SEGUNDO RELATÓRIO PARCIAL - PPD/2023

Envio até 16/10/2023

PPD/2023
Nome dos(a) orientadores(a): MARCOS ANTÔNIO ESTREMOTE
Nome dos(a) alunos(a): **FABRICÍCIO GONÇALVES FERREIRA** **OLGA DE OLIVEIRA**
Alunos voluntários: **MICHAEL R. SANTOS BEZERRA**
Colaboradores internos: **JOSÉ PAULO CODINHOTO**
Colaboradores externos (incluir nome e instituição): **MARCOS GABRIEL DA SILVA ROCHA – FACOM / UFMS**
IDENTIFICAÇÃO DO PROJETO
Título: SIMULAÇÃO E CONTROLE DE ESTABILIDADE DE DRONES QUADRICÓPTEROS ATRAVÉS DE CONTROLE PID UTILIZANDO ARDUINO E ESP32.
Área de Conhecimento: ( ) Ciências sociais e educação ( ) Biológicas e saúde (x) Agrárias, engenharias e tecnologia da informação.
COMITÊ DE ÉTICA EM PESQUISA COM HUMANOS (CEP)
( ) Aprovado - CAAE: _____ ( ) Em processo - CAAE: _____ ( ) Não enviado ( ) Não se aplica

1 – CRONOGRAMA DO PROJETO INICIAL

Conforme definido nas Tabelas 1 e 2, estão expostos em itens no cronograma proposto no projeto entregue ao Programa de Pesquisa Docente 2023.

Este trabalho de pesquisa possui carácter interdisciplinar, onde iniciamos pelas conceituações teóricas, análises matemáticas e desenvolvimento computacional, como forma de

definir e direcionar as sequências de opções mais adequadas e determinação de um melhor resultado. Deste ponto em diante, os esforços estão na transposição das soluções teóricas e virtuais para o sistema real, através das seguintes etapas:

a) Diagramação mecânica e cinemática da estrutura física do aeromodelo, suas características sobre capacidade de sustentação e estabilidade.

Nesta etapa, o aluno realizou o mapeamento de todas as variáveis do aeromodelo tais como: peso dos componentes, peso do modelo completo, verificação da sustentação da estrutura do quadricóptero por parte dos motores, velocidades mínimas e máximas atingidas.

Etapa concluída. Descrição na parte inicial do Desenvolvimento do Projeto.

b) Implementação do algoritmo de controle e sua utilização em plataforma de um grau de liberdade.

Elaboração e implementação do algoritmo na plataforma Arduino e ESP-32 para controle do protótipo com 1 grau de liberdade. Nesta etapa, utilizamos somente 2 motores para a estabilização.

Nesta etapa, foram implementados alguns algoritmos em C/C++ na plataforma Arduino para controle dos motores.

Etapa concluída.

c) Parametrização do algoritmo de interfaceamento sensores-microcontrolador.

Coleta dos valores dos parâmetros dos sensores do quadricóptero através da implementação de algoritmo no Arduino e ESP-32.

Etapa concluída.

***d)* Construção do circuito de desenvolvimento agregado ao microcontrolador, estrutura do aeromodelo com quatro rotores e seus respectivos ESCs (*Electronic Speed Controllers*) e da plataforma de teste para sustentação.**

Elaboração e implementação do algoritmo na plataforma Arduino e ESP-32 para controle do protótipo com 4 motores. Nesta etapa, utilizamos os 4 motores para a estabilização e definição do controle de velocidades.

Etapa concluída.

e) Complementação da estrutura física para seus quatro rotores no modelo completo do quadricóptero.

Construção do Chassi para agregação do circuito de controle, motores e sensores ao quadricóptero.

Etapa concluída.

f) Simulação em ambiente computacional e otimização das equações dinâmicas e seus referentes coeficientes de ganho.

Mapeamento das variáveis para o equacionamento matemático do protótipo para a estabilização através de controle PID.

g) Simulação em ambiente computacional, otimização do controle, seus referentes coeficientes de ganho dos controladores.

Mapeamento das variáveis para otimização do controle do quadricóptero para a estabilização e velocidade do modelo proposto.

h) Implementação do algoritmo de controle na plataforma do *quadricóptero*.

Implementação do programa em Arduino para o controle do quadricóptero em voo e estabilização do conjunto proposto.

i) Análise de resultados e conclusão.

Apresentação dos resultados do projeto, da documentação da pesquisa e possíveis publicações em congressos na área.

Tabela 1- Cronograma proposto no Projeto de Pesquisa Docente (Ano 1).

ANO 1

Atividades	Fev.	Mar	Abr.	Mai	Jun.	Jul	Ago.	Set	Out	Nov.	Dez
a) Diagramação mecânica e cinemática da estrutura física do aeromodelo, suas características sobre capacidade de sustentação e estabilidade											
b) Implementação do algoritmo de controle e sua utilização em plataforma de um grau de liberdade.											
c) Parametrização do algoritmo de interfaceamento sensores-microcontrolador;											
d) Construção do circuito de desenvolvimento agregado ao microcontrolador, estrutura do aeromodelo com quatro rotores e seus respectivos ESCs (Electronic Speed Controllers) e da plataforma de teste para sustentação.											
e) Complementação da estrutura física para seus quatro rotores no modelo completo do quadricóptero;											

Tabela 2 - Cronograma proposto no Projeto de Pesquisa Docente (Ano 2).

ANO 2											
Atividades	**Fev.**	**Mar**	**Abr.**	**Mai**	**Jun.**	**Jul**	**Ago.**	**Set**	**Out**	**Nov.**	**Dez**

f) Simulação em ambiente computacional e otimização das equações dinâmicas e seus referentes coeficientes de ganho;											
g) Simulação em ambiente computacional, otimização do controle, seus referentes coeficientes de ganho dos controladores;											
h) Implementação do algoritmo de controle na plataforma do quadricóptero.											
i) Análise de resultados e conclusão.											

2 - HOUVE ALTERAÇÕES NO SEU PLANO DE TRABALHO (CRONOGRAMA)? QUAIS?

O cronograma inicial foi modificado devido a alteração do calendário proposto pelo Programa de Pesquisa Docente (PPD). As adaptações do novo cronograma estão definidas nas Tabelas 3 e 4.

Tabela 3- Cronograma modificado no Projeto de Pesquisa Docente (Ano 1).

ANO 1											
Atividades	**Fev.**	**Mar**	**Abr.**	**Mai**	**Jun.**	**Jul**	**Ago.**	**Set**	**Out**	**Nov.**	**Dez**
a) Diagramação mecânica e cinemática da estrutura física do aeromodelo, suas características sobre capacidade de sustentação e estabilidade											

b) Implementação do algoritmo de controle e sua utilização em plataforma de um grau de liberdade.											
c) Parametrização do algoritmo de interfaceamento sensores-microcontrolador;											
d) Construção do circuito de desenvolvimento agregado ao microcontrolador, estrutura do aeromodelo com quatro rotores e seus respectivos ESCs (Electronic Speed Controllers) e da plataforma de teste para sustentação.											
e) Complementação da estrutura física para seus quatro rotores no modelo completo do quadricóptero;											

Tabela 4- Cronograma modificado no Projeto de Pesquisa Docente (Ano 2).

ANO 2											
Atividades	**Fev.**	**Mar**	**Abr.**	**Mai**	**Jun.**	**Jul**	**Ago.**	**Set**	**Out**	**Nov.**	**Dez**
e) Complementação da estrutura física para seus quatro rotores no modelo completo do quadricóptero;											
f) Simulação em ambiente computacional e otimização das equações											

dinâmicas e seus referentes coeficientes de ganho;											
g) Simulação em ambiente computacional, otimização do controle, seus referentes coeficientes de ganho dos controladores;											
h) Implementação do algoritmo de controle na plataforma do quadricóptero.											
i) Análise de resultados e conclusão.											

Tabela 5- Cronograma modificado no Projeto de Pesquisa Docente (Ano 3).

ANO 3											
Atividades	**Fev.**	**Mar**	**Abr.**	**Mai**	**Jun.**	**Jul**	**Ago.**	**Set**	**Out**	**Nov.**	**Dez**
g) Implementação do algoritmo para o levantamento do voo do drone e implementação do algoritmo para controle do drone através dos joystics;											
h) Implementação do algoritmo do algoritmo para a comunicação entre o ESP do drone e o ESP da 'manete'.											
i) Implementação do do algoritmo para controle da velocidade do drone e do algoritmo PID para obter estabilidade do drone											

enquanto estiver no ar sofrendo avarias.											

3 - ESTRUTURA DO RELATÓRIO, CONSIDERANDO O ART. 17º DO REGULAMENTO DO PROGRAMA PESQUISADOR DOCENTE

A- INTRODUÇÃO

O município de Santa Fé do Sul está localizado extremo Noroeste Paulista, nas divisas de Minas Gerais e Mato Grosso do Sul. Atualmente, o município é um dos 29 municípios paulistas considerados estâncias turísticas, fortalecendo assim, a indústria do entretenimento, fazendo com que milhares de turistas visitam a cidade todos os anos.

Santa Fé do Sul, também se encontra em uma das maiores regiões produtoras do setor sucroalcooleiro e próxima ao Mato Grosso do Sul, hoje uma das maiores produtoras de papel e celulose do país. Estas indústrias estão em crescente expansão e desenvolvimento, gerando muitos empregos e riquezas para a região.

Para a supervisão e controle dos recursos turísticos, e plantio de cana-de-açúcar e plantação eucalipto, da indústria de papel e celulose, podemos utilizar o monitoramento através de Veículos Aéreos não Tripulados (VANTs) - drones.

Dada a crescente procura por derivados do setor sucroalcooleiro, como açúcar, etanol e energia renovável, verifica-se a grande importância da cana-de-açúcar na economia brasileira. O Brasil destaca-se como maior produtor mundial de cana-de-açúcar, com aproximadamente 640 milhões de toneladas processadas na safra de 2017/2018, de acordo com dados divulgados pela União da Indústria de Cana-de-Açúcar (ÚNICA, 2018), o que resultou em aproximadamente 38,69 milhões de toneladas de açúcar e 27,80 bilhões de litros de etanol para a agroindústria sucroalcooleira (NOVACANA, 2017).

Com a expansão da cultura de cana-de-açúcar no país, aumentando eficiência, produtividade e competitividade, e com o mercado sucroalcooleiro cada vez mais competitivo,

as usinas estão buscando novas alternativas para o desenvolvimento da produtividade agrícola, de forma se estabelecer no mercado de forma competitiva.

Neste cenário de desenvolvimento da produtividade agrícola, os avanços em tecnologias da informação e comunicação (TIC) estão cada vez mais presentes e sendo impactantes, permitindo o armazenamento e processamento de grandes volumes de dados, automatização de processos e o intercâmbio de informações e de conhecimento (MASSRUHÁ; LEITE, 2016). Observa-se atualmente uma forte relação do gerenciamento com a indústria 4.0, que representa novos processos e produtos originados de avanços tecnológicos, com aplicações em várias áreas do conhecimento (SCHWAB, 2016; CONFEDERAÇÃO DA AGRICULTURA E PECUÁRIA DO BRASIL, 2018).

Segundo Lopes (2018) e Simões; Soler; Py (2017), as transformações digitais também chegaram nas atividades rurais, nas quais surge o termo agricultura 4.0, fortalecendo a ligação urbano-rural nas suas diversas abordagens e permitindo dados cada vez mais precisos. A agricultura 4.0 utiliza tecnologias de ponta na agricultura, incentivando processos na cadeia de valor agregado da produção agrícola. Esta tecnologia pode ser uma importante revolução no campo, tornando as lavouras mais eficientes e sustentáveis, bem como auxiliando a troca de informações e os processos com as empresas (SIMÕES; SOLER; PY, 2017).

As transformações digitais e a tecnologia aparecem no setor agrícola de várias formas. A automação do processo, como da mão-de-obra, aconteceu com a ascensão de novas ferramentas, inovadoras e independentes (SIMÕES; SOLER; PY, 2017). Dentre os exemplos de tecnologias e ferramentas utilizadas, encontram-se drones, caracterizados como os veículos aéreos não tripulados (VANT), que são utilizados para monitoramento das lavouras, identificação de falhas no plantio, entre outras atividades.

Segundo a Agência Nacional de Aviação Civil (ANAC, 2017), drones são aeronaves (ou mesmo outro tipo de veículo) que possua alto grau de automatismo. A ANAC caracteriza as aeronaves com finalidade comercial, corporativa ou experimental como *Remotely Piloted Aircraft* (RPA - Aeronaves Remotamente Pilotadas).

O Sindicato Nacional das Empresas de Aviação Agrícola (SINDAG, 2017) supõem que os drones serão substitutos de boa parte dos aparelhos de pulverização, trazendo precisão a segurança na operação, ao mesmo tempo que coletam imagem em um raio maior do que seria possível com acompanhamento da mão-de-obra na lavoura.

A tecnologias abortadas neste projeto são de interesse para a constante evolução das indústrias do turismo, sucro-alcoleira e papel e celulose.

- ✓ **JUSTIFICATIVA**

A ênfase especial que os quadrimotores têm ganhado em cursos de Computação e matérias de sistemas de controle robóticos, estendendo suas aplicações em usos militares, civis, para observação, e até mapeamento de ambientes desconhecidos, é claramente justificável por sua mobilidade aérea.

Para que um drone construído no formato de um quadrimotor possa realizar tarefas de supervisão de ambiente torna-se necessário conhecer o posicionamento e orientação espacial, para que o quadricóptero possa se movimentar neste ambiente.

Logo, a estabilidade do voo do quadricóptero é essencial para que ele possa ser utilizado, impedindo que ele possa colapsar devido a uma forte corrente de vento, ou devido ao seu próprio peso ou até mesmo devido informações com perturbações do empuxo resultante entre as hélices do sistema. Para evitar estes problemas é necessário que se crie um sistema de controle capaz de corrigir as variações devido às perturbações, reposicionando o drone na sua posição de equilíbrio com o menor tempo possível. O controle por PID (*Proportional Integral Derivative* - Proporcional Derivativo e Integrativo) apresenta-se como uma solução de controle linear, utilizado em várias aplicações na indústria de automação, sendo as vantagens deste método segundo Leong (2012), possuir estrutura simples e bom desempenho.

Assim, tem-se como justificativa deste trabalho o estudo e confecção de um quadricóptero com componentes comerciais, e a implementação de um algoritmo de controle computacional

baseado na teoria de sistemas lineares em um sistema dinâmico de estabilidade zero, como os encontrados nos quadrimotores.

✓ PROBLEMATIZAÇÃO

Alguns estudos e trabalhos, como os de Beard (2008) e Zu, Zhang e Shan (2017), dedicam-se à determinação de modelos matemáticos abrangentes e úteis aos problemas de simulação e controle de quadrimotores. Outros, como o de Alves (2012) e Khatoon, Gupta e Das (2014), dedicam-se ao estudo de técnicas para o controle deles, através do uso de técnicas de controle lineares e não lineares.

Esta pesquisa se dedica ao desenvolvimento de estabilização de um drone com 4 motores, envolvendo tanto o desenvolvimento de um protótipo, quanto a modelagem matemática dinâmica do mesmo, o projeto dos controladores de estabilização de voo e o desenvolvimento do software embarcado para controle do drone. Este trabalho também tem como objetivo ser *open source*, o que permite sua replicação, e a um baixo custo, visto que os materiais utilizados foram selecionados baseando-se em seus custos de mercado e nos requisitos mínimos impostos durante a fase de desenvolvimento. O quadrimotor desenvolvido trata-se de um drone em formato de cruz com dois pares de rotores que rotacionam em sentidos opostos a fim de eliminar os torques gerados.

Diferentes controladores para a estabilização de voo de quadrimotores existem, lineares e não lineares, este trabalho visa estabelecer um comparativo entre os resultados obtidos e compará-los com os métodos já elaborados na literatura. Os controladores lineares desenvolvidos utilizarão técnicas conhecidas como PID (*Proportional Integral Derivative* - Proporcional Integral e Derivativo) (Ogata, 2011).

✓ OBJETIVOS

○ OBJETIVO GERAL

Esta pesquisa de pesquisa tem por objetivo a construção completa de um protótipo do tipo quadricóptero, implementar o controle dos sensores e o equacionamento de seu

algoritmo de controle PID, tornando o drone deste projeto, ser capaz de manter voo com a melhor estabilidade.

- **OBJETIVOS ESPECÍFICOS**

Os objetivos específicos definidos na proposta deste trabalho são:

a) Levantamento e equacionamento dinâmico do VANT do tipo quadricóptero.

b) Construção de um algoritmo de controle em Arduino utilizando lei de controle PID.

c) Construção da estrutura (chassi) do quadricóptero.

d) Aquisição e análise de dados pelos sensores utilizados.

e) Simulação utilizando os parâmetros do modelo desenvolvido.

f) Implementação do algoritmo de controle e aplicação no protótipo construído.

B – METODOLOGIA

Foram realizadas pesquisas em livros, artigos científicos e periódicos para conhecimento e aprofundamento do tema.

Esta pesquisa possui carácter interdisciplinar, onde iniciamos pelas conceituações teóricas, análises matemáticas e desenvolvimento computacional, como forma de definir e direcionar as sequências de opções mais adequadas e determinação de um melhor resultado. Deste ponto em diante, os esforços estão na transposição das soluções teóricas e virtuais para o sistema real, através das seguintes etapas:

a) Diagramação mecânica e cinemática da estrutura física do aeromodelo, suas características sobre capacidade de sustentação e estabilidade;

b) Implementação do algoritmo de controle e sua utilização em plataforma de um grau de liberdade;

c) Parametrização do algoritmo de interfaceamento sensores-microcontrolador;

d) Construção do circuito de desenvolvimento agregado ao microcontrolador, estrutura do aeromodelo com quatro rotores e seus respectivos ESCs (*Electronic Speed Controllers* – Controladores Eletrônicos de Velocidade) e da plataforma de teste para sustentação;

e) Complementação da estrutura física para seus quatro rotores no modelo completo do quadricóptero;

f) Simulação em ambiente computacional e otimização das equações dinâmicas e seus referentes coeficientes de ganho;

g) Simulação em ambiente computacional, otimização do controle, seus referentes coeficientes de ganho dos controladores;

h) Implementação do algoritmo de controle na plataforma do *quadricóptero*;

i) Análise de resultados e conclusão.

- **MATERIAL E REQUISITOS TÉCNICOS NECESSÁRIOS PARA O DESENVOLVIMENTO DO PROJETO**

✓ **MATERIAIS UTILIZADOS**

Para a implementação do projeto foram utilizados softwares gratuitos e de código aberto para o desenvolvimento da aplicação. Assim, não sendo necessários a aquisição de licenças de softwares privados.

Para a confecção do drone da pesquisa, serão utilizados os seguintes componentes: ESP32, ESP32-CAM, Módulo Giroscópio com Acelerômetro 3 eixos, 1 Bateria URUAV 11.1V Lipo, 4 Motores *Brushless* 2300 kV e 4 Placas ESC – 30 A. Todos os materiais já foram adquiridos para a realização da pesquisa científica.

✓ **REQUISITOS TÉCNICOS NECESSÁRIOS**

Conhecimentos em Engenharia de Software, Programação de Computadores, Arquitetura de Computadores, Matemática Básica e Física.

Este Projeto de Pesquisa Docente (PPD) deseja estudar e apresentar soluções relevantes sobre a possibilidade de se aplicar o conhecimento adquiridos no curso de Análise e Desenvolvimento de Sistemas, para o equacionamento, implementação e construção de um quadricóptero de baixo custo.

Devido à complexidade da dinâmica envolvida no somatório de fenômenos físicos que variam ao longo do tempo, pode-se perceber que o equacionamento de voo deste sistema, depende de um controle de múltiplos níveis e não estacionário. Este tipo de planta, para ter sua solução ótima, demanda da linearização do sistema e da implementação de um controle robusto.

Algumas literaturas recentes, citam técnicas de inteligência artificial, como possibilidades de soluções que poderiam alcançar a adaptação necessária ao controle de um sistema que possui variação de parâmetros e equações em função do tempo. Sistemas que utilizam técnicas que realizam o recálculo de seus parâmetros, e que as reutilizam como forma de aprendizado, têm maiores chances de alcançar o controle eficaz de um sistema não estacionário. Estas técnicas de programação podem ser exemplificadas pelo estudo das redes neurais, nebulosas, grafos, genéticos e *Fuzzy*, como descrito por este artigo da Universidade do Tehran:

Embora o quadrirotor tenha vantagens em uma construção mecânica fácil contra o helicóptero tradicional, ainda há questões que impedem que ele seja amplamente utilizado em muitos dos campos sugeridos de aplicação. Por exemplo, o controle e orientação do quadrirotor para estabilização, é uma tarefa difícil, devido ao comportamento dinâmico não linear. Métodos de controle convencionais usam teoria linear que é adequado apenas para sistemas lineares. O controle Fuzzy é não-linear, portanto, é adequado para o controle do sistema não-linear (ABBASI; MAHJOOB; YAZDANPANAH, 2013).

Contudo, algoritmos metaheurísticos (como por exemplo Algoritmos Evolutivos ou Genéticos) atuam em espaços de busca relativamente grandes e exigem alta precisão no computar das operações, para muitos casos. Nos casos de algoritmos de inteligência artificial, o problema de custo computacional (processamento) é ainda maior, já que as operações são computacionalmente intensas e exigem altas taxas de leitura e escrita em memória. Esse poder computacional não é totalmente disponível nos processadores e sistemas de embarcados em VANTs. Além disso, operações computacionalmente intensas aumentam o consumo de energia diminuindo a autonomia de voo.

Assim, existem várias formas de se controlar a estrutura PID original, de forma que estas também podem reagir e se adaptarem às mudanças do ambiente em que se encontra. A técnica *Feedforward* de PID pode ser empregada com este intuito, em um sistema que seja possível variar o sinal de controle em um tempo menor que o tempo que um distúrbio leva para afetar a saída, quantificar os distúrbios do sistema e, os efeitos desses distúrbios na resposta depois das análises dos sensores. Assim, este trabalho tem por objetivo, implementar um algoritmo em Arduino e ESP-32 utilizando o controle PID de forma a viabilizar a estabilização do voo de um drone.

C – DESENVOLVIMENTO DO PROJETO

1 - CONSIDERAÇÕES INICIAIS

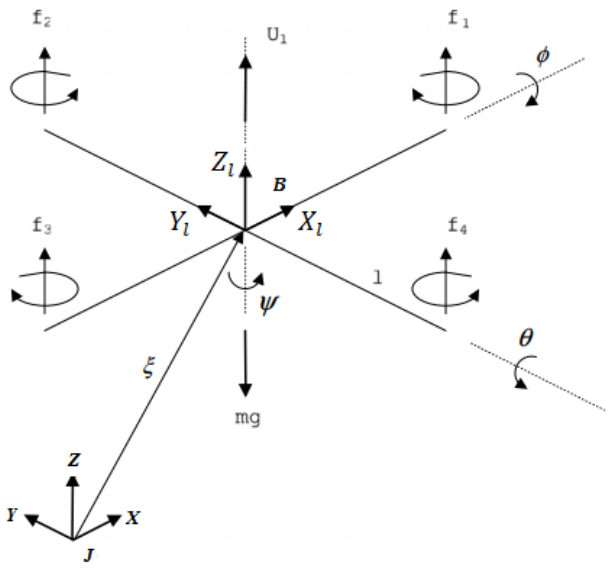
Os drones usam rotores para propulsão e controle. Eles foram projetados de forma que as hélices empurram o ar para baixo. Como as forças vêm em pares, quando o rotor empurra o ar para baixo, o mesmo ar empurra o drone para cima. Assim, quanto mais rápido os rotores giram, maior será a força gerada para subir (ZARDO; REIS; WEBBER, 2021).

Um drone pode executar ações úteis para a compreensão dos fenômenos físicos relacionados a força gravitacional, movimento, aceleração, inércia e rotação. Para exemplificar, a física do drone funciona com os quatro rotores, que estão empurrando o drone para cima, igualando-se à força gravitacional. Para subir, a velocidade dos rotores deve ser aumentada, produzindo uma força ascendente maior que o peso do próprio drone.

Para descer, deve-se diminuir o impulso aos poucos para que a força da gravidade puxe o drone para baixo. Para os movimentos de deslocamento e rotação, os rotores precisam ser coordenados para atuarem em diferentes velocidades, gerando os movimentos desejados do protótipo elaborado (ZARDO; REIS; WEBBER, 2021).

- **Movimentação Vertical:** Os motores são separados da seguinte forma: Sendo que os motores 1-3 sempre rodando em sentido horário e os motores 2-4 em sentido anti-horário. Com os motores rodando as hélices na mesma velocidade e com as rotações dessa forma, o momento angular do drone será completamente 0, como pode observar na Figura 1, (BRAGA, 2013).

Figura 1 – Posição de rotação dos motores



Fonte: (LIMA, 2012).

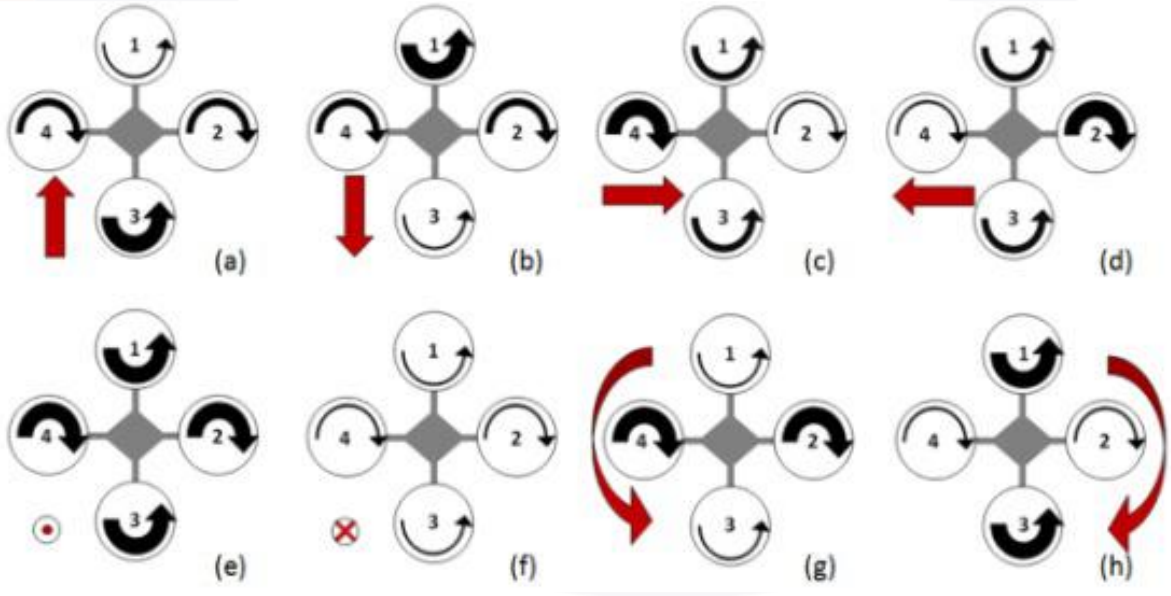
- **Movimentação:** Para a movimentação para frente, a força dos motores Traseiros (3 e 4) tem que aumentar enquanto a força de rotação dos Dianteiros (1 e 2) tem que diminuir. Para ele se movimentar para trás o mesmo processo com as forças invertidas, sendo os traseiros os mais fracos, e os mais fortes, os dianteiros. Pelo fato dos motores traseiros e dianteiros serem um com rotação horário e um anti-

horário, o momento angular do drone continua sendo 0, porém podendo ir para frente ou trás, sem rodar (BRAGA, 2013).

Na Figura 1 são ilustrados os movimentos possíveis, nela a largura da seta é proporcional à velocidade dos rotores.

Os movimentos para direita e esquerda ocorrem quando a velocidade dos motores 1 e 3 permanece a mesma e a velocidade do motor 2 varia em relação a velocidade do motor 4, Figura 2 (c) e 2 (d) (MIRANDA, 2019).

Figura 2 – Posição de rotação dos motores



Fonte: (SÁ, 2012)

O aumento ou a diminuição da velocidade igualmente nos quatro motores faz com que o quadrirotor se mova verticalmente.

A variação de dois rotores situados no mesmo eixo produz um torque em torno do eixo gerando uma aceleração angular. Então, se os dois rotores que giram em sentido anti-horário (rotores 1 e 3) aumentarem suas velocidades angulares e os rotores 2 e 4 permanecessem com velocidade menor, o torque produzido na plataforma faria com que ele gire em sentido horário,

como mostrado na Figura 2(h). O contrário ocorreria se aumentassem os rotores 2 e 4 em relação aos rotores 1 e 3, Figura 2 (g).

1.1 – CÁLCULO PID

Para manter a estabilidade do drone no ar, é de extrema importância utilizar um cálculo PID para atualizar os valores de potência dos motores automaticamente, assim o piloto gozará de maior liberdade no controle do VANT. O cálculo PID funciona da seguinte forma:

• P - representa o valor proporcional, seria o valor que controla a potência de reação dos motores, o que faz o drone sair do chão.

• I - representa o valor integral, é a intolerância ao desnível, onde a leitura da posição atual é diferente do valor esperado (*setpoint*), forçando o motor a trabalhar para alcançar a estabilidade.

• D - representa o valor derivativo, sua função é desacelerar o valor proporcional quando estiver se aproximando do *setpoint*, para assim não ultrapassar o valor inicial, evitando o fenômeno conhecido como *overshoot*, um erro de divergência que atrasa a correção do valor da posição.

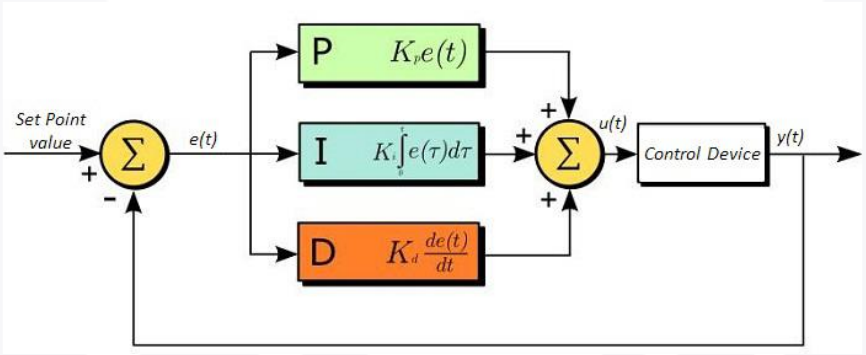
Para melhor visualização do funcionamento do cálculo, observe a Figura 3, que utiliza os dados obtidos através do giroscópio para o controle, em seguida analisa as diferenças entre eles e o *setpoint*, sendo este valor o erro “e(t)”.

O erro “e(t)” é utilizado nas equações representadas na imagem por “P”, “I” e “D”, que por sua vez, o cálculo resulta no sinal de controle “u(t)”, este sinal de controle por sua vez será utilizado no controle de processo utilizando uma representação matemática do sistema, normalmente diferencial, para só assim controlar o processo e alcançar o valor desejado para a saída “y” (*setpoint*).

Por fim, a saída do processo lida mediante a malha de realimentação “y” é subtraída do *setpoint* resultando novamente no sinal de erro “e(t)”, encadeando assim em um *looping* que constantemente corrige a posição do drone, mantendo assim a estabilidade aérea.

Todas as informações descritas estão de acordo com Anon (2016a), consta com adaptações no projeto feitas pelos próprios autores para melhor compreensão do leitor.

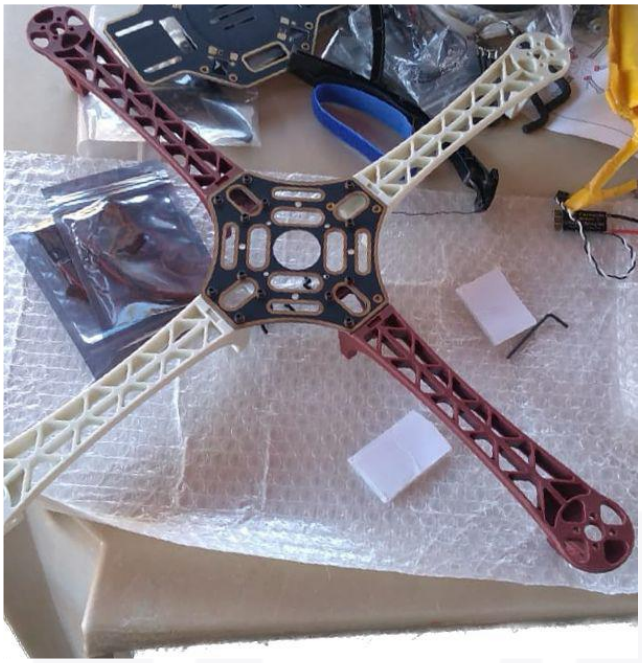
Figura 3 – Alimentação e conexão da controladora com os ESCs



Fonte: (ANON, 2016b)

Na primeira etapa do projeto, optou-se pela aquisição de um chassi pronto, conforme apresentado na Figura 4.

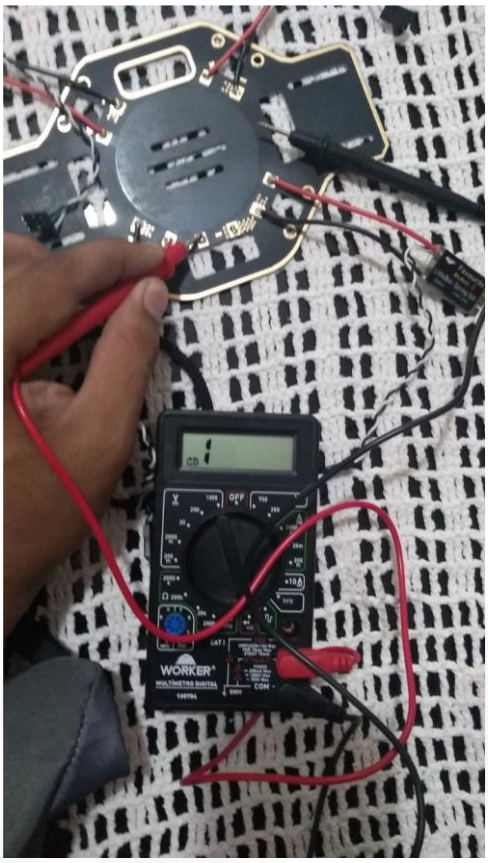
Figura 4 – Chassi do Drone



Fonte: Próprios autores.

Com a chegada das peças iniciou-se o processo de montagem do drone, para assim dar início aos testes. Primeiramente, montou-se o chassi de acordo com o mapa fornecido pelo vendedor, como é apresentado na Figura 4. Em seguida, foram soldados os componentes e feito teste de continuidade com o multímetro para verificar se alguma trilha da placa estava corrompida como mostra a Figura 5.

Figura 5 – Soldagem dos Motores e Conector da Bateria



Fonte: Próprios autores.

Finalizado a etapa de pesquisas e aquisições de equipamentos, iniciou-se os testes de software e hardware, para assim constatar se a comunicação estava funcionando e o controle cumprindo suas funções, o teste foi concluído com êxito observa-se o trabalho de sintonia entre a parte lógica e física do projeto.

1.1.1 AGORITMO 1 – CONTROLE

O algoritmo abaixo tem a função de iniciar o loop de troca de informações entres os ESPs, disparado após enviar dados ao ESP do drone. Além disso, ele permite controlar o drone através do "serial" do *software* "Arduino IDE" e por *joysticks.*

```
#include <esp_now.h>
#include <WiFi.h>
```

```
//MAC ADDRESS DOS ESCRAVOS (SLAVES)
//O endereço de broadcast ({0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}) é um mac andress que qualquer placa vai 'ouvir'
uint8_t MacAndressOfSlaves[][6] = {
  { 0xCC, 0xDB, 0xA7, 0x15, 0x9B, 0x20 }  //Mac address do ESP32 do drone
};

//VARIÁVEIS
const int Canal = 1;  //???????
const int PinosJoystick[4] = { 12, 14, 27, 26 }; //Pinos que estão sendo usados { "JoyStick1X", "JoyStick1Y", "JoyStick2X", "JoyStick2Y" } respectivamente
int QtdPinosJoystick = sizeof(PinosJoystick) / 4; //Quantidade de pinos do joystic
int QtdSlaves = sizeof(MacAndressOfSlaves) / 6; //Quantidade de ESP Escravos, os que vão receber dados

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  Serial.println("Inicializando o modo station");
  Serial.print("Mac Address desse ESP em Station: ");
  Serial.println(WiFi.macAddress());
  Serial.println("================================================================");
  if (esp_now_init() == ESP_OK) {
    Serial.println("ESP Now iniciado com sucesso.");
  } else {
    Serial.println("Não foi possível iniciar o ESPNow.");
    ESP.restart();
  }
  Serial.println("================================================================");
  Serial.println("Configurando Master (Mestre) e Escravos (Slaves)...");
  esp_now_register_send_cb(AoReceberRespostaExecute);  //Registrando a função que é chamada quanto recebe a resposta do envio de dados
  for (int i = 0; i < QtdSlaves; i++) {
    esp_now_peer_info_t Slave = {};
    Slave.channel = Canal;
    Slave.encrypt = 0;
    memcpy(Slave.peer_addr, MacAndressOfSlaves[i], sizeof(MacAndressOfSlaves[i]));
```

```
    esp_now_add_peer(&Slave);
  }
  Serial.println("================================================================
========");
  //CONFIGURANDO PINOS COMO INPUT
  for (int i = 0; i < QtdPinosJoystick; i++) {
    pinMode(PinosJoystick[i], INPUT);
  }
  Serial.println("================================================================
========");
  //INICIA FUNÇÃO QUE COMEÇA A ITERAÇÃO
  EnviarDados();
}

void loop() {
  //MostrarCoordenadas();
}

void EnviarDados() {
  uint8_t Data[QtdPinosJoystick];
  for (int i = 0; i < QtdPinosJoystick; i++) {
    Data[i] = digitalRead(PinosJoystick[i]);
  }
  esp_err_t response = esp_now_send(MacAndressOfSlaves[0], (uint8_t *)&Data,
sizeof(Data));
  Serial.print("Status do envio: ");
  (response == ESP_OK) ? Serial.println("Dados enviados para") :
Serial.println("Dados não enviados para");
  delay(1000);
}

void AoReceberRespostaExecute(const uint8_t *mac_addr, esp_now_send_status_t
response) {
  char macStr[18];
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  Serial.print(macStr);
  Serial.print("Resposta: ");
  (response == ESP_NOW_SEND_SUCCESS) ? Serial.println("Dados recebido") :
Serial.println("Dados não recebido");
  delay(1000);
```

```
  Serial.println("============================================================
========");
  EnviarDados();
}

void MostrarCoordenadas() {
  Serial.print("Joystick 1: X - ");
  Serial.print(analogRead(PinosJoystick[0]));
  Serial.print(" Y - ");
  Serial.print(analogRead(PinosJoystick[1]));
  Serial.print(" || Joystick 2: X - ");
  Serial.print(analogRead(PinosJoystick[2]));
  Serial.print(" Y - ");
  Serial.println(analogRead(PinosJoystick[3]));
  delay(250);
}
```

1.1.2 AGORITMO 2 – DRONE

O algoritmo abaixo tem a função de receber os dados enviados pelo ESP do controle, alterar a velocidade do drone e devolver uma resposta para o ESP do controle. Além disso, é neste ESP que fica o algoritmo para o controle de estabilidade do drone

```
#include <esp_now.h>
#include <WiFi.h>

//SENTIDO DO DRONE
//Para localizar os motores, é necessário tomar a perspectiva de que o drone
está na sua frente e o interruptor da bateria está direção oposta a sua, sendo
o lado do interupitor a 'Frente' e o lado oposto 'Trás'
//21 22
//MPU 3 = 22 = SCL
//MPU 4 = 21 = SDA
//VARIÁVEIS E CONSTANTES
int AntigaVelocidade;
int NovaVelocidade;
const int Frequencia = 50;
const int Resolucao = 12;
const int PinosMotores[4] = { 14, 18, 19, 26 };    //Pino x Motor:
{TrásEsquerda, TrásDireita, FrenteDireita, FrenteEsquerda} respectivamente
```

```
int VelocidadeBase[4] = { 5, 4, 4, 4 };  //3,3,4,4 - A velocidade base é
necessária e individual para cada motor, pois mesmo que os motores sejam
iguais, eles reagem de forma diferente para a mesma quantidade de energia
const int Canais[4] = { 0, 1, 2, 3 };
int QtdPinosMotores = sizeof(PinosMotores) / 4;
//Váriaveis para subir a velocidade do drone pelo Serial
bool isNumber;
char CaracterSerial;
char Escolha;
String TextoSerial = "";

//   VELOCIDADES
// Mínimo |   Máximo
//0 == 205|4095 == 410

void AumentarVelocidade() {
  for (; AntigaVelocidade <= NovaVelocidade; ++AntigaVelocidade) {
    EscreverVelocidade();
  }
}

void DiminuirVelocidade() {
  for (; AntigaVelocidade >= NovaVelocidade; --AntigaVelocidade) {
    EscreverVelocidade();
  }
}

void EscreverVelocidade() {
  for (int i = 0; i < QtdPinosMotores; i++) {
    ledcWrite(i, VelocidadeBase[i] + AntigaVelocidade);
  }
  Serial.println(AntigaVelocidade);
  delay(15);
}

void setup() {
  Serial.begin(9600);
  WiFi.mode(WIFI_STA);
  Serial.println("Inicializando o modo station");
  Serial.print("Mac Address desse ESP em Station: ");
  Serial.println(WiFi.macAddress());
```

```
  Serial.println("================================================================
========");
  if (esp_now_init() == ESP_OK) {
    Serial.println("ESPNow iniciado com sucesso.");
  } else {
    Serial.println("Não foi possível iniciar o ESPNow.");
    ESP.restart();
  }
  Serial.println("================================================================
========");
  Serial.println("Configurando...");
  esp_now_register_recv_cb(AoReceberDadosExecute);  //Registrando a função que
recebe os dados do envio dos dados
  //Adicionando informações aos slaves (ESPs)
  Serial.println("================================================================
========");
  //Seta a velocidade para que ele levanta 10 centimetros
  AntigaVelocidade = map(0, 0, 4095, 205, 410);
  for (int i = 0; i < sizeof(PinosMotores) / 4; i++) {
    pinMode(PinosMotores[i], OUTPUT);
    ledcSetup(i, Frequencia, Resolucao);
    ledcAttachPin(PinosMotores[i], i);
    ledcWrite(i, AntigaVelocidade);
    Serial.println(PinosMotores[i]);
  }
}

void loop() {
  if (Serial.available()) {
    TextoSerial = "";
    isNumber = true;
    delay(1000);
    while (Serial.available()) {
      CaracterSerial = Serial.read();
      if (CaracterSerial != '\n') {
        TextoSerial.concat(CaracterSerial);
        if (!isDigit(CaracterSerial)) {
          isNumber = false;
        }
      }
    }
```

```
    Serial.println("============================================================
==========");
    Serial.print("Texto digitado: ");
    Serial.println(TextoSerial);
    Serial.println("============================================================
==========");
    if (isNumber) {
      NovaVelocidade = TextoSerial.toInt();
      NovaVelocidade = map(NovaVelocidade, 0, 4095, 205, 410);
      Serial.print("Antiga velocidade: ");
      Serial.println(AntigaVelocidade);
      Serial.print("Nova velocidade: ");
      Serial.println(NovaVelocidade);
      Serial.println("============================================================
============");
      if (NovaVelocidade == AntigaVelocidade) {
        Serial.println("A velocidade digitada é a mesma da atual, digite outra
velocidade!!!");
      } else if (NovaVelocidade > AntigaVelocidade) {
        Serial.print("Aumentando a velocidade das hélices PARA ");
        Serial.println(NovaVelocidade);
        for (; AntigaVelocidade <= NovaVelocidade; ++AntigaVelocidade) {
          for (int i = 0; i < QtdPinosMotores; i++) {
            ledcWrite(i, VelocidadeBase[i] + AntigaVelocidade);
          }
          Serial.println(AntigaVelocidade);
          delay(15);
        }
      } else {
        Serial.println("Diminuindo a velocidade das hélices...");
        for (; AntigaVelocidade >= NovaVelocidade; --AntigaVelocidade) {
          for (int i = 0; i < QtdPinosMotores; i++) {
            ledcWrite(i, VelocidadeBase[i] + AntigaVelocidade);
          }
          Serial.println(AntigaVelocidade);
          delay(15);
        }
      }
      AntigaVelocidade = NovaVelocidade;
      for (int i = 0; i < QtdPinosMotores; i++) {
        Serial.print("Velocidade do motor ");
        Serial.print(PinosMotores[i]);
```

```
        Serial.print(" é ");
        Serial.print(VelocidadeBase[i] + AntigaVelocidade);
        Serial.print(" = ");
        Serial.print(VelocidadeBase[i]);
        Serial.print(" + ");
        Serial.println(AntigaVelocidade);
      }
    } else {
      Serial.println("Desculpe, o texto digitado não é um número!!!");
    }
  }
}

void AoReceberDadosExecute(const uint8_t *mac_addr, const uint8_t *Data, int
data_len) {
  char macStr[18];
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  Serial.print("Recebido de: ");
  Serial.println(macStr);
  NovaVelocidade = map(Data[1], 0, 4095, 205, 410);
  for (int i = 0; i < QtdPinosMotores; i++) {
    ledcWrite(i, NovaVelocidade);
    delay(250);
  }
  Serial.println("==========================================================
========");
}
```

1.1.3 ALGORITMO 3 – CÁLCULAR AS ROTAÇÕES REAIS DO DRONE

O algoritmo abaixo tem a função de receber os dados enviados pelo componente MPU6050 que foi pedido pelo drone e calcular a velocidade angular e os ângulos de rotação do *pitch* e do *roll*, com os resultados dos cálculos será possível dar andamento aos cálculos de PIDs para cada eixo de rotação.

```
#define usCiclo 5000  // Ciclo de execução do software em microssegundos
#include <Wire.h>

// MPU6050
#define EnderecoMPU6050 0x68
```

```
float InformacoesMPU6050[7], InformacoesCalibradasMPU6050[7], GyroZ, GyroX,
GyroY, AccTotalVector, AnguloPitchAcc, AnguloRollAcc, AnguloPitch, AnguloRoll,
Temperatura/*, AnguloYaw*/;
bool SetGyroAngles = false;
long TempoExecucao, TempoLoop;
/*
ORDEM DO DADOS NOS ARRAYS
Acelerômetro X
Acelerômetro Y
Acelerômetro Z
Temperatura
Giroscópio X
Giroscópio Y
Giroscópio Z
*/

void setup() {
  Wire.begin();
  Serial.begin(115200);

  // Iniciar sensor MPU6050
  SetupMPU6050();

  //Inicializando arrays
  for(int i = 0; i < 7; i++){
    InformacoesMPU6050[i] = 0;
    InformacoesCalibradasMPU6050[i] = 0;
    Serial.print(InformacoesMPU6050[i]);
    Serial.print(", ");
    Serial.println(InformacoesCalibradasMPU6050[i]);
  }

  // Calibrar giroscópio e acelerômetro. O drone tem que estar em um plano
imóvel. Média de 3000 dados.
  for (int i = 0; i < 3000 ; i ++) {
    LerInformacoesMPU6050();
    for (int i = 0; i < 7; i++) {
      if (i != 3) {
        InformacoesCalibradasMPU6050[i] += InformacoesMPU6050[i];
      }
    }
    delayMicroseconds(50);
```

```
  }

  for (int i = 0; i < 7; i++) {
    if (i != 3) {
      InformacoesCalibradasMPU6050[i] = InformacoesCalibradasMPU6050[i] / 3000;
    }
  }

  TempoLoop = micros();
}

void loop() {
  //Ciclo
  while (micros() - TempoLoop < usCiclo);
  TempoExecucao = (micros() - TempoLoop) / 1000;
  TempoLoop = micros();

  //Ler e cálcular
  LerInformacoesMPU6050();
  CalcularInformacoesMPU6050();

  //Mostrar ângulos
  Serial.print(AnguloPitch);
  Serial.print("\t");
  Serial.println(AnguloRoll);
}

void SetupMPU6050() {
  Wire.beginTransmission(EnderecoMPU6050);
  Wire.write(0x6B);                          // PWR_MGMT_1 registro 6B hex
  Wire.write(0x00);                          // 00000000 para ativar
  Wire.endTransmission();

  Wire.beginTransmission(EnderecoMPU6050);
  Wire.write(0x1B);                          // GYRO_CONFIG registro 1B hex
  Wire.write(0x08);                          // 00001000: 500dps
  Wire.endTransmission();

  Wire.beginTransmission(EnderecoMPU6050);
  Wire.write(0x1C);                          // ACCEL_CONFIG registro 1C hex
  Wire.write(0x10);                          // 00010000: +/- 8g
  Wire.endTransmission();
```

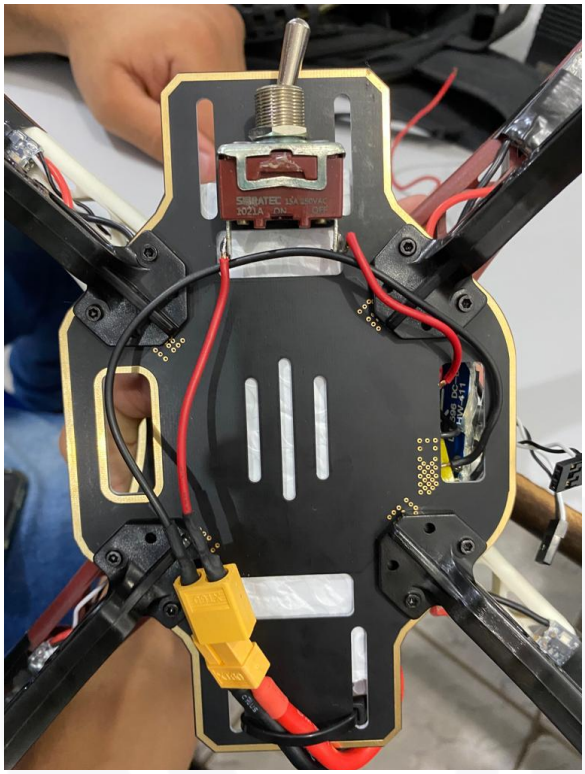
```
}

void LerInformacoesMPU6050() {
  //Os dados do giroscópio vai do 3B a 14
  Wire.beginTransmission(EnderecoMPU6050);  // Iniciar comunicação
  Wire.write(0x3B);                         // Pedir o registro 0x3B (AcX)
  Wire.endTransmission();
  Wire.requestFrom(EnderecoMPU6050, 14);    // Solicitar um total de 14
registros, 2 para cada informação
  while (Wire.available() < 14);            // Espera até receber os 14 dados

  //Atribuição
  for (int i = 0; i < 7; i++) {
    if (i != 3) {
      InformacoesMPU6050[i] = Wire.read() << 8 | Wire.read();
    }else{
      Temperatura           = Wire.read() << 8 | Wire.read();
    }
  }

  //A ORDEM
  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
}

// Cálculo de velocidade angular (º/s) e ângulo (º)
void CalcularInformacoesMPU6050() {
  //Subtrair os valores de calibração do acelerômetro
  InformacoesMPU6050[0] -= InformacoesCalibradasMPU6050[0];
  InformacoesMPU6050[1] -= InformacoesCalibradasMPU6050[1];
  InformacoesMPU6050[2] -= InformacoesCalibradasMPU6050[2];
  InformacoesMPU6050[2]  = InformacoesMPU6050[2] + 4096;

  //Subtrair os valores de calibração do giroscópio e calcular
  //Velocidade angular em º/s. Ler 65.5 em raw equivale a 1º/s
  GyroX = (InformacoesMPU6050[4] - InformacoesCalibradasMPU6050[4]) / 65.5;
  GyroY = (InformacoesMPU6050[5] - InformacoesCalibradasMPU6050[5]) / 65.5;
```

```
  GyroZ = (InformacoesMPU6050[6] - InformacoesCalibradasMPU6050[6]) / 65.5;

  //Calcular ângulo de inclinação com dados do giroscópio
  //0.000000266 = TempoExecucao / 1000 / 65.5 * PI / 180
  AnguloPitch += GyroX * TempoExecucao / 1000;
  AnguloRoll  += GyroY * TempoExecucao / 1000;
  AnguloPitch += AnguloRoll  * sin((InformacoesMPU6050[6] -
InformacoesCalibradasMPU6050[6]) * TempoExecucao  * 0.000000266);
  AnguloRoll  -= AnguloPitch * sin((InformacoesMPU6050[6] -
InformacoesCalibradasMPU6050[6]) * TempoExecucao  * 0.000000266);

  //Cálcular vetor de aceleração
  //57.2958 = Converter de radianos para graus 180/PI
  AccTotalVector  = sqrt(pow(InformacoesMPU6050[1], 2) +
pow(InformacoesMPU6050[0], 2) + pow(InformacoesMPU6050[2], 2));
  AnguloPitchAcc  = asin(InformacoesMPU6050[1] / AccTotalVector) * 57.2958;
  AnguloRollAcc   = asin(InformacoesMPU6050[0] / AccTotalVector) * -57.2958;

  //Filtro complementar
  if (SetGyroAngles) {
    AnguloPitch = AnguloPitch * 0.99 + AnguloPitchAcc * 0.01;
    AnguloRoll  = AnguloRoll  * 0.99 + AnguloRollAcc  * 0.01;
  }else {
    AnguloPitch = AnguloPitchAcc;
    AnguloRoll  = AnguloRollAcc;
    SetGyroAngles = true;
  }
}
```

1.2 - PLACA PCB COM INTERRUPTOR

Foi adaptado um interruptor ao chassi, isto foi feito com o intuito de permitir ligar e desligar o drone para testes de forma mais rápida e simples, pois, se isso não fosse feito, teríamos que sempre ficar conectando e desconectando os pinos macho e fêmea do conector XT60. Mesmo sendo um interruptor de boa qualidade estando desligado ele permite que seja medida uma tensão de aproximadamente de 0.8V, por conta disso, quando os testes forem parados é aconselhável desconectar o pino da bateria do pino do chassi, como pode-se observar na Figura 6.

Figura 6 – Adaptação do Botão Tic-Tac



Fonte: Próprios autores.

1.3 - BRAÇO COM MOTOR E ESC

Neste braço (parte parafusada na peça central do drone, elas acoplam os motores em suas pontas), como nos outros, foi preso em sua ponta um motor sem escova (motores que não usam escovas de contato elétrico para energizar os campos magnéticos) rs2205 2300kv 2205 cW/ccW, ele foi preso com parafusos que vieram junto com o conjunto do *chassi* que compõem o módulo. Para fazer o controle desse tipo de motor precisamos utilizar um ESC (*Eletronic Speed Controller*, Controlador Eletrônico de Velocidade) o qual foi preso na parte de baixo do braço com seus terminais soldados na placa de baixo do chassi - placa PCB (*Printed Circuit Board* - Placa de Circuito Impresso) onde estão presos os pés do drone – e seus pinos de controle, onde será recebido o pulso PWM (*Pulse Width Modulation* – Modulação com largura de pulso), será preso na placa de controle, como pode-se observar na Figura 7.

Figura 7 – Braço com Motor e ESC

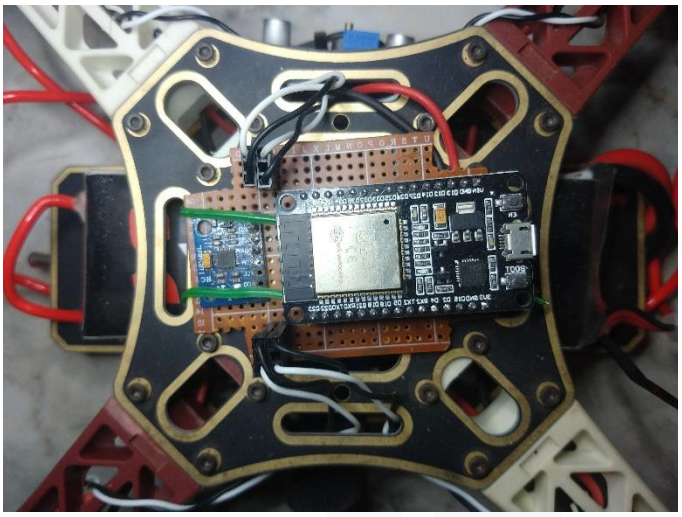
Fonte: Próprios autores.

1.4 - FIXAÇÃO DAS PLACAS AO CHASSI

Para terminar a montagem do drone em si, anexou-se a placa de controle e o servidor web ESP32-CAM ao drone. Portanto, para o ESP32-CAM foi utilizado fita dupla face e uma cinta de velcro azul que veio junto com a compra do chassi. Já para colocar a placa de controle foi utilizado fita dupla face e para evitar o contato da parte eletrônica que fica embaixo da placa de controle com a parte metálica do chassi, foi anexado entre ambos um pedaço de papelão – preso à placa com cola quente – e preso ao papelão a dupla face que o fixou ao chassi, como pode observar nas Figuras 8 e 9.

Figura 8 – Fixação das hélices

Figura 9 – Fixação das placas ao Chassi

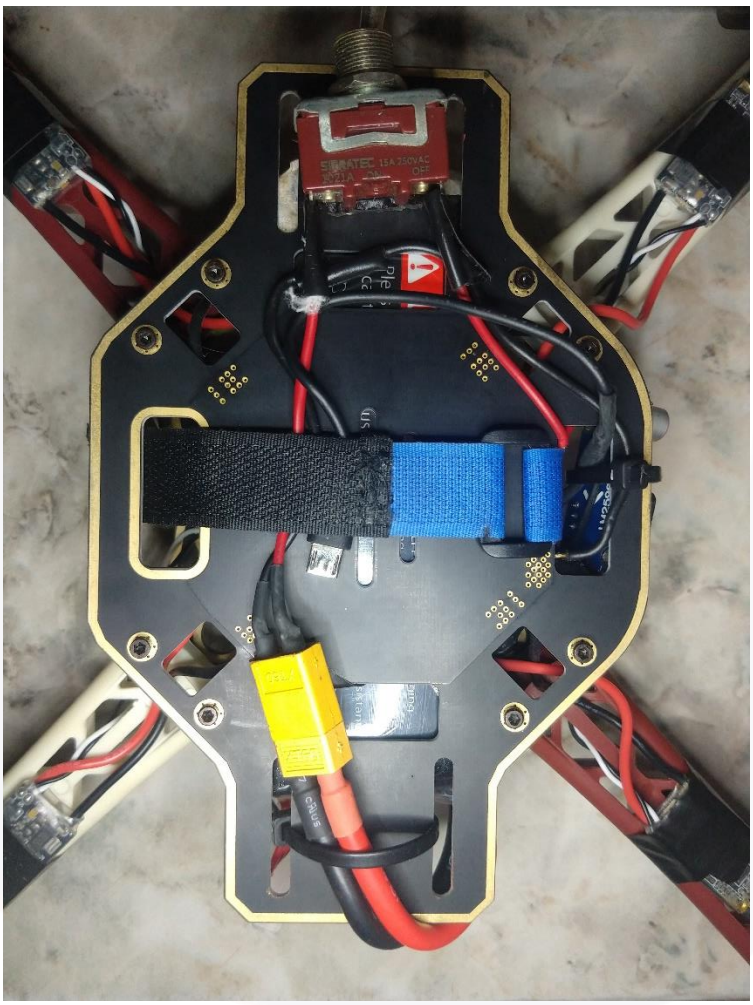


Fonte: Próprios autores.

1.5 - ALIMENTAÇÃO E CONEXÃO DA CONTROLADORA COM OS ESCS

Após prender as duas placas, realizou-se a parte de alimentação de ambas, onde foi soldado a placa, terminais de saída do regulador de tensão que envia para placa uma tensão de 5V. Da placa, a energia é enviada para o ESP32 – por meio de fios soldados aos terminais GND e VIN – e para o ESP32-CAM – conectado a um cabo *micro-usb* que tem seus terminais conectados aos pinos GND e VIN do ESP32. Também foi feita a conexão entre a placa de controle e os ESCs, onde cada pino de entrada de dados de cada ESC foi colocado em cada terminal específico, como pode observar a Figura 10.

Figura 10 – Alimentação e conexão da controladora com os ESCs



Fonte: Próprios autores.

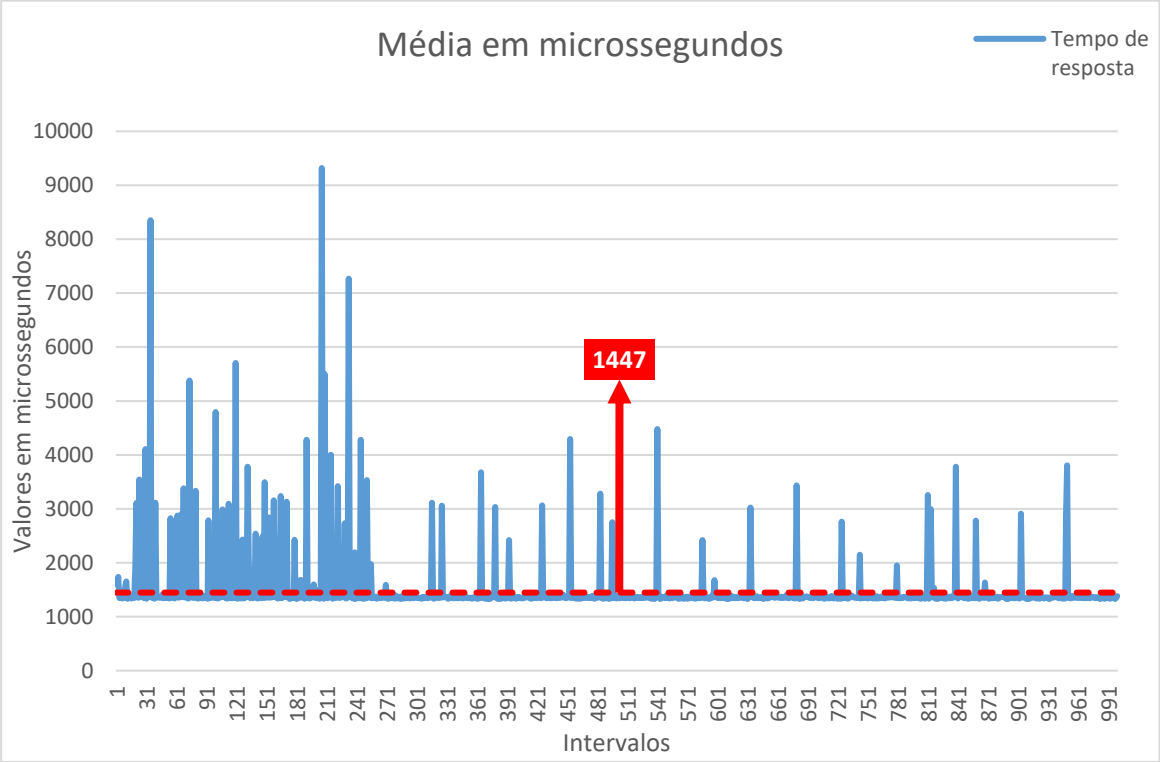
1.6 – ANÁLISE INICIAL

1.6.1 – TEMPO DE COMUNICAÇÃO

O tempo de comunicação entre o controle e o drone é uma boa informação para entender se o tempo resposta está adequado, se o controle não está demorando muito para se comunicar com o drone. Abaixo pode ser visto a equação para o cálculo da média do tempo de comunicação.

$$f(x) = \frac{1}{n} \times \left(\sum_{i}^{n} t \right)$$

Vale considerar que diversas outras coisas podem influenciar no tempo de comunicação entre ambos, como distância entre eles, funções executas não essências antes da resposta do drone, interferências no meio do caminho e outros. Depois de executar algumas vezes o algoritmo, obtive o tempo de comunicação de 1447 microssegundos ou 0,001447 segundos. Veja o gráfico de média dos valores do tempo de resposta.



1.7 - CONSIDERAÇÕES FINAIS

No próximo relatório dar-se-á ênfase os algoritmos e componentes eletrônicos utilizados na etapa **(f)** do cronograma. A junção dos itens propostos no cronograma deste relatório, definirá a composição do texto desta pesquisa científica.

REFERÊNCIAS BIBILIOGRÁFICAS

ABBASI E.; MAHJOOB M. J.; YAZDANPANAH R. **Controlling of Quadrirotor UAV Using a Fuzzy System for Tuning the PID Gains in Hovering Mode.** School of Mechanical Engineering, University of Tehran, Iran, 2013.

ALVES, Ana Sophia Cavalcanti. **Estudo e Aplicação de Técnicas de Controle Embarcadas para Estabilização de Voo de Quadricópteros**, 2012, 121 f. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Juiz de Fora.

ANAC, 2017. Regras sobre drones. Disponível em: http://www.anac.gov.br/noticias/2017/regras-da-anac-para-uso-de-drones-entram-em-vigor/release_drone.pdf. Acesso em: 20 de Dez. 2022.

ANON. **Controlador PID. Você sabe o que é e como funciona? Aula 01 - Teoria**. 2016. Url https://www.youtube.com/watch?v=MPkEPvs8rec.

ANON. **CONTROLADORES PID**. 2016. Url https://smarti.blog.br/controladores-pid/.

BEARD, R. W. **Quadrotor Dynamics and Control**, 2008.

BRAGA, A. P. **Simulação e controle de um quadcopter utilizando lógica fuzzy**. Universidade Estadual Paulista (Unesp), 2013.

KHATOON, S; GUPTA, D.; DAS, L. K. **PID & LQR Control for a Quadrotor: Modeling and Simulation**, IEEE Robotics & Automation Magazine, 2014.

LEONG, Bernard Tat Meng; LOW, Sew Ming; OOI, Melanie Po-Leen. Low-Cost Microcontroller-based Hover Control Design of a Quadcopter. **Procedia Engineering.** V. 41, p. 458-464, 2012.

LOPES, M. A. Agricultura 4.0: o agronegócio também na rota do desenvolvimento tecnológico. 2018. Disponível em: https://www.techminds.info/2018/05/10/agricultura-4-0-oagronegocio-tambem-na-rota-do-desenvolvimento-tecnologico. Acesso em: 02 de Jan. 2023.

MASSRUHÁ, S. M. F. S.; LEITE, M. A. A. Agricultura Digital. RECoDAF – Revista Eletrônica Competências Digitais para Agricultura Familiar, Tupã, v. 2, n. 1, p. 72-88, jan./jun. 2016.

MIRANDA, B. d. A. S. **Montagem de uma plataforma para testes de controle de um veículo aéreo não tripulado quadrimotor**. Dissertação (B.S. thesis) − Universidade Tecnológica Federal do Paraná, 2019.

NOVACANA, 2017. Conab divulga dados finais de 2016/17 e 1º levantamento da safra 2017/18 de cana-de-açúcar. Disponível em: https://www.novacana.com/n/cana/safra/ conabdados-finais-2016-17-levantamento-safra-2017-18-cana-de-acucar-180417/. Acesso em: 10 Jan. 2023.

OGATA, K. **Engenharia de Controle Moderno**. 5th ed. 2011.

SÁ, R. C. **Construção, modelagem dinâmica e controle PID para estabilidade de um veículo aéreo não tripulado do tipo quadrirotor**. 2012.

LIMA, Gabriela et al. Modelagem dinâmica de um veículo aéreo não tripulado do tipo quadricóptero. **ResearchGate**, 2014. Disponível em: https://www.researchgate.net/publication/305318229_MODELAGEM_DINAMICA_DE_UM_VEICULO_AEREO_NAO_TRIPULADO_DO_TIPO_QUADRICOPTERO_DYNAMIC_MODELING_OF_AN_UNMANNED_AERIAL_VEHICLE_QUADROTOR-TYPE. Acesso em: 22 de nov. 2023.

SCHWAB, K. The Fourth Industrial Revolution, WEF, 2016.

SIMÕES, M.; SOLER, L; S.; PY, H. Tecnologias a serviço da sustentabilidade e da agricultura. Boletim informativo da SBCS. Mai-ago, 2017.

SINDAG, 2017. Relatório de Atividades – Brasil 2016. Disponível em: http://sindag.org.br/wp-content/uploads/2016/12/SINDAG-Relatorio-de-Atividades-Abril2016.pdf.

UNICA, 2018. Relatório final da safra 2017/2018. Disponível em: http://www.unicadata.com.br/listagem.php?idMn=102.

ZARDO, M.; REIS, C. E. R. dos; WEBBER, C. G. **Como um drone voa? descubra na aula de física!** RENOTE, v. 19, n. 2, p. 292–301, 2021.

ZU, Y.; ZHANG, W.; SHAN J. **A trajectory design method for quadrotor based on DMOC method and Dubins path algorithm, 36th Chinese Control Conference**, 2017.