



PROJETO DE ESTABILIDADE DE UM MULTI ROTOR

Projeto de Graduação do Curso de Engenharia e Automação

Monografia apresentada como requisito parcial para obtenção do grau de Engenheiro pelo Programa de Graduação em Engenharia de Controle e Automação da PUC-Rio.

Coordenador curso:

Mauro Speranza neto

Orientador:

Marco Antonio Meggiolaro

Aluno:

Bruno Vasconcelos de Freitas Cavalcanti

Matrícula: 07102183

Rio de Janeiro

Dezembro de 2011

A Deus e Minha Família.

AGRADECIMENTOS

Quero agradecer aos professores Dr. Marco Antônio Meggiolaro, Dr. Mauro Speranza Neto, pelo apoio, pelos conhecimentos ensinados e pelas oportunidades oferecidas em minha vida acadêmica.

Ao Dr. Pedro Blois por todo conhecimento fornecido e incentivo para o desenvolvimento deste projeto.

Aos professores da PUC – RIO, principalmente o Mauro schwanck pelo ensino.

A equipe Riobotz.

A Nathalia Morreira Leahy por tornar minha vida mais feliz e ter me dado forças para sempre seguir em frente.

À PUC-Rio, por ser meu segundo lar.

Aos meus familiares,

A minha mãe e irmão, por tudo.

RESUMO

O presente relatório fala sobre Veículo **Aéreo Não Tripulado**, também chamado **UAV**. O uso de UAVs multi-rotores tem sido objeto de diversos estudos e testes nos últimos anos devido ao fato de serem relativamente fáceis de construir, baratos e por voarem como um helicóptero, os mesmos podem ser operados em espaços restritos como um laboratório de protótipos.

O sensoriamento de UAVs baseia-se principalmente na leitura dos gyrometros e acelerômetros embarcados. A unidade composta por estes dois sensores é chamada IMU (inertial measurement unit), junto com o controle PID tornam-se bastante efetivos para a estabilização do UAV desde que se garanta que os erros dos ângulos não cresçam demais.

LISTA DE FIGURAS

Figura 1 - MQ-9 Reaper, caçador de vigilância UAV	10
Figura 2– UAV para uso em pesquisas científicas, aplicações comerciais e estatais.	12
Figura 3 - Aeryon scout, produzido pela empresa canadense Aeryon labs, conta com uma poderosa câmera com zoom óptico de 10x, e uma câmera térmica utilizada para obter imagens noturnas. Pode atingir velocidades de até 50Km/h e persistir em uma posição estável mesmo em condições de ventos de até 80Km/h.	13
Figura 4– Sentido de rotação dos motores.....	17
Figura 5 – Dranganflyer Possui uma IMU com acelerômetros e giroscópios, de 3 eixos.	18
Figura 6 - Esquema das forças produzidas pela rotação dos propulsores a igual velocidade.....	19
Figura 7– Movimento no eixo Z, Altitude.....	20
Figura 8– Roll (<i>movimento a azul</i>).	20
Figura 9– Pitch (<i>movimento a azul</i>).....	21
Figura 10– Yaw (<i>movimento a azul</i>)	22
Figura 11– Referenciais B (<i>vermelho</i>), E (<i>preto</i>) e o vector posição linear T^E (<i>azul</i>).....	23
Figura 12– funcionamento do acelerometro.	33
Figura 13– funcionamento do acelerometro.	33
Figura 14 - Comparação das leituras acelerometro e gyro	40
Figura 15- Modelo de um quadri-rotor.....	41
Figura 16– Conceito de uma gangorra.....	42
Figura 17– Princípio de funcionamento de uma alavanca.....	42
Figura 18– Princípio de funcionamento do experimento.....	43
Figura 19 – Gráfico sinal PPM x EMPUXO	44
Figura 20– Medição de empuxo.	44
Figura 21– Eletronica usada.	45
Figura 22– Protótipo montado para simular estabilidade de um multirotor.	46
Figura 23– Protótipo montado para simular estabilidade de um multirotor.	46
Figura 24- Central inercial (SEN-10321).	47
Figura 25 - esquemático Central inercial (SEN-10321).....	48
Figura 26- Protocolo I2C.	49
Figura 27- TowerPro Brushless Outrunner 2410-08T 890kv	50
Figura 28 – Arduino nano.....	52

LISTA DE TABELAS

Tabela 1– Classificação pelo peso.....	14
Tabela 2– Classificação pelo alcance e resistência de voo.....	14
Tabela 3– Classificação pela altura máxima	14
Tabela 4– Classificação pela carga das Asas	15

Sumário

1	OBJETIVO.....	9
	<i>DESCRIÇÃO GERAL DAS ATIVIDADES</i>	9
	<i>ORGANIZAÇÃO DA MONOGRAFIA</i>	9
2	- <i>INTRODUÇÃO</i>	10
2.1	- VEÍCULOS AÉREOS NÃO TRIPULADOS (UAVs)	10
2.2	PLATAFORMAS	12
2.3	CLASSIFICAÇÃO DE UAVs.....	14
3	- <i>MULTIROTOR</i>	16
3.1	PLATAFORMAS MULTI-ROTORES.....	16
3.2	QUADROTOR.....	16
3.3	APLICAÇÕES DO QUADROTOR	18
3.4	PRINCIPIO DE FUNCIONAMENTO DO QUADROTOR	19
3.5	Modelo de Newton – Euler.....	22
4	- <i>SENSORIAMENTO E CONTROLE PID</i>	32
4.1	SENSORES USADOS PARA ESTABILIDADE	32
4.2	FILTROS.....	35
4.3	CONTROLE PID:.....	40
5	- <i>SIMULAÇÃO DE ESTABILIDADE</i>	42
5.1	PROTÓTIPO	42
5.2	CENTRAL INERCIAL.....	47
5.2.1	Protocolo I2C.....	49
5.3	MOTOR BRUSHLESS.....	50
5.4	MICROCONTROLADORES	51
5.5	SPEED CONTROL (Controlador de velocidade)	53
6	CONCLUSÃO	54
7	REFERÊNCIAS	55
	Apêndice A (especificação arduino nano).....	56
	Apêndice B (especificação motor).....	59
	Apêndice C (especificação speed)	60
	Apêndice D (especificação bateria).....	61
	Apêndice E (especificação helice)	62
	Apêndice F (datasheet acelerometro)	63
	Apêndice G (datasheet gyro).....	65

Apêndice H (software)	67
-----------------------------	----

1 OBJETIVO

O objetivo desta monografia é estudar os conceitos de estabilidade de UAVs, Utilizando-os para desenvolver um protótipo que simule a estabilidade de um multirotor.

DESCRIÇÃO GERAL DAS ATIVIDADES

As atividades desenvolvidas durante o período de desenvolvimento desta monografia foram as seguintes:

- Estudo sobre veículos aéreos não tripulados (UAV).
- Estudo sobre sensoriamento de UAVs.
- Estudo do controle PID em um quadrotor.
- Montagem de um protótipo.

ORGANIZAÇÃO DA MONOGRAFIA

Esta monografia está dividida em cinco partes: a primeira parte consiste de uma breve introdução sobre UAVs, a segunda parte descreve o estudo sobre quadrotor, a terceira fala sobre os sensores usados para estabilidade e controle PID, a quarta parte sobre o protótipo desenvolvido para simular a estabilidade em apenas um eixo, e a última parte conclusão.

2 - INTRODUÇÃO

2.1 - VEÍCULOS AÉREOS NÃO TRIPULADOS (UAVs)

Veículos aéreos não tripulados (UAV), também conhecido como um sistema de aeronaves não tripuladas, é uma máquina que funciona tanto pelo controle remoto de um navegador ou de forma autônoma. Estes estão se tornando amplamente utilizados, para missões de vários tipos como, reconhecimento e combate militar, investigação de fenômenos atmosféricos em grande altitude, investigação em zonas de difícil alcance, comunicações, operações de salvamento e vigilância de espaços.

Estes veículos aéreos não tripulados têm propulsão própria utilizando forças aerodinâmicas que provocam a sua sustentação e não possuem cabine de pilotagem, pois podem ser controlados á distância ou podem possuir algoritmos sofisticados de voo que não requerem a intervenção humana. Assim os mesmos proporcionam uma medida adicional de segurança e conveniência quando aplicada a inúmeras situações que antes exigiam uma aeronave de tamanho normal com piloto, cujos custos são muito maiores.



Figura 1 - MQ-9 Reaper, caçador de vigilância UAV

A indústria aeronáutica aumentou os investimentos no estudo e desenvolvimento deste tipo de aeronaves, mas o setor militar continua a ser o maior investidor. Embora o setor militar seja o maior investidor, estas aeronaves não são apenas utilizadas em situações militares, mas também são utilizadas na observação de fenômenos meteorológicos onde não é possível a presença humana, em zonas de vigilância prolongadas, em operações de

resgate e busca e em laboratórios e universidades para fins científicos e educativos.

O desenvolvimento crescente deste tipo de aeronaves deve-se ao facto de possuírem diversas configurações e capacidades para enfrentarem situações em que os humanos teriam limitações a nível físico e psicológico para concluírem as mesmas tarefas de uma maneira eficiente.

Muitos UAVs atuais possuem sistemas on-board de controle que reduzem a quantidade de controle exigido a partir do operador da estação de terra. Um exemplo típico de uma estação de controle de um UAV moderno incluiria observação da atitude do UAV e transmissão de alta velocidade. A tendência do desenvolvimento dos UAVs modernos se dá na direção de embarcar a maior quantidade possível de controle para que recaia sobre o operador poucas funções de controle de baixo nível, permitindo a ele controlar mais eficientemente e nos casos mais avançados, um só operador controlar diversas aeronaves com diferentes rotas e objetivos de voo a serem cumpridos. Tarefas como a estabilização, controle de atitude, controle de trajetória e até mesmo evitar a colisão com obstáculos podem ser processadas a bordo, sem que o operador precise interferir.

Tal nível de sofisticação no controle embarcado permite ainda que se diminua a quantidade de informações que devem ser enviadas à base e retornadas à aeronave, isso pode ser crucial já que um eventual atraso de envio ou perda de informação não alteraria o controle de estabilidade e atitude que exige uma taxa de atualização elevada. Isto permite que se empreguem sistemas de transmissão mais baratos e de alcance maior.

2.2 PLATAFORMAS

Aeronaves de asa fixa, de tamanho médio a grande requerem pistas para decolagem e pousos, a fim de desenvolver velocidade suficiente para gerar sustentação enquanto alguns UAVs são pequenos o suficiente para serem jogados estes ainda requer espaço suficiente na frente do lançador para a aeronave ganhar velocidade suficiente para decolar.

Proporcionar espaço suficiente para a aeronave decolar e pousar em terra é na maioria dos casos inconveniente e até mesmo impossível em algumas áreas urbanas ou densamente arborizadas. Para o caso de operação em locais aonde não há espaço para pouso pode ser utilizada uma rede para capturar a aeronave. Contudo, esta técnica imprime grandes esforços à estrutura da aeronave e não dificilmente danifica-a.



Figura 2– UAV para uso em pesquisas científicas, aplicações comerciais e estatais.

Além destas características, as aeronaves com asas fixas necessitam de uma velocidade mínima constante, e por isso não podem fazer curvas muito fechadas, voar para trás ou mesmo manter uma posição fixa. Devido a esta limitação, operações de vigilância exigem das aeronaves que se mantenham voando em círculo, e sobre um alvo um número de vezes. Devido a esta necessidade de grande espaço para decolar ou fazer complexas manobras, UAVs de asas fixas também não podem observar objetos em estreita proximidade, normalmente recorrem à manutenção de uma órbita de altitude sobre o objeto de interesse, exigindo uma câmera móvel para manter o objeto em vista e tirar fotos distantes. Embora isso funcione bem para a observação de uma vasta área, é difícil obter imagens detalhadas de um objeto de interesse sem uma pesada câmera de alta resolução e lente de alto ganho.



Figura 3 - Aeryon scout, produzido pela empresa canadense Aeryon labs, conta com uma poderosa câmera com zoom óptico de 10x, e uma câmera térmica utilizada para obter imagens noturnas. Pode atingir velocidades de até 50Km/h e persistir em uma posição estável mesmo em condições de ventos de até 80Km/h.

Uma plataforma de asas rotativas ou hUAV, fornece a capacidade de mover-se muito mais perto de um objeto, tirar fotos detalhadas, pairar no lugar, fazer curvas apertadas, e se mover em qualquer direção.

Inspeção rigorosa de materiais ou situações perigosos, vigilância em ambientes fechados ou ao ar livre, monitoramento estacionário de um objeto ou cena, e videografia estacionários são apenas algumas aplicações onde um hUAV poderia ser usado e um UAV convencional não poderia.

2.3 CLASSIFICAÇÃO DE UAVs

Os Veículos Aéreos não Tripulados podem ser classificados em função das características da aeronave. Existem dois tipos de classificação possíveis que podem ser devido ao desempenho ou à missão para que foram projectadas.

Quanto ao desempenho pode ser classificado de acordo com as seguintes características, peso, alcance, resistência do voo, altura máxima, carga máxima das asas, tipo de motor e potência.

- **Classificação pelo peso.**

Tabela 1– Classificação pelo peso

Categoria	Peso
Super Pesado	>2000Kg
Pesado	200-2000Kg
Médio	50-200Kg
Leve	5-50Kg
Micro	<5Kg

- **Classificação pelo alcance e resistência do voo**

Tabela 2– Classificação pelo alcance e resistência de voo

Categoria	Resistência	Alcance
Alta	>24h	>1500km
Média	5-24h	100-400km
Baixa	<5h	<100km

- **Classificação pela altura máxima**

Tabela 3– Classificação pela altura máxima

Categoria	Altura	Atmosfera
Alta	>10000m	Estratosfera
Média	1000-10000m	Troposfera
Baixa	<1000	Troposfera

- **Classificação pela carga máxima das Asas**

Tabela 4– Classificação pela carga das Asas

Categoria	Carga máxima da asa Kg/m²
Alta	>100
Média	50-100
Baixa	<50

Quanto á classificação do tipo de motor, esta pode ser extensa, dependendo das características do motor, do tipo de energia que utiliza, ou mesmo do tipo de aeronave e dos objectivos para a qual é projectada.

Os UAVS ainda podem ser classificados de acordo com a missão que cumprem, podendo ser:

- Vigilância e Reconhecimento.
- Combate Aéreo.
- Podem ainda englobar as duas características anteriores.
- Aterragem e descolagem vertical. (Utilizadas em operações de salvamento e transporte)
- Comunicações e radar.
- Entrega e reabastecimento de energia.

3 - MULTIROTOR

3.1 PLATAFORMAS MULTI-ROTORES

As plataformas multi-rotor são um alvo de interesse relativamente novo na área de controle. Os multi-rotor têm seis graus de liberdade, yaw, pitch, Roll, X (movimento na direção para frente do avião), y (movimento em direção ao lado esquerdo da nave) e z (altitude). Estes seis graus de liberdade devem ser controlados usando apenas quatro atuadores. Isto permite rotinas mais simples de controle (comandos iguais precisam ser enviados na mesma magnitude para todos os atuadores), mas propiciam uma área interessante de estudo sobre como separar o controle para permitir o controle estável de todos os seus seis graus de liberdade.

Plataformas multi-rotor fornecem uma perspectiva de design interessante também. Como seu uso nos últimos anos é pequeno para direcionar futuras aplicações, multirotores é uma área pouco explorada.

A simetria do projeto permite o escalonamento e a centralização dos sistemas de controle e do posicionamento da carga útil. Os rotores de um multi-cóptero permitem uma quantidade maior de empuxo do que um helicóptero típico o que permite payloads maiores e sistemas de controle embarcado mais complexos, especialmente importantes em aplicações de UAVs. Plataformas multi-rotor também podem ser facilmente cobertas com uma proteção de segurança, tornando-se mais seguros para operação em ambientes com pouco espaço, ou em ambientes populosos, o que um helicóptero padrão com grandes rotores expostos não poderia fazer.

3.2 QUADROTOR

O Quadrotor é uma aeronave de pequena dimensão, com uma configuração de asa rotativa, constituída por quatro motores e respectivos propulsores que asseguram a sua sustentação em voo. Os motores são normalmente instalados nos quatro cantos de uma estrutura cruzada possuindo no seu centro de massa todos os equipamentos de medição, controle, comunicação e energia. Esta aeronave pode ser totalmente autónoma em voo, pode ser semi-autónoma permitindo a um piloto com menos experiência controlar a aeronave, mas sempre com um sistema paralelo que assegure um voo estável ou ser totalmente manual permitindo ao piloto assegurar todas as manobras de controlo da aeronave.

Esta aeronave tem algumas vantagens em relação ao helicóptero convencional, sendo duas delas a facilidade de construção não existindo sistemas mecânicos complicados e frágeis e devido ao facto de possuir propulsores de menor dimensão o que facilita o voo em espaços reduzidos.

A mesma permite descolagens, aterragens verticais bem como manobras em espaços reduzidos com obstáculos podendo o seu controlo de estabilidade e posição se basear apenas na variação de velocidade dos quatro motores.

Podemos verificar que os motores 1 e 3 rodam no sentido contra horário e os motores 2 e 4 rodam no sentido horário, isto porque os motores em rotação criam não só forças verticais responsáveis pela sustentação mas também forças horizontais que criam um movimento de rotação do Quadrotor sobre o seu eixo central (Z). O facto de existir um par de motores rodando em num sentido e outro par a rodando em outro sentido cria duas forças horizontais contrárias permitindo assim controlar o movimento de rotação do Quadrotor sobre o seu eixo central aumentando assim a controlabilidade do ângulo Yaw em voo.

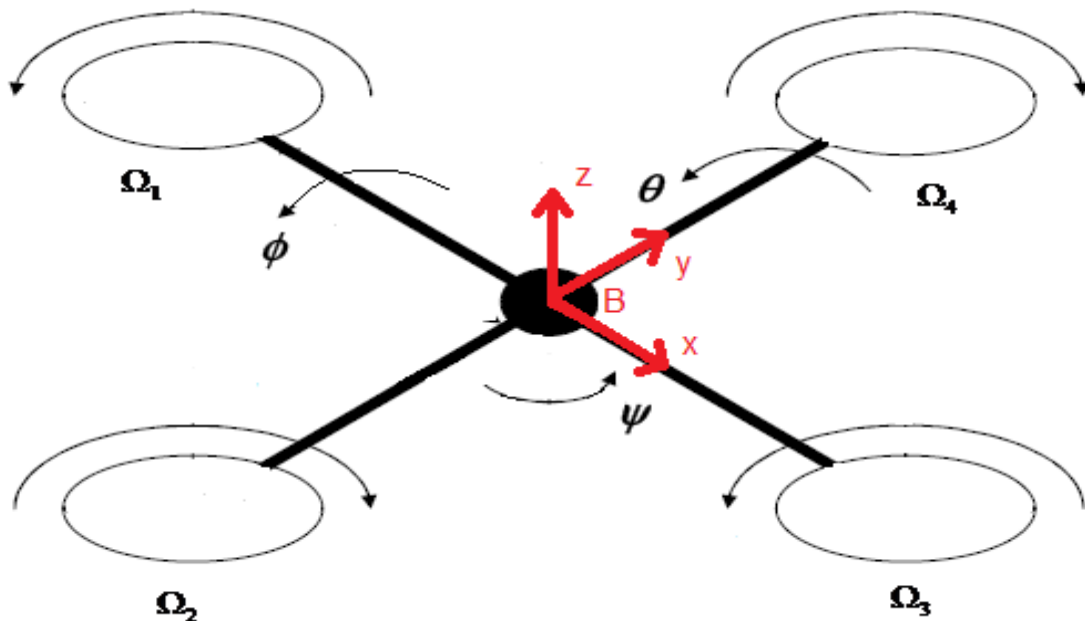


Figura 4– Sentido de rotação dos motores

3.3 APLICAÇÕES DO QUADROTOR

Este veículo aéreo está sofrendo um aumento contínuo de utilizadores e funcionalidades podendo ser utilizado nas mais diversas áreas desde, fotografia e vídeo, actividades governamentais, militar, industriais e educacionais.

Nas áreas de fotografia e vídeo, podem ser usados para observar vida selvagem, paisagens, e até criar anúncios promocionais reduzindo os custos e aumentando a produtividade.

Na área governamental pode ser usado num conjunto de acções e actividades extensas desde investigação de cenas onde ocorreram crimes, investigação de acidentes de tráfico, monitorização e análise do tráfico facilitando assim a deslocação de viaturas policiais ou de emergência, ajudando no desmantelamento de bombas voando sobre essas zonas identificando as áreas afetadas, controle de multidões facultando imagens aéreas que permitirão às autoridades o controle da situação, ajuda na observação e identificação de pessoas em zonas de desastres naturais permitindo o seu salvamento e controle de fogos identificando os focos e a propagação desses mesmos.

Na área militar tem aplicações estratégicas desde vigilância, reconhecimento espacial ou até de combate dependendo das suas características e dimensões.

Nas actividades industriais podem ter também diversas utilizações, sobretudo na construção de edifícios, pontes estradas e na manutenção dos mesmos podendo ser equipados com equipamentos para a detecção de falhas.

Estes veículos também podem ser usados por entidades educacionais desde testes aerodinâmicos a algoritmos de voo, descolagem, aterragem, exploração geológica e atmosférica e ambiente.



Figura 5 – **Dranganflyer** Possui uma IMU com acelerómetros e giroscópios, de 3 eixos.

3.4 PRINCIPIO DE FUNCIONAMENTO DO QUADROTOR

O Quadrotor possui uma estrutura em cruz, cujos braços estão dispostos com ângulos de 90° entre si, possuindo nas extremidades motores eléctricos com propulsores directamente acoplados responsáveis por forças horizontais e verticais.

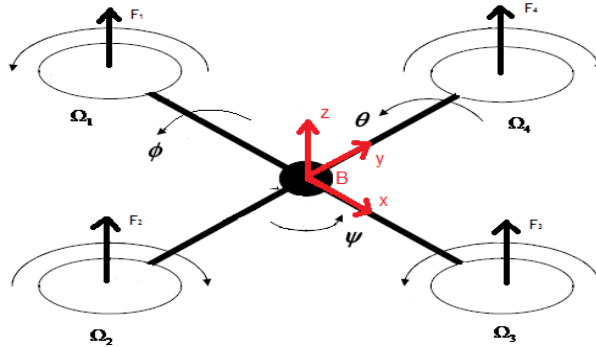


Figura 6 - Esquema das forças produzidas pela rotação dos propulsores a igual velocidade

Os motores e os propulsores são os responsáveis pela criação das forças que originam os movimentos da aeronave em voo. Desta maneira podemos afirmar que o controle da aeronave está directamente ligado ao controle de velocidade de cada um dos motores e respectivo propulsor.

Na Figura 6 observa-se o sentido de rotação dos propulsores e os vetores das forças criadas por cada um dos propulsores. Os vetores das forças criadas pelos propulsores são todos iguais pois giram todos á mesma velocidade. O facto de as forças serem iguais garante que os binários sobre os eixos, X, Y e Z sejam nulos garantindo assim estabilidade ao nível do *roll*, *pitch* e *yaw*.

- **Altitude (U_1 [N])**

Para garantir que o Quadrotor tenha um movimento vertical ascendente em relação á superfície terrestre basta garantir que o somatório das forças (1) produzidas pelos quatro propulsores seja superior ao peso P [N] da aeronave. No caso do movimento vertical descendente basta garantir que o somatório das forças produzidas pelos quatro propulsores seja inferior ao peso da aeronave.

$$U_1 = F_1 + F_2 + F_3 + F_4 \quad (1)$$

Condição para uma aceleração positiva (*Subida*): $P < U_1$

Condição para uma aceleração negativa (*Descida*): $P > U_1$

Garantindo as condições anteriores garantimos que a aeronave está subindo ou descendo, contudo é possível controlar a velocidade de subida e descida da aeronave através de incrementos $\Delta\Omega$ [$rad\ s^{-1}$] positivos ou negativos nas velocidades dos propulsores, controlando assim a respectiva força destes de igual forma. (Figura 7)

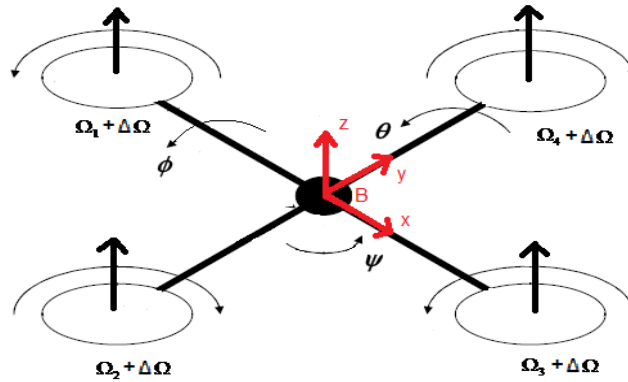


Figura 7– Movimento no eixo Z, Altitude.

- **Roll (U_2 [N m])**

Este comando é responsável pela criação de um binário em relação ao eixo X, permitindo que o Quadrotor rode em torno desse mesmo eixo. (2)

$$U_2 = F_4 - F_2 \quad (2)$$

Ao aumentar a velocidade no propulsor 4 e diminuir no propulsor 2 cria-se um movimento positivo. Uma diminuição da velocidade no propulsor 4 e respectivo aumento no propulsor 2 cria um movimento negativo. (Figura 8)

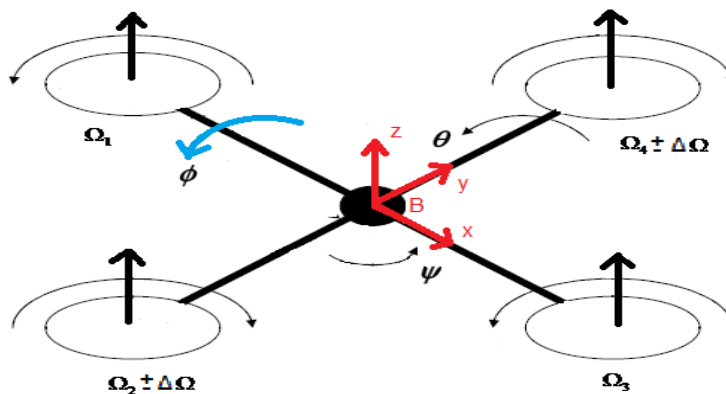


Figura 8– Roll (movimento a azul).

- **Pitch (U_3 [N m])**

Este comando é responsável pela criação de um binário em relação do eixo Y, permitindo que o Quadrotor rode em torno desse mesmo eixo. (3)

$$U_3 = F_3 - F_1 \quad (3)$$

Aumentando a velocidade no propulsor 3 e diminuindo no propulsor 1 cria-se um movimento positivo. Uma diminuição da velocidade no propulsor 3 e respectivo aumento no propulsor 1 cria um movimento negativo. (Figura 9)

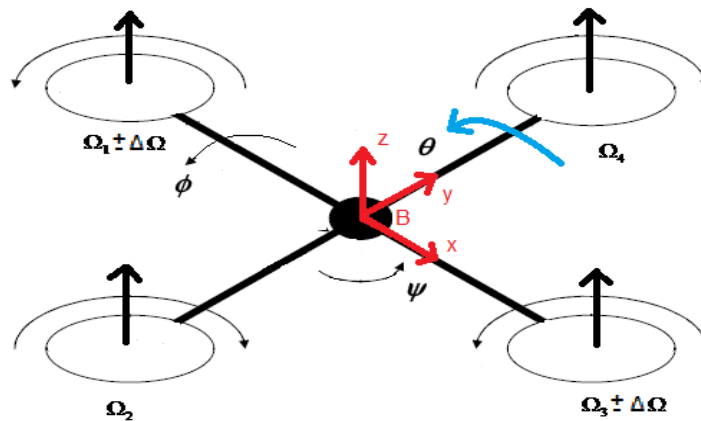


Figura 9– Pitch (movimento a azul)

- **Yaw (U_4 [N.m])**

Este comando é responsável pela criação de um binário em relação do eixo do Z, permitindo que o Quadrotor rode em torno desse mesmo eixo. (4)

$$U_4 = -F_1 + F_2 - F_3 + F_4 \quad (4)$$

Aumentando as velocidades dos propulsores pares e diminuindo a velocidade dos propulsores ímpares cria-se um movimento negativo (horário). Uma diminuição da velocidade nos propulsores pares e um respectivo aumento nos propulsores ímpares resultam na criação de um movimento positivo (contra horário). (Figura 10)

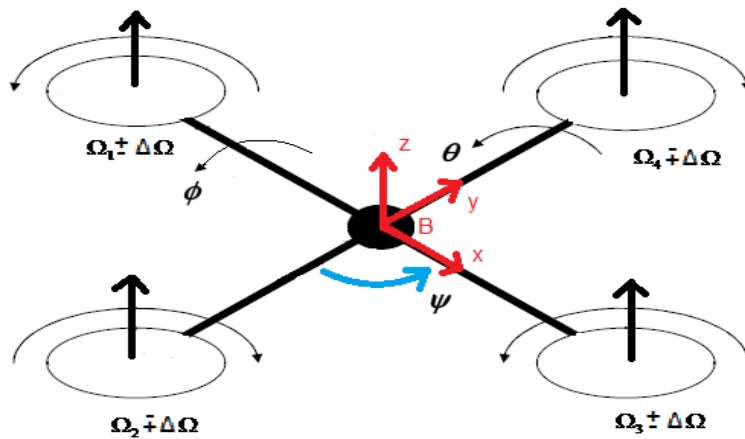


Figura 10– Yaw (movimento a azul)

3.5 Modelo de Newton – Euler

O modelo desenvolvido assume que o Quadrotor tem uma estrutura rígida e simétrica, o referencial fixado a essa mesma estrutura tem a sua origem a coincidir com o centro de massa da estrutura, os eixos do referencial fixado a essa estrutura coincidem com os eixos de inércia do veículo, os propulsores são rígidos e as forças originadas pelos propulsores são proporcionais ao quadrado da velocidade destes.

Antes da obtenção das respectivas equações que identificam o modelo é necessário primeiro identificar o sistema de coordenadas a ser usados. Dois sistemas de coordenadas são suficientes para analisar a dinâmica do Quadrotor.

- O referencial fixo Terra (E)
- O referencial fixo á estrutura do Quadrotor (B)

O referencial Terra (x_E, y_E, z_E) é um referencial fixo e o referencial do Quadrotor (x_B, y_B, z_B) é um referencial móvel, cuja dinâmica pode ser escrita em relação ao referencial Terra.

O primeiro referencial (E) está ligado á superfície da Terra com o eixo x_E apontando para Norte, o eixo y_E apontando para Leste e o eixo z_E apontando para cima em relação á Terra. Este referencial será usado para determinar a posição linear $r^E [m]$ e angular $\Theta^E [rad]$ do Quadrotor.

O segundo referencial (B) está ligado á estrutura do Quadrotor em que, x_B aponta para o propulsor 3, y_B aponta para o propulsor 4, z_B aponta para cima e a origem deste referencial coincide com o centro de massa do Quadrotor. Este referencial será usado para definir a velocidade linear $V^B [m s^{-1}]$, angular $\omega^B [rad s^{-1}]$, as forças $F^B [N]$ e os binários $M^B [N m]$.

A posição linear Γ^E do quadrotor é determinada pelas coordenadas do vector entre a origem do referencial E e a origem do referencial B em relação ao referencial E , de acordo com a seguinte equação (3.5).

$$\Gamma^E = [XYZ]^T \quad (5)$$

A Figura 11 mostra a relação entre os dois referenciais.

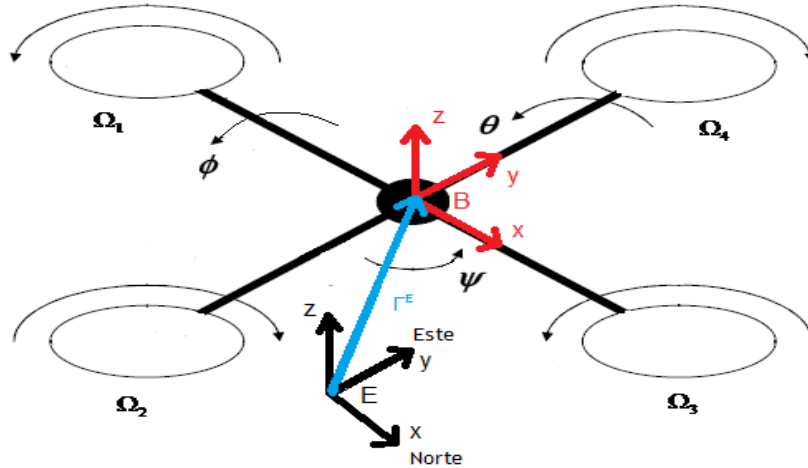


Figura 11– Referenciais B (vermelho), E (preto) e o vector posição linear Γ^E (azul)

A atitude Θ^E [rad] do Quadrotor é definida pelas rotações do referencial B em relação ao referencial E . Existem 3 rotações consecutivas do referencial B em relação ao E , ou seja, os movimentos *roll*, *pitch* e *yaw*.

A equação (6) mostra o vector da atitude.

$$\Theta^E = [\phi \ \theta \ \psi]^T \quad (6)$$

A matriz de rotação R_Θ do referencial B em relação E é obtida pela multiplicação das três matrizes de rotação dos 3 eixos.

- Rotação sobre o eixo z_E do ângulo ψ (*yaw*).

$$R(\psi, z) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7)$$

- Rotação sobre o eixo y_E do ângulo θ (*pitch*).

$$R(\theta, y) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad (8)$$

- Rotação sobre o eixo x_E do ângulo θ (*roll*).

$$R(\phi, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \quad (9)$$

Multiplicando as três matrizes resulta R_{Θ} (10) (11).

$$R_{\Theta} = R(\psi, z)R(\theta, z)R(\phi, z) \quad (10)$$

$$R_{\Theta} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \quad (11)$$

A velocidade linear V^B e a velocidade angular ω^B são expressas pelas seguintes equações (12) (13) respectivamente.

$$V^B = [uvw]^T \quad (12)$$

$$\omega^B = [pqr]^T \quad (13)$$

Combinando as grandezas lineares e angulares, obtêm-se dois vectores que representam de forma generalizada o vector da posição ξ e da velocidade do Quadrotor v de acordo com as seguintes equações (14) (15).

$$\xi = [\Gamma^E \Theta^E]^T = [XYZ\phi\theta\psi]^T \quad (14)$$

$$v = [V^B \omega^B]^T = [uvw pqr]^T \quad (15)$$

A relação entre a velocidade no referencial B e a velocidade no referencial E é expressa da seguinte forma (16).

$$\dot{\Gamma}^E = R_{\Theta} V^B \quad (16)$$

A mesma relação é possível para a velocidade angular através da matriz de transferência T_{Θ} . A equação seguinte mostra essa relação (17).

$$\dot{\Theta}^E = T_{\Theta} \omega^B \quad (17)$$

A matriz de transferência pode ser determinada a partir da resolução da seguinte equação (18) (19) (20).

$$\omega^B = T_{\Theta}^{-1} \dot{\Theta}^E \quad (18)$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R(\phi, x)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi, x)^{-1} R(\phi, y)^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = T_{\Theta}^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (19)$$

$$T_{\Theta} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \frac{\sin \phi}{\cos \theta} & \frac{\cos \phi}{\cos \theta} \end{bmatrix} \quad (20)$$

Desta maneira, juntando tudo numa única expressão resulta a velocidade linear e angular do referencial E a partir da velocidade linear e angular do referencial B resulta (21).

$$\dot{\xi} = J_{\Theta} v \quad (21)$$

A matriz J_{Θ} é expressa pela seguinte equação (3.22).

$$J_{\Theta} = \begin{bmatrix} R_{\Theta} & 0_{3 \times 3} \\ 0_{3 \times 3} & T_{\Theta} \end{bmatrix} \quad (22)$$

O modelo do sistema do Quadrotor terá doze estados, seis estados para a posição e atitude, outros seis para a velocidade linear e angular, representando assim os seis graus de liberdade do Quadrotor.

A fim de obter o modelo dinâmico do Quadrotor é necessário aplicar o formalismo de Newton – Euler, cujas equações representam o total das forças

externas e momentos de inércia que actuam que actuam no centro de massa da estrutura do Quadrotor (23).

$$\begin{bmatrix} F^B \\ M^B \end{bmatrix} = \begin{bmatrix} \omega^B \times (mV^B) \\ \omega^B \times (I\omega^B) \end{bmatrix} + \begin{bmatrix} mI_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} \begin{bmatrix} \dot{V}^B \\ \dot{\omega}^B \end{bmatrix} \quad (23)$$

Generalizando, pode escrever-se da seguinte forma. (24)

$$M_B \dot{v} + C_B(v) = \Lambda \quad (24)$$

Onde M_B é a matriz que contém a diagonal constituída pela massa do veículo e pelos momentos de inércia dos eixos X, Y e Z do veículo (25).

$$M_B = \begin{bmatrix} mI_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \quad (25)$$

Por sua vez a matriz $C_B(v)$ é a matriz que resulta da resolução dos produtos vectoriais entre os vectores (26).

$$C_B(v) = \begin{bmatrix} \omega^B \times (mV^B) \\ \omega^B \times (I\omega^B) \end{bmatrix} = \begin{bmatrix} mwq - mvr \\ -mwp + mur \\ mvp - muq \\ I_{zz}rq - I_{yy}qr \\ -I_{zz}rp + I_{xx}pr \\ I_{yy}qp - I_{xx}pq \end{bmatrix} \quad (26)$$

O vector Λ contém as forças que actuam no Quadrotor (27).

$$\Lambda = [F^B M^B]^T = [F_x F_Y F_Z M_x M_Y M_Z]^T \quad (27)$$

As forças que actuam no Quadrotor podem ser divididas em três componentes, de acordo com a dinâmica deste.

A primeira componente resulta do efeito da aceleração da gravidade $g [m s^{-2}]$ no Quadrotor. Este efeito está directamente ligado aos parâmetros lineares excluindo qualquer efeito sobre os parâmetros angulares. Desta maneira pode escrever-se a seguinte equação (28) a fim de obter $G_B(\xi)$.

$$G_B(\xi) = \begin{bmatrix} F_G^B \\ 0_{3 \times 1} \end{bmatrix} = \begin{bmatrix} R_\Theta^{-1} F_G^E \\ 0_{3 \times 1} \end{bmatrix} = \begin{bmatrix} R_\Theta^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \\ 0_{3 \times 1} \end{bmatrix} = \begin{bmatrix} mg \sin \theta \\ -mg \cos \theta \sin \phi \\ -mg \cos \theta \sin \phi \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (28)$$

Onde F_G^B é a força gravítica em relação ao referencial B e F_G^E em relação ao referencial E . Outra propriedade que surgiu na equação (28) foi o facto de a inversa da matriz R_Θ ser igual á transposta da mesma, pois é uma matriz ortogonal normalizada.

A segunda componente deve-se ao efeito giroscópio produzido pela rotação dos propulsores. Este efeito surge quando a soma da velocidade dos quatro propulsores não é igual a zero e quando a velocidade angular sobre o *roll* e o *pitch* são diferentes de zero.

Surge assim a seguinte equação (29).

$$G_{Bgyro}(v)\Omega = \begin{bmatrix} 0_{3 \times 1} \\ -\sum_{i=1}^{i=4} J_T \left(\omega^B \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) (-1)^i \Omega_i \end{bmatrix} = \begin{bmatrix} 0_{3 \times 1} \\ J_T \begin{bmatrix} -q \\ p \\ 0 \end{bmatrix} \end{bmatrix} \Omega \quad (29)$$

Onde $J_T [N m s^2]$ é o momento de inércia total dos propulsores e $\Omega [rad s^{-1}]$ é a matriz de velocidades dos diversos propulsores (30). Observa-se que os efeitos giroscópios estão directamente relacionados com os movimentos angulares, daí os primeiros três elementos da coluna da matriz ser nula, ou seja, não existe relação com os movimentos lineares.

$$\Omega = \begin{bmatrix} -\Omega_1 \\ \Omega_2 \\ -\Omega_3 \\ \Omega_4 \end{bmatrix} \quad (30)$$

Onde $\Omega_1 [rad s^{-1}]$ corresponde a velocidade do propulsor 1, $\Omega_2 [rad s^{-1}]$ corresponde a velocidade do propulsor 2, $\Omega_3 [rad s^{-1}]$ corresponde a velocidade do propulsor 3 e $\Omega_4 [rad s^{-1}]$ corresponde a velocidade do propulsor 4.

A terceira componente está relacionada com as forças produzidos pela rotação dos propulsores. A próxima equação tem em conta a distância l [m] dos propulsores ao centro de massa do Quadrotor, o coeficiente de impulso vertical c [$N s^2$] e o coeficiente de arrastamento d [$N m s^2$] criado pela rotação dos propulsores.

A equação (31) apresenta as forças criadas.

$$U_B(\Omega) = \begin{bmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ c(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ cl(\Omega_4^2 - \Omega_2^2) \\ cl(\Omega_3^2 - \Omega_1^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \quad (31)$$

Encontradas as três componentes é possível reescrever a equação (23) descrevendo assim a dinâmica do Quadrotor (32).

$$M_B \dot{v} + C_B(v) = G_B(\xi) + G_{Bgyro}(v)\Omega + U_B(\Omega) \quad (3.32)$$

Escrevendo em ordem á aceleração \dot{v} do referencial B , resulta a equação (33).

$$\dot{v} = M_B^{-1} (G_B(\xi) + G_{Bgyro}(v)\Omega + U_B(\Omega) - C_B(v)) \quad (33)$$

A partir da resolução da equação anterior (33) resulta o seguinte sistema de equações referente ao referencial B (34).

$$\begin{cases} \dot{u} = (vr - wq) + g\sin\theta \\ \dot{v} = (wp - ur) - g\cos\theta\sin\phi \\ \dot{w} = (uq - vp) - g\cos\theta\sin\phi + \frac{U_1}{m} \\ \dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}} qr - \frac{J_T}{I_{xx}} q\Omega + \frac{U_2}{I_{xx}} \\ \dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}} pr + \frac{J_T}{I_{yy}} p\Omega + \frac{U_3}{I_{yy}} \\ \dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{U_4}{I_{zz}} \end{cases} \quad (34)$$

Em que Ω [$rad\ s^{-1}$] corresponde á seguinte equação (35).

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \quad (35)$$

A equação (34) representa a dinâmica do Quadrotor mas em relação ao referencial B , contudo será necessário escrever as equações lineares em ordem ao referencial E e as equações angulares em ordem ao referencial B , isto porque será mais fácil expressar a aceleração linear no referencial E em termos da identificação dos parâmetros de controlo. Desta maneira foi criado um referencial H que combina os movimentos lineares no referencial E e os movimentos angulares no referencial B .

A equação (36) mostra a velocidade ζ no referencial H .

$$\zeta^H = \begin{bmatrix} \dot{\Gamma}^E \omega^B \end{bmatrix}^T = \begin{bmatrix} \dot{X} \dot{Y} \dot{Z} p q r \end{bmatrix}^T \quad (36)$$

Escrevendo a equação que representa a dinâmica do Quadrotor para o referencial H , resulta (37).

$$\begin{bmatrix} F^E \\ M^B \end{bmatrix} = \begin{bmatrix} 0_{3 \times 3} \\ \omega^B \times (I \omega^B) \end{bmatrix} + \begin{bmatrix} m I_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} \begin{bmatrix} \dot{\zeta}^H \end{bmatrix} \quad (37)$$

Estruturando em ordem á aceleração (38).

$$\dot{\zeta}^H = M_H^{-1} (G_H + G_{Hgyro}(\zeta)\Omega + U_H(\Omega) - C_H(\zeta)) \quad (38)$$

As equações em 25, 26, 27, 28 e 29 foram escritas em relação ao referencial B , deste modo e como foi definido um novo referencial que combina o referencial E e B para descrever o modelo do Quadrotor será necessário escrever essas mesmas equações em ordem a este referencial (*referencial H*).

Escrevendo a matriz M_H em ordem ao referencial H resulta (39).

$$M_H = M_B = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \quad (39)$$

A matriz M_H não sofre qualquer alteração pois apresenta parâmetros físicos da estrutura do Quadrotor que não são alterados. Por sua vez a matriz C_H sofre alterações (40).

$$C_H(\zeta) = \begin{bmatrix} 0_{3 \times 3} \\ \omega^B \times (I \omega^B) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ I_{zz}rq - I_{yy}qr \\ -I_{zz}rp + I_{xx}pr \\ I_{yy}qp - I_{xx}pq \end{bmatrix} \quad (40)$$

A matriz G_H apresenta o efeito da gravidade no referencial H, este efeito apenas se faz sentir no eixo Z. (41)

$$G_H(\zeta) = \begin{bmatrix} F_G^E \\ 0_{3 \times 1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (41)$$

Os efeitos giroscópios não sofrem qualquer alteração logo a matriz será igual (42)

$$G_{Hgyro}(\zeta)\Omega = \begin{bmatrix} 0_{3 \times 1} \\ J_T \begin{bmatrix} -q \\ p \\ 0 \end{bmatrix} \end{bmatrix} \Omega \quad (42)$$

As forças produzidas pelas rotações dos propulsores vão sofrer uma alteração ao nível da força total U_1 no referencial H, sofrerá o efeito da rotação do referencial B em relação ao referencial E. Esta rotação é expressa pela matriz de rotação R_Θ multiplicada pela força U_1 . (43)

$$U_H(\Omega) = \begin{bmatrix} R_\Theta & 0_{3 \times 3} \\ 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} U_B(\Omega) = \begin{bmatrix} (\sin \psi \sin \phi + \cos \psi \sin \theta \cos \phi) U_1 \\ (-\cos \psi \sin \phi + \sin \psi \sin \theta \cos \phi) U_1 \\ \cos \theta \cos \phi U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \quad (43)$$

Resolvendo a equação (38) resulta o seguinte sistema de equações (44) que descreve a dinâmica do Quadrotor em relação ao referencial H.

$$\left\{ \begin{array}{l} \ddot{X} = (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi) \frac{U_1}{m} \\ \ddot{Y} = (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi) \frac{U_1}{m} \\ \ddot{Z} = -g + (\cos\theta\cos\phi) \frac{U_1}{m} \\ \dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}} qr - \frac{J_T}{I_{xx}} q\Omega + \frac{U_2}{I_{xx}} \\ \dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}} pr + \frac{J_T}{I_{yy}} p\Omega + \frac{U_3}{I_{yy}} \\ \dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{U_4}{I_{zz}} \end{array} \right.$$

4 - SENSORIAMENTO E CONTROLE

PID

4.1 SENSORES USADOS PARA ESTABILIDADE

O principal sensor utilizado em controle de estabilidade de UAVs é a IMU ou central de medições inércias. Ela tipicamente possui 6 ou 9 graus de liberdade, sendo a de 6 composta por um giroscópio e um acelerômetro de 3 eixos (X, Y, Z) e a de 9 graus de liberdade, que possui além dos mesmos giroscópios e acelerômetros uma bússola eletrônica ou magnetômetro de 3 eixos.

Os magnetômetros de três eixos são pouco utilizados para o controle de UAVs de pequeno porte, sobretudo nas plataformas do tipo multi-rotor porque os componentes eletrônicos, principalmente os controladores de velocidade dos motores elétricos, e os próprios motores elétricos geram grandes distorções no campo magnético percebido por estes sensores. Os motores elétricos utilizados nessas aeronaves são geralmente de alto desempenho e utilizam ímãs conhecidos como terra-rara que tem altíssimo grau de magnetismo. Além disto, as correntes tipicamente envolvidas nestas aeronaves de pequeno porte estão na ordem de grandeza de dezenas de amperes, chegando até mesmo a ultrapassar uma centena de amperes nos modelos de maior porte. Esta corrente alta é chaveada (PWM) pelos controladores de velocidade gerando também uma grande quantidade de interferências magnéticas. Devido às pequenas dimensões destes modelos, é impossível distanciar os sensores das fontes de interferência o que torna praticamente inviável o seu uso.

Os acelerômetros de 3 eixos medem as acelerações em cada um dos eixos separadamente, ou seja, a cada instante de medição ele “imprime” na sua saída 3 valores de acelerações, respectivamente x, y, z. Por exemplo, um acelerômetro estático, parado sobre uma mesa mediria apenas uma aceleração de 1g em z negativo. Isto se deve ao fato de que os acelerômetros medem a força inercial, abaixo está um exemplo gráfico de seu funcionamento:

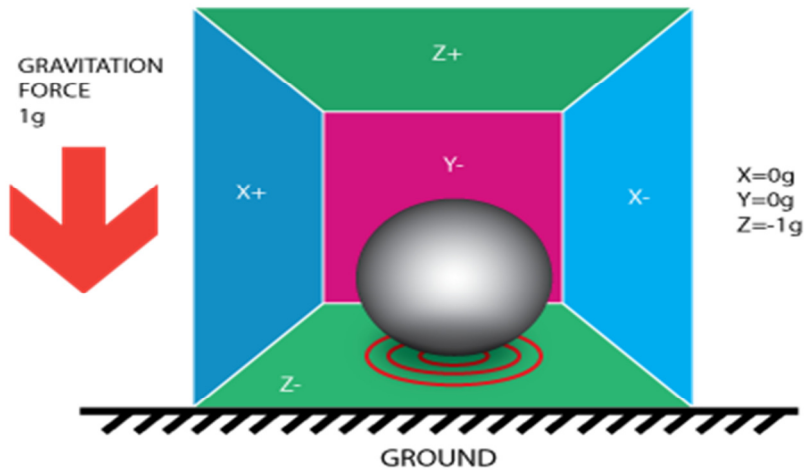


Figura 12– funcionamento do acelerometro.

Caso haja mais de uma força agindo sobre a IMU ou apenas uma força que não esteja agindo paralela a um dos eixos coordenados, haverá uma decomposição vetorial das forças de aceleração. Para uma aceleração atuando em R como no modelo abaixo teríamos;

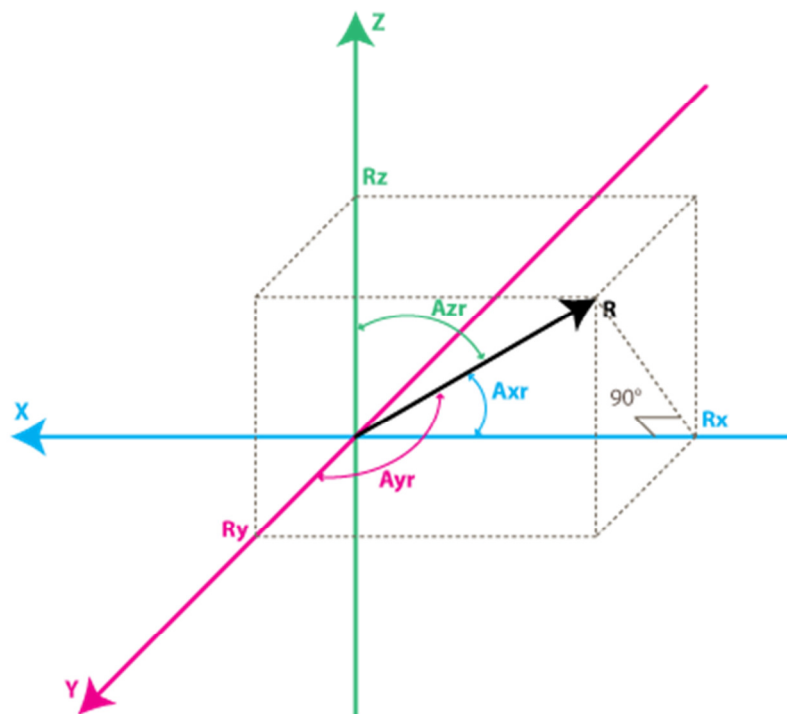


Figura 13– funcionamento do acelerometro.

Aonde:

$$R^2 = R_x^2 + R_y^2 + R_z^2$$

$$\cos(A_{xr}) = R_x / R$$

$$\cos(A_{yr}) = R_y / R$$

$$\cos(A_{zr}) = R_z / R$$

$$A_{xr} = \arccos(R_x/R)$$

$$A_{yr} = \arccos(R_y/R)$$

$$A_{zr} = \arccos(R_z/R)$$

$$\cos X = \cos(A_{xr}) = R_x / R$$

$$\cos Y = \cos(A_{yr}) = R_y / R$$

$$\cos Z = \cos(A_{zr}) = R_z / R$$

$$\text{SQRT}(\cos X^2 + \cos Y^2 + \cos Z^2) = 1$$

Juntamente com o acelerômetro, o outro principal sensor que compõe uma IMU é o giroscópio ou simplesmente gyro. Ele é responsável por medir as taxas de giro, ou velocidades angulares da central.

$$\text{RateAxz} = (A_{xz1} - A_{xz0}) / (t_1 - t_0).$$

Suas medições se dão em componentes desta medição para cada um dos eixos, assim como as medições feitas pelo acelerômetro.

O uso do giroscópio se faz necessário porque os dados do acelerômetro não são suficientes para se medir a atitude de um corpo em movimento no espaço. Como visto acima, para um corpo estático apenas com as informações do acelerômetro seria possível obter a atitude, mas como ele também medirá as acelerações causadas pelo movimento do corpo, caso exista, e medirá ruídos intrínsecos a qualquer medição, devemos utilizar o sinal de um gyro para corrigir esta medição. Esta é a principal razão pela qual as maiorias dos sistemas IMU usam um giroscópio para suavizar os erros acelerômetro.

O giroscópio por sua vez, também não está livre de ruídos, no entanto como as medidas de rotação são menos sensíveis a movimentos mecânicos lineares, o tipo de ruído que acelerômetro sofre, pode auxiliar na compensação de erros. Por outro lado, os giroscópios têm outros tipos de problemas como o desvio, por exemplo: Não voltar a taxa de valor zero quando a rotação cessa. No entanto por dados acoplados que vem do acelerômetro e giroscópio podemos obter uma estimativa relativamente melhor da inclinação ou atitude do dispositivo atual do que poderíamos obter usando apenas os dados do acelerômetro ou do giroscópio sozinhos.

4.2 FILTROS

Existem diversos modelos de filtros de controle que podem ser implementados de forma a obter medições mais precisas de atitude em aeronaves. Cada um deles tem as suas peculiaridades, dificuldades e benefícios. Para o uso em um UAV com processamento embarcado que é o caso deste estudo, devemos procurar soluções que não dependam de modelos extremamente fiéis do objeto a ser controlado já que isto pode ser difícil de ser levantado de um modelo possível de criado em um laboratório universitário. Deve-se também evitar modelos de filtros que exijam grandes quantidades de cálculos do processador, ou a manipulação de vastas matrizes já que isto encareceria o processador a ser especificado para o projeto e demandaria mais custos de energia o que levaria a mais peso inerente ao sistema de controle, fato que deve sempre ser evitado no projeto de aeronaves.

Um algoritmo interessante pela sua simplicidade, que foi inspirado por algumas idéias usadas em filtro de Kalman, mas, no entanto, é de longe mais simples e mais fácil de implementar em dispositivos embarcados será apresentado abaixo.

A princípio vamos definir o que queremos que o nosso algoritmo calcule ou estime. O principal é a direção do vetor força gravitacional $R = [R_x, R_y, R_z]$ a partir do qual podemos derivar os outros valores como A_x, A_y, A_z or $\cos X, \cos Y, \cos Z$ que dará uma idéia sobre a inclinação do dispositivo em relação ao referencial fixo (terra).

Sendo:

$$R_{acc} = [R_{xAcc}, R_{yAcc}, R_{zAcc}]$$

O vetor de acelerações medidas pelo acelerômetro e:

$$R_{acc}(\text{normalizado}) = [R_{xAcc}/|R_{acc}|, R_{yAcc}/|R_{acc}|, R_{zAcc}/|R_{acc}|]$$

O vetor de acelerações medidas pelo acelerômetro normalizado. Este será o vetor de entrada do filtro já que é um vetor de medições. Este filtro terá como saída o vetor de acelerações estimadas:

$$R_{est} = [R_{xEst}, R_{yEst}, R_{zEst}]$$

Temos como condição inicial que o vetor de saída será igual ao vetor de entrada:

$$R_{est}(0) = R_{acc}(0)$$

Aonde $R_{est}(t)$ é o valor do vetor em um tempo discreto genérico "t"

Para a correção do valor da aceleração pelo filtro será usado a informação de taxa de giro obtida pela leitura do giroscópio:

$$R_{gyro} = [R_xGyro, R_yGyro, R_zGyro]$$

Observando a relação no modelo do gyro acima, do ângulo à direita do triângulo formado R_z and R_xz podemos calcular:

$$\tan(A_{xz}) = R_x/R_z \Rightarrow A_{xz} = \text{atan2}(R_x, R_z)$$

Aonde atan2 é uma função trigonométrica similar à arcotangente (atan) mas que retorna valores entre $(-\pi, \pi)$ e não $(-\pi/2, \pi/2)$ como a arcotangente e tem dois argumentos ao invés de um. Esta função permite converter dois valores de R_x, R_z para ângulos entre $(-\pi$ to $\pi)$.

Conhecendo os valores de $R_{xEst}(n-1)$, e $R_{zEst}(n-1)$ obtém-se:

$$A_{xz}(n-1) = \text{atan2}(R_{xEst}(n-1) , R_{zEst}(n-1)).$$

Como o gyro mede a taxa de variação do angulo A_{xz} . Pode-se estimar o novo valor do ângulo (n) como:

$$A_{xz}(n) = A_{xz}(n-1) + \text{Rate}A_{xz}(n) * T$$

Pode-se ainda aplicar um filtro que calcule a média dos valores medidos a fim de reduzir os efeitos de ruídos de medição:

$$\text{Rate}A_{xz}Avg = (\text{Rate}A_{xz}(n) + \text{Rate}A_{xz}(n-1)) / 2$$

$$A_{xz}(n) = A_{xz}(n-1) + \text{Rate}A_{xz}Avg * T$$

Das equações acima, podemos obter o modulo do vetor R_{gyro} pela equação:

$$|R_{gyro}| = \text{SQRT}(R_xGyro^2 + R_yGyro^2 + R_zGyro^2)$$

Como este vetor foi normalizado antes, pode ser reescrito como:

$$|R_{gyro}| = 1$$

Convencionando para facilidade de notação:

$$x = R_xGyro , y = R_yGyro, z = R_zGyro$$

E usando as equações logo acima temos:

$$x = x / |R_{gyro}| = x / \text{SQRT}(x^2 + y^2 + z^2)$$

Dividindo pelo módulo $\text{SQRT}(x^2 + z^2)$ para normalizar:

$$x = (x / \text{SQRT}(x^2 + z^2)) / \text{SQRT}((x^2 + y^2 + z^2) / (x^2 + z^2))$$

Sendo:

$$x / \text{SQRT}(x^2 + z^2) = \sin(\text{Axz})$$

Então:

$$x = \sin(\text{Axz}) / \text{SQRT} (1 + y^2 / (x^2 + z^2))$$

Multiplicando o numerador e denominador dentro da raiz quadrada por z^2 :

$$x = \sin(\text{Axz}) / \text{SQRT} (1 + y^2 * z^2 / (z^2 * (x^2 + z^2)))$$

Como:

$$z / \text{SQRT}(x^2 + z^2) = \cos(\text{Axz}) \text{ e } y / z = \tan(\text{Ayz})$$

Então:

$$x = \sin(\text{Axz}) / \text{SQRT} (1 + \cos(\text{Axz})^2 * \tan(\text{Ayz})^2)$$

O que voltando à notação inicial pode ser reescrito como:

$$RxGyro = \sin(\text{Axz}(n)) / \text{SQRT} (1 + \cos(\text{Axz}(n))^2 * \tan(\text{Ayz}(n))^2)$$

Assim como:

$$RyGyro = \sin(\text{Ayz}(n)) / \text{SQRT} (1 + \cos(\text{Ayz}(n))^2 * \tan(\text{Axz}(n))^2)$$

Como dito anteriormente, ao se fazer uso de processamento de sinais e controle embarcados na aeronave é sempre interessante que se leve em conta o custo computacional. Levando isto em conta podemos simplificar as equações a seguir:

Dividindo as duas partes da fração por **(Axz(n))**:

$$RxGyro = 1 / \text{SQRT} (1 / \sin(\text{Axz}(n))^2 + \cos(\text{Axz}(n))^2 / \sin(\text{Axz}(n))^2 * \tan(\text{Ayz}(n))^2)$$

$$RxGyro = 1 / \text{SQRT} (1 / \sin(\text{Axz}(n))^2 + \cot(\text{Axz}(n))^2 * \sin(\text{Ayz}(n))^2 / \cos(\text{Ayz}(n))^2)$$

Desenvolvendo:

$$\cos(\text{Axz}(n))^2 / \sin(\text{Axz}(n))^2 = \cot(\text{Axz}(n))^2$$

Temos:

$$R_{xGyro} = 1 / \text{SQRT} (1/ \sin(A_{xz}(n))^2 - \cos(A_{xz}(n))^2 / \sin(A_{xz}(n))^2 + \cot(A_{xz}(n))^2 * \sin(A_{yz}(n))^2 / \cos(A_{yz}(n))^2 + \cot(A_{xz}(n))^2)$$

Agrupando os termos da equação:

$$R_{xGyro} = 1 / \text{SQRT} (1 + \cot(A_{xz}(n))^2 * \sec(A_{yz}(n))^2)$$

onde:

$$\cot(x) = 1 / \tan(x) \text{ and } \sec(x) = 1 / \cos(x)$$

Esta última fórmula utiliza somente duas funções trigonométricas, o que resulta em um processamento mais eficiente do que a escrita anteriormente.

Considerando:

$$R_{zGyro} = \text{Sign}(R_{zGyro}) * \text{SQRT}(1 - R_{xGyro}^2 - R_{yGyro}^2).$$

Aonde $\text{Sign}(R_{zGyro}) = 1$ quando $R_{zGyro} \geq 0$, e $\text{Sign}(R_{zGyro}) = -1$ quando $R_{zGyro} < 0$.

O que pode ser estimado fazendo-se:

$$\text{Sign}(R_{zGyro}) = \text{Sign}(R_{zEst}(n-1))$$

Deve-se ter cuidado quando $R_{zEst}(n-1)$ assume valores próximos de 0. O filtro pode pular de sentido de giro neste caso, e atribuir:

$$R_{gyro} = R_{est}(n-1).$$

R_z é usado como uma referência para o cálculo dos ângulos A_{xz} e A_{yz} e quando está próximo de 0, os valores podem explodir e provocar resultados errados. Estas contas os cálculos das funções $\tan()$ / $\text{atan}()$ geralmente não tem precisão.

Assim, em um tempo genérico "t" temos.

R_{acc} – Leitura atual do acelerômetro

R_{gyro} – Calculado pelo $R_{est}(n-1)$ e a leitura atual do gyro.

Para o cálculo de $R_{est}(n)$ utilizaremos uma relação de pesos w_1 e w_2 :

$$R_{est}(n) = (R_{acc} * w_1 + R_{gyro} * w_2) / (w_1 + w_2)$$

O que pode ser simplificado como:

$$\text{Rest}(n) = (\text{Racc} * w1/w1 + \text{Rgyro} * w2/w1) / (w1/w1 + w2/w1)$$

Que convencionando a relação:

$$w2/w1 = w\text{Gyro}$$

Obtém-se:

$$\text{Rest}(n) = (\text{Racc} + \text{Rgyro} * w\text{Gyro}) / (1 + w\text{Gyro})$$

Aonde $w\text{Gyro}$ é um coeficiente que indica o quão confiável é a medição do gyro em comparação com a do acelerômetro. Este valor é usualmete obtido experimentalmente, mas para a maioria dos hUAVs de pequeno porte utilizando IMUs convencionais assume valores entre 5 e 20.

Simplificadamente, a diferença deste modelo de filtro para o filtro de Kalman é que neste modelo apresentado este coeficiente é fixo e no modelo de filtro de Kalman ele é atualizado a cada iteração de controle baseado nas informações de ruído vindas das medições do. O filtro de Kalman convencional proporcina um controle melhor mas é mais caro computacionalmente e depende de informações sobre os sensores que usualmente são difíceis de ser obtidas em um ambiente de laboratório de universidade. Além disso, para projetos não muito complexos este filtro “adaptado” é satisfatório.

Além disso, pode ser interessante incluir neste algoritmo apresentado uma lei de atualização deste coeficiente segundo informações dos sensores sem muita dificuldade.

Assim:

$$\text{RxEst}(n) = (\text{RxAcc} + \text{RxGyro} * w\text{Gyro}) / (1 + w\text{Gyro})$$

$$\text{RyEst}(n) = (\text{RyAcc} + \text{RyGyro} * w\text{Gyro}) / (1 + w\text{Gyro})$$

$$\text{RzEst}(n) = (\text{RzAcc} + \text{RzGyro} * w\text{Gyro}) / (1 + w\text{Gyro})$$

Aonde, normalizando temos:

$$R = \text{SQRT}(\text{RxEst}(n)^2 + \text{RyEst}(n)^2 + \text{RzEst}(n)^2)$$

E:

$$\text{RxEst}(n) = \text{RxEst}(n)/R$$

$$\text{RyEst}(n) = \text{RyEst}(n)/R$$

$$\text{RzEst}(n) = \text{RzEst}(n)/R$$

Na figura abaixo pode-se observar a diferença entre as leituras no tempo de um acelerômetro com a leitura após o processamento com o uso do filtro acima descrito:

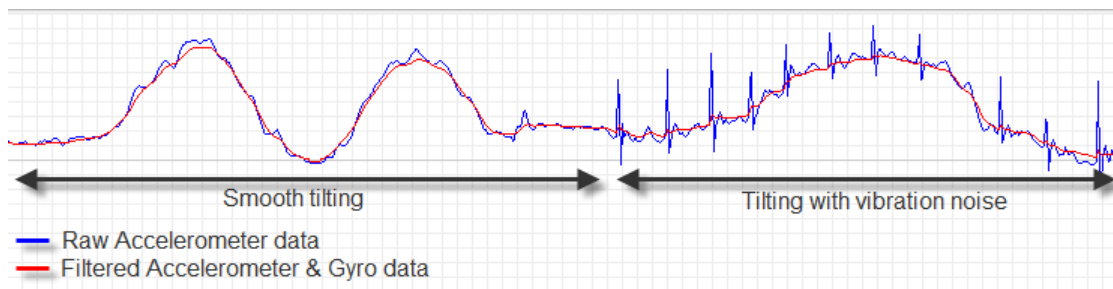


Figura 14 - Comparação das leituras acelerometro e gyro .

Pode-se notar uma grande melhora no sinal, o que se torna ainda mais evidente e importante na presença de ruídos (à direita).

4.3 CONTROLE PID:

As compensações de erros de atitude para aeronaves do tipo multi-rotor exigem em geral apenas cálculos para pequenos ângulos uma vez que por ser uma plataforma inerentemente instável, um grande ângulo de atitude pode ser dificilmente revertido ou compensado, assim, estes ângulos nunca irão chegar a um ângulo em que a formulação de Euler se aproximaria das condições de singularidade. Por este motivo, uma formulação baseada na teoria de quatérnios torna-se complexa demais propiciando poucos ganhos em relação à modelagem clássica com ângulos de Euler.

O controle do tipo PID consiste em uma lei de controle composta por 3 constantes K_p , K_d , K_i , que multiplicam os erros, as derivadas dos erros e suas integrais. Para alguns casos, a constante K_i pode ser suprimida, transformando o controle em um do tipo PD que é satisfatório para diversos tipos de hUAVs.

As equações que regem este tipo de controle são:

$$effort_d = K_d * \frac{\delta error_s}{\delta t}, \delta error_s = (error_{s_t} - error_{s_{t-1}})$$

$$effort_i = K_i * \delta t * error_s + effort_{i_{t-1}}$$

$$effort_p = K_p * error_s$$

Aonde:

$$error_s \triangleq \begin{pmatrix} error_x \\ error_y \\ error_z \\ error_\theta \\ error_\phi \\ error_\psi \end{pmatrix}$$

$$error_{\{x,y\}} = desired_{\{x,y\}} - measured_{\{x,y\}}$$

$$error_{\{\psi,z\}} = desired_{\{\psi,z\}} - measured_{\{\psi,z\}} + trim_{\{\psi,z\}}$$

A estrutura inicial do PID para um multi-rotor tem como função computar correções para os erros de pitch e roll, e converter estes erros em um valor que possa ser enviado aos controladores de cada um dos motores. Já que neste tipo de aeronave é a diferença dos empuxos dos motores que atuam sobre ele para movimenta-lo, como pode ser visto na figura abaixo para um exemplo de quadri-rotor:

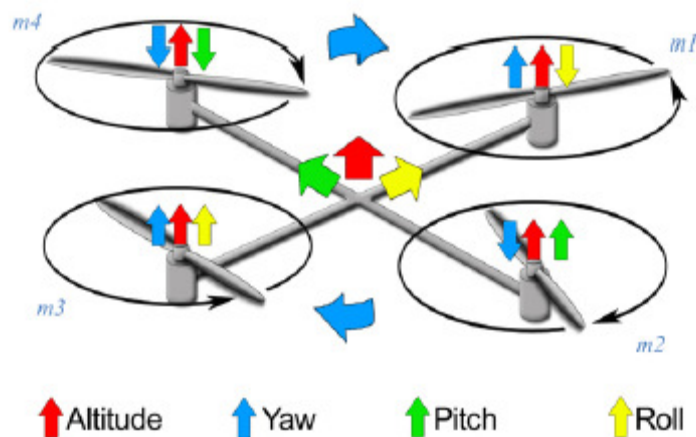


Figura 15- Modelo de um quadri-rotor

5 - SIMULAÇÃO DE ESTABILIDADE

5.1 PROTÓTIPO

A fim de estudar os conceitos apresentados foi construído um protótipo de uma gangorra para simular a estabilidade de um quadrotor. A gangorra simula apenas a estabilidade em um eixo usando apenas dois motores.

A figura abaixo apresenta a ideia para simular a estabilidade em apenas um eixo de um quadrotor.

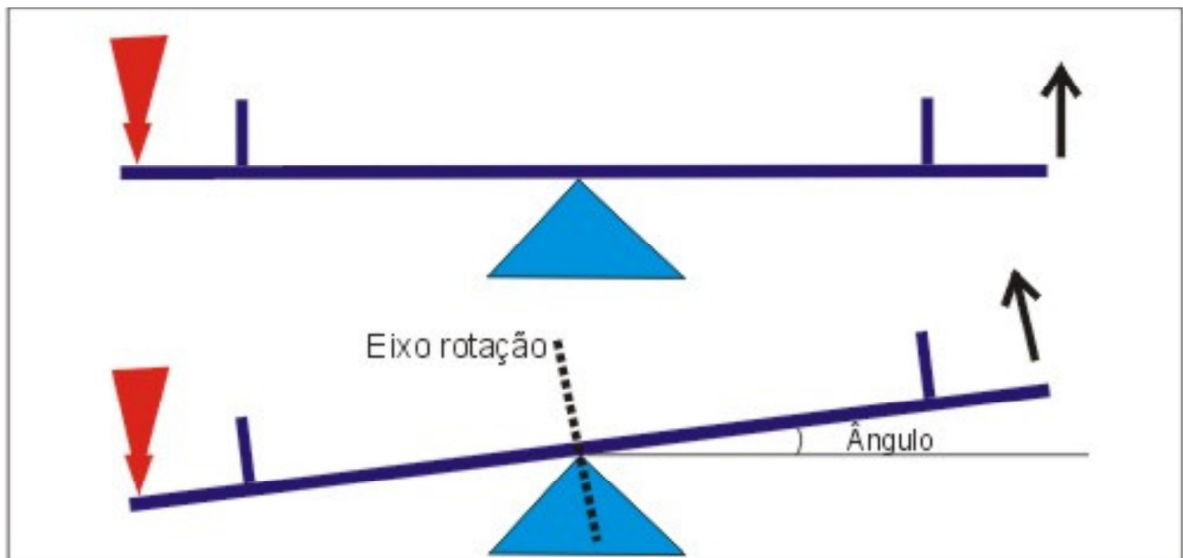


Figura 16- Conceito de uma gangorra.

A gangorra pode ser vista como uma alavanca. Na física, a alavanca é um objeto rígido que é usado com um ponto fixo apropriado (fulcro) para multiplicar a força mecânica que pode ser aplicada a outro objeto (resistência). Isto é denominada também vantagem mecânica, e é um exemplo do princípio dos momentos.

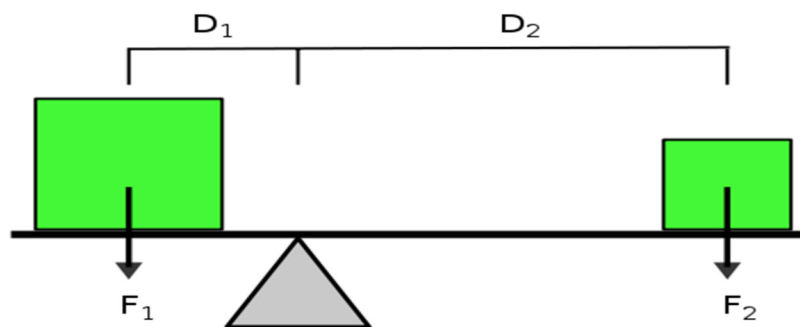


Figura 17- Princípio de funcionamento de uma alavanca.

A força aplicada em pontos de extremidade da alavanca é proporcional à relação do comprimento do braço de alavanca medido entre o fulcro e o ponto da aplicação da força aplicada em cada extremidade da alavanca.

A equação fundamental das alavancas é: $F_p \times BP = F_r \times BR$

onde:

- **F_p** é a força potente;
- **F_r** é a força resistente;
- **BP** é o braço potente; e
- **BR** é o braço resistente.

A gangorra é uma alavanca interfixa, pois seu ponto fixo fica entre as duas forças que atuam sobre esta máquina.

Para que, ocorra equilíbrio entre os lados, o produto do braço pela força resultante deve ser igual em ambas às extremidades.

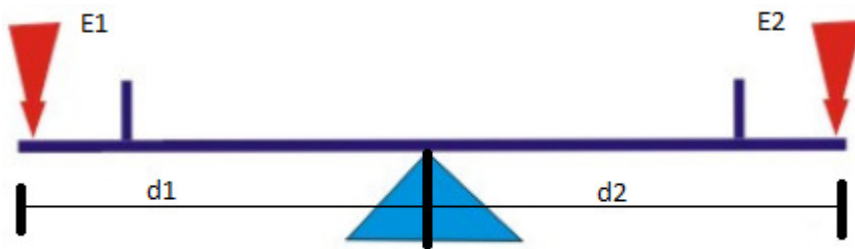


Figura 18– Princípio de funcionamento do experimento.

Assim para manter o equilíbrio na gangorra temos $E1 \times d1 = E2 \times d2$.

Como, $d1 = d2$ temos que $E1 = E2$.

Onde:

E1 → empuxo do motor um.

E2 → empuxo do motor dois.

d1 → distancia do ponto fixo ao motor um.

d2 → distancia do ponto fixo ao motor dois.

Para movimentar a barra é necessario que E1 seja maior que E2, ou E2 maior que E1.

Para saber o empuxo gerado pelo motor mais helice 9x5, foi colocado uma balança em uma das extremidades (figura 19). Variando o sinal PPM enviado para o speed foi possivel traçar um grafico SINAL PPM x EMPUXO (g).

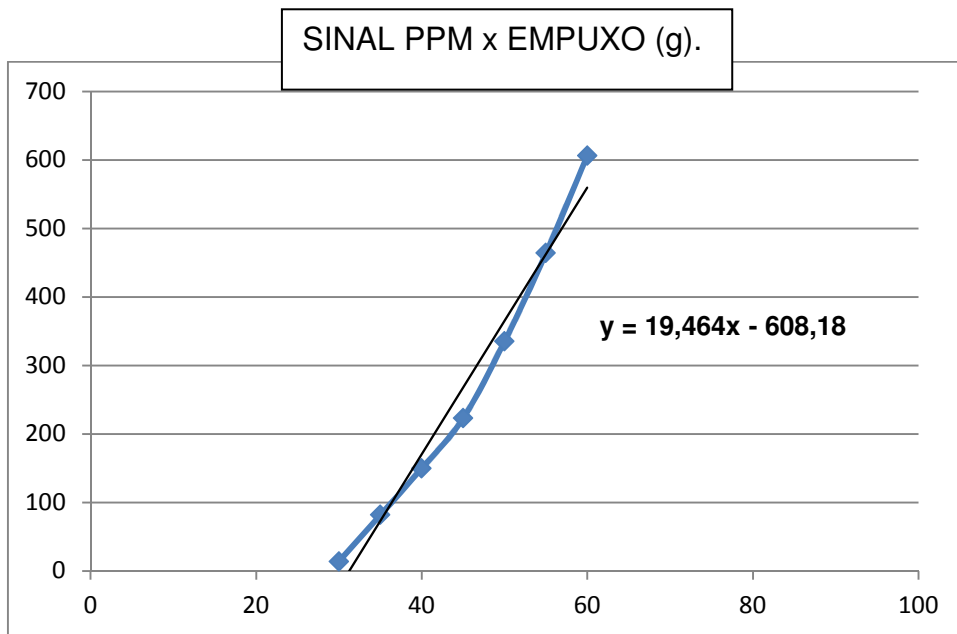


Figura 19 – Gráfico sinal PPM x EMPUXO

O sinal PPM varia entre 1 milissegundo a 2 milissegundos, usando a função map do arduino, o sinal foi mapeado entre 0 a 100.

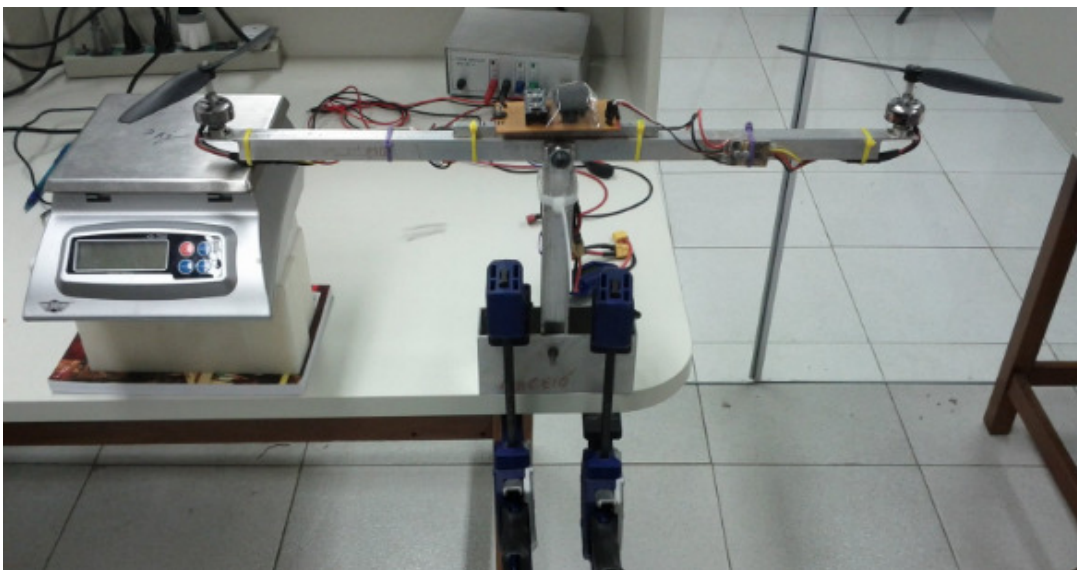


Figura 20– Medição de empuxo.

O prototipo é constituído de:

- Uma estrutura mecânica de alumínio.
- Uma central inercial com acelerómetro e giro.
- Dois motores brushless.
- Dois speeds para controlar os motores.
- Um microcontrolador arduino nano.
- Uma bateria nanotech 3s.
- Uma hélice 9x5.
- Uma hélice 9x5 reversa.

Implementado os conceitos vistos acima na parte de sensores é possível obter o ângulo da barra, através da leitura do acelerómetro e giro, o valor deste ângulo é usado no controle PID onde sua saída será a velocidade de cada motor para manter a gangorra estável.

A figura 20 mostra a eletrônica usada na experiência, uma placa universal com o microcontrolador arduino nano, a central inercial da sparkfun, e os conectores para os speed.

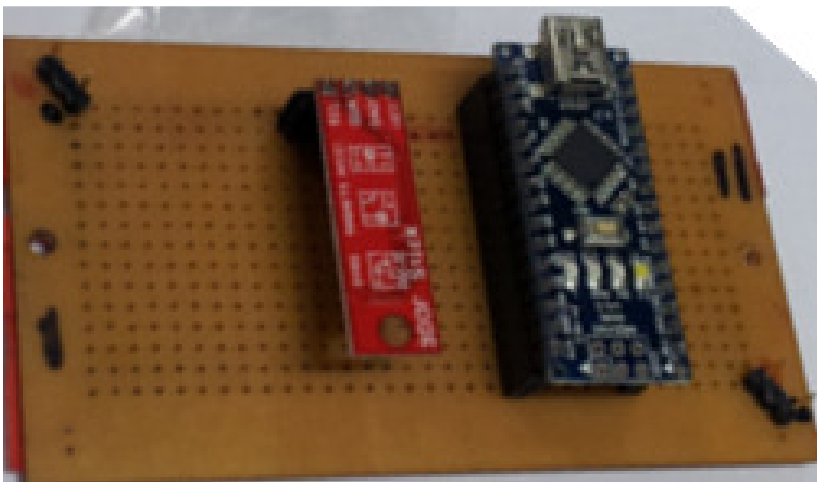


Figura 21– Eletrônica usada.

A estrutura montada é basicamente um projeto de uma gangorra, o material usado foi alumínio, no eixo de rotação foram colocados rolamentos agulha para diminuir o atrito.

As figuras 18 e 19 mostram o protótipo para simular estabilidade.

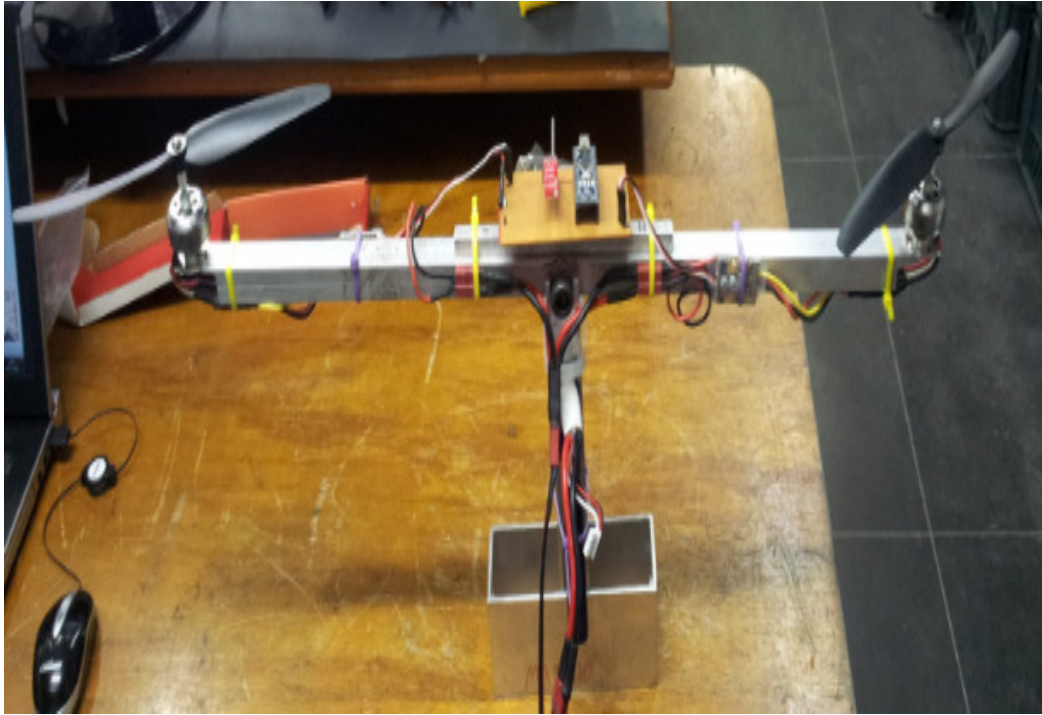


Figura 22– Protótipo montado para simular estabilidade de um multirotor.



Figura 23– Protótipo montado para simular estabilidade de um multirotor.

5.2 CENTRAL INERCIAL

A central inercial usada na experiência foi a, IMU da sparkfun uma placa de sensor muito pequeno, com 9 graus de liberdade. Tem uma interface simples I2C, e um orifício de montagem para anexá-lo ao seu projeto.

Ela inclui:

- Acelerômetro de três eixos ADXL345 com resolução de 13 bits, e medição de até ± 16 g, saída digital no formato de 16 bits complemento de dois e é acessível através de qualquer um SPI (3 ou 4 fios) ou I2C interface digital. Para obter o valor G é preciso dividir o valor lido pelo acelerômetro pela sensibilidade. A valor da sensibilidade varia com a resolução que foi escolhida.
- Magnetômetro HMC5843 de três eixos, que não sera usado.
- Giroscópio-3200 ITG de três eixos, integrado 16-bit ADCs fornecer amostragem simultânea de gyro, não requerendo nenhum multiplexador externo. Para obter o valor deg/s é preciso dividir o valor lido pelo gyro pela sensibilidade.



Figura 24- Central inercial (SEN-10321).

O circuito se comunica com os sensores através do protocolo I2C. Para isso, o circuito deve possuir um barramento ligado ao SCL e ao SDA do microcontrolador que foram implementados por software.

A figura 25 mostra o esquemático do circuito.

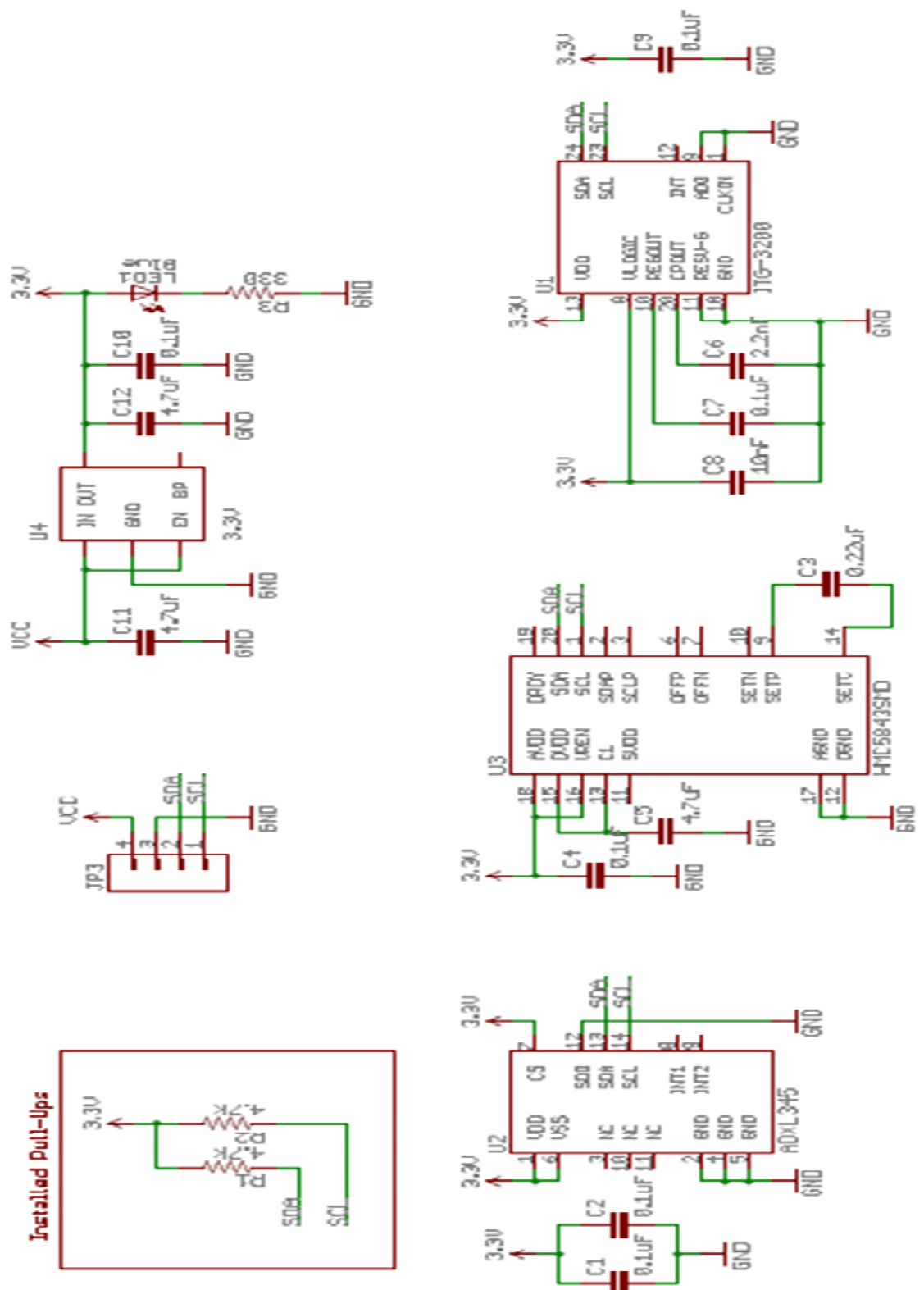


Figura 25 - esquema Central inercial (SEN-10321).

5.2.1 Protocolo I2C

O protocolo de comunicação em sinais I2C foi originalmente desenvolvido pela Philips em meados de 1996. Atualmente este protocolo está amplamente difundido e interconecta uma ampla gama de dispositivos eletrônicos. Dentre estes encontramos vários dispositivos de controle inteligente, normalmente microcontroladores e microprocessadores assim como outros circuitos de uso em geral, como drivers LCD, portas de I/O, memórias RAM EEPROM ou conversores de dados.

O procedimento (Figura) de comunicação do protocolo I2C é extremamente simples. Basicamente temos 6 itens para análise:

1. O dispositivo master ajusta a condição inicial.
2. O dispositivo master envia 7 bits de endereçamento.
3. O dispositivo master envia o 8º bit, RW/
4. O dispositivo slave envia o sinal de ACK (Acknowledge)
5. O dispositivo master (ou slave) envia pacotes de 8 bits de dados, sempre seguidos de um sinal ACK enviado pelo dispositivo slave (ou master) confirmando a recepção.
6. O dispositivo master encerra a comunicação.

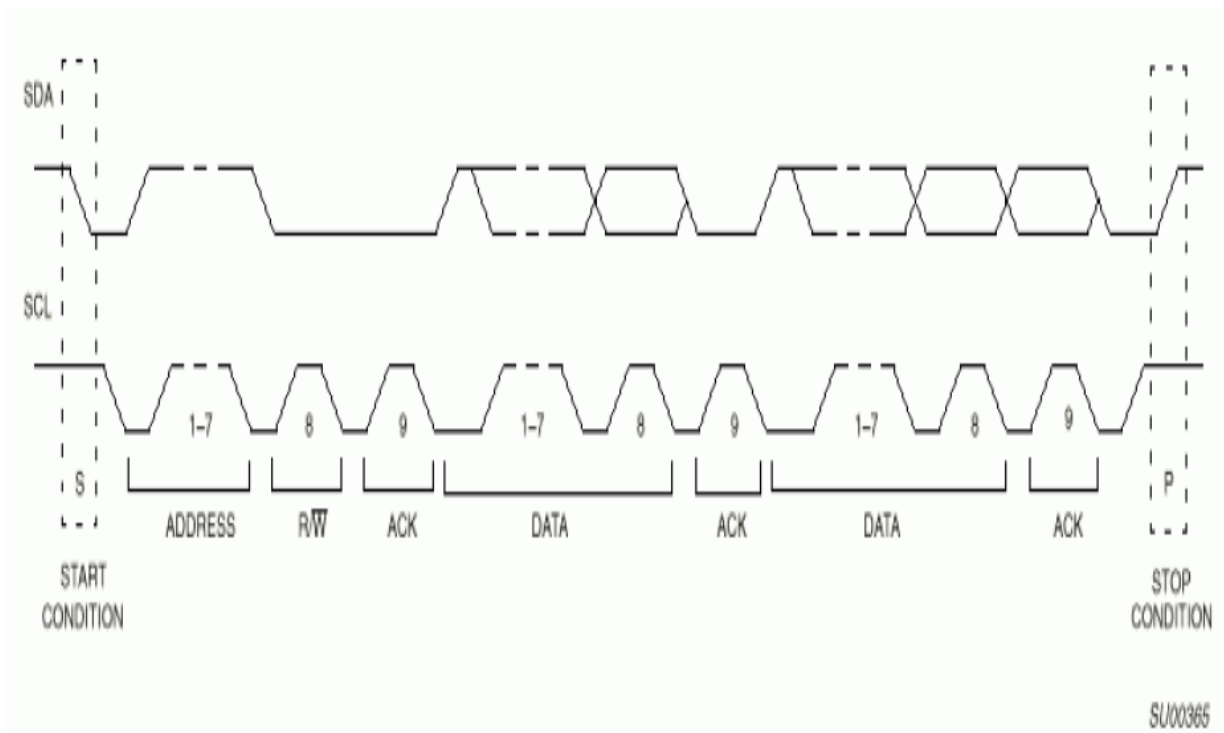


Figura 26- Protocolo I2C.

5.3 MOTOR BRUSHLESS

Motores Brushless DC são motores elétricos alimentados por corrente contínua de eletricidade (DC), com sistemas de comutação eletrônica, ao invés de mecânica comutadores e escovas. Os relacionamentos atuais para torque e frequência na velocidade são lineares. Possuem ímãs permanentes colados no rotor, geralmente com quatro ímãs em torno do perímetro. O estator do motor é composto por electroímãs, geralmente quatro deles, colocados em cruz formando um ângulo de 90 graus entre eles. Como o rotor contém somente os ímãs permanentes, não precisa de alimentação, assim não é necessária nenhuma ligação para o rotor.

Essas características comparadas aos motores com escovas, garantem ao motor brushless uma confiabilidade elevada, redução no ruído, vida útil mais longa, eliminação da ionização do comutador, e a redução total de interferência eletromagnética.

O motor usado no prototipo foi o TowerPro Brushless Outrunner 2410-08T 890kv, apesar de não ser um motor muito robusto ele é suficiente para o projeto devido ao seu baixo custo.



Figura 27- TowerPro Brushless Outrunner 2410-08T 890kv

5.4 MICROCONTROLADORES

Um microcontrolador é um computador inteiro em um único chip, contendo um processador, memória e funções de entrada e saída. Além dos componentes lógicos e aritméticos usuais de um microprocessador de uso geral, o microcontrolador integra elementos adicionais, tais como memória RAM, memória FLASH e/ou memória EEPROM para armazenamento de dados ou programas, dispositivos periféricos de interfaces de E/S que podem ir de um simples pino digital do componente a uma interface USB Ethernet ou CAN nos mais avançados.

Com frequências de clock de poucos MHz ou ainda mais baixas, os microcontroladores são considerados lentos se comparados aos microprocessadores modernos, mas isso é perfeitamente adequado para aplicações típicas. Eles consomem relativamente pouca energia (miliwatts), e geralmente possuem a capacidade de entrar em modo "sleep" enquanto aguardam que aconteça algum evento interessante provocado por um periférico, tal como o pressionar de um botão, que os colocam novamente em atividade. O consumo de energia enquanto estão em modo "sleep" pode ser de nanowatts, tornando-os ideais para aplicações de baixa energia e que economizem bateria.

Existem diversos fabricantes de microcontroladores, dentre elas, as mais populares são: Atmel, Microchip (fabricante da família PIC) e Philips NXT. Os fabricantes disponibilizam diversos modelos de microcontroladores cada qual com suas funções e periféricos específicos. Então, durante qualquer projeto é essencial a escolha do modelo que mais se adéqua a aplicação desejada.

Para usar um microcontrolador, o código a ser desenvolvido deve ser escrito, testado, e armazenado na ROM do microcontrolador. Normalmente o software é escrito e compilado em um PC e então carregado na ROM como código de máquina. Se o programa é escrito em linguagem assembler, o PC tem que ter um compilador para gerar o código de máquina para o microcontrolador.

O microcontrolador usado no projeto foi o arduino nano, este é responsável pela leitura dos sensores e controle de rotação dos motores.

Arduino é uma plataforma open-source de protótipos eletrônicos baseados em hardware e software flexível e de fácil uso. Este pode perceber o ambiente ao receber como entrada de uma variedade de sensores. O microcontrolador na placa é programado usando uma linguagem de programação Arduino e o ambiente de desenvolvimento Arduino.

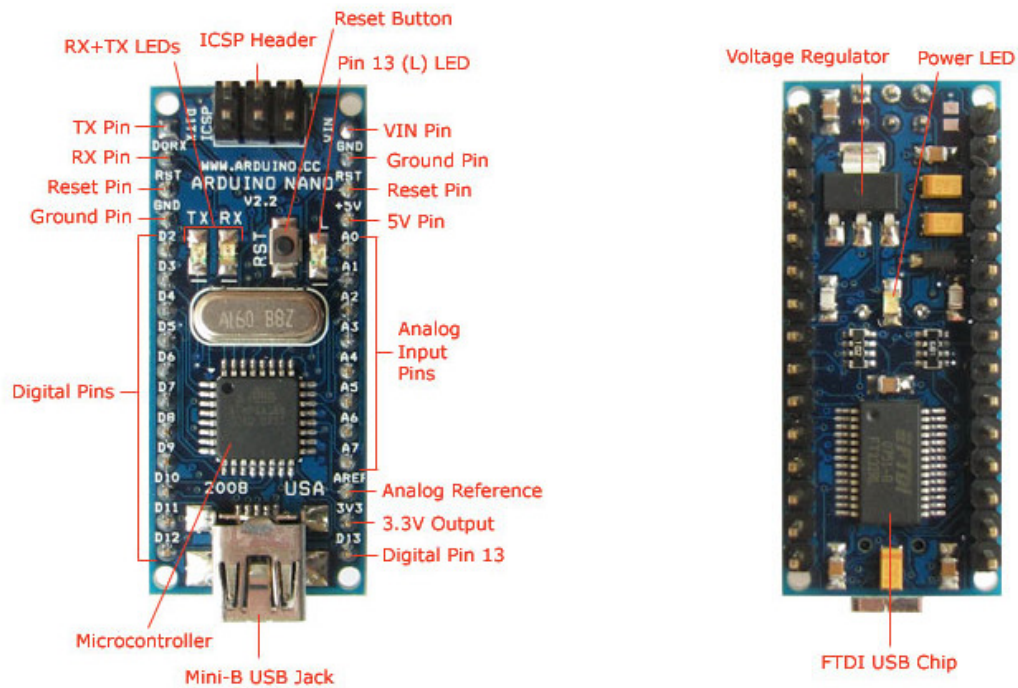


Figura 28 – Arduino nano.

5.5 SPEED CONTROL (Controlador de velocidade)

Para controlar os motores brushless é necessário de um speed control, o microcontrolador envia um sinal PPM para o speed, e o mesmo envia a informação para o motor através de uma corrente trifásica para o motor a partir da corrente contínua da bateria.

É um circuito eletrônico micro processado que permite a aceleração proporcional (0% até 100%).

O speed control funciona por pulsos, variando a quantidade de energia que é transferida da bateria para o motor, possibilitando aceleração proporcional conforme a posição do stick do rádio. Abaixo estão relacionadas algumas destas funções.

Função BEC (battery eliminator circuit): esta função serve para alimentar o receptor e os servos sem a necessidade de uma bateria separada. Um circuito interno separa parte da voltagem da bateria de lipo, em torno de 5 volts, para alimentar o sistema.

Função Auto Cut Off : como o sistema utiliza apenas uma bateria para motor e eletrônicos é importante não deixar que o motor esgote totalmente a bateria fazendo com que os comandos venham a falhar por falta de energia. Para que isto não aconteça, assim que ele detecta que a voltagem esta abaixo de determinado limite, o sistema corta progressivamente o motor avisando que esta na hora de pousar. Assim, mesmo que não tenha mais força para girar a hélice, você sempre terá uma reserva de energia (em torno de 5 minutos) para poder comandar o aeromodelo com segurança ate a pista.

Função ON-OFF: esta função impede que o motor gire quando a bateria é conectada.

Função BRAKE (Freio): a maioria dos speed control possui esta função, que permite frear a hélice quando cortamos a aceleração do motor para que este não toque o solo durante o pouso.

Cada motor tem um consumo diferente, medido em kw/volts, tendo um consumo em amperes/hora de acordo com seu tamanho e helice utilizada, por isso necessitamos de speed controls com diferentes capacidades.

Ex: um motor brushless de 1700 Kv[Rpm/V] ira consumir em torno de 4/13 amperes. Nesse caso, o ideal para ter uma margem de segurança é usar um speed control acima de 18 amperes e assim progressivamente.

6 CONCLUSÃO

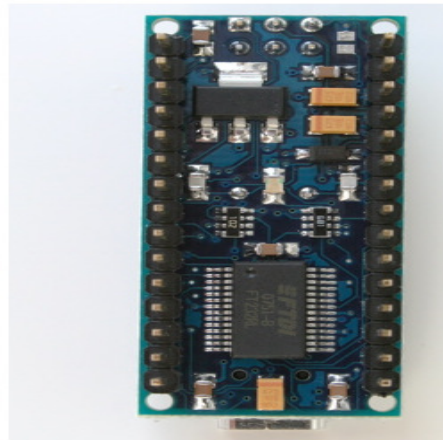
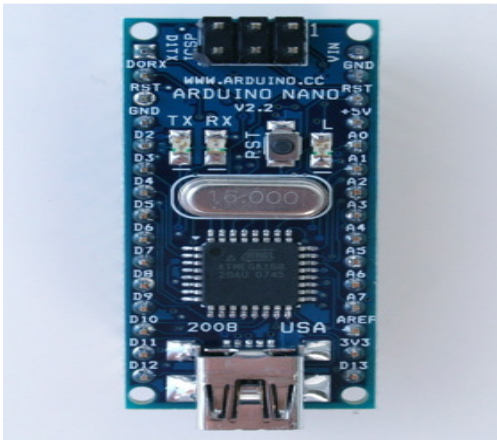
A experiência para simular a estabilidade de um multi rotor saiu como esperado, foi possível obter uma leitura ótima do ângulo no eixo x. Usando o controle PID foi possível controlar a velocidade de cada motor. Os ajustes dos ganhos KP, KI e KD foram feitos por tentativa e erro, os mesmos forneceram ao controle uma resposta boa porem é possível melhorar o desempenho do mesmo.

Já que os resultados foram satisfatórios o próximo passo será montar uma estrutura igual a um quadrotor, e fixar em baixo da estrutura uma junta universal. Com isso será possível simular a estabilidade em todos os eixos

7 REFERÊNCIAS

- Ogata, K; Engenharia de Controle Moderno, 4ª edição, Prentice Hall 2007
- Meggiolaro , Marco Antonio; “RioBotz Combot Tutorial, version 2.0”, 2009;
- [http://www.wpi.edu/Pubs/E-project/Available/E-project-030609-124019/unrestricted/MQP_Report_Quadrotor_Final\[1\].pdf](http://www.wpi.edu/Pubs/E-project/Available/E-project-030609-124019/unrestricted/MQP_Report_Quadrotor_Final[1].pdf)
- www.starlino.com
- <http://www.wikipedia.org/>
- www.google.com
- Datasheet ADXL345
- Datasheet ITG3200
- www.arduino.com

Apêndice A (especificação arduino nano)



Especificações:

Microcontrolador	Atmel ATmega168 ou ATmega328
Tensão operacional (nível lógico)	5 V
Tensão de entrada (recomendado)	12/07 V
Tensão de entrada (limites)	20/06 V
E / S digitais Pinos	14 (dos quais 6 oferecem saída PWM)
Analógica dos Pinos de Entrada	8
DC Current per I / O Pin	40 mA
Memória Flash	16 KB (ATmega168) ou 32 KB (ATmega328), dos quais 2 KB usados por bootloader
SRAM	1 KB (ATmega168) ou 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) ou 1 KB (ATmega328)
Velocidade do relógio	16 MHz

Dimensões 0,73 "x 1,70"

Alimentação

O Arduino Nano pode ser alimentado através do Mini-B conexão USB, 6-20V da fonte de alimentação externa não regulamentada (pino 30), ou regulamentadas 5V fonte de alimentação externa (pino 27). A fonte de energia é automaticamente selecionado para a fonte maior de tensão.

Memória

O ATmega168 tem 16 KB de memória flash para armazenamento de código (dos quais 2 KB é usado para o bootloader), o ATmega328 tem 32 KB, (também com 2 KB utilizada para o bootloader). O ATmega168 tem 1 KB de SRAM e 512 bytes de EEPROM (que pode ser lido e escrito com a [biblioteca EEPROM](#)), o ATmega328 tem 2 KB de SRAM e 1 KB de EEPROM.

Entrada e Saída

Cada um dos 14 pinos digitais do Nano pode ser usado como uma entrada ou saída, usando as funções [pinMode \(\)](#), [digitalWrite \(\)](#), e [digitalRead \(\)](#). Eles operam com 5 volts. Cada pino pode fornecer ou receber um máximo de 40 mA e tem um resistor pull-up interno (desconectado por padrão) de 20-50 kOhms.

Além disso, alguns pinos têm funções especializadas:

Serial:. 0 (RX) e 1 (TX) Usado para receber (RX) e transmitir dados (TX) TTL serial. Estes pinos são conectados aos pinos correspondentes do chip FTDI USB-to-TTL Serial.

Interrupções externas:. 2 e 3 Estes pinos podem ser configurados para disparar uma interrupção por um valor baixo, uma borda de subida ou queda, ou uma alteração no valor.

PWM:. 3, 5, 6, 9, 10 e 11 Fornecer de 8 bits de saída PWM com a função [analogWrite \(\)](#).

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estes pinos suporta comunicação SPI, que, embora fornecido pelo hardware subjacente, não está incluído na linguagem Arduino.

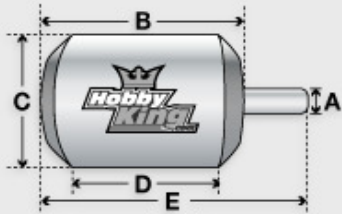
LED: 13. Há um built-in LED conectado ao pino digital 13. Quando o pino está em HIGH o LED está ligado, quando o pino é LOW, ele está fora.

O Nano tem 8 entradas analógicas, cada uma das quais com 10 bits de resolução (1024 valores diferentes). Por padrão, eles medem do terra para 5 volts, embora seja possível mudar o limite superior de sua faixa usando a função o analogReference().

I²C: 4 (SDA) e 5 (SCL) Suporte I²C de comunicação (TWI) usando a biblioteca Wire.

AREF. Tensão de referência para as entradas analógicas. Usado com analogReference ().

Apêndice B (especificação motor)



Kv (rpm/v)	890
Weight (g)	66
Max Current (A)	12
Resistance (mh)	0
Max Voltage (V)	11
Power(W)	0
Shaft A (mm)	-
Length B (mm)	35
Diameter C (mm)	31
Can Length D (mm)	10
Total Length E (mm)	62



Configuração:	Hélice:	Volts:	Amps:	Empuxo:
Direct Drive	1047	7.4V	6.0A	12,3 oz / 344,4 gramas
Direct Drive	1047	8.4V	7.3A	14,7 oz / 411,6 gramas
Direct Drive	1047	9.6V	8.6A	17,6 oz / 492,8 gramas
Direct Drive	1047	11.1V	10.9A	21,5 oz / 602 gramas

Apêndice C (especificação speed)

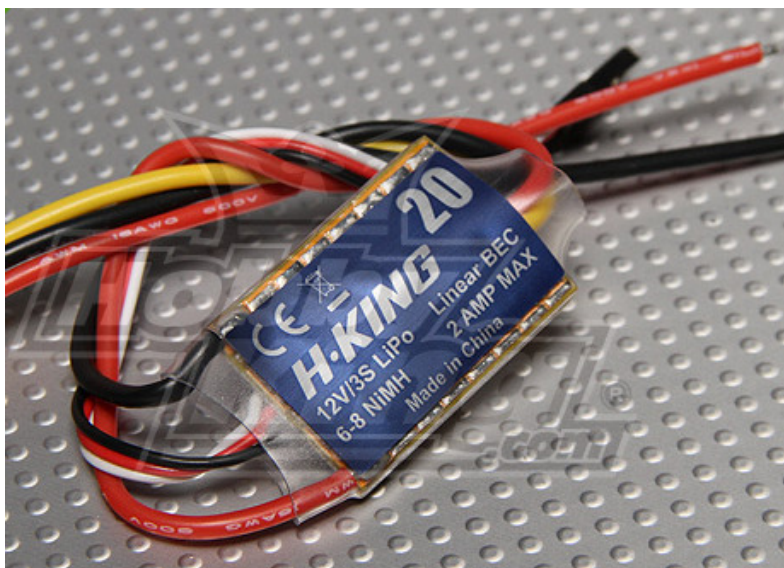


Figura 20 – Speed control usado na experiência

Amp rating: **20A**

Taxa de Burst (15seg): **25A**

BEC Current: **2A, 5V (Linear)**

Tensão: **(2-3 Lipo celular)**

Dimensões: **34x23x8mm**

Peso: **19g**

Apêndice D (especificação bateria)



Capacidade : 2650mAh

Voltagem : 3S1P / 3 Cell / 11.1V

Descarga : 25C Constante Explosão / 50C

Peso : 215g (incluindo plug-fio, e caso)

Dimensões : 148x44x18mm

Plug-Balance : JST-XH

Apêndice E (especificação helice)

Na experiência foram usadas duas hélices, uma 9x5 Propeller Standard e a outra 9x5 Propeller Counter Rotating.



Length (Inch [X])	9
Pitch (Inch [Y])	5
	0
	0
	0
	0
	0
	0

Apêndice F (datasheet acelerometro)



3-Axis, $\pm 2\text{ g}/\pm 4\text{ g}/\pm 8\text{ g}/\pm 16\text{ g}$ Digital Accelerometer

ADXL345

FEATURES

Ultralow power: as low as 40 μA in measurement mode and 0.1 μA in standby mode at $V_s = 2.5\text{ V}$ (typical)

Power consumption scales automatically with bandwidth
User-selectable resolution

Fixed 10-bit resolution

Full resolution, where resolution increases with g range, up to 13-bit resolution at $\pm 16\text{ g}$ (maintaining 4 mg/LSB scale factor in all g ranges)

Embedded, patent pending FIFO technology minimizes host processor load

Tap/double tap detection

Activity/inactivity monitoring

Free-fall detection

Supply voltage range: 2.0 V to 3.6 V

I/O voltage range: 1.7 V to V_s

SPI (3- and 4-wire) and I²C digital interfaces

Flexible interrupt modes mappable to either interrupt pin

Measurement ranges selectable via serial command

Bandwidth selectable via serial command

Wide temperature range (-40°C to $+85^\circ\text{C}$)

10,000 g shock survival

Pb free/RoHS compliant

Small and thin: 3 mm \times 5 mm \times 1 mm LGA package

APPLICATIONS

Handsets

Medical instrumentation

Gaming and pointing devices

Industrial instrumentation

Personal navigation devices

Hard disk drive (HDD) protection

Fitness equipment

GENERAL DESCRIPTION

The ADXL345 is a small, thin, low power, 3-axis accelerometer with high resolution (13-bit) measurement at up to $\pm 16\text{ g}$. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I²C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0° .

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion and if the acceleration on any axis exceeds a user-set level. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm \times 5 mm \times 1 mm, 14-lead, plastic package.

FUNCTIONAL BLOCK DIAGRAM

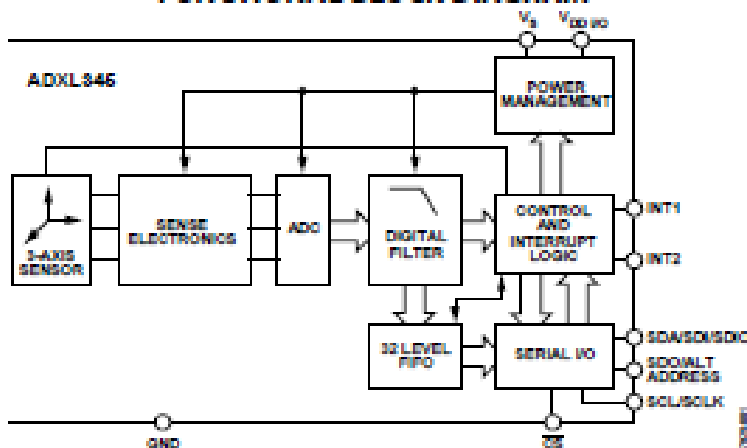


Figure 1.


SPECIFICATIONS

$T_A = 25^\circ\text{C}$, $V_S = 2.5\text{ V}$, $V_{DDIO} = 1.8\text{ V}$, acceleration = 0 g, $C_S = 1\text{ }\mu\text{F}$ tantalum, $C_{IO} = 0.1\text{ }\mu\text{F}$, unless otherwise noted.

Table 1. Specifications¹

Parameter	Test Conditions	Min	Typ	Max	Unit
SENSOR INPUT					
Measurement Range	Each axis User selectable		$\pm 2, \pm 4, \pm 8, \pm 16$		g
Nonlinearity	Percentage of full scale		± 0.5		%
Inter-Axis Alignment Error			± 0.1		Degrees
Cross-Axis Sensitivity ²			± 1		%
OUTPUT RESOLUTION					
All g Ranges	Each axis 10-bit resolution		10		Bits
$\pm 2\text{ g}$ Range	Full resolution		10		Bits
$\pm 4\text{ g}$ Range	Full resolution		11		Bits
$\pm 8\text{ g}$ Range	Full resolution		12		Bits
$\pm 16\text{ g}$ Range	Full resolution		13		Bits
SENSITIVITY					
Sensitivity at $X_{out}, Y_{out}, Z_{out}$	Each axis $\pm 2\text{ g}$, 10-bit or full resolution	232	256	286	LSB/g
Scale Factor at $X_{out}, Y_{out}, Z_{out}$	$\pm 2\text{ g}$, 10-bit or full resolution	3.5	3.9	4.3	mg/LSB
Sensitivity at $X_{out}, Y_{out}, Z_{out}$	$\pm 4\text{ g}$, 10-bit resolution	116	128	143	LSB/g
Scale Factor at $X_{out}, Y_{out}, Z_{out}$	$\pm 4\text{ g}$, 10-bit resolution	7.0	7.8	8.6	mg/LSB
Sensitivity at $X_{out}, Y_{out}, Z_{out}$	$\pm 8\text{ g}$, 10-bit resolution	58	64	71	LSB/g
Scale Factor at $X_{out}, Y_{out}, Z_{out}$	$\pm 8\text{ g}$, 10-bit resolution	14.0	15.6	17.2	mg/LSB
Sensitivity at $X_{out}, Y_{out}, Z_{out}$	$\pm 16\text{ g}$, 10-bit resolution	29	32	36	LSB/g
Scale Factor at $X_{out}, Y_{out}, Z_{out}$	$\pm 16\text{ g}$, 10-bit resolution	28.1	31.2	34.3	mg/LSB
Sensitivity Change Due to Temperature			± 0.01		%/ $^\circ\text{C}$
0 g BIAS LEVEL					
0 g Output for X_{out}, Y_{out}	Each axis	-150	± 40	+150	mg
0 g Output for Z_{out}		-250	± 80	+250	mg
0 g Offset vs. Temperature for x-, y-Axis			± 0.8		mg/ $^\circ\text{C}$
0 g Offset vs. Temperature for z-Axis			± 4.5		mg/ $^\circ\text{C}$
NOISE PERFORMANCE					
Noise (x-, y-Axis)	Data rate = 100 Hz for $\pm 2\text{ g}$, 10-bit or full resolution		<1.0		LSB rms
Noise (z-Axis)	Data rate = 100 Hz for $\pm 2\text{ g}$, 10-bit or full resolution		<1.5		LSB rms
OUTPUT DATA RATE AND BANDWIDTH					
Measurement Rate ³	User selectable	6.25		3200	Hz
SELF-TEST⁴					
Output Change in x-Axis	Data rate $\geq 100\text{ Hz}$, $2.0\text{ V} \leq V_S \leq 3.6\text{ V}$	0.20		2.10	g
Output Change in y-Axis		-2.10		-0.20	g
Output Change in z-Axis		0.30		3.40	g
POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range (V_{DDIO})	$V_S \leq 2.5\text{ V}$	1.7	1.8	V_S	V
	$V_S \geq 2.5\text{ V}$	2.0	2.5	V_S	V
Supply Current	Data rate > 100 Hz		145		μA
	Data rate < 10 Hz		40		μA
Standby Mode Leakage Current			0.1	2	μA
Turn-On Time ⁵	Data rate = 3200 Hz		1.4		ms
TEMPERATURE					
Operating Temperature Range		-40		+85	$^\circ\text{C}$
WEIGHT					
Device Weight			20		mg

Apêndice G (datasheet gyro)

	ITG-3200 Product Specification	Document Number: PS-ITG-3200A-00-01.4 Revision: 1.4 Release Date: 03/30/2010
---	--------------------------------	--

2 Features

The ITG-3200 triple-axis MEMS gyroscope includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyros) on one integrated circuit with a sensitivity of 14.375 LSBs per °/sec and a full-scale range of $\pm 2000^\circ/\text{sec}$
- Three integrated 16-bit ADCs provide simultaneous sampling of gyros while requiring no external multiplexer
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Low frequency noise lower than previous generation devices, simplifying application development and making for more-responsive motion processing
- Digitally-programmable low-pass filter
- Low 6.5mA operating current consumption for long battery life
- Wide VDD supply voltage range of 2.1V to 3.6V
- Flexible VLOGIC reference voltage allows for I²C interface voltages from 1.71V to VDD
- Standby current: 5 μ A
- Smallest and thinnest package for portable devices (4x4x0.9mm QFN)
- No high pass filter needed
- Turn on time: 50ms
- Digital-output temperature sensor
- Factory calibrated scale factor
- 10,000 g shock tolerant
- Fast Mode I²C (400kHz) serial interface
- On-chip timing generator clock frequency is accurate to +/-2% over full temperature range
- Optional external clock inputs of 32.768kHz or 19.2MHz to synchronize with system clock
- MEMS structure hermetically sealed and bonded at wafer level
- RoHS and Green compliant

3 Electrical Characteristics

3.1 Sensor Specifications

Typical Operating Circuit of Section 4.2, VDD = 2.5V, VLOGIC = 1.71V to VDD, TA=25°C.

Parameter	Conditions	Min	Typical	Max	Unit	Note
GYRO SENSITIVITY						
Full-Scale Range	FS_SEL=3		±2000		°/s	4
Gyro ADC Word Length			16		Bits	3
Sensitivity Scale Factor	FS_SEL=3		14.375		LSB/(°/s)	3
Sensitivity Scale Factor Tolerance	25°C	-6		+6	%	1
Sensitivity Scale Factor Variation Over Temperature			±10		%	2
Nonlinearity	Best fit straight line; 25°C		0.2		%	6
Cross-Axis Sensitivity			2		%	6
GYRO ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance			±40		°/s	1
ZRO Variation Over Temperature	-40°C to +85°C		±40		°/s	2
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.2V		0.2		°/s	5
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.2V		0.2		°/s	5
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.2V		4		°/s	5
Linear Acceleration Sensitivity	Static		0.1		°/s/g	6
GYRO NOISE PERFORMANCE						
Total RMS noise	FS_SEL=3 100Hz LPF (DLPFCFG=2)		0.38		°/s-rms	1
Rate Noise Spectral Density	At 10Hz		0.03		°/s/√Hz	2
GYRO MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	1
Y-Axis		27	30	33	kHz	1
Z-Axis		24	27	30	kHz	1
Frequency Separation	Between any two axes	1.7			kHz	1
GYRO START-UP TIME						
ZRO Settling	DLPFCFG=0 to ±1°/s of Final		50		ms	6
TEMPERATURE SENSOR						
Range			-30 to +85		°C	2
Sensitivity			280		LSB/°C	2
Temperature Offset	35°C		-13,200		LSB	1
Initial Accuracy	35°C		TBD		°C	
Linearity	Best fit straight line (-30°C to +85°C)		±1		°C	2, 5
TEMPERATURE RANGE						
Specified Temperature Range		-40		85	°C	

Apêndice H (software)

```
#include <Wire.h>                // I2C library, gyroscope
#include <Servo.h>

/* *Accelerometer ADXL345 */

#define ACC (0x53)                // ADXL345 ACC address
#define A_TO_READ (6)           // Num of bytes we are going to read each time (two
                                // bytes for each axis)

/* *Gyroscope ITG3200 */

#define GYRO 0x68                // gyro address, binary = 0110 1000 when AD0 is connected
                                // to GND (see schematics of your breakout board)

#define G_SMPLRT_DIV 0x15
#define G_DLPF_FS 0x16
#define G_INT_CFG 0x17
#define G_PWR_MGM 0x3E
#define G_TO_READ 8              // 2 bytes for each axis x, y, z

Servo myservo1;                 // create servo object to control a servo
Servo myservo2;                 // create servo object to control a servo

/* *Parâmetros para PID */

double Input;
double Output_Motor_Left;
```

```

double Output_Motor_Right;

double Setpoint;

double error;

double M1 = 33;           //valor inicial do motor 1
double M2 = 32;           //valor inicial do motor 2

/* Leitura inicial para remover o offset */

int x0_ACCEL;             //primeira leitura do acelerômetro eixo X.
int y0_ACCEL;             //primeira leitura do acelerômetro eixo Y.
int z0_ACCEL;             //primeira leitura do acelerômetro eixo Z.

int x0_GYRO;              //primeira leitura do gyro eixo X.
int y0_GYRO;              //primeira leitura do gyro eixo Y.
int z0_GYRO;              //primeira leitura do gyro eixo Z.

char str[512];            //string buffer to transform data before sending it to the serial port

boolean firstSample = true;

float RwAcc[3];           //projection of normalized gravitation force vector on x/y/z axis,
                           as measured by accelerometer

float Gyro_ds[3];         //Gyro readings
float RwGyro[3];          //Rw obtained from last estimated value and gyro movement
float Awz[2];             //angles between projection of R on XZ/YZ plane and Z axis (deg)
float RwEst[3];

```

```
int lastTime = 0;
int interval = 0;
float wGyro = 10.0;
```

```
/* FUNÇÃO PARA INICIALIZAR O ACELEROMETRO */
```

```
void initAcc()
```

```
{
```

```
    /* Turning on the ADXL345 */
```

```
    writeTo(ACC, 0x2D, 0);
```

```
    writeTo(ACC, 0x2D, 16);
```

```
    writeTo(ACC, 0x2D, 8);
```

```
    /* by default the device is in +-2g range reading */
```

```
}
```

```
/* FUNÇÃO PARA PEGAR OS DADOS DO ACELEROMETRO, E TRANSFORMAR OS  
DADOS DE 2 BYTES EM UM INT */
```

```
void getAccelerometerData(int * result)
```

```
{
```

```
    int regAddress = 0x32;    //first axis-acceleration-data register on the ADXL345
```

```
    byte buff[A_TO_READ];
```

```
    readFrom(ACC, regAddress, A_TO_READ, buff);    //read the acceleration data from  
                                                    the ADXL345
```

```
    //each axis reading comes in 10 bit resolution, ie 2 bytes. Least Significant Byte first!!
```

```
    //thus we are converting both bytes in to one int
```

```
    result[0] = (((int)buff[1]) << 8) | buff[0] - x0_ACCEL;
```

```
    result[1] = (((int)buff[3])<< 8) | buff[2] - y0_ACCEL;
```

```

    result[2] = (((int)buff[5] << 8) | buff[4]) - z0_ACCEL;

    /* Ler os valores na serial */
    // Serial.print(result[0]);
    // Serial.print(" ");
    // Serial.print(result[1]);
    // Serial.print(" ");
    // Serial.println(result[2]);

}

/* FUNÇÃO PARA PEGAR OS DADOS DO ACELEROMETRO EM G, OS DADOS SÃO
DIVIDIDOS PELA Sensitivity=256 LSB/g */
void rawAccToG(int * raw, float * RwAcc)
{
    RwAcc[0] = ((float) raw[0]) / 256.0;
    RwAcc[1] = ((float) raw[1]) / 256.0;
    RwAcc[2] = ((float) raw[2]) / 256.0;

    /* Ler os valores na serial */
    // Serial.print(RwAcc[0]);
    // Serial.print(" ");
    // Serial.print(RwAcc[1]);
    // Serial.print(" ");
    // Serial.println(RwAcc[2]);
}

```

```
/** FUNÇÃO PARA INICIALIZAR O GYROSCOPIO **/
```

```
void initGyro()
```

```
{
```

```
/******
```

```
* ITG 3200
```

```
* power management set to:
```

```
* clock select = internal oscillator
```

```
* no reset, no sleep mode
```

```
* no standby mode
```

```
* sample rate to = 125Hz
```

```
* parameter to +/- 2000 degrees/sec
```

```
* low pass filter = 5Hz
```

```
* no interrupt
```

```
*****/
```

```
//This register (PWR_MGM=0x3E) is used to manage the power control, select the clock source, and to issue a master reset to the device.
```

```
writeTo(GYRO, G_PWR_MGM, 0x00);
```

```
// This register (SMPLRT_DIV=0x15) determines the sample rate of the ITG-3200 gyros (= 125Hz, or 8ms per sample).// EB, 50, 80, 7F, DE, 23, 20, FF
```

```
writeTo(GYRO, G_SMPLRT_DIV, 0x07);
```

```
// Digital low pass filter (5Hz),internal sampling rate (1kHz); Full scale selection for gyro sensor data (+/- 2000 dgrs/sec)
```

```
writeTo(GYRO, G_DLPF_FS, 0x1E);
```

```
writeTo(GYRO, G_INT_CFG, 0x00);
```

```
}
```

```
/* FUNÇÃO PARA PEGAR OS DADOS DO GYROSCOPIO, E TRANSFORMAR OS
DADOS DE 2 BYTES EM UM INT */
```

```
void getGyroscopeData(int * result)
```

```
{
```

```
  /******
```

```
  Gyro ITG-3200 I2C
```

```
  registers:
```

```
  temp MSB = 1B, temp LSB = 1C
```

```
  x axis MSB = 1D, x axis LSB = 1E
```

```
  y axis MSB = 1F, y axis LSB = 20
```

```
  z axis MSB = 21, z axis LSB = 22
```

```
  *****/
```

```
  int regAddress = 0x1B;
```

```
  int temp, x, y, z;
```

```
  byte buff[G_TO_READ];
```

```
  readFrom(GYRO, regAddress, G_TO_READ, buff);           //read the gyro data from the
                                                           ITG3200
```

```
  result[0] = ((buff[2] << 8) | buff[3]) - x0_GYRO;
```

```
  result[1] = ((buff[4] << 8) | buff[5]) - y0_GYRO;
```

```
  result[2] = ((buff[6] << 8) | buff[7]) - z0_GYRO;
```

```
  result[3] = (buff[0] << 8) | buff[1];                 // temperature
```



```
/* Ler os valores na serial */
```

```
// Serial.print(result[0]);
```

```
// Serial.print(" ");
```

```
// Serial.print(result[1]);
```

```
// Serial.print(" ");
```

```
// Serial.println(result[2]);
```

```
}
```

```
/* FUNÇÃO PARA PEGAR OS DADOS DO GYROSCOPIO EM degrees/sec, OS DADOS  
SÃO DIVIDIDOS PELA Sensitivity=14.375 LSBs per °/sec */
```

```
void rawGyroToDegsec(int * raw, float * gyro_ds)
```

```
{
```

```
    gyro_ds[0] = ((float) raw[0]) / 14.375;
```

```
    gyro_ds[1] = ((float) raw[1]) / 14.375;
```

```
    gyro_ds[2] = ((float) raw[2]) / 14.375;
```

```
/* Ler os valores na serial */
```

```
// Serial.print(gyro_ds[0]);
```

```
// Serial.print(" ");
```

```
// Serial.print(gyro_ds[1]);
```

```
// Serial.print(" ");
```

```
// Serial.println(gyro_ds[2]);
```

```
}
```

```
/* FUNÇÃO PARA NORMALIZAR OS DADOS // Racc(normalized) = [RxAcc/|Racc| ,  
RyAcc/|Racc| , RzAcc/|Racc|] */
```

```
void normalize3DVec(float * vector)
```

```
{
```

```
float R; // R = |Racc|
```

```
||Racc| = SQRT(RxAcc^2 +RyAcc^2 + RzAcc^2)
```

```
R = sqrt(vector[0]*vector[0] + vector[1]*vector[1] + vector[2]*vector[2]);
```

```
vector[0] /= R; // RxAcc/|Racc|
```

```
vector[1] /= R; // RyAcc/|Racc|
```

```
vector[2] /= R; // RzAcc/|Racc|
```

```
}
```

```
/* equation everytime "squared" is used */
```

```
float squared(float x)
```

```
{
```

```
return x*x; // x^2
```

```
}
```

```
void getInclination()
```

```
{
```

```
int w = 0;
```

```
float tmpf = 0.0;
```

```
int currentTime, signRzGyro;
```

```
currentTime = millis();
```

```
interval = currentTime - lastTime;
```

```

lastTime = currentTime;

if (firstSample)
{
    for(w=0;w<=2;w++)
    {
        RwEst[w] = RwAcc[w];           //initialize with accelerometer readings
    }
}
Else
{
    //evaluate RwGyro vector
    if(abs(RwEst[2]) < 0.1)
    {
        //Rz is too small and because it is used as reference for computing Axz, Ayz it's error
        //fluctuations will amplify leading to bad results

        //in this case skip the gyro data and just use previous estimate

        for(w=0;w<=2;w++)
        {
            RwGyro[w] = RwEst[w];     //initialize with gyro readings
        }
    }
    else
    {
        //get angles between projection of R on ZX/ZY plane and Z axis, based on last RwEst
        for(w=0;w<=1;w++)
        {

```

```

tmpf = Gyro_ds[w];           //get current gyro rate in deg/s
tmpf *= interval / 1000.0f;  //get angle change in deg
Awz[w] = atan2(RwEst[w],RwEst[2]) * 180 / PI; //get angle and convert to degrees;
                                           atan2 returns values between [-PI,PI]
Awz[w] += tmpf;             //get updated angle according to gyro movement
}

```

//estimate sign of RzGyro by looking in what quadrant the angle Axz is, RzGyro is positive if Axz in range -90 ..90 => cos(Awz) >= 0

```

signRzGyro = ( cos(Awz[0] * PI / 180) >= 0 ) ? 1 : -1; //+1 when signRzGyro<=0; -1 when
                                                    signRzGyro<0

```

//reverse calculation of RwGyro from Awz angles.

```

for(w=0;w<=1;w++)

```

```

{

```

```

    RwGyro[0] = sin(Awz[0] * PI / 180); //RxGyro; sin(Axz(x)) in rad

```

// RxGyro/[(1+cos(Axz(x))^2]*[tan(Ayz(y))^2]

```

    RwGyro[0] /= sqrt( 1 + squared(cos(Awz[0] * PI / 180)) * squared(tan(Awz[1] * PI / 180)) );

```

```

    RwGyro[1] = sin(Awz[1] * PI / 180); //RyGyro; sin(Ayz(y)) in rad

```

//RyGyro/[(1+cos(Ayz(y))^2]*[tan(Axz(x))^2]

```

    RwGyro[1] /= sqrt( 1 + squared(cos(Awz[1] * PI / 180)) * squared(tan(Awz[0] * PI / 180)) );

```

```

}

```

```

    RwGyro[2] = signRzGyro * sqrt(1 - squared(RwGyro[0]) - squared(RwGyro[1])); //To find
                                                    RzGyro

```

```

}

```

```
//combine Accelerometer and gyro readings
```

```
for(w=0;w<=2;w++)
```

```
{
```

```
RwEst[w] = (RwAcc[w] + wGyro * RwGyro[w]) / (1 + wGyro); //to update estimated values
```

```
}
```

```
normalize3DVec(RwEst); //to normalize the above vector
```

```
/* Ler os valores na serial */
```

```
// Serial.print(RwEst[0]);
```

```
// Serial.print(" ");
```

```
// Serial.print(RwEst[1]);
```

```
// Serial.print(" ");
```

```
// Serial.println(RwEst[2]);
```

```
}
```

```
firstSample = false;
```

```
}
```

```
/* arm the speed controller, modify as necessary for your ESC */
```

```
void arm()
```

```
{
```

```
setSpeed(25);
```

```
delay(1000); //delay 1 second, some speed controllers may need longer
```

```
}
```

// speed is from 0 to 100 where 0 is off and 100 is maximum speed the following maps speed values of 0-100 to angles from 0-180, some speed controllers may need different values, see the ESC instructions

```
void setSpeed(int speed)
{
  int angle = map(speed, 0, 100, 0, 180);

  myservo1.write(angle);
  myservo2.write(angle);
}
```

/**PID**/

```
void Compute()
{
  double error_der = 0.0;
  static double error_acc = 0.0;
  static double error_old = 0.0;

  Setpoint = 0;
  Input = RwEst[0];

  error = Setpoint - Input;
  error_acc += error;
  error_der = error_old - error;

  double Kp = 1.5;
  double Kd = 2.0;
  double Ki = 0.01;
```

```
double U = Kp * error + Ki * error_acc + Kd * error_der;
```

```
Output_Motor_Left = (U) + M1;
```

```
Output_Motor_Right = -(U) + M2;
```

```
error_old = error;
```

```
int Mleft = map(Output_Motor_Left, 0, 100, 0, 180);
```

```
int Mrigth = map(Output_Motor_Right, 0, 100, 0, 180);
```

```
myservo1.write(Mleft < 30 ? 30 : Mleft);
```

```
myservo2.write(Mrighth < 30 ? 30 : Mrighth);
```

```
/* Ler os valores na serial */
```

```
Serial.print(Mleft );
```

```
Serial.print(" ");
```

```
Serial.print(Mrighth );
```

```
Serial.println(" ");
```

```
// Serial.print(Ki*error_acc );
```

```
// Serial.print(" ");
```

```
// Serial.print(Kd*error_der );
```

```
// Serial.print(" ");
```

```
// Serial.println(Kp*error );
```

```
}
```

```

void setup()
{
  Serial.begin(9600);
  Wire.begin();

  byte buff_G[G_TO_READ];
  byte buff_A[A_TO_READ];

  readFrom(GYRO, 0x1B, G_TO_READ, buff_G); // primeira leitura do sensor para verificar
                                             o offset

  x0_GYRO = ((buff_G[2] << 8) | buff_G[3]);
  y0_GYRO = ((buff_G[4] << 8) | buff_G[5]);
  z0_GYRO = ((buff_G[6] << 8) | buff_G[7]);

  /* Ler os valores na serial */
  // Serial.print(x0_GYRO);
  // Serial.print(" ");
  // Serial.print(y0_GYRO);
  // Serial.print(" ");
  // Serial.println(z0_GYRO);

  readFrom(ACC, 0x32, A_TO_READ, buff_A); //read the acceleration first data from the
                                             ADXL345

  x0_ACCEL = (((int)buff_A[1]) << 8) | buff_A[0];
  y0_ACCEL = (((int)buff_A[3]) << 8) | buff_A[2];
  z0_ACCEL = (((int)buff_A[5]) << 8) | buff_A[4] + 256; //soma 256 referente a 1 g em z

```



```
/* Ler os valores na serial */  
// Serial.print(x0_ACCEL );  
// Serial.print(" ");  
// Serial.print(y0_ACCEL );  
// Serial.print(" ");  
// Serial.println(z0_ACCEL );
```

```
delay(1000);
```

```
initAcc();
```

```
initGyro();
```

```
myservo1.attach(3);
```

```
myservo2.attach(11);
```

```
arm();
```

```
}
```

```
void loop()
```

```
{
```

```
if(!Serial.available())
```

```
{
```

```
int acc[3];
```

```
int gyro[4];
```

```
getAccelerometerData(acc);
```

```
rawAccToG(acc, RwAcc);
```

```
    normalize3DVec(RwAcc);
    getGyroscopeData(gyro);
    rawGyroToDegsec(gyro, Gyro_ds);
    getInclination();
    Compute();
  }
}
```

```
/* Writes val to address register on ACC */
```

```
void writeTo(int DEVICE, byte address, byte val)
```

```
{
  Wire.beginTransmission(DEVICE);    //start transmission to ACC
  Wire.send(address);                // send register address
  Wire.send(val);                    // send value to write
  Wire.endTransmission();            //end transmission
}
```

```
/* reads num bytes starting from address register on ACC in to buff array */
```

```
void readFrom(int DEVICE, byte address, int num, byte buff[])
```

```
{
  Wire.beginTransmission(DEVICE);    //start transmission to ACC
  Wire.send(address);                //sends address to read from
  Wire.endTransmission();            //end transmission

  Wire.beginTransmission(DEVICE);    //start transmission to ACC
  Wire.requestFrom(DEVICE, num);     // request 6 bytes from ACC
}
```

```
int i = 0;
while(Wire.available())           //ACC may send less than requested (abnormal)
{
    buff[i] = Wire.receive();     // receive a byte
    i++;
}
Wire.endTransmission();         //end transmission
}
```