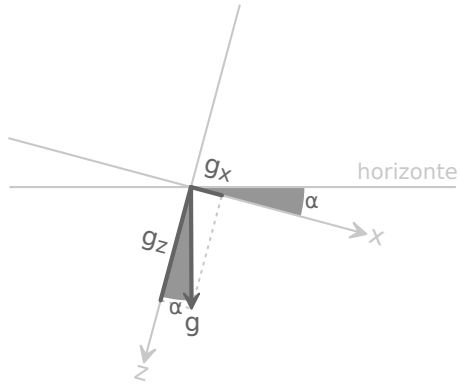

Figura 3.12: Projeção de *g* sobre o eixo X quando o quadroto inclina.

de 1,23V. Ou então, se a saída amplificada do giroscópio (x4) for utilizada, uma velocidade de até 1,4°/s poderá ser detectada. E se ainda for necessário realizar uma conversão com maior precisão, isto é, em 10 *bits* o presente conversor, poderá ser detectada uma velocidade de até 0,35°/s.

3.2.2 Captura dos Canais do Rádio Receptor

Com o *hardware* apresentado na seção 3.1.3, é possível implementar a rotinas para capturar os canais do rádio receptor de duas formas: fazendo a leitura dos canais paralelamente, isto é, através de 4 pinos de I/O (um para cada canal do rádio) fazendo *polling*[19], ou então, serialmente através do pino INT0 gerando interrupção (como elucidado na seção 3.1.3). Por levar menos tempo de processamento, tal rotina foi implementada a partir desta segunda opção.

Para capturar os canais serialmente através do pino INT0, este foi conectado à saída serial criada no rádio Futaba (seção 3.1.2) e configurado para gerar interrupção de *hardware* quando um transição positiva (borda de subida) do sinal digital (carrilhão) acontecer.

A cada borda de subida, Figura 2.8, que corresponde ao termino de um canal e início do seguinte, o valor de *timer1* é capturado dentro da rotina de interrupção. A diferença de tempo entre o instante de uma borda de subida e a anterior corresponde ao tempo de duração de um canal, isto é, o tempo que este deve permanecer em nível lógico alto na saída do rádio. Se este tempo for menor que o tempo mínimo (1ms) de duração de um canal, este é considerado como ruído, portanto, não é atribuído a nenhum canal. E se for maior que o tempo máximo de duração de canal (2ms), este é considereado como fim do carrilhão (sincronismo). O fluxograma da Figura 3.13 ilustra a implementação do rotina para a coleta dos canais serialmente, que

relógio com movimento contínuo.

[19] *Polling*: técnica de leitura de um sinal digital de entrada onde se checa de tempo em tempo o nível lógico em que este se encontra.

corresponde à função ISR(INT0_vect) da Tabela 3.3.

Como o rádio transmissor utilizado envia 6 canais, dois deles são ignorados, onde estes correspondem aos canais 5 e 6 identificado respectivamente por res.(5) e res.(6) no fluxograma da Figura 3.13.

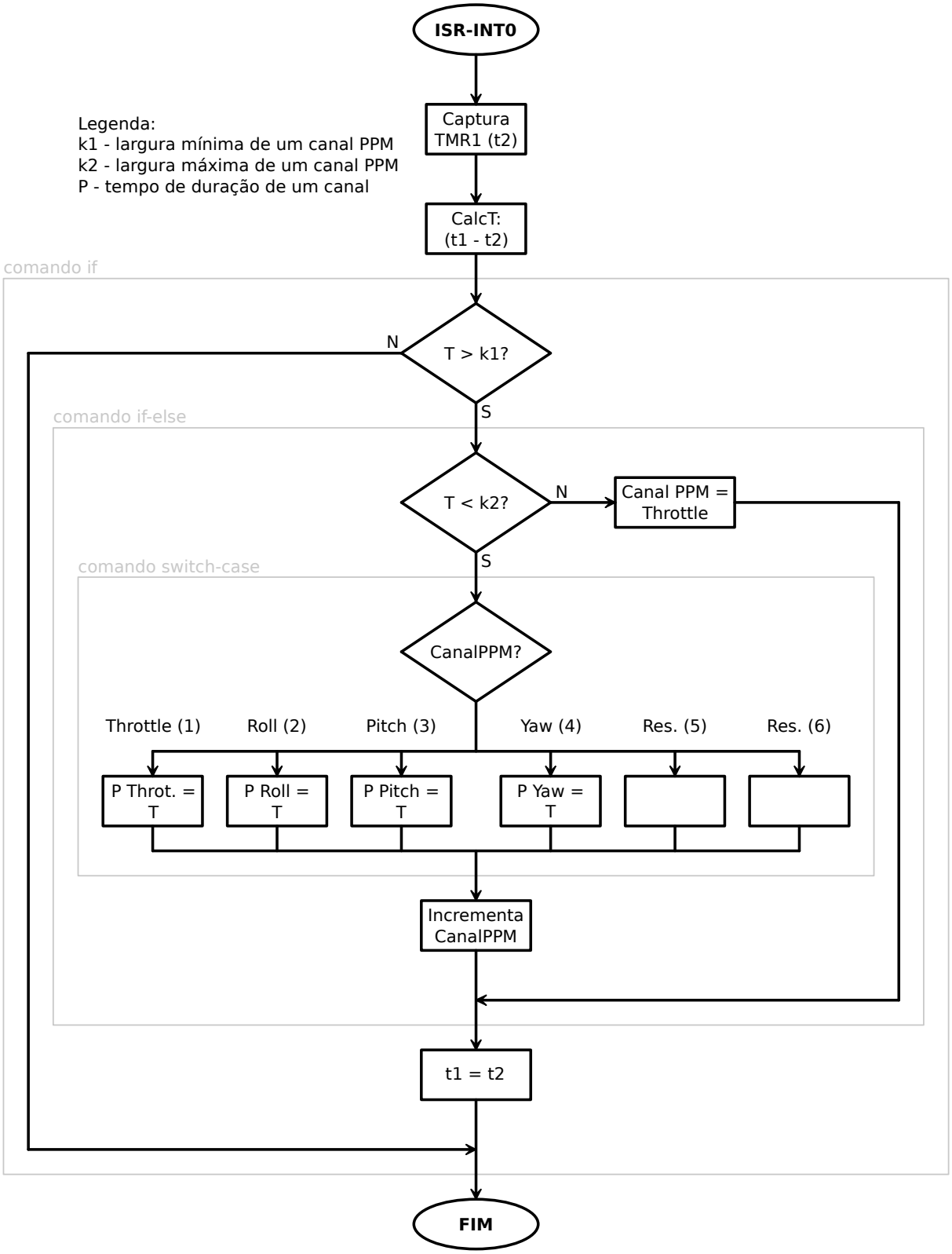

Figura 3.13: Fluxograma para a rotina de captura dos sinais de PPM do rádio receptor por interrupção.

3.2.3 Filtragem Digital

Nem todos os paradigmas de controle lidam bem com sinais de entrada ruidosos, sendo uma exceção os controladores baseados no paradigma nebuloso [Mendel, J. M. 1995]. Portanto, para o *firmware* dar suporte à implementação de controladores baseado em diversos paradigmas, um filtro digital de média (filtro FIR [Antoniou, A. 2006]) foi elaborado para atenuar ruídos de alta frequência oriundos dos próprios sensores ou externos a estes, quando necessário .

Um filtro de média de ordem m (com $m = n+1$), consiste em calcular a média $v_m(t)$ de m valores amostrados conforme a equação 3.4:

$$v_m(t) = \frac{v(t) + v(t-1) + v(t-2) + ... + v(t-n)}{n+1}, \tag{3.4}$$

onde, $v(t)$ é o último valor amostrado (mais recente) acrescentado ao *buffer* e $v(t-n)$ o mais antigo.

Logo, para a implementação via programa, deve-se reservar um *array* (*buffer* circular de memória) para armazenar as m amostras de um dado sinal a ser filtrado. Conforme o fluxograma da Figura 3.14 (correspondente à função Filter da Tabela 3.3), ao invés de realizar a soma de todos os valores contidos no *buffer*, isto é, computar o somatório de m termos (um *loop* de m iterações) toda vez que uma nova amostra sobrescrever a mais antiga no *buffer*, optou-se por subtrair a amostra mais antiga e adicionar o mais nova a fim de computar o numerador da equação 3.4, que tem o mesmo efeito.

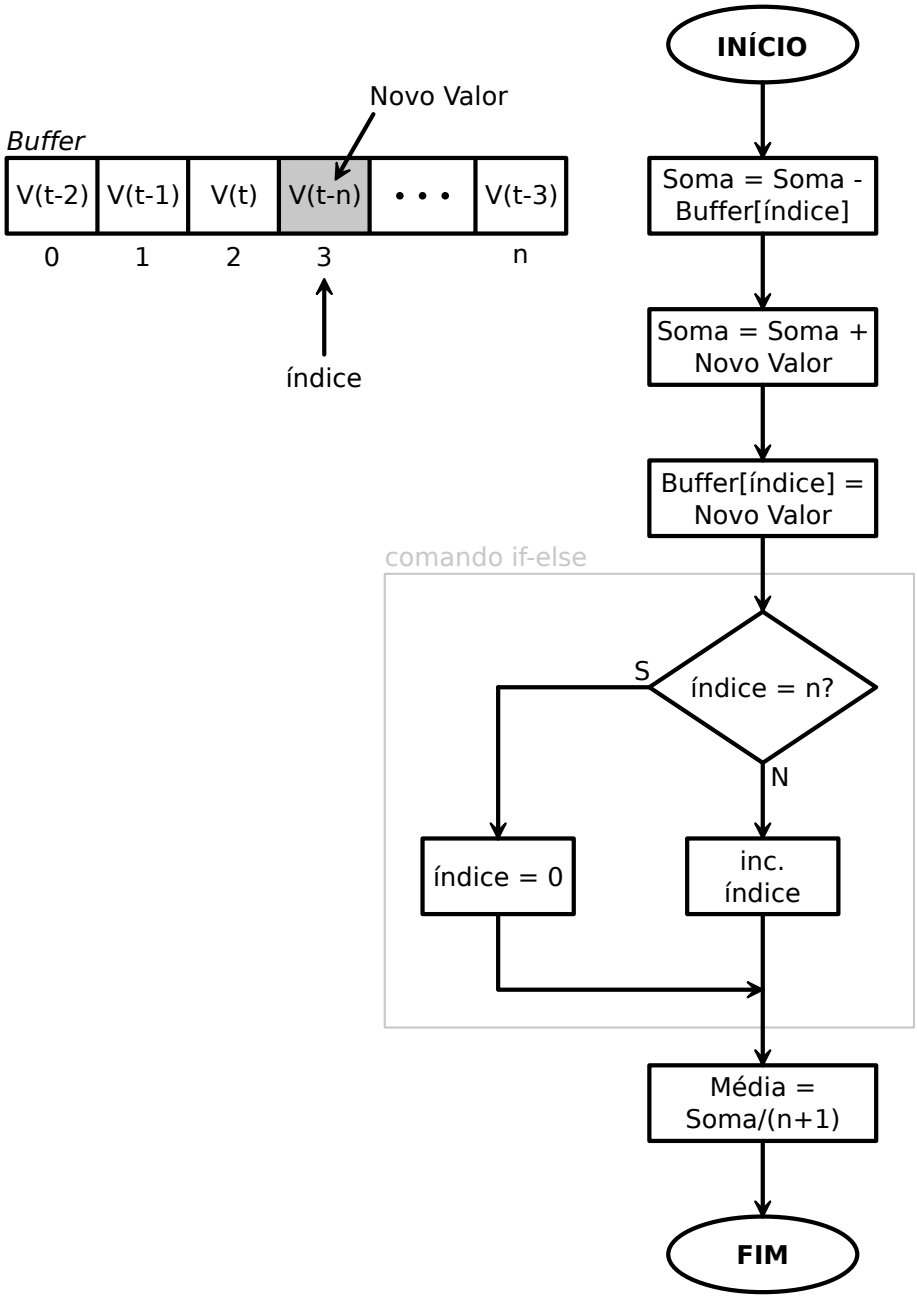

Figura 3.14: Fluxograma para a rotina referente ao filtro digital de média.

3.2.4 Programa Principal

Para a estabilização do quadrotor via controle automático (desconsiderando um controlador de altitude), pode-se dividir tal controle em três partes: um controlador para o eixo X, controlando a arfagem; um controlador para o eixo Y, controlando a rolagem; e um controlador para o eixo Z, controlando a guinada.

Uma vez capturado os quatro canais do rádio conforme, descrito na seção 3.2.2, os canais referentes ao comando *pitch*, *roll* e *yaw* podem ser utilizados como referência, isto é, *set point*

para cada controlador (X, Y e Z) conforme a Tabela 3.5. Em outras palavras, o valor de um canal corresponde ao ângulo que o quadrotor deve permancer em relação ao horizonte no caso dos eixos X e Y, ou com que velocidade angular deve permanecer em relação ao eixo Z.

Comando	***Set Point* para:**
Pitch	Eixo X
Roll	Eixo Y
Yaw	Eixo Z

Tabela 3.5: Comandos e respectivos *set points*.

Assim, cada controlador deve gerar uma ação de controle, que quando combinadas segundo o princípio de funcionamento do quadrotor, ou seja, conforme as equações 3.5, 3.6, 3.7 e 3.8, cada uma dessas corresponderá ao tempo que o PWM que um determinado ESC deverá permanecer em nível lógico alto, isto é, a velocidade de cada motor.

$$ESC1 = T_{throttle} - (AC_{controladorX}) - (AC_{controladorZ}), \tag{3.5}$$

$$ESC2 = T_{throttle} + (AC_{controladorX}) - (AC_{controladorZ}), \tag{3.6}$$

$$ESC3 = T_{throttle} - (AC_{controladorY}) + (AC_{controladorZ}), \tag{3.7}$$

$$ESC4 = T_{throttle} + (AC_{controladorY}) + (AC_{controladorZ}), \tag{3.8}$$

onde, *AC* corresponde a uma ação de controle tomada por um dado controlador, sendo que sua magnitude deve ser compatível com a excursão máxima de um sinal de PWM utilizado em RC, isto é, deve estar dentro da faixa de 0 a 1ms. E, $T_{throttle}$ corresponde ao tempo de duração do comando *throttle*.

A título de ilustração, considerando um controlador como o da Figura 3.15 para o eixo X, baseado apenas em um acelerômetro como sensor de inclinação, se o *stick* B for empurado para frente, o *set point* do controlador X mudará para um valor correspondente ao deslocamento deste *stick*, isto é, $k_x * stick$, que deve corresponder ao ângulo no qual a plataforma deverá permanecer. Isto resultará em um erro, e_x, que corresponde à diferença do novo *set point* e a posição atual do quadrotor α_x e pode ser expresso pela equação 3.9 ou pela equação 3.10.

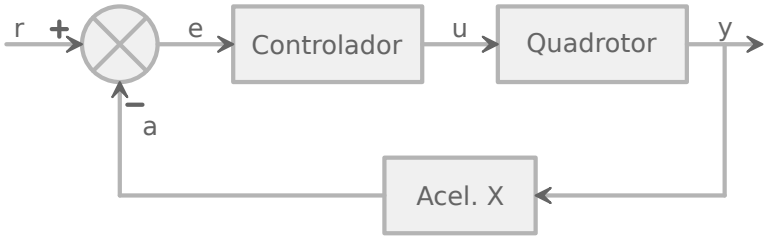

Figura 3.15: Malha fechada para um controle automático no eixo X.

$$e_x = setpoint_x - a_x, \tag{3.9}$$

$$e_x = (k_x * stick_x) - a_x. \tag{3.10}$$

Lembrando que neste trabalho, a_x é um valor de tensão analógica convertida (despresando o *offset*, 1,65V para os acelerômetros) para digital proporcional à projeção da aceleração da gravidade sobre o eixo X, g_x na Figura 3.16 ($a_x \propto g_x$).

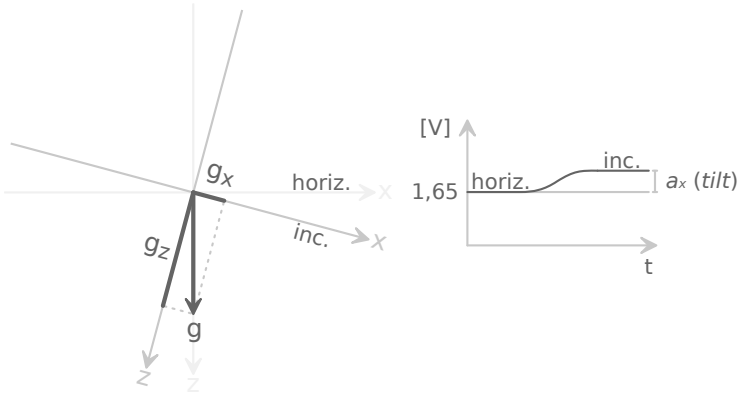

Figura 3.16: Projeção da aceleração da gravidade sobre o eixo X para o quadrotor inclinado para frente.

Baseado neste erro, equação 3.10, o controlador X — seja qual for o paradigma de controle utilizado na implementação deste — deve gerar uma ação de controle tal que aumente a velocidade do motor 2 e diminua a do motor 1 (segundo as equações 3.5 e 3.6) de forma a arfar o quadrotor — conforme ilustra a Figura 3.16 —, consequentemente, resultando em um movimento avante. Quando o ângulo de inclinação medido pelo acelerômetro corresponder à posição (ângulo) do *stick* B, o erro será zero e o quadrotor permanecerá inclinado para frente (em movimento). Ao soltar o *stick*, este retorna para sua posição central, automaticamente, restaurando seu *set point default* (posição original), isto é, um *set point* tal que o controlador ponha o quadrotor nivelado com o horizonte.

Caso um controlador para o eixo Y e outro para o eixo Z (estabilização de giro) também esteja funcionando simultaneamente, para que o quadrotor permaneça sobrevoando um ponto de forma estável (*hovering*), basta o piloto controlar o comando *throttle* através do *stick* A (movimento vertical deste *stick* — para cima e para baixo). Tarefa esta que poderia ser feita por um quarto controlador — controlador de altitude implementado a partir de um medidor de pressão atmosférica, por exemplo. Os demais movimentos dos *sticks* podem permanecer em suas posições originais.

Portanto, um fluxograma que descreva o algoritmo para o programa principal é ilustrado na Figura 3.17:

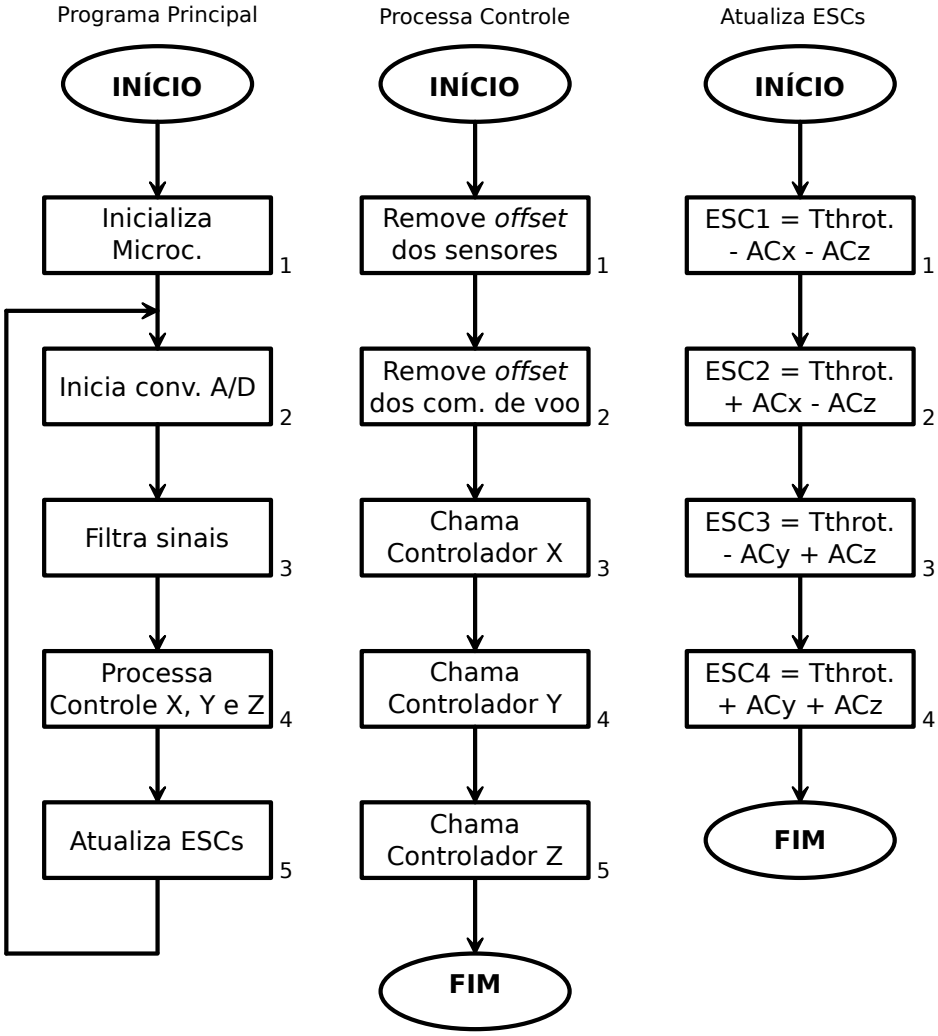

Figura 3.17: Fluxograma para o programa principal.

Onde os blocos 3, 4 e 5 do fluxograma "processa controle"da Figura 3.17 correspondem ao bloco 5 da Figura 3.1.

3.2.5 Temporização dos Sinais de PWM

As equações 3.5, 3.6, 3.7 e 3.8 armazenam o tempo que o PWM de cada ESC deve permanecer em nível lógico alto dentro de um período típico de RC (20ms). Para que estes valores se tornem de fato sinais de PWM nos pinos PD7, PD6, PB1 e PB0 do microcontrolador, dois deles devem ser copiados para os registradores de comparação OCR0A e OCR0B, referentes ao *timer0* e os outros dois para os registradores de comparação OCR2A e OCR2B, referentes ao *timer2*. Logo, as variáveis ESC1 a ESC4 devem ser de 8 *bits*.

Os *timers 0* e *2* iniciam as suas contagens toda vez que uma interrupção de comparação do *timer1* acontece, isto é, quando a contagem do *timer1* se iguala ao conteúdo de OCR1A: instante *t* da Figura 3.18 e t_1 da Figura 3.19. Neste mesmo instante e dentro da referente rotina interrupção, os quatro sinais de PWM são colocados em nível lógico alto e permanecem assim até que o valor do *timer0* alcance o valor de OCR0A e OCR0B, para os PWMs dos ESCs 1 e 2; e até que o valor do *timer2* alcance o valor de OCR2A e OCR2B, para os PWMs dos outros dois ESCs.

A Figura 3.19 ilustra a contagem de um *timer* de 8 *bits* e dois sinais de PWM, PWMA e PWMB, gerados por meio deste *timer* com auxílio de seus registradores de comparação OCRnA e OCRnB; onde *n* pode ser 0 ou 2 (*timer0 ou timer2*). Quando o valor da contagem deste *timer* se iguala com um destes registradores, uma interrupção de comparação é gerada, onde o referente sinal de PWM é colocado em nível lógico baixo.

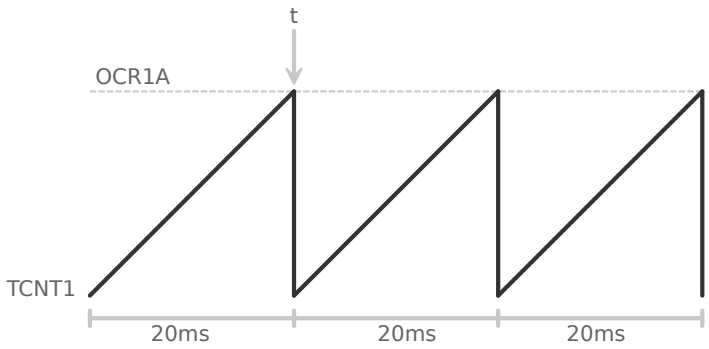

Figura 3.18: Temporização do período de 20ms com o *timer1*.

Quando um *timer* de 8 *bits* "estoura"sua contagem (*overflow*), instante t_2 da Figura 3.19, uma interrupção de *hardware* é gerada, onde o mesmo é desligado; e, sua contagem só reinicia quando há uma interrupção de comparação do *timer1*, isto é, início de um novo período de 20ms.

Para obter esta temporização os *timers* foram configurados para incrementar a partir da frequência da CPU (20MHz) divida por um fator de 256. Assim, o *timer1* precisa fazer 1563

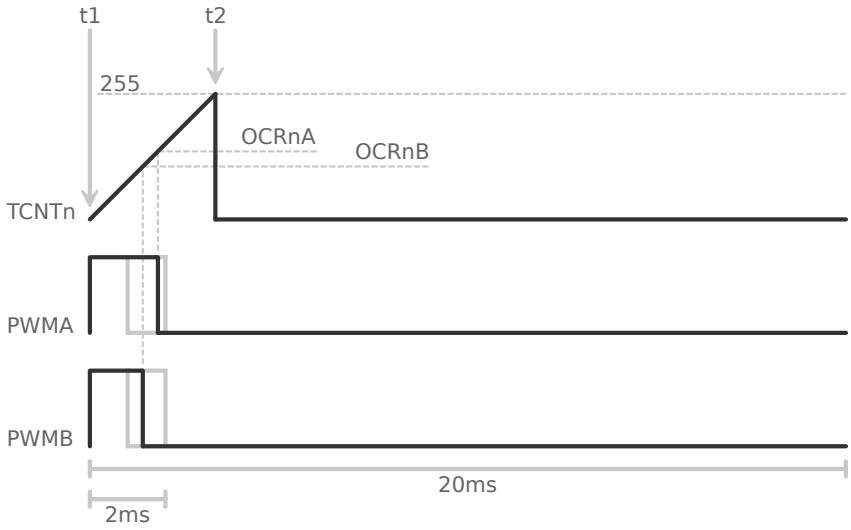

Figura 3.19: Temporização com um *timer* de 8 *bits*.

incrementos para contar 20ms; e os *timers* 0 e 2, 160 incrementos para contar 2ms. Qualquer outro fator de divisão menor que 256 para os *timers* de 8 *bits* faria com que estes "estourassem"sua contagem antes de 2ms, causando erro na temporização. Maiores detalhes de configuração destes temporizadores podem ser encontrados na listagem de código do anexo A.

As rotinas de interrupções utilizadas com o *timer0*, interrupção comparação e de *overflow* são ilustradas através dos fluxogramas da Figuras 3.20; a rotina interrupção de comparação do *timer1* na Figura 3.21; e as pertinentes ao *timer2*, interrupções de comparação e de *overflow*, assim como para o *timer0*, estão na Figura 3.22.

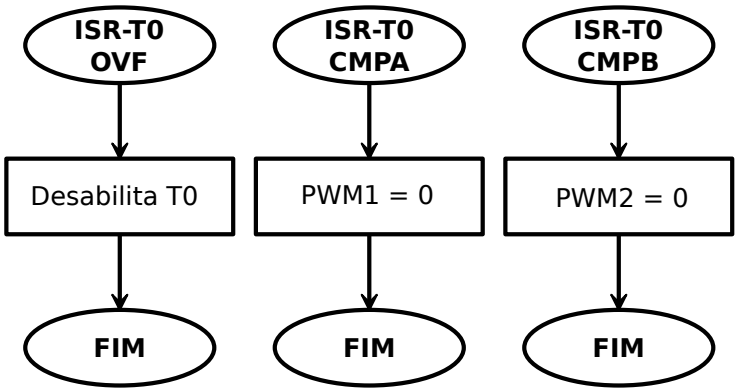

Figura 3.20: Fluxograma para as rotinas de interrupção do *timer0*.

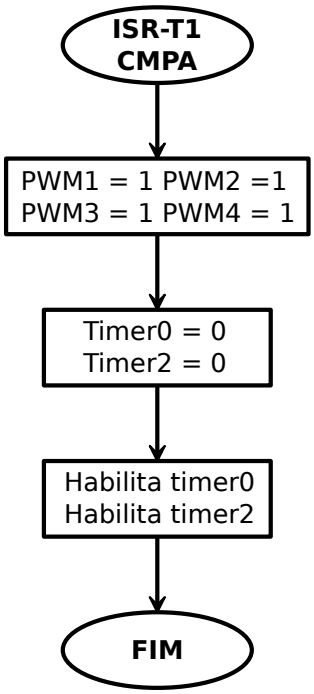

Figura 3.21: Fluxograma para a rotina de interrupção do *timer1*.

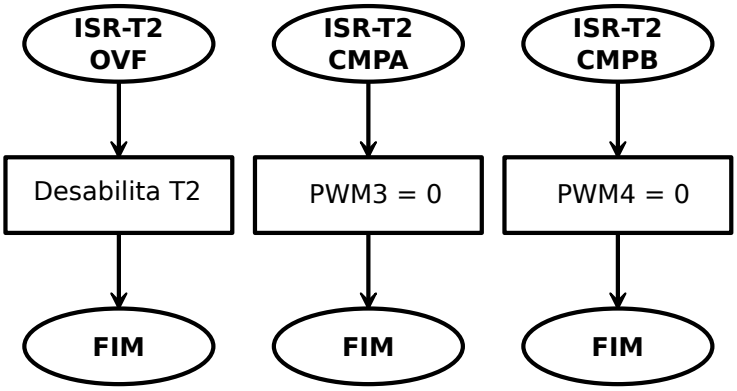

Figura 3.22: Fluxograma para as rotinas de interrupção do *timer2*.

3.2.6 Coleta de Dados - Log

No caso dos sinais dos sensores precisarem ser visualizados em um computador, uma rotina foi implementada fazendo uso da USART para transferí-los, além dos canais capturados do rádio receptor ou qualquer outra informação de interesse, como a ação de controle de um dado controlador que venha ser implementado no quadrotor. Para isto, tais informações devem ser colocadas dentro do *buffer* de *log* para que tal rotina, como a ilustrada pelo fluxograma na Figura 3.23, possa enviá-lo de forma assíncrona *byte* a *byte* a uma taxa de transmissão de 57600bps.

A USART do microcontrolador foi configurada para gerar interrupção toda vez que seu registro de transmissão esvaziar. Assim, para dar início à transmissão, um procedimento a

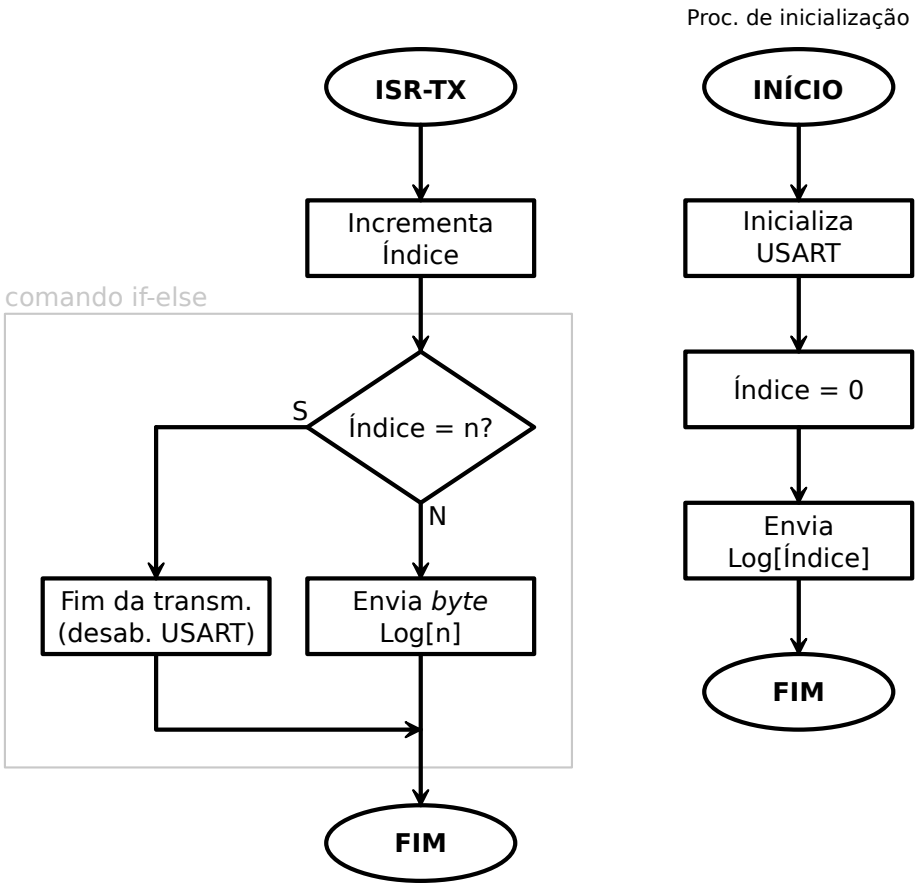

Figura 3.23: Fluxograma para a rotina de *log*.

parte é realizado — proc. de inicialização na Figura 3.23 —, onde a USART é incializada e o primeiro *byte* do *log* é movido para o registro de transmissão da USART, e daí em diante, o processo de envio torna-se automático através da rotina de interrupção identificada como ISR-TX no fluxograma da Figura 3.23. A transmissão *byte* a *byte* permanece até que o índice utilizado para apontar para uma posição do *buffer* do *log* se torne igual ao tamanho *buffer*, isto é, igual a *n*.

Um programa rodando no computador (vide anexo A a sua correspondente listagem na liguagem de programação Python), que lê a porta serial, aguarda pela chegada de um *byte* de informação por um tempo de 10s. Caso nenhum *byte* chegue, o programa é abortado, o que corresponde ao microcontrolador não está operando ou que o *firmware* não está funcionamdo devidademente.

Na medida em que os *bytes* de um *log* vão chegando, estes vão sendo armazenados em *buffers* diferenciados segundo a ordem em que os dados no *log* foram armazenados pelo *firmware* e enviados por este. A ordem de envio escolhida ao se enviar todos os sinais analógicos e todos os canais do rádio é a seguinte: AN0, AN1, ..., AN5, *throttle*, *pitch*, *roll* e *yaw*.

4 Testes e Resultados

Para testar o funcionamento do *hardware* os seguintes procedimentos foram realizados em malha aberta:

- inclinando o quadrotor (girando-o em torno de seus eixos) para coletar os sinais (resposta) dos acelerômetros e dos giroscópios. Conforme a Figura 4.1, f corresponde a uma força aplicada no quadrotor de forma a girá-lo em torno de um dos seus eixos inclinando-o e s corresponde à inclinação i adquirida pelo quadrotor coletada por um sensor;
- movendo os *sticks* do rádio transmissor para cima e para baixo, para um lado e para o outro capturando os comandos de voo referente a tais movimentos, já na saída do rádio receptor, através do pino INT0 de forma serial;
- geração de 4 sinais de PWM, um para cada ESC. Assim, os comandos de voo *pitch*, *roll* e *yaw* fornecidos manualmente e coletados no rádio rececptor podem simular as ações de controle para estabilização tomadas nas equações 3.5, 3.6, 3.7 e 3.8 por cada controlador, junto com o comando *throttle*.

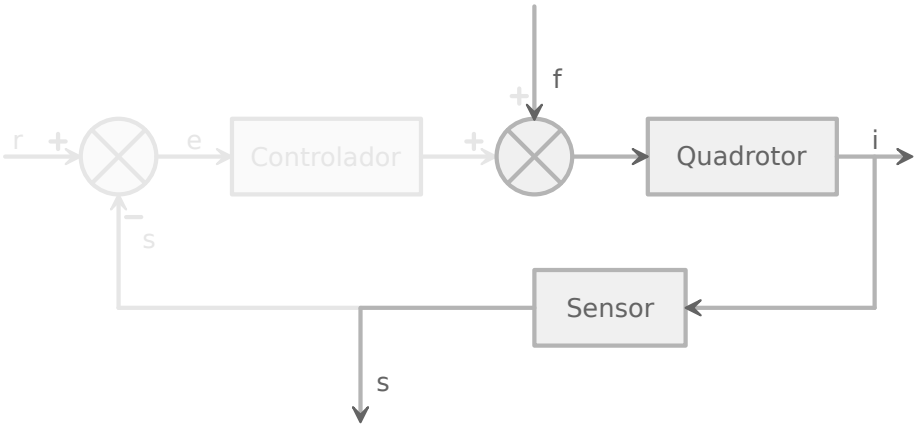

Figura 4.1: Malha aberta — parte realçada da figura.

4.1 Log dos Sinais Analógicos

Para observar as respostas dos sensores perante à movimentação da plataforma, cada canal foi amostrado de 20 em 20ms com o conversor A/D configurado para converter em 8 *bits*, ou seja, com os valores de aceleração e giro após a conversão compreendidos entre 0 e 255, conforme o eixo vertical das Figuras 4.2, 4.3, 4.5 e 4.6.

À medida que o sinal de um dado sensor é amostrado, este é enviado para o computador, onde um programa de comunição serial armazena-o em um *buffer*, para posterior plotagem. As respotas para os acelerômetros X e Y podem ser observadas respectivamente nas Figuras 4.2 e 4.3. Como os acelerômetros são inerentemente ruidosos [Melo, Marco 2004], foi necessário a utilização do filtro de média da seção 3.2.3.

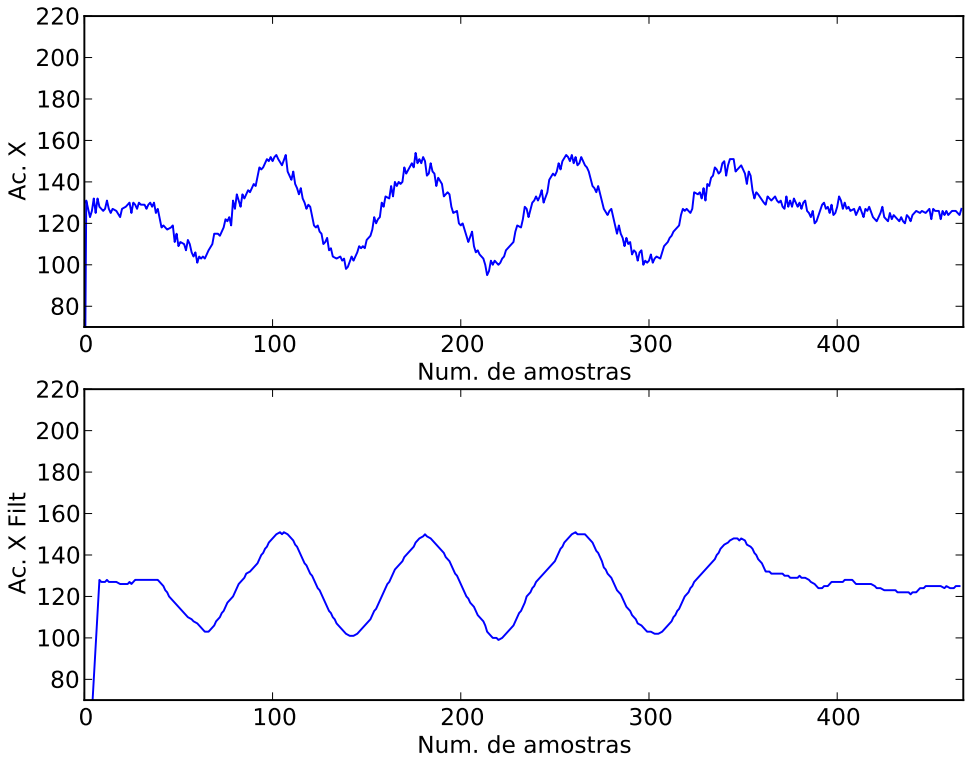

Figura 4.2: Log do acelerômetro X e sua respectiva filtragem (no microcontrolador).

Tais respostas equivalem aos movimentos de inclinar o quadrotor para frente e para trás, conforme a Figura 4.4 (giro em torno do eixo Y), e para um lado e para o outro (giro em torno do eixo X).

O acelerômetro Z, por sua vez, por estar alinhado com o vetor da aceleração da gravidade, possui um *offset* maior que os outros dois — próximo de 200, segundo a Figura 4.5. Logo,

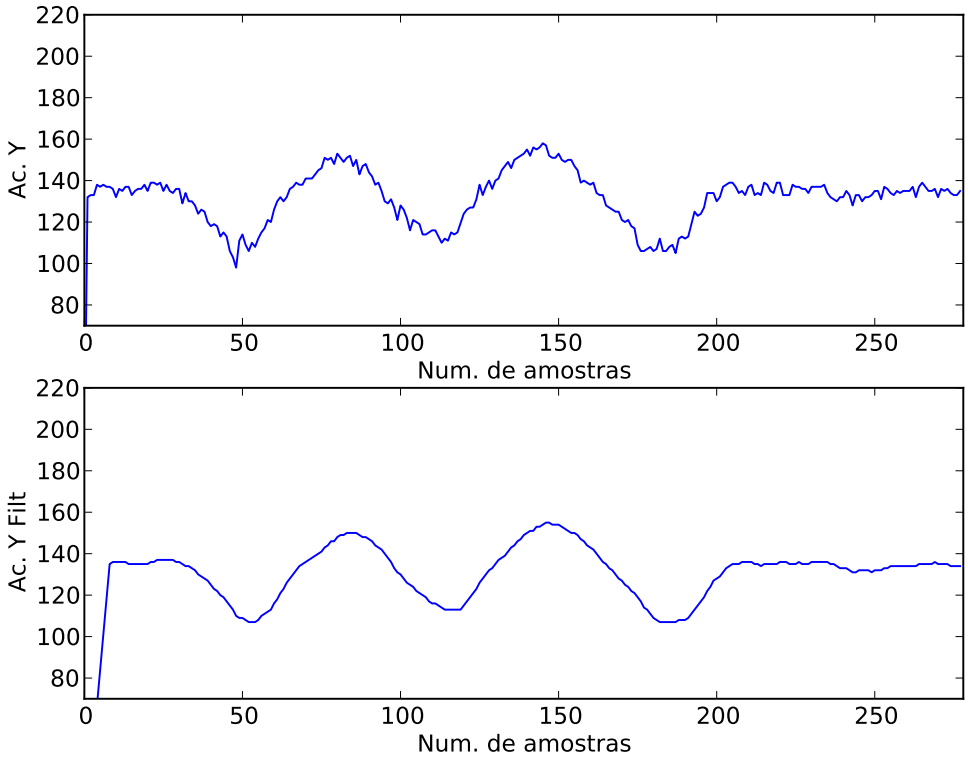

Figura 4.3: Log do acelerômetro Y e sua respectiva filtragem (no microcontrolador).

Figura 4.4: Movimento oscilatório equivalente a empurar as estremidades do quadrotor para cima e para baixo.

qualquer inclinação (até 90°) feita nos eixos X e Y do quadrotor só fará com que este *offset* caia. Isto pode ser visto na Figura 4.5: à medida que o tempo evolui (amostras são feitas), conforme se observa na Figura 4.4, a plataforma foi inclinada para um lado e para o outro, para frente e para trás.

Quanto aos giros X, Y e Z, pode-se observar na Figura 4.6 que estes têm uma resposta menos ruidosa que o acelerômetro e, embora o procedimento de movimentação (giro) e captura tenham sido os mesmos utilizados para com os acelerômetros. Neste caso, os eixos verticais da Figura 4.6 correspodem à coversão A/D de cada giroscópio, equivalendo a suas respectivas velocidades angulares em °/s. Quando não há presença de giro, o valor da conversão fica em torno de 95 (*offset* de 1,23V).

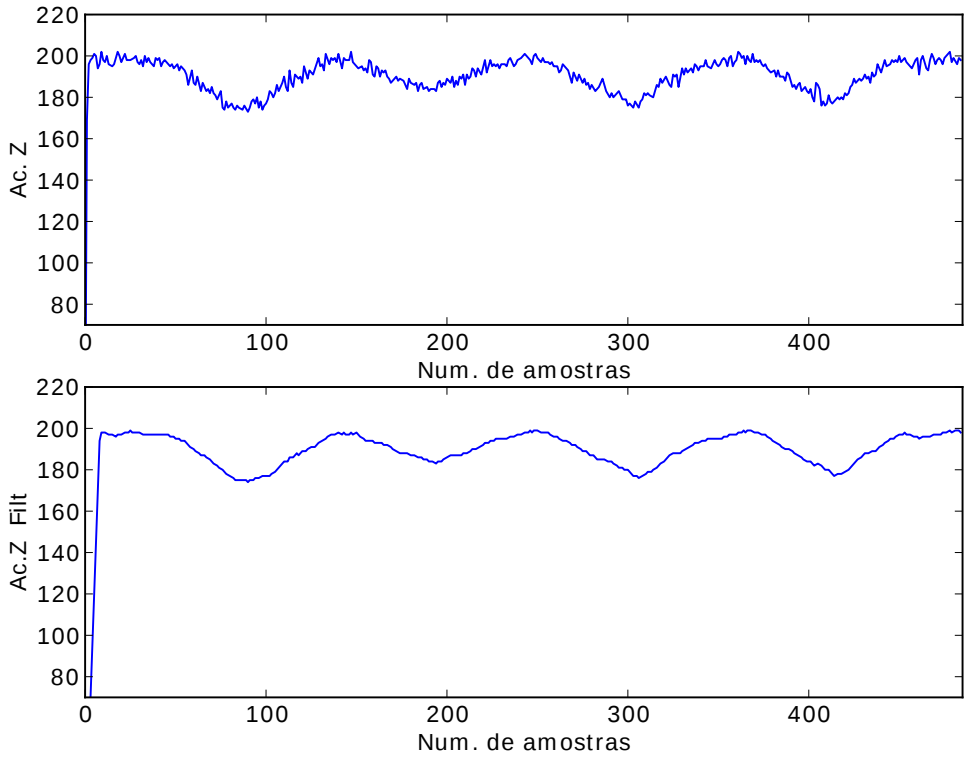

Figura 4.5: Log do acelerômetro Z e sua respectiva filtragem (no microcontrolador).

4.2 Log dos Canais do Rádio Receptor

Como o tamanho de um quadro PPM é 20ms, e o carrilhão de canais do rádio receptor é coletado via interrupção de *hardware*, cada canal é amostrado automaticamanete de 20ms em 20ms. Durante a captura, para observar a resposta dos canais frente a um comando manual, moveu-se os *sticks* A e B para cima e para baixo, para um lado e para outro, resultado no gráfico da Figura 4.7. Onde os eixos verticais das plotagens correspondem à temporização (tempo em nível lógico alto) de um sinal de PWM — 80 para 1ms e 160 para 2ms —, os horizontais, o número de amostras acumuladas ao longo do tempo.

Inicialmente os *sticks* estão em suas posições centrais, o que corresponde a um *offset* igual a 120 (em torno de 1,5ms) na Figura 4.7. Em seguida o *stick* A foi empurrado para baixo, permancendo nesta posicição do instante em que é feita a amostra de número 100 até o instante próximo do instante da amostra 150. Em seguida o *stick* é empurrado para cima, retornando para a posição central, permanecendo aí por um curto intervalo de tempo. O mesmo é feito com este *stick*, porém empurrando para cima. O mesmo procedimento foi feito para os outros canais *roll*, *pitch* e *yaw*, nesta ordem.

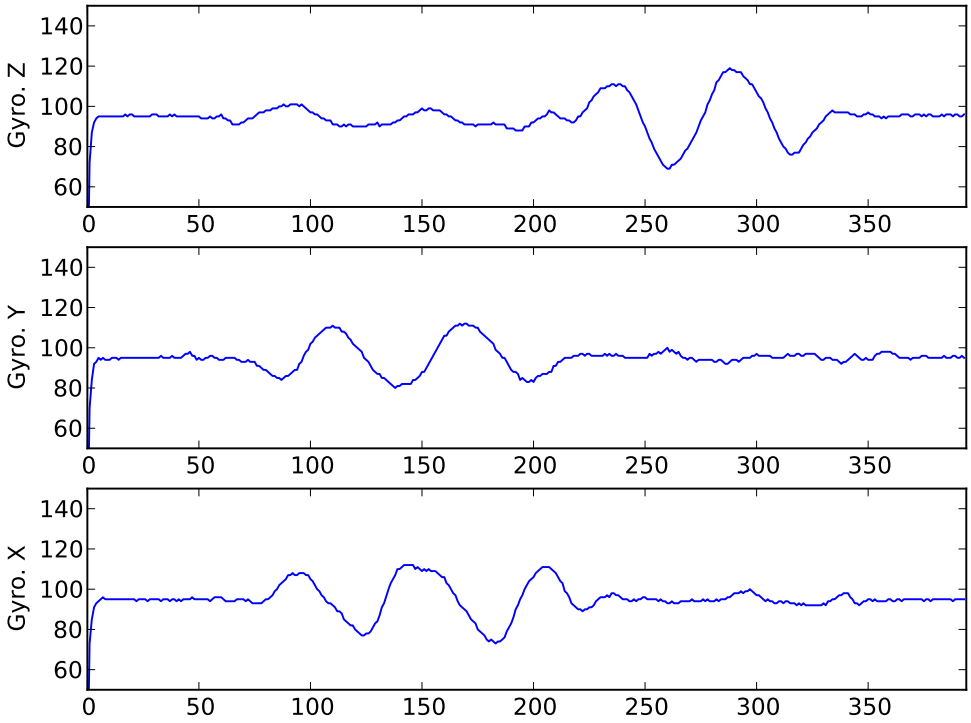

Figura 4.6: *Log* dos giroscópios.

4.3 Teste dos Sinais de PWM

Embora o sinal de PWM para um ESC deva ser gerado a partir das ações de controle dos controladores X, Y e Z, segundo as equações 3.5, 3.6, 3.7 e 3.8 (para um controle automático), no lugar destas, foram usados os próprios sinais capturados, isto é, os comandos de voo manual *throttle*, *pitch*, *roll* e *yaw*, com os três últimos sem os seus respectivos *offsets*. Este teste foi realizado apenas com o intuito de validar a temporização de cada PWM gerado segundo a técnica apresentada na seção 3.2.5, equivalendo a um controle puramente manual do quadrotor, onde o correspondente resultado pode ser visto na Figura 4.8

Observando a Figura 4.8, incialmente todos os sinais de PWM para os ESCs estão com uma largura de pulso entre 80 e 90 incrementos do *timer* (em torno de 1ms). A partir da amostragem de número 50, através do movimento vertical (para cima) do *stick* A (comando *throttle*), adiciona-se um *offset* a todos os sinais de PWM, o que corresponde a todos os motores aumentarem as suas velocidades simultaneamente (quadrotor subindo). Logo após, o mesmo *stick* é reposicionado para uma posição intermaediária (próximo do centro), diminuindo assim, o PWM de todos os ESCs.

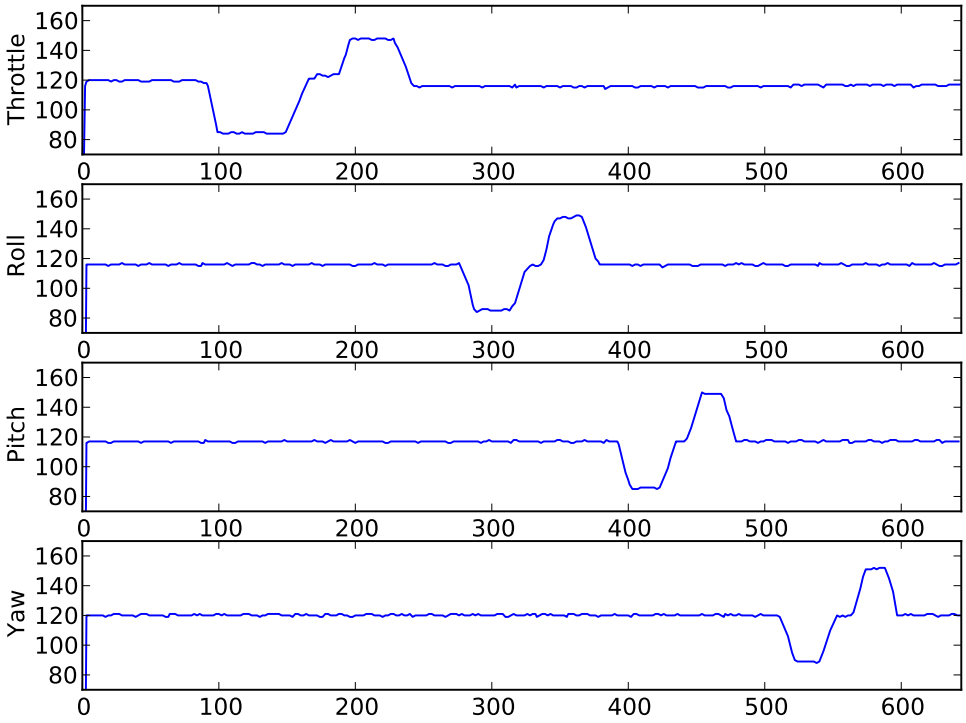

Figura 4.7: Log do acelerômetro Z e sua respectiva filtragem (no microcontrolador).

Em seguinda, um movimento vertical foi imposto ao *stick* B, empurrado-o para baixo, fazendo com que o sinal de PWM para o ESC1 aumente no mesmo tanto que o sinal do ESC2 diminui (equações 3.5, 3.6). Logo após este *stick* retorna a sua posição original (central), seguido de um movimento para cima, o que corresponde à situação anterior de forma inversa. Após este comando, realizou-se o movimento do mesmo *stick*, porém na horizontal: para a esquerda, dimuindo o PWM referente ao ESC3 e aumentando o referente ao ESC4; para a direita, aumentando o PWM referente ao ESC3 e diminuido o referente ao ESC4.

Por último, foi realizado o movimento horizontal do *stick* A para esquerda e para direita, correspondendo ao comando *yaw*, que nesta ordem, resulta em: aumentar simultâneamente os PWMs referentes aos ESCs 1 e 2 no mesmo tanto que os PWMs para os ESCs 3 e 4 diminuem (para esquerda); diminuir simultâneamente os PWMs referentes aos ESCs 1 e 2 no mesmo tanto que os para ESCs 3 e 4 aumentam (para direita).

Tal procedimento foi realizado aproveitando o *firmware* utilizado na coleta de dados, listegem A.1, adicionando à rotina principapal deste, as linhas de código 13 a 21 da listagem A.2, que corresponde ao bloco 5 do fluxograma "programa principal"na Figura 3.17, além de remover o *offset* (1,5ms, que corresponde a posição central do *stick*, ou seja, 120 nesta aplicação) dos

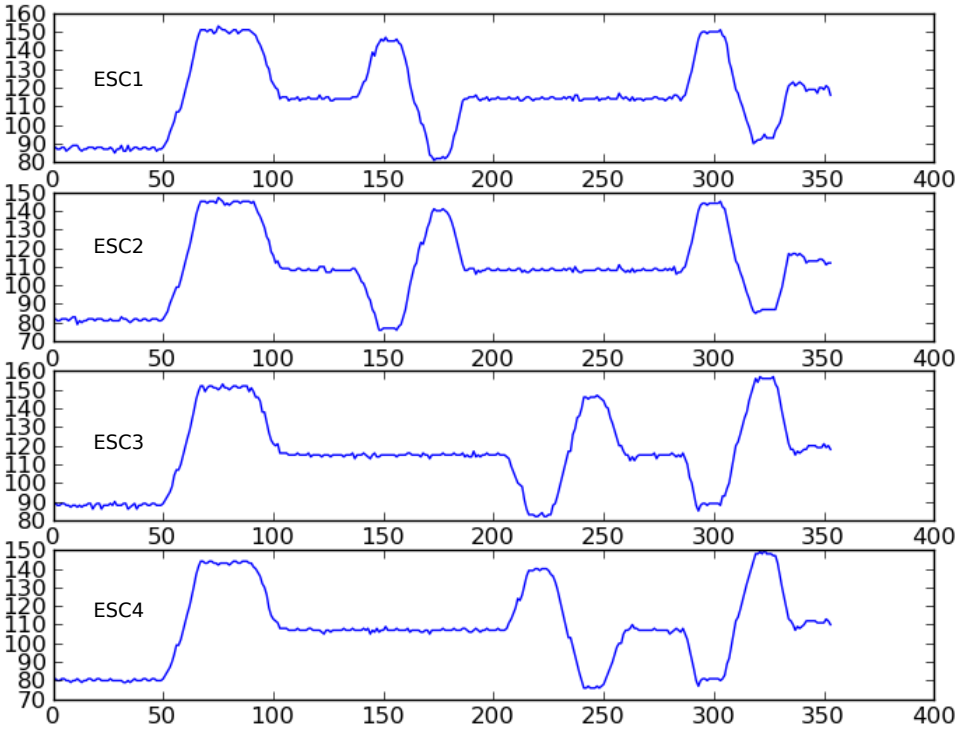

Figura 4.8: Log dos sinais de PWM para os ESCs 1, 2, 3 e 4.

comandos *pitch*, *roll* e *yaw*.

5 *Conclusão*

A proposta de desenvolvimento do *firmware*, como uma interface, permite a elaboração de controladores que utilizem as informações pré-processadas pela bloco 4 (bloco de *firmware* da Figura 3.1), equivalente ao *driver*. Desta forma, foi possível isolar a parte de acionamento e instrumentação da parte de controle. Assim, o projetista de controle abstrai dos detalhes de captura de sinais de sensores e acionamento de motores, preocupando-se com o controlador propriamente dito.

Este *firmware* ocupou aproximadamente 2KB/16KB da memória de programa do micro-controlador, permitindo que todo o espaço restante seja utilizado para implementação de um controlador. Quanto a memória RAM, destinada a armazenar variáveis do *fimware* e de um controlador teve o seu uso limitado em menos de 50B de total de 1K.

Com relação ao *hardware*, nesse trabalho preocupou-se em utilizar dispositivos (sensores) do tipo MEMS com objetivo de fazer uma montagem com o máximo de integração possível com relação à instrumentação. Ou seja, utilizou-se um dispositivo MEMS com três acelerômetros dispostos ortogonalmente entre si. Para o caso específico dos giroscópios, embora um deles (giroscópio Z) não esteja integrado aos demais, ambos são são próprios para montagem em um mesmo plano, ou seja, na mesma PCI, ao contrário de outros dispositivos em que cada eixo está integrado em um encapsulamento a parte, o que exige montagem tri-dimensional. Como vantagem, os CIs MEMS utilizados dispensam qualquer procedimento de aferição mecânica para correção de *offset*.

Como demonstrado nos resultados apresentados nas Figuras 4.2, 4.3 e 4.5, o uso do filtro de média apresentou resultados satisfatórios, contornando o problema de ruído inerente aos acelerômetros. Por natureza, os giroscópios demonstraram serem mais robustos a ruídos que os acelerômetros. Portanto, não foi necessário utilizar o mesmo filtro com estes dispositivos, nem com os canais do rádio receptor.

O protótipo foi elaborado de modo a permitir que ações de controle proporcionada por um especialista humano possam ser capturadas para posteriormente servir como parâmetro de

treinamento supervisionado de um sistema inteligente de controle como o neural, ou então um sistema neural e *fuzzy*).

Como melhorias, sugere-se a substituição do presente microcontrolador, o Atmega168-20PU, pelo Atmega328-20PU, por possuir os mesmos módulos internamente, ser compatível pino a pino e possuir 32KB de memória de programa ao invés de 16KB. Isto permite a implementação de controladores com código fonte maior, dando assim, maior flexibilidade em projetos futuros.

Além disto, sugere-se também a troca dos ESCs com entrada em PWM por outros com entrada digitais usando o barramento I2C, o que fará com que a placa microcontrolada tenha menos trilhas e pinos favorencendo a sua compactação. Isto também permite que parâmetros tal como o período de atualização do ESC seja facilmente reconfigurado pelo mesmo barramento I2C (o microcontrolador do ESC esteja preparado para tal operação), permitindo testes com frequências de atualização diferentes de 50Hz (padrão em RC).

Visto que a captura dos canais do rádio receptor foi satisfatória via interrupção de *hardware*, pode-se liberar três dos quatro pinos de I/O reservados para a captura por meio de *polling* (um dos quatro é o pino PD2 (INT0) é utilizado para capturar o carrilhão serialmente). Assim, estes pinos podem ser utilizados em outras ocasiões.

O fato de ter separado a placa de processamento da placa de sensoriamento, permite maior flexibilidade para se agregar outros tipos de sensores, substituindo a presente placa de sensoreamento, ou então, adicionando uma terceira placa por cima desta através dos conectores CN10 e CN10 da Figura 3.10.

Ao longo da pesquisa e desenvolvimento do protótipo, pode-se constatar que muitos dos conceitos aplicados e circuitos utilizados no quadrotor podem ser reutilizados em outras UAVs, onde a maior parte das adaptações podem ser feitas a nível de *firmware*, bloco 4 da Figura 3.1. Pois, independente do aspecto construtivo, qualquer aeronave precisa de um circuito de sensoriamento possuindo basicamente acelerômetros e giroscópios; e um outro microcontrolado para coletar sinais de comando de voo via rádio, além dos sinais dos sensores, processar algum algoritmo de controle e acionar ESCs e/ou servo-motores, via PWM ou outra forma de sinal.

Para a realização deste trabalho como um todo foram utilizados somente *software* livre, exceto o editor de *layout* para PCIs, Eagle, que por sua vez, foi utilizado na sua versão *freeware* (com limitações). Os demais *software* utilizados são: o compilador para linguagem C avr-gcc; *software* programador Avrdude junto com o *hardware* programador USBasp (projeto *open source* [Fischl, T. 2010]) para programar o microcontrolador; interpretador Python para rodar

o programa cliente feito para a coleta de dados no computador e plotá-los; Gimp para edição de fotografias; Inkscape para edição figuras vetoriais; editor de textos Gedit e o compilador de textos LaTex utilizando a classe ABNTex. Todos eles rodando sobre o sistema operacional GNU/Linux, distribuição Ubuntu versão 9.10. Tais ferramentas serviram satisfatoriamente para o desenvolvimento deste trabalho.

Outras informações e fotos com vistas de detalhes do quadrotor podem ser encontrados no sítio www2.ele.ufes.br/˜alexandre/quadrotor .

Referências Bibliográficas

[Ailson Rosetti de Almeida 2004]Ailson Rosetti de Almeida. *Apostila ADDA*. 2004. URL: http://www2.ele.ufes.br/ ailson. Última vista: 14/06/2010.

[Alexandre Secchin de Melo 2010]Alexandre Secchin de Melo. *Quadrotor*. 2010. URL: http://www2.ele.ufes.br/ alexandre/quadrotor. Última vista: 14/06/2010.

[Antoniou, A. 2006]Antoniou, A. *Digital Signal Processing, Signal Systems and Filters*. [S.l.]: Mc Graw Hill, 2006.

[Atmel 2007]ATMEL, C. *8-bit AVR Microcontroller*. [S.l.]: datasheet, 2007.

[Boanerges, A. V.]Boanerges, A. V. *Aerodinâmica de Helicópteros*. [S.l.]: Editor Rio.

[Bouabdallah, S. Murrieri, P. 2004]Bouabdallah, S. Murrieri, P., . S. *Design and Control of an Indoor Micro Quadrotor*. [S.l.], 2004.

[Bouabdallah, S. 2004]Bouabdallah, S., N. . S. *PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor*. [S.l.], 2004.

[Bresciani, T. 2008]Bresciani, T. *Modelling, Identification and Control of a Quadrotor Helicopter*. [S.l.]: Department of Automatic Control, Lund University, 2008.

[Draganflyer 2010]Draganflyer. *Draganflyer X6*. 2010. URL: http://www.draganfly.com/uav-helicopter/draganflyer-x6/. Última visita: 10/03/2010.

[Fischl, T. 2010]Fischl, T. *USBasp - USB programmer for Atmel AVR controllers*. 2010. URL:http://www.fischl.de/usbasp/. Última visita: 15/06/2010.

[Hartman, E. Keeler, J. D. 1989]Hartman, E. Keeler, J. D., . K. *Layered neural networks with Gaussian hidden units as universal approximation. Neural computation, 2, 210-215*. [S.l.], 1989.

[Hoffmann, G. et all 2004]Hoffmann, G. et all. The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In: *In Proceedings of the 23rd Digital Avionics Systems Conference*. [S.l.: s.n.], 2004.

[Hornik, M. 1989]Hornik, M., S. . W. *Multilayer feedforward networks are universal approximators. Neural Networks, 2, 359-366*. [S.l.], 1989.

[Melo, Marco 2004]Melo, Marco, K. Filtragem de sinais do acelerômetro pela difusão anisotrópica. In: *Anais do CBA 2004, Gramado*. [S.l.]: Sociedade Brasileira de Automatica, 2004.

[Mendel, J. M. 1995]Mendel, J. M. Fuzzy logic systems for engineering: A tutorial. In: . [S.l.]: BirkHauser, 1995. p. 345–377.

[Newman 2003]NEWMAN, M. E. J. *The Structure and Function of Complex Networs*. [S.l.], 2003.

[Paul Mc Whorter 2008]Paul Mc Whorter. *About MEMS*. 2008. URL: http://www.memx.com/. Última visita: 10/03/2010.

[Perspectives Aerials 2010]Perspectives Aerials. *Perspectives Aerials*. 2010. URL: http://www.perspectiveaerials.com/. Última visita: 10/03/2010.

[Prinz, F. B. 1999]Prinz, F. B., K. . *The Mesicopter: A Meso-Scale Flight Vehicle*. [S.l.], 1999.

[Quemel, P. H. R. 2009]Quemel, P. H. R., S. . B. Modelagem e controle de quadrirrotores. In: *Grupo de Robótica, Automação e Visão Computacional (GRAV), Depertamento de Engenharia Elétrica, UNB, Brasília, DF*. [S.l.: s.n.], 2009.

[Ryan, M. 2002]Ryan, M., F. *Communication and Information Systems*. [S.l.]: Argor Press, 2002.

[Santos, I. F. 2001]Santos, I. F. . *Dinâmica de Sistemas Mecânicos, Modelagens, Simulação, Visualização, Verificação*. [S.l.]: Makron Books, 2001.

[Sontag, E. D. 1993]Sontag, E. D. Neural networks for control. In: *In Essays on Controal: Perspectives in Theory and its Applications*. [S.l.]: BirkHauser, 1993. p. 339–380.

[Stingu, E]Stingu, E, L. *Quad-Rotor Specifications*. [S.l.].

[Waslander, S. L. Hoffmann, G. M. 2005]Waslander, S. L. Hoffmann, G. M., J. . T. Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learnig*. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2005.

[Wikipedia 2010]Wikipedia. *Quadrotor*. 2010. URL: http://www2.ele.ufes.br/ ailson. Última vista: 14/06/2010.

ANEXO A – Código Fonte para a Validação do Hardware

A listagem A.1, na linguagem C, é referente ao *firmware* utilizado para validar o *hardware*, podendo ser utilizado como driver para algum controlador que venha ser implementado.

```
1  #define F_CPU                                    20000000UL
2  #define Ttop                                     1600
3  #define USART_BAUD                               57600UL
4  #define USART_UBBR_VALUE                         ((F_CPU/(USART_BAUD<<4))-1)
5  #define ADC0                                     0
6  #define ADC1                                     1
7  #define ADC2                                     2
8  #define ADC3                                     3
9  #define ADC4                                     4
10 #define ADC5                                     5
11 #define RCThrottleT                              6
12 #define RCPitchT                                 7
13 #define RCRollT                                  8
14 #define RCYawT                                   9
15
16 #include <avr/io.h>
17 #include <avr/interrupt.h>
18
19 unsigned char Data[10], Log[10];
20
21 unsigned char Flag;
22
23 unsigned char Counter;
```

```
24
25 unsigned int     t, Previoust;
26
27 unsigned char    PPMChannel;
28
29 unsigned char AD[6];
30
31 unsigned char Buffer[3][8], Index[3];
32 //unsigned char Buffer0[8], Buffer1[8], Buffer2[8], Index0, Index1, Index2;
33
34 void Config(void) {
35
36         // PWM Outputs
37         DDRD |= (1<<7) | (1<<6);
38         DDRB |= (1<<1) | (1<<0);
39
40         // Enable INT0
41         EICRA = 3;
42         EIMSK = 1;
43
44         // Timer1 period for PWM1, PWM2, PWM3 and PWM4
45         TCCR1A = 0x00;
46         TCCR1B = 0x0C; // F_CPU/256
```

```
47
48          OCR1AH = 0x06;
49          OCR1AL = 0x40;
50
51          TIMSK1 = 0x02; // Timer1 Interrupt On Match A
52
53          // Timer0 – PWM1 and PWM2
54          TCCR0A = 0;
55          TCCR0B = 0;
56
57          TIMSK0 = 0x07; // Timer Overflow, Output Compare A and B int. ON
58
59          OCR0A = 80;
60          OCR0B = 80;
61
62          // Timer2 – PWM3 and PWM4
63          TCCR2A = 0;
64          TCCR2B = 0;
65
66          TIMSK2 = 0x07; // Timer Overflow, Output Compare A and B int. ON
67
68          OCR2A = 80;
69          OCR2B = 80;
```

```
70
71          // Set baud rate
72          UBRR0H = (uint8_t) (USART_UBBR_VALUE>>8);
73          UBRR0L = (uint8_t) USART_UBBR_VALUE;
74
75          // Set frame format to 8 data bits, no parity, 1 stop bit
76          UCSR0C = (0<<USBS0)|(1<<UCSZ01)|(1<<UCSZ00);
77
78          // Enable transmitter
79          UCSR0B = (1<<TXEN0) | (1<<TXCIE0);
80          //UCSR0B = (1<<TXEN0) | (1<<RXCIE0) | (1<<RXEN0);
81
82          // Global Enable Interrup
83          sei();
84
85  }
86
87  unsigned int GetTime(void) {
88
89          return ((unsigned int)(TCNT1L) | (unsigned int)(TCNT1H << 8));
90
91  }
92
```

```
93 unsigned int CalcT(unsigned int t1, unsigned int t2) {
94
95         if (t1 > t2)
96                 return ((Ttop - t1) + t2);
97
98         return (t2 - t1);
99
100 }
101
102 ISR(TIMER1_COMPA_vect) {
103
104         PORTD |= (1<<7) | (1<<6);
105
106         PORTB |= (1<<1) | (1<<0);
107
108
109         TCNT0 = 0;
110         TCNT2 = 0;
111
112         // Timer0 ON (clk = F_CPU/256)
113         TCCR0B = 0x04;
114
115         // Timer2 ON (clk = F_CPU/256)
```

```
116             TCCR2B = 0x06;
117
118             Flag = 1;
119
120 }
121
122 ISR(TIMER0_OVF_vect) {
123
124             // Stop Timer0
125             TCCR0B = 0;
126
127 }
128
129 ISR(TIMER2_OVF_vect) {
130
131             // Stop Timer2
132             TCCR2B = 0;
133
134 }
135
136 ISR(TIMER0_COMPA_vect) {
137
138             // PWM1 Off;
```

```
139             PORTD &= ~(1<<6);
140
141 }
142
143 ISR(TIMER0_COMPB_vect) {
144
145             // PWM2 Off;
146             PORTD &= ~(1<<7);
147
148 }
149
150 ISR(TIMER2_COMPA_vect) {
151
152             // PWM3 Off
153             PORTB &= ~(1<<0);
154
155 }
156
157 ISR(TIMER2_COMPB_vect) {
158
159             // PWM4 Off
160             PORTB &= ~(1<<1);
161
```

```
162 }
163
164 ISR(INT0_vect) {
165
166         unsigned int    T;
167
168         t = GetTime();
169         T = CalcT(Previoust, t);
170         if ((T > 70) && (T < 170)) {
171                 switch (PPMChannel) {
172                         case 0:
173                                 Data[RCThrottleT] = (unsigned char) T;
174                                 break;
175                         case 1:
176                                 Data[RCRollT] = (unsigned char) T;
177                                 break;
178                         case 2:
179                                 Data[RCPitchT] = (unsigned char) T;
180                                 break;
181                         case 3:
182                                 Data[RCYawT] = (unsigned char) T;
183                                 break;
184                         default:        // Channels 5 and 6
```

```
185                                     ;
186                     }
187                     PPMChannel++;
188                     Previoust = t;
189             }
190             else if (T > 200) {
191                     PPMChannel = 0; //Throttle channel;
192                     Previoust = t;
193             }
194 }
195
196 ISR(ADC_vect){
197
198             unsigned char ADResult;
199             unsigned char    ADMuxChannel;
200
201             ADResult = ADCH;
202             ADMuxChannel = ADMUX & 0x07;
203             ADMUX++;
204             ADCSRA |= (1<<ADSC);
205             switch (ADMuxChannel)    {
206                     case 0:
207                             //Data[ADC0] = ADResult;
```

```
208                 AD[0] = ADResult;
209                 break;
210             case 1:
211                 //Data[ADC1] = ADResult;
212                 AD[1] = ADResult;
213                 break;
214             case 2:
215                 //Data[ADC2] = ADResult;
216                 AD[2] = ADResult;
217                 break;
218             case 3:
219                 Data[ADC3] = ADResult;
220                 //AD[3] = ADResult;
221                 break;
222             case 4:
223                 Data[ADC4] = ADResult;
224                 //AD[4] = ADResult;
225                 break;
226             case 5:
227                 Data[ADC5] = ADResult;
228                 //AD[5] = ADResult;
229                 ADCSRA &= ~(1<<ADEN);    // disable A/D converter
230                 break;
```

```
231                     default:
232                             ;
233             }
234 }
235
236 void StartADConversions() {
237
238             // set ADMUX Register: channel 0 selected and result left justified
239             ADMUX = (1<<ADLAR);
240
241             // set ADCSRA Register: AD Enable, ADStar, ADInterruptEnable...
242             ADCSRA = (1<<ADEN) | (1<<ADSC) | (1<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
243
244 }
245
246 void EndOfTransmission() {
247
248             UCSR0B = (1<<RXCIE0) | (1<<RXEN0);
249
250 }
251
252 ISR(USART_TX_vect) {
253
```

```
254         Counter++;
255
256         if (Counter == 10)
257                 EndOfTransmission();
258         else {
259                 UDR0 = Data[Counter];
260         }
261
262 }
263
264 void StartOfTransmission() {
265
266         UCSR0B |= (1<<TXEN0) | (1<<TXCIE0);
267         UDR0 = Data[0];
268         Counter = 0;
269 }
270
271
272 void Filter(unsigned int *PointerToSum, unsigned int *PointerToBuffer, unsigned int *
      PointerToIndex, unsigned int *Value) {
273
274         *PointerToSum = *PointerToSum - *(PointerToBuffer + *(PointerToIndex)) + *Value;
275         *(PointerToBuffer + *(PointerToIndex)) = *Value;
```

```
276         *PointerToIndex = ((*PointerToIndex) - 1) & 0x07;
277         *Value = ((*PointerToSum) / 3);
278 }
279
280
281 /*
282 void Filter(unsigned char *PointerToBuffer, unsigned char *PointerToIndex, unsigned char *Value)
        {
283
284         unsigned char i,j;
285         unsigned int    Sum;
286
287         Sum = 0;
288         j = *PointerToIndex;
289         *(PointerToBuffer + j) = *Value;
290         for(i=0;i<8;i++) {
291                 Sum = Sum + *(PointerToBuffer + j);
292                 if (j == 7)
293                         j = 0;
294                 else
295                         j++;
296         }
297         *PointerToIndex = ((*PointerToIndex) + 1) & 0x07;
```

```
298             *Value = Sum >> 3;
299
300     }
301     */
302
303     int main(void) {
304
305             unsigned char i;
306
307             PPMChannel = 0;
308
309             Config();
310
311             while (1) {
312
313                     StartADConversions();
314
315                     // Filter Accel. x, y, z
316                     for (i=0;i<3;i++) {
317                             //Filter(&Buffer[i][0], &Index[i], &AD[i]);
318                             Filter(&Sum[i], &Buffer[i][0], &Index[i], &AD[i]);
319                     }
320
```

```
321                 Data[0] = AD[0];
322                 Data[1] = AD[1];
323                 Data[2] = AD[2];
324
325                 StartOfTransmission();
326
327                 // Wait to reach 20ms
328                 while (!Flag);
329
330                 Flag = 0;
331
332         }
333
334         return 0;
335
336 }
```

Listing A.1: Listagem do *firmware*.

O trecho de código da listagem A.2 é uma sugestão para uma chamada à um controlador (bloco 5 da Figura 3.1) baseado no *tilt* dos eixos X e Y e giro do eixo Z, lembrando que as variáveis a mais contidas nesse trecho de código devem ser declaradas. Onde as variáveis AccelX, AccelY e GyroZ corresponde às variáveis posições 1, 2 e 3 do *array* Data da listagem A.1. Já os endereços &CtrlX, &CtrlY e &CtrlZ, corresponde ao início de uma estrutura (*struct*) que contenha todos os parâmetro do controlador. Assim, o mesma função para um dado algoritmo de controle pode usada para o controle dos três eixos.

```
1                SetPointX = (signed char) ((RCPitchT – OffsetPitch) * kX);
2                SetPointY = (signed char) ((RCRollT  – OffsetRoll)  * kY);
3                SetPointZ = (signed char) ((RCYawT   – OffsetYaw)   * kZ);
4
5                TiltX = (signed char) (AccelX – OffsetAccelX);
6                TiltY = (signed char) (AccelY – OffsetAccelY);
7                GyroZ = (signed char) (GyroZ  – OffsetGyroZ);
8
9                ControlActionX = Control(SetPointX, TiltX, &CtrlX);
10               ControlActionY = Control(SetPointY, TiltY, &CtrlY);
11               ControlActionZ = Control(SetPointZ, GyroZ, &CtrlZ);
12
13               ESC1 = RCThrottleT – ControlActionX – ControlActionZ;
14               ESC1 = RCThrottleT + ControlActionX – ControlActionZ;
15               ESC1 = RCThrottleT – ControlActionY + ControlActionZ;
16               ESC1 = RCThrottleT + ControlActionY + ControlActionZ;
17
18               OCR0A = ESC1;
19               OCR0B = ESC2;
20               OCR2A = ESC3;
21               OCR2B = ESC4;
22
23 char Control(signed char SetPoint, signed char Sensor, struct Controller *p) {
```

```
24
25          // controller
26
27          return ControlAction;
28
29  }
```

Listing A.2: Listagem para chamada a um controlador.

ANEXO B – Código Fonte para o Software do Computador

A listagem B.1, na linguagem Python, refere-se ao *software* utilizado para fazer a captura de todos os sinais de interesse para a validação do *hardware* do quadrotor e plotá-los. Para utilizar este programa, basta digitar o seguinte comando em um terminal: `log.py`. Após pressionar a tecla *enter*, o programa arguardará o envio de informação de informação por 10s (*timeout* = 10). Caso nenhum *byte* chege, o programa e abortado.

Se a transmissão iniciar antes de 10s, o programa capturará dados até que a tecla *enter* seja pressionada novamente. Uma vez interrompido com a tecla *enter*, os dados são plotados pelo mesmo programa.

```
1  import serial
2  import sys
3  import time
4  import glob
5  import numpy as np
6  import matplotlib.pyplot as plt
7  import select
8  import operator
9
10 def heardEnter():
11     i,o,e = select.select([sys.stdin],[],[],0.0001)
12     for s in i:
13         if s == sys.stdin:
14             input = sys.stdin.readline()
15             return True
16     return False
17
18
19 def main():
20
21         BaudRate = 57600
22
23         # Search for a serial port
```

```
24      SerialPorts =  glob.glob('/dev/ttyUSB*') + glob.glob('/dev/ttyS*')
25      if SerialPorts == []:
26              sys.exit("There isn't serial port avaible.")
27      else:
28              SerialPort = SerialPorts[0]
29              print "Serial port", SerialPort, "was found."
30
31      ser = serial.Serial(SerialPort, BaudRate, bytesize=serial.EIGHTBITS, parity='N',
           stopbits=serial.STOPBITS_ONE, timeout=10)
32
33      ADC0 = []
34      ADC1 = []
35      ADC2 = []
36      ADC3 = []
37      ADC4 = []
38      ADC5 = []
39      RCThrottleT = []
40      RCRollT = []
41      RCPitchT = []
42      RCYawT = []
43
44      Cycles = 0
45
```

```
46      Data = [ADC0, ADC1, ADC2, ADC3, ADC4, ADC5, RCThrottleT, RCRollT, RCPitchT, RCYawT]
47
48      while 1:
49              for i in Data:
50                      Char = ser.read()
51                      i.append(ord(Char))
52
53              if heardEnter():
54                      Cycles = len(ADC0)
55
56                      plt.figure(1)
57
58                      plt.subplot(3,1,1)
59                      plt.plot(ADC0)
60                      plt.ylabel('Accel. Z')
61                      plt.axis([0, Cycles, 70, 220])
62
63                      plt.subplot(3,1,2)
64                      plt.plot(ADC1)
65                      plt.ylabel('Accel. Y')
66                      plt.axis([0, Cycles, 70, 220])
67
68                      plt.subplot(3,1,3)
```

```
69          plt.plot(ADC2)
70          plt.ylabel('Accel. X')
71          plt.axis([0, Cycles, 70, 220])
72
73          plt.figure(2)
74
75          plt.subplot(3,1,1)
76          plt.plot(ADC3)
77          plt.ylabel('Gyro. Z')
78          plt.axis([0, Cycles, 50, 150])
79
80          plt.subplot(3,1,2)
81          plt.plot(ADC4)
82          plt.ylabel('Gyro. Y')
83          plt.axis([0, Cycles, 50, 150])
84
85          plt.subplot(3,1,3)
86          plt.plot(ADC5)
87          plt.ylabel('Gyro. X')
88          plt.axis([0, Cycles, 50, 150])
89
90          plt.figure(3)
91
```

```
92          plt.subplot(4,1,1)
93          plt.plot(RCThrottleT)
94          plt.ylabel('Throttle')
95          plt.axis([0, Cycles, 70, 170])
96
97          plt.subplot(4,1,2)
98          plt.plot(RCRollT)
99          plt.ylabel('Roll')
100         plt.axis([0, Cycles, 70, 170])
101
102         plt.subplot(4,1,3)
103         plt.plot(RCPitchT)
104         plt.ylabel('Pitch')
105         plt.axis([0, Cycles, 70, 170])
106
107         plt.subplot(4,1,4)
108         plt.plot(RCYawT)
109         plt.ylabel('Yaw')
110         plt.axis([0, Cycles, 70, 170])
111
112         plt.show()
113
114         print Cycles
```

```
115                             ser.close()
116                             break;
117
118 if __name__ == "__main__":
119
120         main()
```

Listing B.1: Progama para fazer *log* de dados.

ANEXO C – Layout das PCIs

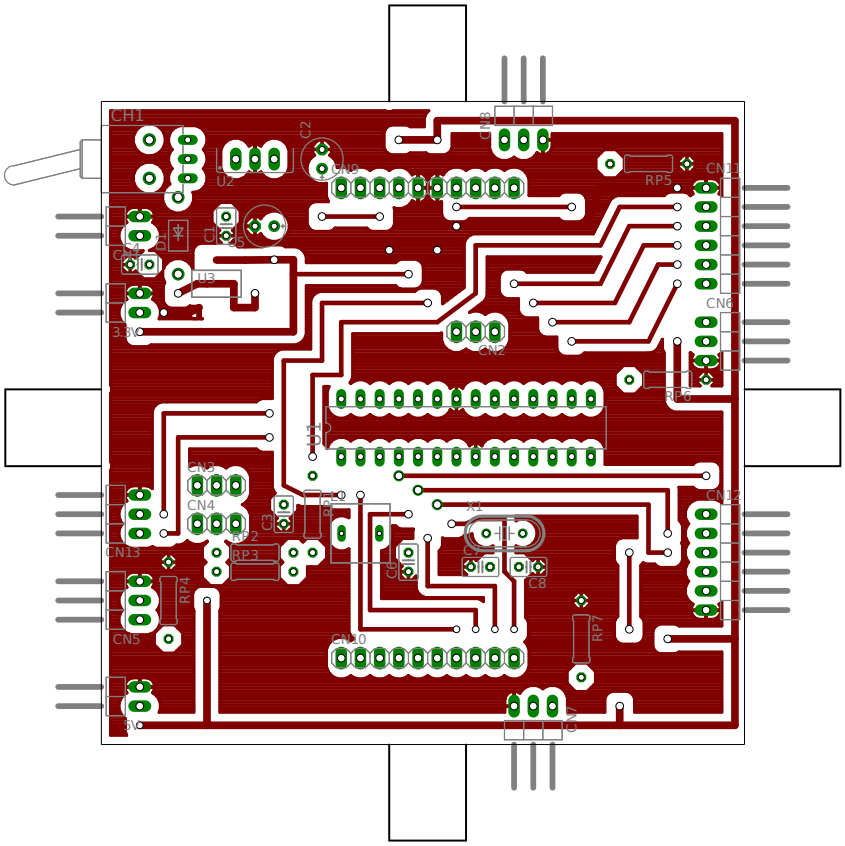

Figura C.1: Camada superior da placa microcontroladora.

Rótulo	**Componente**	**Código**	**Valor/característica**
U1	Microcontrolador	Atmega168-20PU	—
U2	Regulador de tensão	7805	5V
U3	Regulador de tensão LDO	LD111t-33	3,3V
D1	Diodo	1N400x	—
C1	Capacitor	—	100nF
C2	Capacitor	—	10μF
C3	Capacitor	—	100nF
C4	Capacitor	—	100nF
C5	Capacitor	—	10μF
C6	Capacitor	—	100nF
C7	Capacitor	—	27pF
C8	Capacitor	—	27pF
Rp	Resistor	—	10kΩ
L1	Indutor	—	470μH
CN1	Conector Bateria	—	—
CN2	Conector Seleção de Aref	—	3,3V/5V
CN3	Conector Sleção de GIO ou I2C	—	AD4/SDA
CN4	Conector Sleção de GIO ou I2C	—	AD5/SCL
CN5	Conector PWM1	—	—
CN6	Conector PWM2	—	—
CN7	Conector PWM3	—	—
CN8	Conector PWM4	—	—
CN9	Conector Expansão	—	—
CN10	Conector Expansão	—	—
CN11	Conector SPI/ISP	—	—
CN12	Conector Rádio Recep.	—	—
CN11	Conector SPI/ISP	—	—
CN13	Conector USART.	—	—
Hn	Furo	—	3mm

Tabela C.1: Componentes da placa microcontrolada.

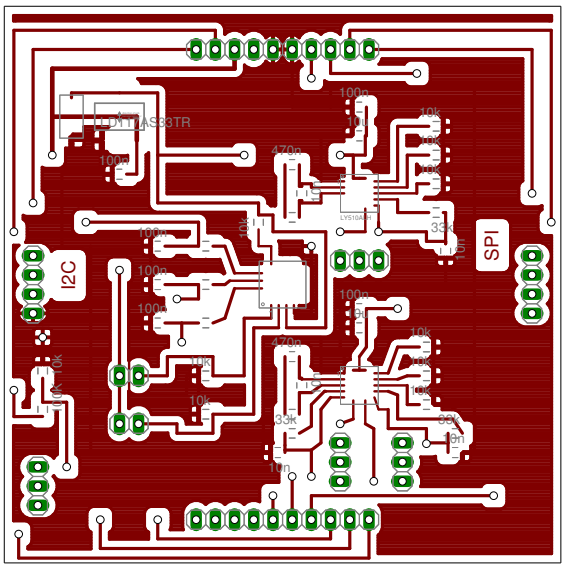

Figura C.2: Camada superior da placa de sensoriamento.

Rótulo	**Componente**	**Código**	**Valor/característica**
U1	Sensor giroscópio	LY510LH	2,5 ou 10mV/°s/s
U2	Sensor giroscópio	LPR510AL	2,5 ou 10mV/°s/s
U3	Sensor acelerômetro	MMA7260Q	até 800mV/g
U4	Regula de Tensão LDO	LD1117-33	3,3V
C1	capacitor	—	10nF
C2	capacitor	—	100nF
C3	capacitor	—	470nF
C4	capacitor	—	10nF
C5	capacitor	—	2,2μF
C6	capacitor	—	2,2μF
C7	capacitor	—	100nF
C8	capacitor	—	100nF
C9	capacitor	—	470nF
C10	capacitor	—	10nF
C11	capacitor	—	100nF
C12	capacitor	—	100nF
C13	capacitor	—	100nF
C14	capacitor	—	100nF
C15	capacitor	—	100nF
C16	capacitor	—	10μF
R1	Resistor	—	10kΩF
R2	Resistor	—	10kΩF
R3	Resistor	—	10kΩF
R4	Resistor	—	10kΩF
R5	Resistor	—	33kΩF
R6	Resistor	—	33kΩF
R17	Resistor	—	10kΩF
R18	Resistor	—	10kΩF
R19	Resistor	—	10kΩF
R10	Resistor	—	10kΩF
R11	Resistor	—	33kΩF
R12	Resistor	—	1kΩF
R13	Resistor	—	1kΩF
R14	Resistor	—	1kΩF

Tabela C.2: Componentes da placa de sensoreamento.

APÊNDICE A – Princípios Sobre Motores Brushless

Partindo da definição de motor,

"*tudo em mecânica que imprime movimento, como o vapor, a água, o vento, etc*";

e dos seguintes princípios:

- de que pólos magnéticos diferentes (norte e sul) se atraem e iguais se repelem (norte e norte ou sul e sul);
- que todo condutor quando percorrido por uma corrente elétrica gera um campo magnético;

é possível construir um motor elétrico rudimentar, isto é, um dispositivo eletro-mecânico (que converte energia elétrica em mecânica) de posse de dois ímãs, uma barra de material ferromagnético (material permeável ao campo magnético e que se magnetiza em sua presença) e condutores.

Enrolando um condutor elétrico encapado (com isolante elétrico) em forma de espiral para formar uma bobina e o energizando através de uma fonte elétrica, surgirá nas proximidades do condutor, conforme a Figura A.1, um campo magnético proporcional à corrente que circula por ele com dois pólos: norte e sul. Obs: por convenção, as linhas de campo magnético sempre saem do pólo norte e chegam no sul, quando se observa externamente um objeto magnetizado, tal como na Figura A.1.

Os pólos se estabelecem na bobina conforme o sentido da corrente que a percorre. Em outras palavras, para uma corrente elétrica circulando em um dado sentido, cria-se um norte em um lado e um sul no outro — Figura A.1 a); e, para uma corrente elétrica circulando no sentido contrário, os pólos invertem de posição — Figura A.1 b). Para determinar onde o pólo norte e sul estão em função da corrente que percorre o condutor, usa-se a regra da mão direita[1].

[1]Regra da mão direita: o polegar aponta para o norte enquanto os outros quatro dedos contornam a(s) espira(s) formada(s) pelo condutor no mesmo sentido que a corrente convencional a(s) percorre.

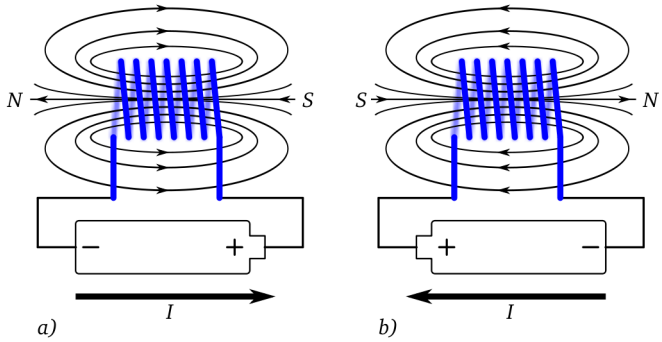

Figura A.1: a) Pólo norte abaixo; b) Pólo norte acima.

Considerando o imã da Figura A.2 a) fixo e uma bobina posicionada ao acaso em suas proximidades, livre para se movimentar, ao energizá-la, esta criará um campo magnético que interagirá com o campo do ímã movimentando-a de forma a se alinhar magneticamente. O alinhamento se dará de forma que o pólo sul da bobina se aproxime do pólo norte do ímã, e o pólo norte da bobina se alinha que o pólo sul do ímã.

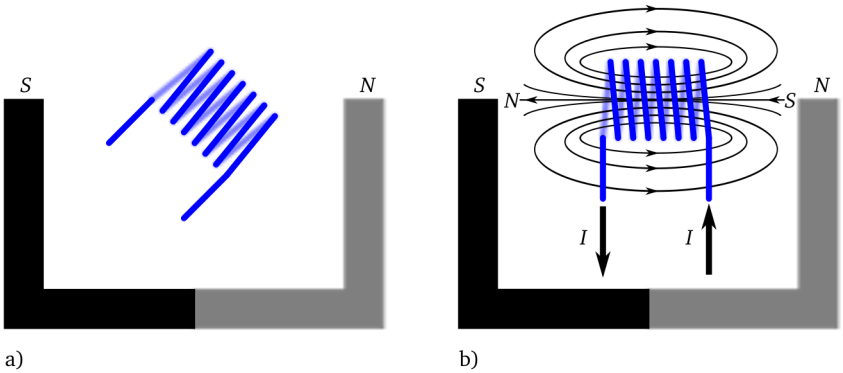

Figura A.2: a) Bobina desligada nas proximidades do ímã; b) Bobina energizada com pólos magnéticos arificiais alinhado com o ímã devido a interação de força magnéticas.

Entretanto, a interação das forças magnéticas pode se tornar mais forte caso a bobina tenha em seu interior um núcleo ferromagnético, pois este se magnitizará na presença do campo magnético da bobina, isto é, se tornará um ímã artificial, intensificando o campo no interior da mesma, além de tornar os pólos magnéticos norte e sul mais bem definidos. A esta configuração dá-se o nome de eletroímã.

Assim, a partir do princípio de funcionamento do eletroímã, pode-se obter um motor de dois pólos rudimentar, cujas partes constituintes, dividem-se em duas: estator e rotor. O estator é a parte estática onde se encontram os ímãs[2] e se apoia o rotor. Já o rotor, pode ser construído

[2]Em pequenos motores, os ímãs do estator normalmente são permanente, tal como nos motores usados em

com duas bobinas, uma para cada pólo, ligadas em série — Figura A.3 — com o mesmo núcleo ferromagnético, cujo centro, é atravessado transversalmente por um eixo, que por sua vez, se apoia no estator.

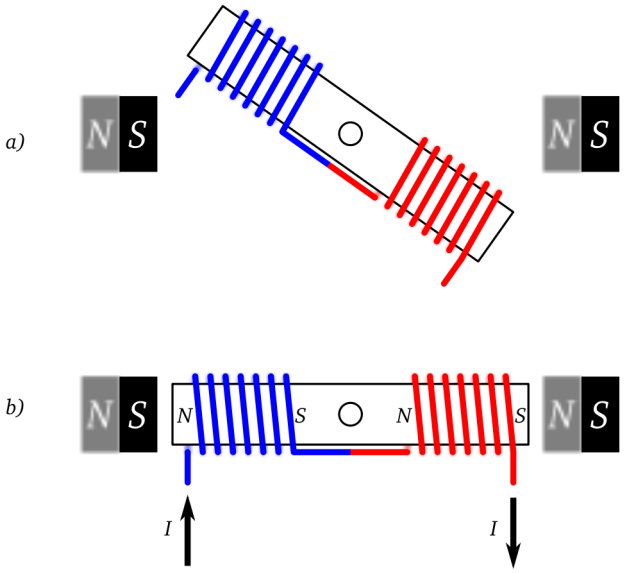

Figura A.3: a) Rotor sem corrente e desalinhado; b) Alinhamento do rotor quando percorrido por corrente.

Para um motor como o da Figura A.3, uma vez energizado, o movimento do rotor cessa assim que o eletroímã formado pelas duas bobinas se alinha aos ímãs — Figura A.3 b). Entretanto, o intuído é permanecer girando. Então, deve-se adicionar a este motor um dispositivo que faça a inversão de polaridade magnética do bastão toda vez que este se alinhar aos ímãs — tal como na Figura A.4 a) e b) — , mantendo assim o movimento giratório.

Tal dispositivo — Figura A.4 c) — é constituído basicamente por escovas, fixadas no estator, e comutadores fixados no rotor. As escovas, contatos feitos normalmente de grafite, possuem conexão permanente com a fonte de energia, estão fixadas no estator, defasadas entre si 180° e pressionadas contra os comutadores. Já os comutadores, arcos metálicos, estão fixados no eixo do rotor através de um material isolante elétrico, defasados entre si 180° e conectados permanentemente aos terminais do enrolamento do rotor. Assim, a polaridade nas bobinas do motor inverte a cada 180° percorrido pelo rotor a partir do alinhamento magnético, conforme ilustra a Figura A.5.

Neste dispositivo de comutação, as escovas devem ser menores que o interstício existente entre um comutador e outro para que não se curto-circuite a fonte de energia elétrica. Em consequência, se o motor estiver desligado (em repouso) e o rotor alinhado com os ímãs, Figura

briquedos ou pequenos equipamentos, enquanto motores maiores podem ter ímãs artificiais em seu estator.

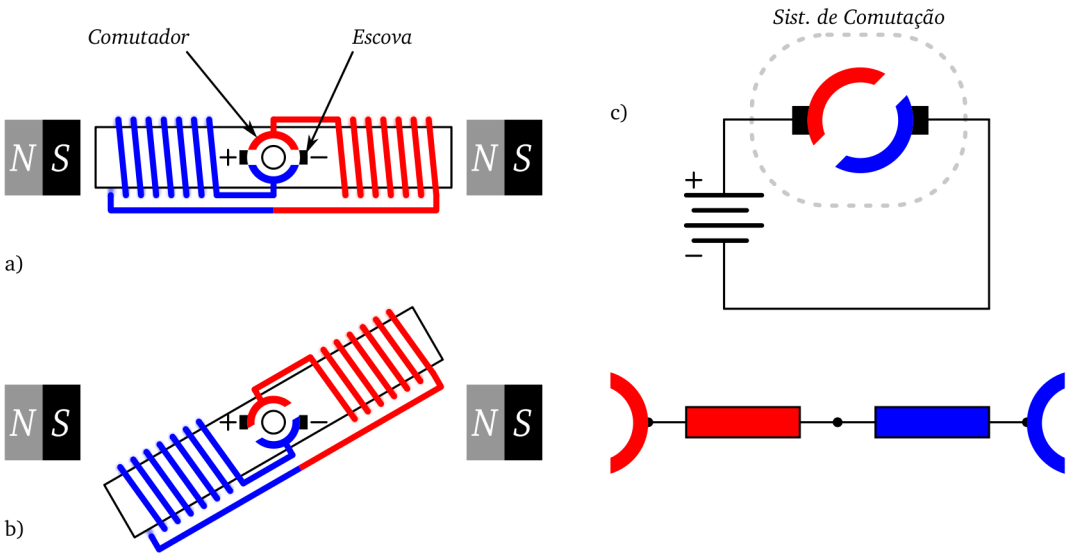

Figura A.4: Motor de 2 pólos: a) Comutadores e Escovas adicionados; b) Comutadores fazendo contato com as escovas; c) Circuito elétrico do motor.

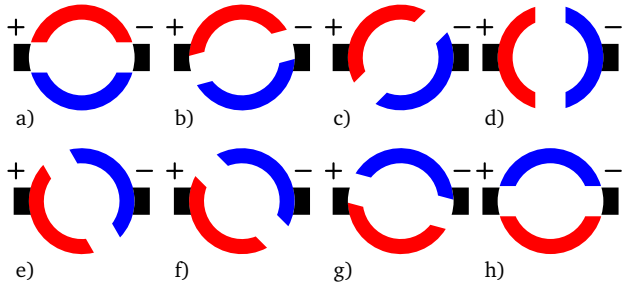

Figura A.5: giro no rotor de 180° no sentido anti-horário.

A.5 a) e Figura A.5 h), para dar início ao movimento giratório é necessário um agente externo que lhe aplique um torque no eixo para conectar as bobinas do rotor à fonte.

Para evitar este inconveniente, adiciona-se mais um pólo no rotor, conforme a Figura A.6, dispostos entre si com ângulo de 120° com suas respectivas bobinas ligadas em estrela[3] e escovas com tamanho maior que o interstício entre cada comutador. Assim, pelo menos dois pólos sempre estarão em contato com as escovas fazendo que o motor gire por si só quando as escovas forem conectadas à fonte, e sem a possibilidade desta ser curto-circuitada.

Os motores elétricos que funcionam segundo o princípio até aqui apresentado são chamados de motores de corrente contínua com escova ou simplesmente motores DC (*Direct Current*) por serem alimentados com uma fonte de corrente contínua.

[3]Ligação estrela: um terminal de cada bobina ligado em um ponto comum e cada um dos outros terminais ligados em um dos comutadores.

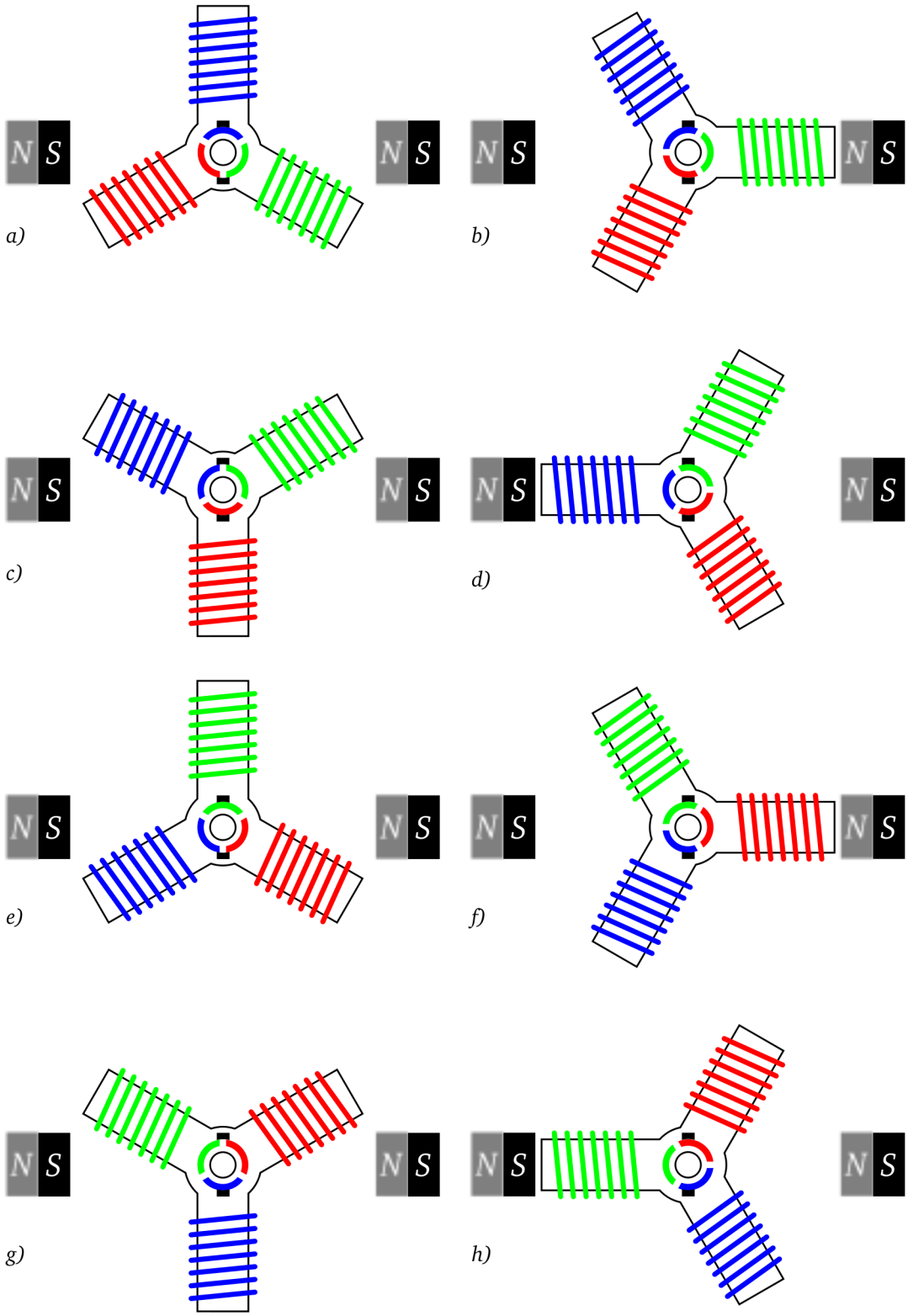

Figura A.6: Motor com rotor de três pólos.

Devido a forma como o sistema de comutação de um motor DC de três pólos é construído, nunca as três bobinas são alimentadas simultaneamente com corrente máxima I, pois, tal sistema é constituído por três arcos metálicos (comutadores) defasados entre si 120° e equidistantes do

eixo do rotor, junto com duas escovas (terminais positivo e negativo da fonte DC) defazadas entre si 180°, também equidistantes do eixo do rotor conforme a Figura A.6 a). Assim, as duas possíveis formas de se alimentar as bobinas desta máquina elétrica, resusme-se nas seguintes situações:

- cada escova em contato exclusivo com uma bobina, isto é, duas das três bobinas energizadas — Figura A.6 b). Em consequência, nas bobinas energizadas circulará uma corrente I, que corresponde a qualquer instante que não esteja dentro do intervalo T_n da Figura A.7, com n igual a $1, 2, 3...$;
- comutação — uma escova em contato exclusivo com uma das três bobinas, e a outra em contato com as outras duas bobinas — Figura A.6 a). Neste caso, em uma bobina, a que está em contato exclusivo com uma das escovas, terá uma corrente I e, as outras duas terão uma corrente $I/2$ por ficarem em paralelo, o que corresponde a qualquer intervalo T_n da Figura A.7.

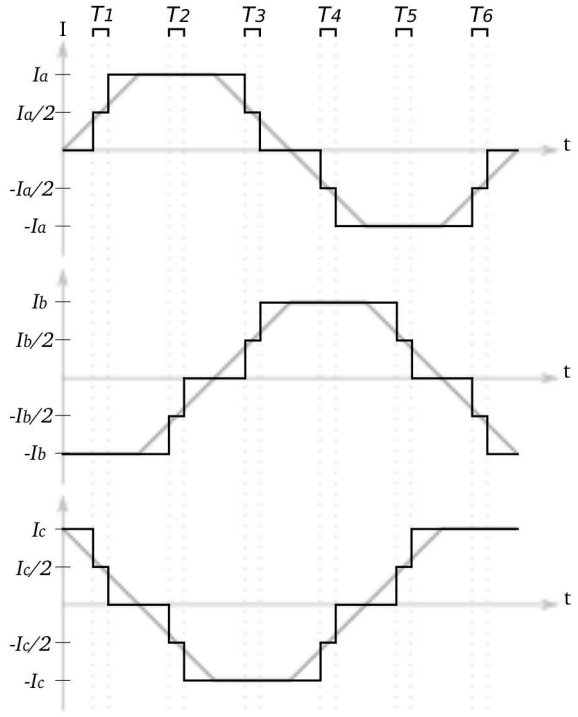

Figura A.7: Corrente nas bobinas de um motor DC de três pólos, onde *a*, *b* e *c* correspondem às bobinas do motor. Obs: o intervalo T corresponde à comutação.

No entanto, algumas observações podem ser feitas à respeito do uso de um motor DC na prática:

- por conta da comutação mecânica, os comutadores e principalmente as escovas se desgastam com o uso, exigindo manutenção periódica;

- centelhas (que implicam em perdas) e ruídos são gerados durante a comutação por interromper a corrente abruptamente em uma dada bobina (indutor), gerando assim ruídos que podem interferir no funcionamento de circuitos eletrônicos em suas proximidades (como é o caso do quadrotor, que leva abordo vários circuitos eletrônicos);

- os comutadores e escovas impõe não só um limite de velocidade de operação, mas também um limite no números de pólos de um motor DC, pois quanto mais pólos, mais comutadores serão necessários com escovas cada vez menores chegando ao ponto de ser impraticável.

A.1 Motor DC sem Escovas

Uma alternativa à comutação mecânica com escovas é a comutação eletrônica. Aproveitando o motor DC descrito até aqui, pode-se construir um motor DC sem escovas (*brushless*), isto é, um motor DC com comutação eletrônica. No entanto, agora as bobinas do motor do motor DC não podem mais ter movimento devido os fios que as conectam com o circuito eletrônico de comutação. Logo, para resolver este impasse, as bobinas passam a ser o estator, e os ímãs, o rotor.

Uma vez que a comutação em um motor DC *brushless* é eletrônica, não necessariamente esta precisa ser feita abruptamente como no comutador mecânico, mas pode ser feita de uma forma mais suave e controlada aplicando-se uma outra forma de onda, por exemplo, trapezoidal — Figura A.7 — ou até mesmo senoidal, desde que as formas de onda de cada bobina continuem defasadas de 120° entre si para o motor de três bobinas (fases) da Figura A.6.

Entretanto, para se obter maior precisão e torque, os motores *brushless* possuem mais que três pólos e mais que dois ímãs mantendo o mesmo número de fases, ou seja, mais de um pólo por fase. Lembrando que, por não ter comutadores e escovas, consegue-se construir motores *brushless* com maior número de pólos.

Enfim, baseado no que foi descrito nesta seção e na anterior, torna-se mais adequado o uso do motor DC *brushless* num quadrotor microcontrolado quando comparado com um motor DC com escovas, pois:

- por se controlar a energização das bobinas eletrônicamente, o motor *brushless* gera menos

ruído que o motor DC com escovas. Isto é importante pois os motores estão próximos dos circuitos de instrumentação do quadrotor;

- possuem rendimento superior aos motores DC com escova (bom para aplicações cuja fonte de energia é uma bateria).

Existe ainda a versão *inrunner* do motor DC *brushless*, porém não será explanada aqui por fugir do escopo do trabalho, visto que o motor escolhido é *outrunner* por possuir mais torque que um *inrunner* do mesmo porte (sem levar em conta o tipo de ligação das bobinas - estrela ou triângulo, que influencia no torque). Isto se deve ao fato do motor *outrunner* possuir seus ímãs mais distantes do eixo.