

Laboratório de Sistemas Embarcados - ES070

Relatório Final

EH - DRONE

Implementação utilizando ESP-32

ALUNOS

Giácomo Antonio Dollevedo (RA: 173029)

Gustavo Lino Fernandes (RA: 174167)

PROFESSOR

Prof. Rodrigo Moreira Bacurau

Campinas
Janeiro de 2021

1. Resumo	3
2. Introdução	4
3 - Planejamento	5
3.1 - Requisitos do Projeto	5
3.1.1 Requisitos Funcionais	5
3.1.2 Requisitos Não Funcionais	5
3.1.3 Requisitos Adicionais	5
3.2 - Plataformas e Ferramentas para Desenvolvimento	5
3.2.1 - Desenvolvimento de Software	5
3.2.2 - Integração do Hardware	5
3.3 - Modelo de Desenvolvimento	6
3.4 - Divisão de Atividades e Cronograma	6
4.1 Principais Componentes	7
4.1.1 Frame	7
4.1.2 Motores, Hélices e ESCs	8
4.1.3 IMU	10
4.1.4 Microcontrolador	10
4.2 Diagrama de Blocos	11
4.3 Esquema Elétrico	12
4.4 Orçamento	13
5. Projeto de Software do Sistema	14
5.1 Diagrama de camadas	14
5.2 Fluxograma	15
5.3 Diagrama de Classes	17
6.1 Inicialização do Sistema	18
Após isso, o Drone pode ser ligado.	18
6.2 Modo de Testes	20
6.3 Modo Joystick	21
8. Problemas Identificados	25
8.1 Problemas identificados pelos alunos	25
8.1.1 Resolvidos	25
8.1.2 Não Resolvidos	25

1. Resumo

Neste projeto foi desenvolvido um Drone Quadcóptero de forma integral, contemplando a escolha e a integração dos componentes de Hardware, bem como a implementação em Software para garantir o funcionamento do Drone. Durante o projeto, houve a implementação de códigos para realizar a interface do microcontrolador com os principais componentes de hardware escolhidos: quatro motores *Brushless DC*, seus respectivos controladores de velocidade (*ESCs*), um módulo sensor Giroscópio/Acelerômetro, e o módulo WiFi integrado ao microcontrolador. Todo o projeto de Software se encontra documentado em seu repositório no GitHub.

O sistema em questão foi construído para ser capaz de elevar uma carga de ~700 g até uma altura de 1 m, e controlar sua velocidade de giro nos eixos de *Roll* e *Pitch* através de um controlador PID implementado digitalmente. Também permite o monitoramento dos seus parâmetros e envio de comandos de testes pelo usuário, além do controle remoto manual através da comunicação WiFi. Para isso, também houve a implementação de um Servidor para ser utilizado em um computador pessoal.

2. Introdução

Como forma de aplicar e aprofundar ainda mais o conhecimento de sistemas embarcados e microcontroladores obtidos durante a disciplina de Sistemas Embarcados (ES-670), foi projetado um Drone Quadróptero. Com isso, também foi possível implementar técnicas de Controle na prática, contribuindo ainda mais para o aprendizado dos alunos. Por fim, as aplicações de um sistema como esse são de grande interesse em diferentes setores, como agrícolas, logísticos e militares.

Neste documento será apresentado boa parte da maneira que o projeto foi idealizado e a documentação que permitiu a organização dos códigos, como os diagramas UML, o manual de orientação para utilização, a técnica para o controle PID aplicado, a interface com o WiFi de interação com o usuário, que pode ser um usuário final (apenas utilização) ou um usuário técnico (análise e configuração).

Faz parte integral deste documento a descrição do Hardware, do seu funcionamento e de como sua escolha influencia o design geral do projeto Além disso, também cabe a descrição das principais implementações em Software e de seu planejamento, que devem ser encontrados no mesmo diretório digital (GitHub) que este documento foi obtido. Deve-se ainda ressaltar que o desenvolvimento deste projeto enfrentou cenários inesperados, os quais interferiram diretamente no planejamento e propostas iniciais.

Uma seção dedicada para os problemas identificados, e como foi possível (ou não) solucioná-los foi separada ao final para reconhecer adequadamente as limitações e dificuldades superadas.

3 - Planejamento

3.1 - Requisitos do Projeto

Os requisitos do projeto podem ser separados em requisitos funcionais (aqueles que definem o funcionamento do sistema de forma completa e concisa), e em não funcionais (se referem às restrições do projeto que afetam uma ou mais de suas funcionalidades).

3.1.1 Requisitos Funcionais

- Controle de taxa de giro (estabilização)
- Permitir o monitoramento de parâmetros por Wi-Fi
- Permitir a configuração de parâmetros através do WiFi

3.1.2 Requisitos Não Funcionais

- Frame em "X"
- Suportar uma carga mínima de 700g
- Alimentação através de baterias recarregáveis
- Atingir ao menos 1m de altura de voo

3.1.3 Requisitos Adicionais

- Permitir o controle remoto pelo usuário através do WiFi

3.2 - Plataformas e Ferramentas para Desenvolvimento

3.2.1 - Desenvolvimento de Software

A maior parte deste projeto foi desenvolvida em linguagem C/C++, uma vez que esta é a linguagem para operação do microcontrolador escolhido.

Como ferramenta de desenvolvimento, utilizou-se majoritariamente o software Visual Studio Code, que conta com várias extensões que auxiliam a tarefa de programação. Além disso, também foi utilizado o Arduino IDE para que o código fosse exportado para o microcontrolador.

Para controle de versão dos códigos, e também para seu fácil armazenamento e compartilhamento, utilizou-se a plataforma GitHub. Todo o projeto, incluindo as rotinas de testes realizadas, pode ser encontrado no repositório.

3.2.2 - Integração do Hardware

Para integração e interface de todo o hardware do sistema foi utilizado o microcontrolador ESP-32. Os componentes integrados junto a uma melhor descrição do microcontrolador se encontram na próxima seção (**4 - Projeto de Hardware do Sistema**).

3.3 - Modelo de Desenvolvimento

Para execução do projeto, seguiu-se um modelo do tipo “cascata”, ilustrado, em sua forma mais usual, na figura a seguir (Figura 1).

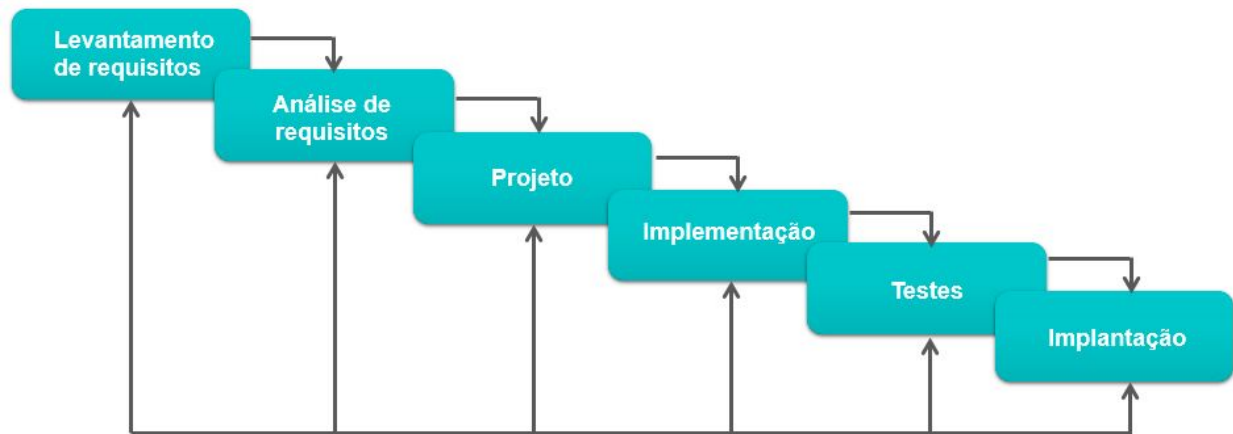


Figura 1 - Modelo de Desenvolvimento em Cascata.

Este tipo de modelo foi escolhido principalmente pela necessidade de progressão linear deste projeto. Como a principal funcionalidade do sistema (voo e estabilização do Drone) depende da integração prévia de todos os componentes, os testes precisam ser realizados ao final do projeto. Assim, modelos que preveem iterações rápidas de prototipagem e análise (como “Espiral” ou “Ágil”) não seriam a escolha ideal para o projeto.

Como mostra a Figura 1, as primeiras etapas contemplam o levantamento e análise de requisitos, descritos anteriormente em 3.1. A etapa de Projeto foi separada em projeto de Hardware (seção 4) e de Software (seção 5). As implementações e rotinas de testes podem ser encontradas no repositório do GitHub deste projeto.

3.4 - Divisão de Atividades e Cronograma

A principal decisão para nortear a separação das atividades, foi quem ficaria com a montagem dos componentes, a finalização do hardware propriamente dito do sistema, uma vez que o trabalho foi desenvolvido de forma remota entre os dois estudantes e não seria uma estratégia viável montar dois drones, seja por questões financeiras ou logísticas.

O cronograma foi realizado levando em conta a programação de entrega parcial das atividades para avaliação, mais especificamente denominadas Etapa1, Etapa2, Etapa3, Etapa4 e Etapa5. A descrição do que era esperado em cada uma dessas etapas estão no arquivo “Aula1.pdf”, o cronograma feito utilizando a ferramenta Project Libre pode ser encontrado no formato PDF também no arquivo “Cronograma.pdf”, paralelamente foi utilizado o Trello para acompanhamento das atividades e discussões, mas que se mostrou pouco eficiente, uma vez que a maior parte era tratada e resolvida pelo canal de comunicação direto através do Discord.

De maneira geral, o cronograma conseguiu ser seguido com algumas ressalvas flexibilizados, o atraso de uma semana prevista foi readequado e as atividades que não foram entregues especificamente na data prevista pelo cronograma também não impediram a entrega das atividades seguintes e não impactaram de forma mais representativa no cronograma final.

4. Projeto de Hardware do Sistema

Nesta seção será apresentado o projeto de hardware do sistema. Em um projeto como este, a escolha de componentes deve ser feita de maneira cautelosa, já que as características de desempenho final do sistema estão muito atreladas aos próprios itens de hardware. Além disso, cada um dos componentes influencia diretamente na escolha dos outros, já que deve ser levado em consideração a compatibilidade e o desempenho entre o próprio hardware.

4.1 Principais Componentes

A seguir, estão descritos em maiores detalhes o papel de cada um dos principais componentes de um Drone, quais foram as escolhas para este projeto e sua justificativa, e as características de cada uma delas. Além disso, o peso é uma das características mais importantes de cada componente, já que um dos objetivos finais é a capacidade de elevação de carga.

4.1.1 Frame

O “*Frame*”, ou corpo do Drone, é a base para montagem de todo o sistema. Nele deve haver encaixes para todos os demais componentes de hardware (principalmente os motores), e ele deve ser suficientemente grande para comportar as hélices, e leve para que o conjunto hélice + motor seja capaz de erguer sua carga.

Neste projeto, foi escolhido um frame feito através de impressão 3D em ABS. Este tipo de manufatura foi escolhida por permitir uma alta customização e custo relativamente baixo. O modelo utilizado para impressão é baseado no apresentado na figura 2, com uma pequena modificação no braço para que os furos encaixem com o motor escolhido.

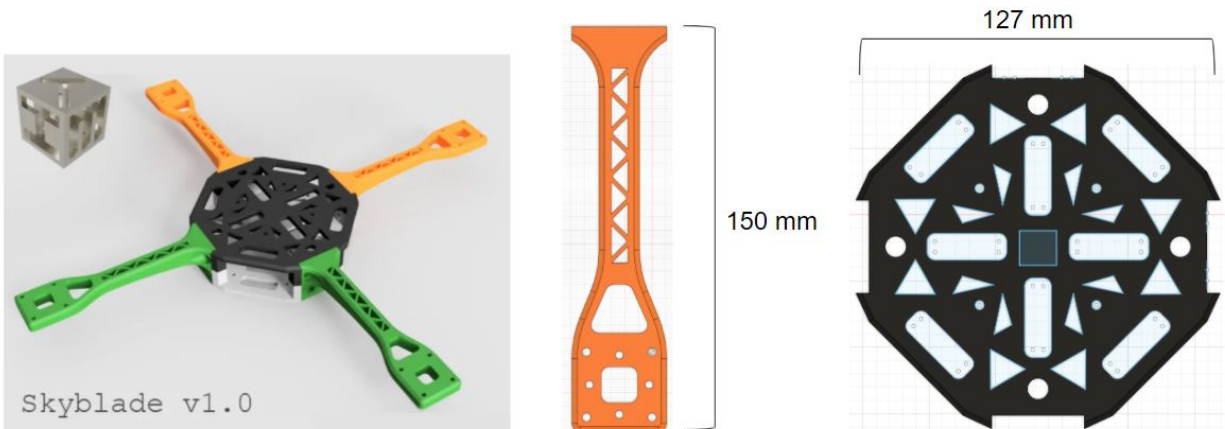


Figura 2 - Modelo 3D do Frame utilizado. Este modelo se encontra disponível em [1].

O peso aproximado deste frame, impresso em ABS, é de 200 g.

4.1.2 Motores, Hélices e ESCs

O conjunto de Motores, hélices e ESCs (controladores eletrônicos de velocidade) devem ser escolhidos de forma a serem compatíveis, conforme descrito anteriormente. Este conjunto é responsável pelo desempenho em relação a elevação de carga do sistema.

Neste projeto foram escolhidos motores do tipo “*Brushless*” (sem escovas), de modelo Emax-Cf2822 (Figura 3). Este motor opera tipicamente com tensões de 11.1 V, apresentando uma resposta de 1200 rpm/V, ou 1200 kV, e pesa aproximadamente 39 g.

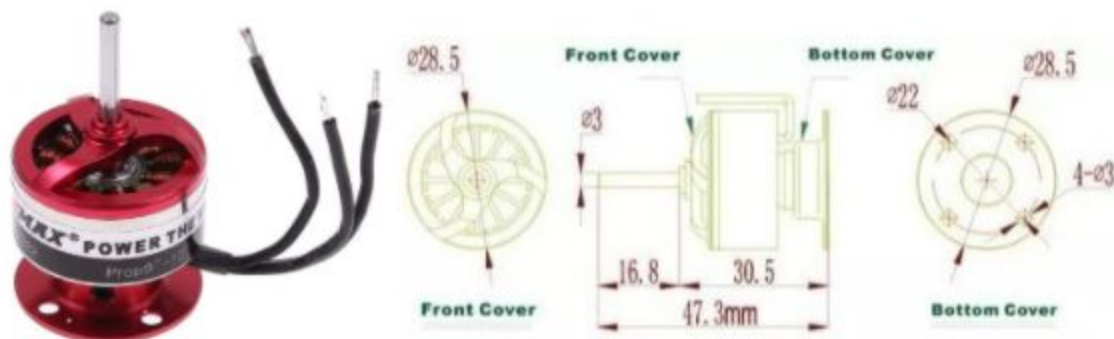


Figura 3 - Motor Emax-Cf2822 e duas dimensões.

Com o motor escolhido, pode-se utilizar seu datasheet (figura 4) para checar seu desempenho em função das hélices.

Model	Cell Count	RPM/V	Prop	RPM	Current	Thrust
CF2822	3S	1200	10X5	7100	14.5A	710g/1.5lb.
			10X4.7(SLOW)	6100	14.5A	745g/1.64

Figura 4 - Recorte do datasheet do motor.

Conforme mostra a figura 4, com hélices de tamanho 10x4.7" (figura 5), o conjunto motor + hélice é capaz de propulsionar, idealmente, 745 g. Como o projeto propõe a construção de um quadcóptero, com um conjunto de 4 motores + hélices, o Drone seria capaz de elevar aproximadamente 2.98 kg.

Também é necessário que o conjunto de hélices contenha um par desenhado para operar em rotação-horária (CW), e um par para rotações anti-horárias (CCW). O peso das hélices escolhidas é de 12 g/hélice.



Figura 5 - Conjunto de Hélices 10x4.7".

Por fim, para realizar o controle de velocidade dos motores é necessário um circuito capaz de modular a tensão de entrada em cada motor com base na entrada dada pelo microcontrolador. Para isso, existem *ESCs* (*Electronic Speed Controllers* - *Figura 6*), que são circuitos pré-fabricados que permitem essas duas funcionalidades de forma fácil. Assim, foram empregados quatro *ESCs*, um para cada motor, e seu peso é de 25 g/*ESC*.

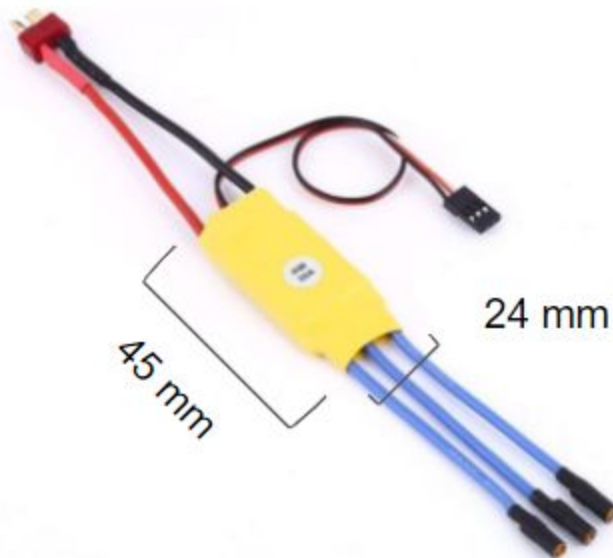


Figura 6 - ESC 30 A Hobbywing.

4.1.3 IMU

Outro componente importante para o funcionamento deste sistema é a Unidade de Medição de Inércia, ou *IMU*. Este componente é o sensor principal, responsável por medir as velocidades angulares e acelerações do Drone, que possibilitam o cálculo de sua posição angular. O sensor escolhido para este projeto foi o MPU-6050 (Figura 7), que é um giroscópio junto a um acelerômetro nos eixos X, Y e Z.

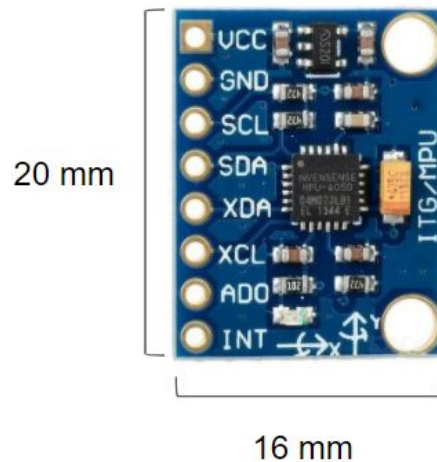


Figura 7 - Sensor MPU-6050.

Este sensor é lido através do protocolo I2C, e permite leituras em velocidades de 400 kHz. Os sensores, por sua vez, são atualizados a 8 kHz e 1 kHz, giroscópio e acelerômetro, respectivamente. Seu peso é de 2.1 g.

4.1.4 Microcontrolador

Finalmente, o componente que integra e comanda todos os outros é o microcontrolador. Para este projeto, foi escolhido o ESP-32, representado na figura 8. A escolha foi baseada levando em consideração o requisito comunicação WiFi (seção 3.1.1), uma vez que o ESP-32 possui uma antena WiFi integrada, o que facilita sua implementação. Além disso, este microcontrolador conta com uma velocidade de processamento relativamente alta (80 a 240 MHz), quando comparado às alternativas mais populares (ex. Arduino - Atmega).

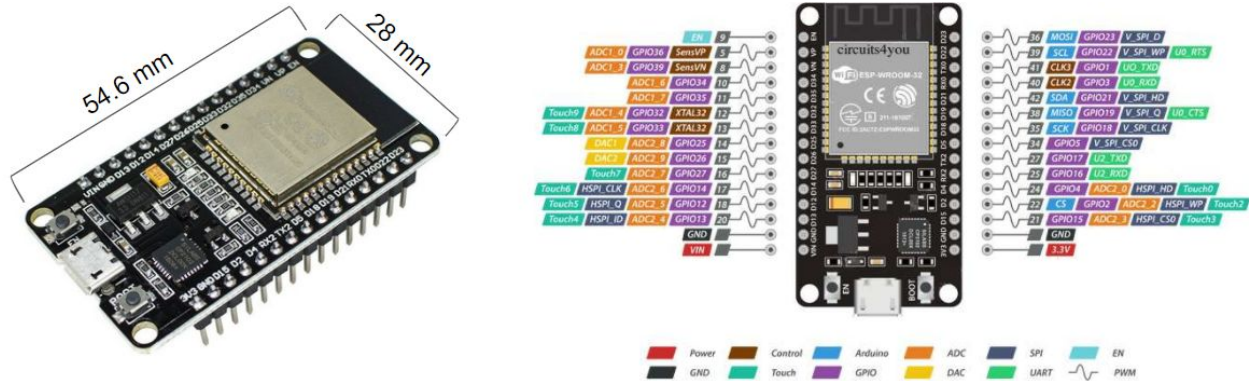


Figura 8 - ESP-32 Devkit V1 e suas dimensões (esquerda). Pinagem do ESP-32 (direita).

Como mostra a figura acima, o ESP-32 conta com 32 GPIOs, 2 interfaces I2C, até 18 canais de ADC, etc., em uma placa de desenvolvimento (Devkit V1) com aproximadamente 9 g. Tudo isso permite uma maior flexibilidade no decorrer do projeto.

4.2 Diagrama de Blocos

Abaixo, na figura 9, encontra-se o planejamento das interfaces de hardware do sistema. Este é o projeto inicial de Hardware que mostra como será a integração e o relacionamento entre os principais componentes do sistema.

Apesar da figura mostrar um “computador pessoal” como parte do diagrama, ele não compõe o sistema embarcado, mas é indispensável para seu funcionamento proposto.

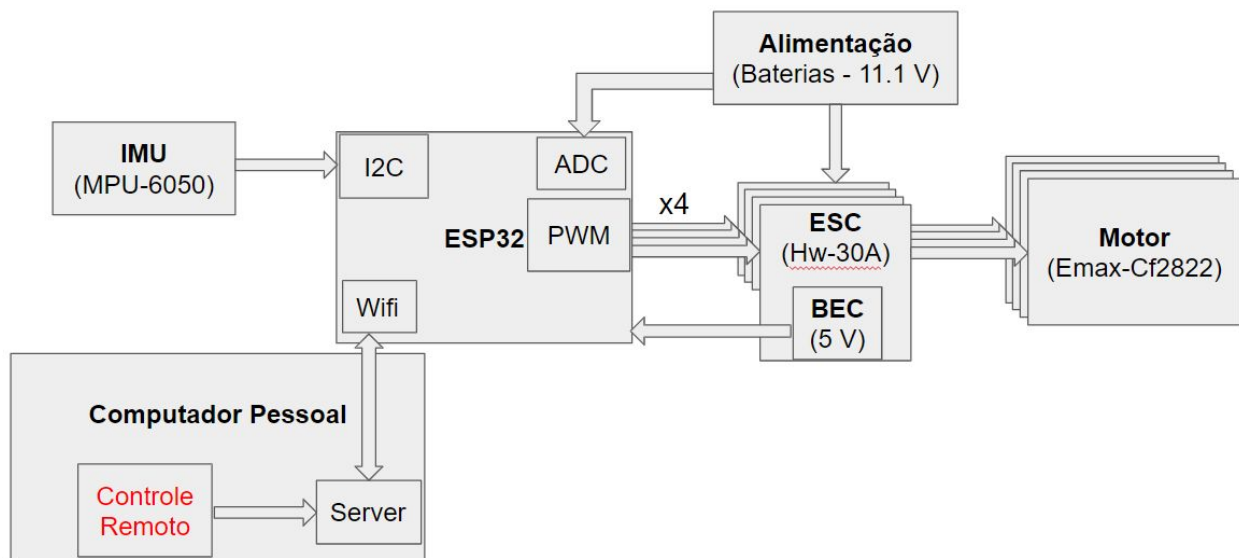


Figura 9 - Diagrama de Blocos do sistema. Destaque para o requisito adicional (vermelho).

4.3 Esquema Elétrico

A partir do diagrama de blocos e das necessidades dos componentes levantadas no Datasheet, o esquema elétrico para contemplar as interligações foi feita utilizando o software Proteus, a visualização das conexões permitiu prever a necessidade da confecção de uma placa de circuito impresso para realizar a distribuição da alimentação 12V do sistema. O esquema elétrico está representado na Figura 10. A interface de alimentação foi produzida utilizando uma placa de fenolite e percloroeto de ferro, de maneira manual, com o layout o mais próximo possível do exposto na Figura 11(fora de escala), na próxima página.

No mesmo diretório de acesso em que este documento foi originalmente disponibilizado, estará uma pasta chamada “Arquivos de Mídia”, onde haverão diversas imagens do processo de construção e montagem do drone.

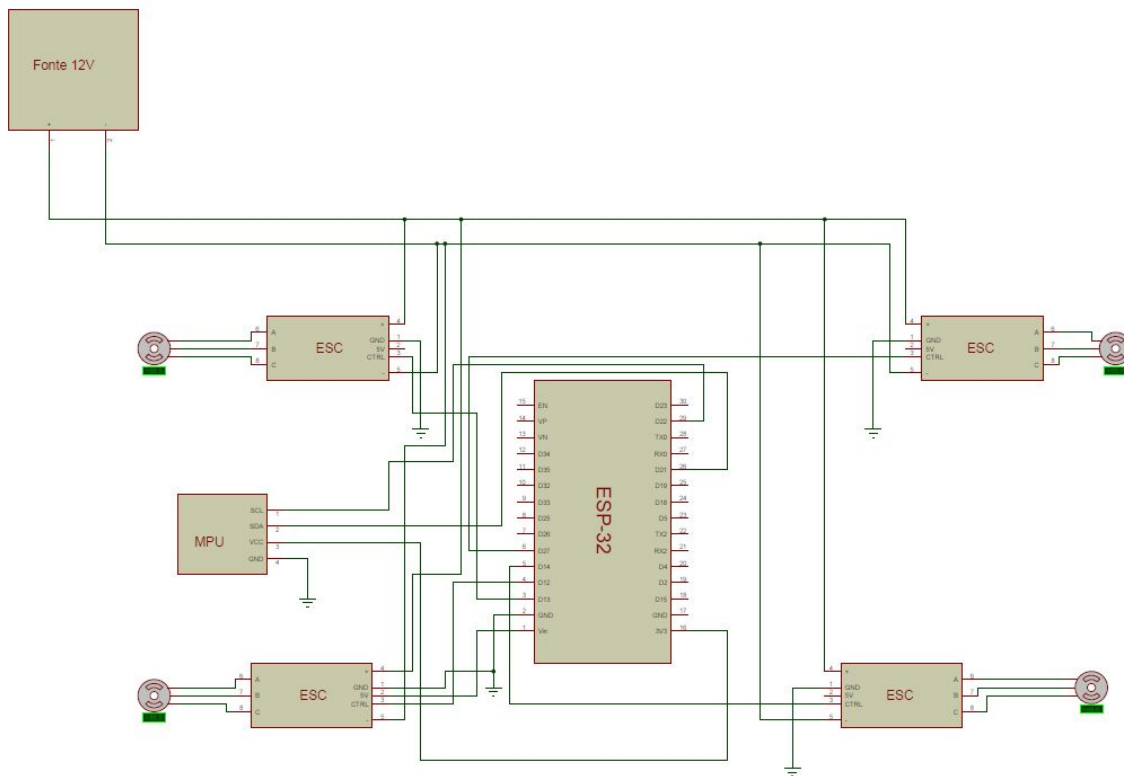


Figura 10 - Esquema de Ligações dos componentes no microcontrolador.

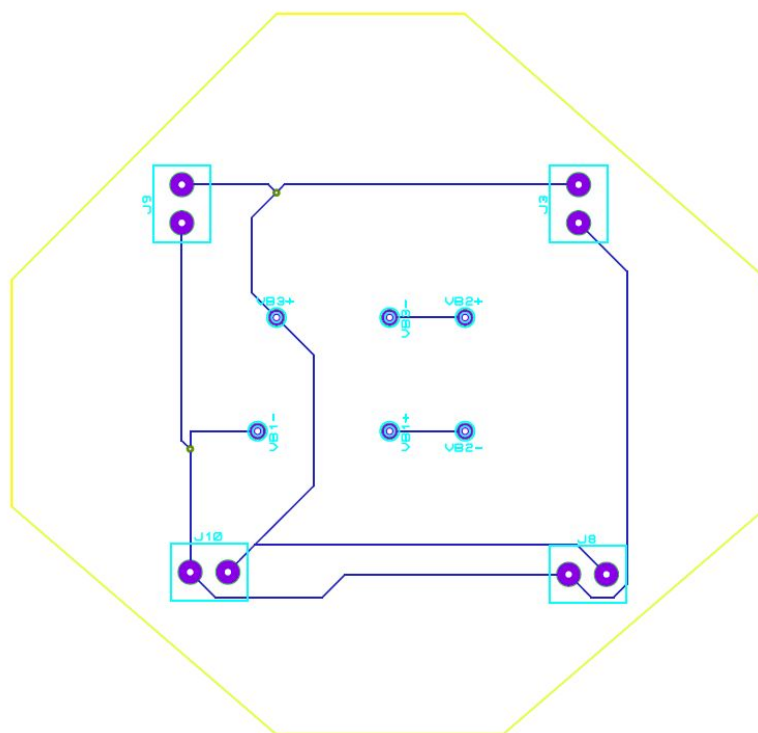


Figura 11 - Esquema da placa de circuito impresso para interface de alimentação.

4.4 Orçamento

O custo efetivo de fabricação do drone está descrito na Tabela I abaixo, o orçamento ficou dentro do previsto. Majoritariamente, com exceção dos itens diversos para montagem, a aquisição dos componentes foi feita através de compras online pela plataforma Mercado Livre.

Componente	Preço (un)	Quantidade	SubTotal
Motor Brushless CF2822	R\$ 49,99	4	R\$ 199,96
ESP32	R\$ 54,99	1	R\$ 54,99
2 Pares de Hélices (+) (-)	R\$ 50,00	1	R\$ 50,00
Frame Drone	R\$ 100,00	1	R\$ 100,00
ESC 30A	R\$ 49,90	4	R\$ 199,60
Bateria + Carregador	R\$ 154,90	1	R\$ 154,90
IMU	R\$ 19,90	1	R\$ 19,90
Itens Diversos para Montagem	257,52	1	257,52
Total		R\$ 1.036,87	

Tabela 1 - Orçamento de montagem.

5. Projeto de Software do Sistema

A seguir, será descrita a documentação do sistema projetado. Todas as imagens referentes aos diagramas do sistema (modelagem e UML) estão disponíveis no repositório do GitHub em melhor resolução.

5.1 Diagrama de camadas

A fim de compreender melhor como é dada a interação entre as diferentes camadas que vão do hardware do microcontrolador até os processos finais de aplicação do projeto, foi confeccionado um diagrama de camadas, conforme figura abaixo.

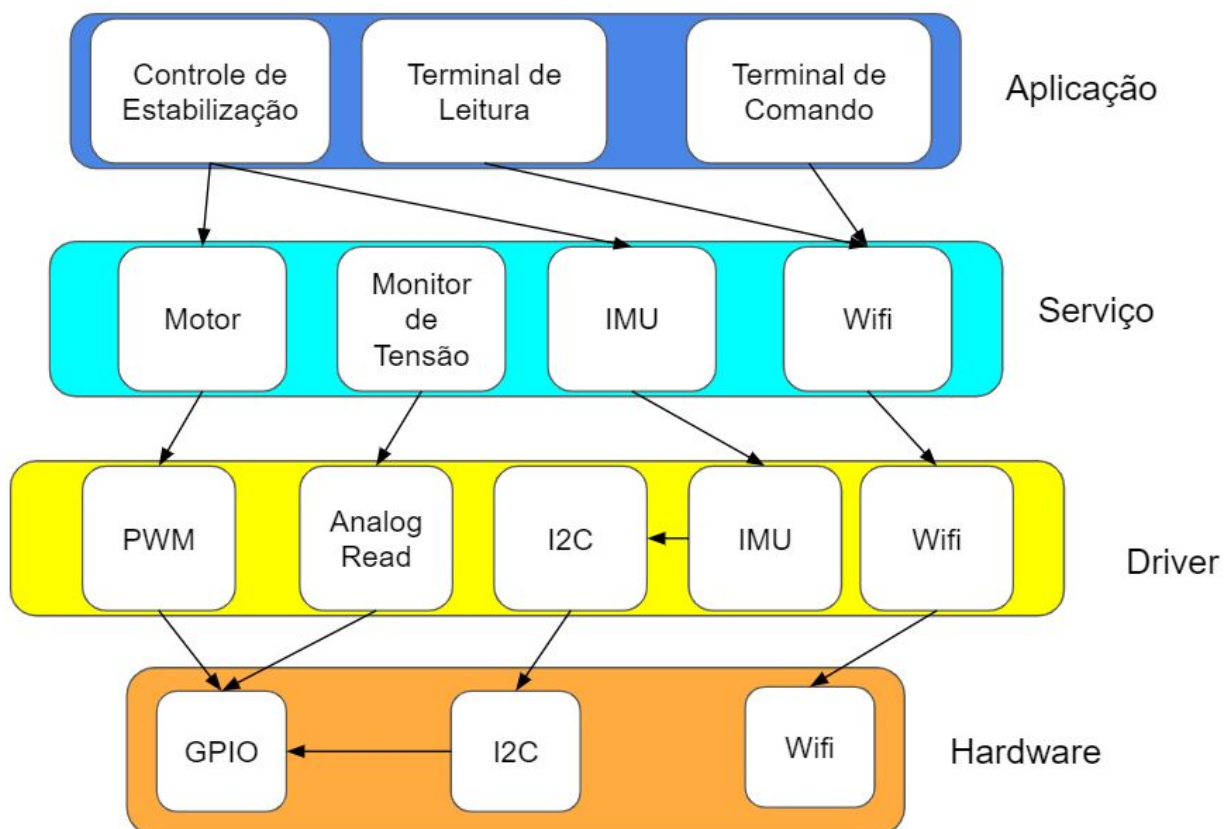


Figura 12 - Diagrama de Camadas.

A aplicação responsável pela estabilização da velocidade do drone, faz uso dos serviços de IMU para capturar os valores de inércia necessários para monitoramento e controle, além disso também utiliza os motores para balancear os movimentos necessários que garantirão a estabilização. O Terminal de Leitura é uma aplicação em que o usuário (técnico ou final) para monitorar os parâmetros do drone, maiores detalhes serão abordados na seção 6 -

Manual de Utilização, esta aplicação utiliza o serviço wifi para permitir a sua execução, de maneira análoga a aplicação de Terminal de Comando utiliza dos mesmos princípios, exceto que possibilita ao usuário atuar diretamente nos parâmetros do drone.

O serviço de motor utiliza os sinais controlados por pwm através do Driver pwm disponível no controlador, para dosar a potência adequada ao motor. O serviço de IMU também faz uso de boa parte dos drivers já nativos do microcontrolador utilizado e também foi implementado dentro deste objeto os tratamentos de sinal adequado para permitir o uso otimizado. O ESP-32 possui um módulo integrado de wifi, e este serviço faz acesso direto ao seu driver, com boa parte das funções já disponíveis como biblioteca de programação, foi adicionado a este driver também o tratamento adequado da comunicação com o servidor Windows.

Tanto os drivers PWM quanto os drivers de leitura Analog Read utilizam diretamente as portas de hardware de GPIO do microcontrolador. O driver da IMU utiliza o driver de comunicação I2C do microcontrolador, que por sua vez está diretamente atrelado também às portas GPIOs, por fim o driver Wifi não possui nenhum outro mediador para o hardware que produz o sinal de comunicação wifi.

É importante destacar que embora previsto inicialmente, o serviço de monitoramento de tensão não foi implementado, maiores detalhes serão abordados na seção 8 - Problemas Identificados.

5.2 Fluxograma

O fluxograma compreende o fluxo de operação do sistema. Nele, a representação da lógica, juntamente com sua interpretação, torna-se evidente. Como a programação em microcontroladores normalmente compreende uma rotina de inicialização (chamada de “*setup*”), e uma que será executada continuamente após a inicialização (chamada de “*loop*”), os fluxogramas foram separados para uma representação mais clara, como mostram as figuras 13 e 14.

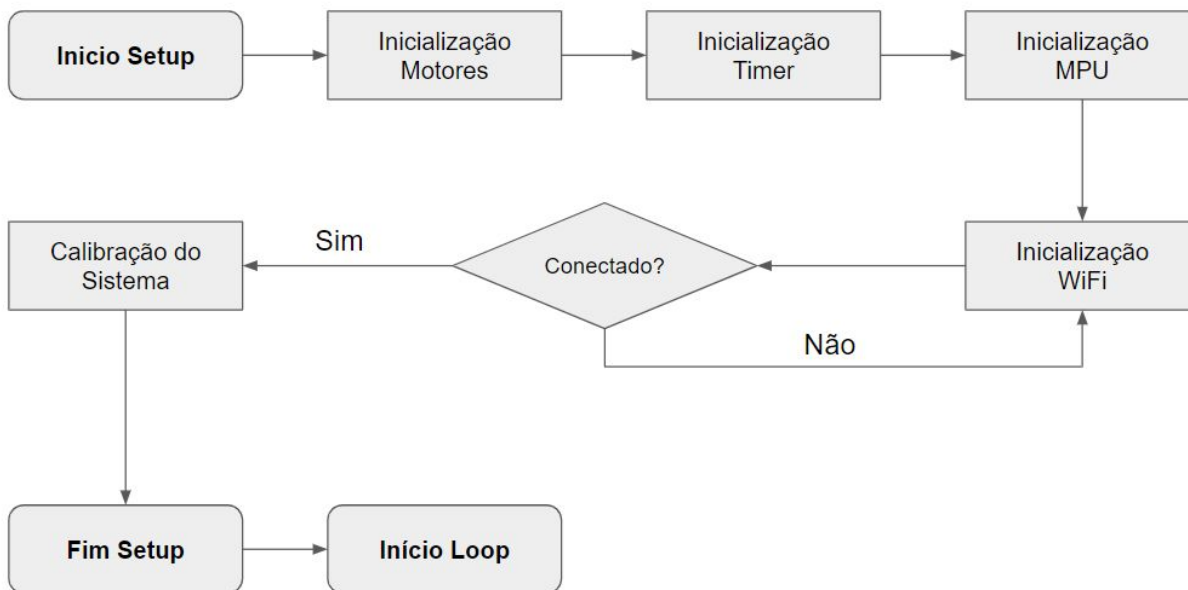


Figura 13 - Fluxograma da rotina de *Setup*.

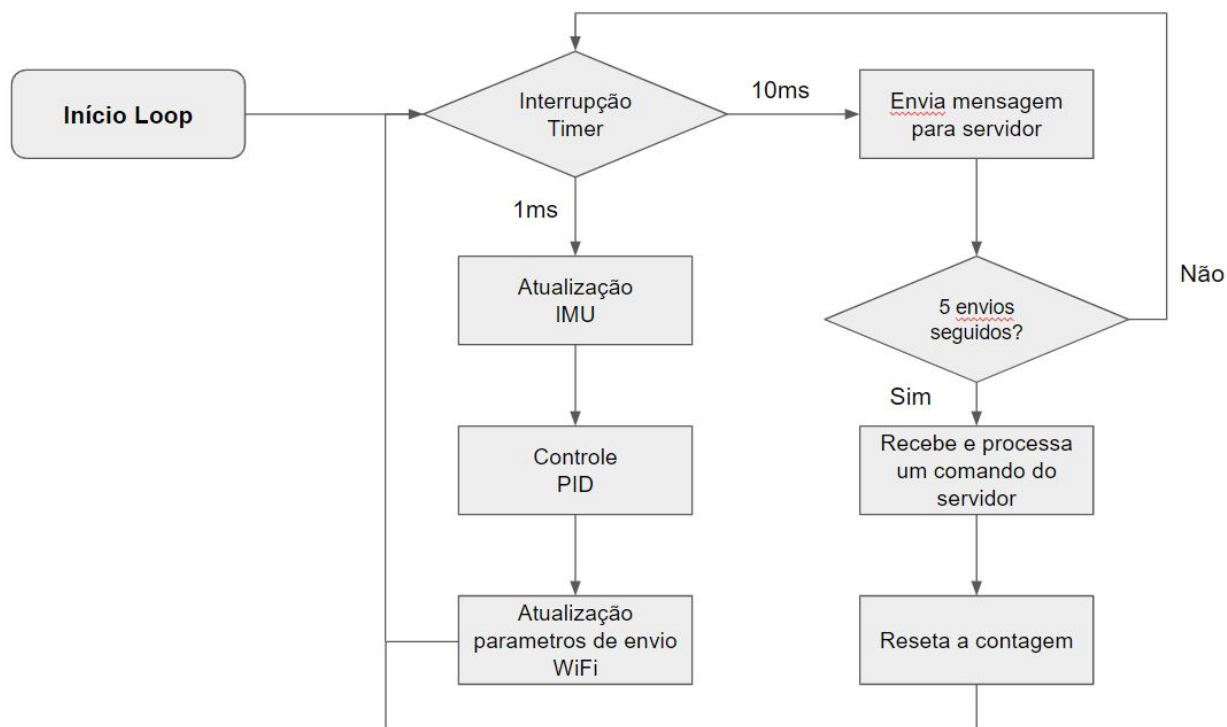


Figura 14 - Fluxograma da rotina de *Loop*.

5.3 Diagrama de Classes

Abaixo se encontra o diagrama de classes do sistema (figura 15). Este diagrama mostra como as principais implementações em Software se associam, e quais funções e variáveis cada uma delas contém. Uma imagem em melhor resolução pode ser encontrada no repositório do GitHub deste projeto.

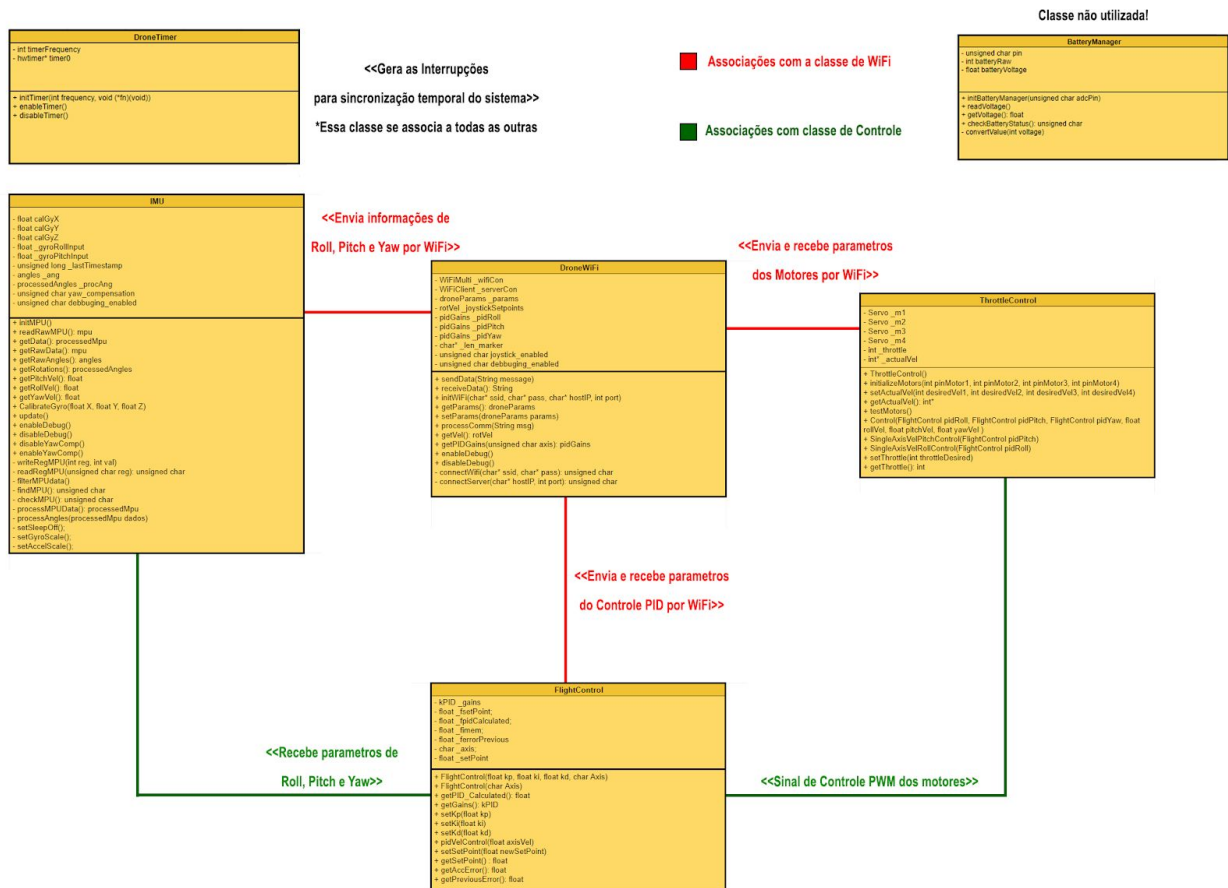


Figura 15 - Diagrama de Classes do Sistema

Em resumo, a classe **DroneWiFi**, cuida de toda comunicação com a rede e Servidor, atualizando as demais classes de acordo com os comandos do usuário (Seção 6), e enviando os parâmetros de interesse para o computador. A classe **IMU** é responsável por calcular os parâmetros de Roll, Pitch e Yaw. A classe **ThrottleControl** atua sobre os motores, para alteração de potência. E a classe **FlightControl** implementa o controlador PID (Seção 7) de velocidade. Por fim, a classe **DroneTimer** gera as interrupções para sincronização temporal do sistema.

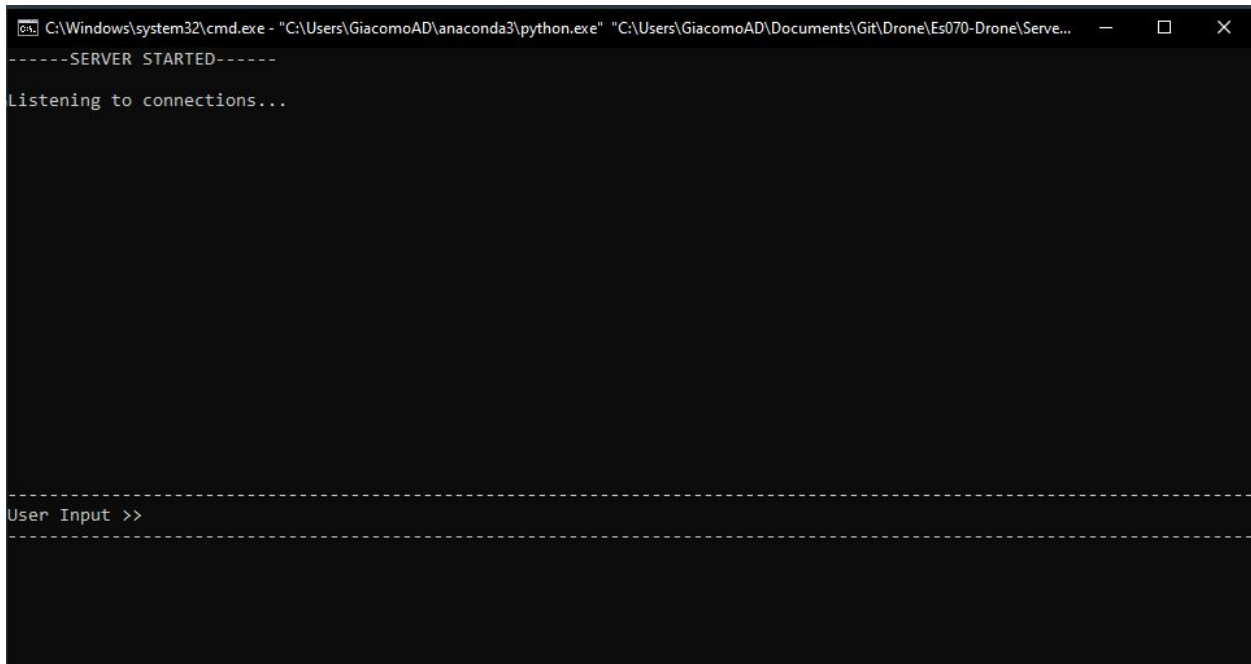
*A classe **BatteryManager**, apesar de implementada, não foi utilizada no projeto. Veja a Seção 8 para mais informações.

6. Manual de Utilização

Nesta seção se encontra o manual de utilização do sistema, incluindo os comandos de operação para usuários técnicos e final.

6.1 Inicialização do Sistema

Primeiramente, o Drone deve ser configurado (via código) para se conectar a rede WiFi na qual o servidor estará disponível localmente. Para isso é necessário conhecer o SSID (nome) da rede e sua senha. Em seguida, com o computador conectado a esta rede, deve-se iniciar o servidor, executando o código em Python 3. A seguinte tela (figura 16) deve ser exibida:



```
C:\Windows\system32\cmd.exe - "C:\Users\GiacomoAD\anaconda3\python.exe" "C:\Users\GiacomoAD\Documents\Git\Drone\E5070-Drone\Serve...
-----SERVER STARTED-----
Listening to connections...

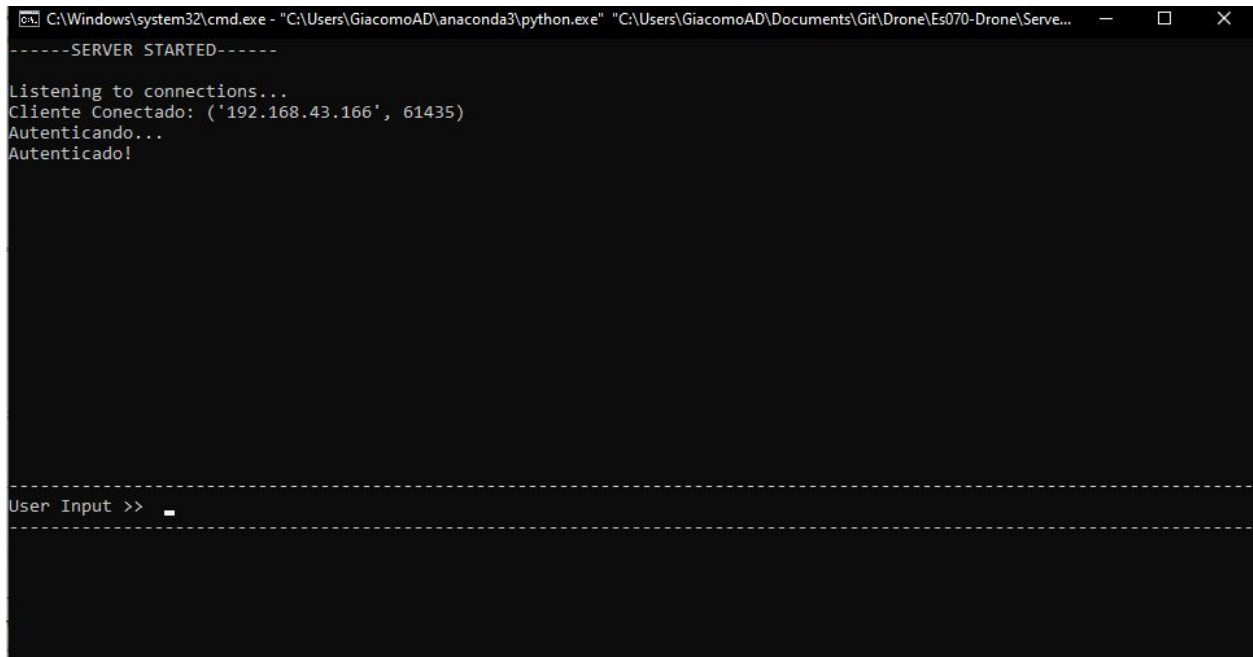
-----
User Input >>
-----
```

Figura 16 - Inicialização do Servidor

Após isso, o Drone pode ser ligado.

Ao ligar, o sistema se conectará a rede WiFi pré-programada e, em seguida, ao servidor que está sendo executado no computador. Com isso, o servidor deverá exibir uma mensagem indicando a conexão (figura 17) e, para inicialização do sistema, o usuário pode inserir o comando de inicialização:

“Comando”	 	Operação
“#G”	 	Comando de Inicialização (“Go”)



```
C:\Windows\system32\cmd.exe - "C:\Users\GiacomoAD\anaconda3\python.exe" "C:\Users\GiacomoAD\Documents\Git\Drone\Es070-Drone\Serve...
-----SERVER STARTED-----
Listening to connections...
Cliente Conectado: ('192.168.43.166', 61435)
Autenticando...
Autenticado!

-----
User Input >> _
-----
```

Figura 17 - Indicação de conexão do Drone ao Servidor.

Após conectado, o Drone irá realizar sua rotina de calibração. Enquanto isso, é de suma importância que o Drone permaneça **completamente estático** sobre uma superfície nivelada, garantindo uma boa calibração.

A seção destinada às entradas do usuário ("*User Input*") será, então, habilitada. Após a inserção do comando de inicialização ("*#G*"), o sistema entra em modo de testes e a tela do servidor exibida será como mostra a figura 18.

Além disso, o servidor sempre exibe o comando que está sendo enviado, a mensagem de resposta do Sistema com alguns parâmetros para monitoramento, e também o último comando inserido pelo usuário.

6.2 Modo de Testes

```
C:\Windows\system32\cmd.exe - "C:\Users\GiacomoAD\anaconda3\python.exe" "C:\Users\GiacomoAD\Documents\Git\Drone\Es070-Drone\Serve...
-----SERVER STARTED-----
Listening to connections...
Cliente Conectado: ('192.168.43.166', 61435)
Autenticando...
Autenticado!

ENVIANDO COMANDO ---> #K
MENSAGEM RECEBIDA ---> Roll:-75.52    Pitch:-3.70
                        Velocidade GyX:-0.02    GyY:-0.14    GyX2:-0.03    GyY:-0.02
ULTIMO COMANDO INSERIDO --->#G

-----
User Input >> _
-----
JOYSTICK DESABILITADO
'#J' = HABILITAR JOYSTICK
'#SGaxis;kp;ki;kd' = SETAR GANHO CONTROLADOR POR EIXO
'#STxxxx;yyyy;zzzz;kkkk' = SETAR THROTTLE
'#R' = RESETAR MOTORES PARA VELOCIDADE BASE
```

Figura 18 - Servidor exibindo o Modo de Testes.

O modo de testes é sempre o modo em que o sistema inicializa. Nele, o usuário pode inserir comandos mais complexos para controlar diferentes parâmetros do Drone. Neste modo, o servidor aguarda o usuário pressionar a tecla “**Enter**” para enviar o comando. A lista de comandos que podem ser usados neste modo é exibida abaixo do campo onde o usuário pode digitar. Além disso, a mesma lista é dada a seguir:

“Comando”	Operação
“#J”	Habilita o modo de Joystick;
“#SGaxis;kp;ki;kd”	Troca os Ganhos kp, ki e kd do controlador de velocidade no eixo axis ;
“#STxxxx;yyyy;zzzz;kkkk”	Troca a velocidade (Throttle) dos quatro motores individualmente. As velocidades devem ser valores entre 1000 e 2000;
“#R”	Reseta as velocidades dos motores.

6.3 Modo Joystick

Quando habilitado, o servidor troca a exibição dos comandos disponíveis para aqueles que permitem o controle por Joystick. Neste modo de operação, **o usuário não pode mais inserir comandos complexos, e a entrada é lida assim que uma tecla é pressionada**. Isto é feito para garantir uma boa velocidade entre a entrada do usuário e o envio do comando.

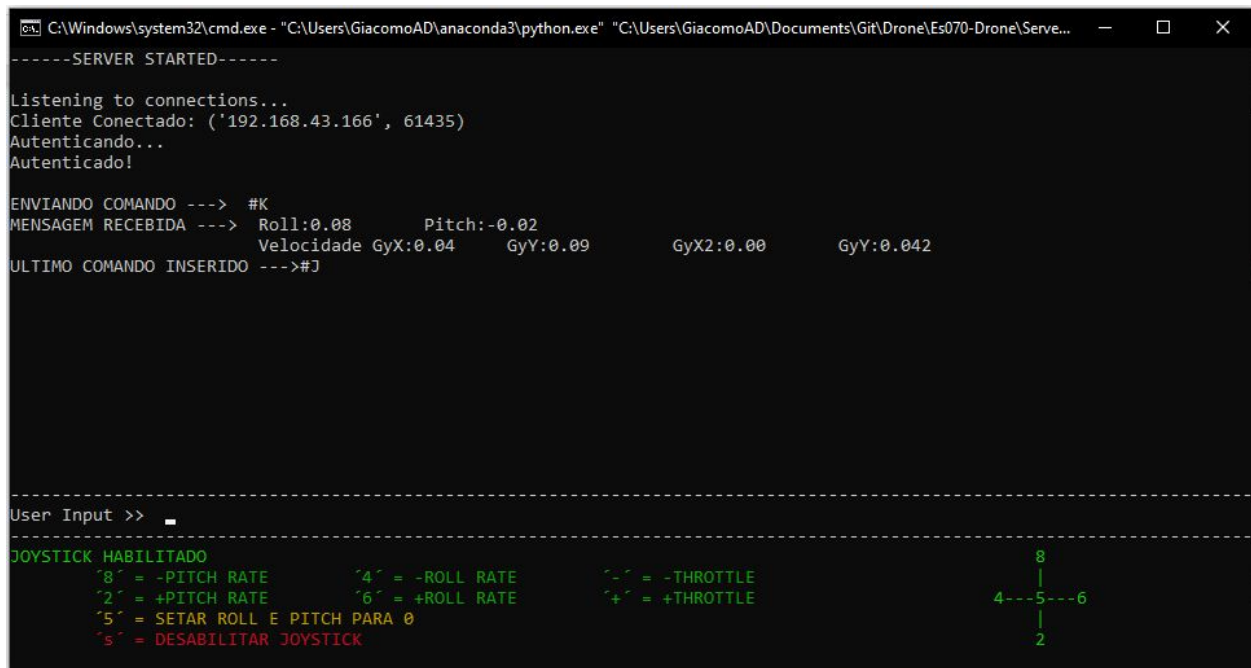


Figura 19 - Servidor exibindo o Modo Joystick.

"Comando"	Operação
"8"	Diminui o Setpoint de Velocidade de Pitch;
"2"	Aumenta o Setpoint de Velocidade de Pitch;
"4"	Diminui o Setpoint de Velocidade de Roll;
"6"	Aumenta o Setpoint de Velocidade de Pitch;
"5"	Reseta o Setpoint de Velocidade de Pitch e Roll para 0;
"+"	Eleva a potência dos motores igualmente;
"_"	Diminui a potência dos motores igualmente;
"s"	Retorna para o Modo de Testes;

7. Controlador PID

A rotina responsável por tratar o sinal de controle adequado, conforme demonstrado no fluxograma, é realizada a cada 1ms. Nesta mesma frequência é adquirido os sinais da IMU que serão utilizados para os cálculos. Os valores de setpoint serão sempre os inseridos por último pelo usuário, e caso não houver alteração permanecerão os mesmos. Abaixo estão as principais linhas de comando utilizadas e as variáveis que permitem o controle.

```
error_temp = fgyro - _setPoint;
```

```
_fimem += _gains.fki * error_temp;
```

```
_fpidCalculated = _gains.fkp*error_temp + _fimem + _gains.fkd*(error_temp - _errorPrevious);
```

```
_errorPrevious = error_temp;
```

O erro atual (temporário até a próxima medição) é calculado diretamente pela medição da velocidade atual realizada através da IMU (de acordo com eixo) subtraída da velocidade desejada, o setpoint. O erro acumulado é somado a cada interação, multiplicando o erro atual pela constante de ganho integrativo, essa variável está limitada para saturar em uma margem segura que poderá ser consultada nos códigos disponibilizados. O sinal de saída do controlador então será o ganho proporcional multiplicado pelo erro atual, somado a isso o erro acumulado e mais a constante derivativa multiplicada pela diferença entre o erro atual e o erro anterior.

Ao final do cálculo do sinal de controle, o erro anterior é atualizado com o erro atual para permitir a replicabilidade do código na próxima interação do controlador.

Os sinais de controle são então enviados para o gerenciador de potência dos motores, para permitir o controle dos movimentos de Pitch, Roll e Yaw adequados, conforme abaixo:

```
desiredVel1 = _throttle - pidPitch.getPID_Calculated() - pidRoll.getPID_Calculated() - pidYaw.getPID_Calculated();
```

```
desiredVel2 = _throttle - pidPitch.getPID_Calculated() + pidRoll.getPID_Calculated() + pidYaw.getPID_Calculated() ;
```

```
desiredVel3 = (_throttle +30) + pidPitch.getPID_Calculated() - pidRoll.getPID_Calculated() + pidYaw.getPID_Calculated();
```

```
desiredVel4 = _throttle + pidPitch.getPID_Calculated() + pidRoll.getPID_Calculated() - pidYaw.getPID_Calculated();
```

Apesar de calibrar todos os ESCs, um dos motores estava apresentando desempenho abaixo do esperado, fez-se necessário aumentar sua potência base (_throttle) para compensar esse baixo desempenho. A diferença entre os sinais de PID calculado para cada eixos, é baseado na orientação de movimentação definida e na ordem dos motores. A figura abaixo exemplifica esse padrão, os números na figura representam a identificação dos motores.

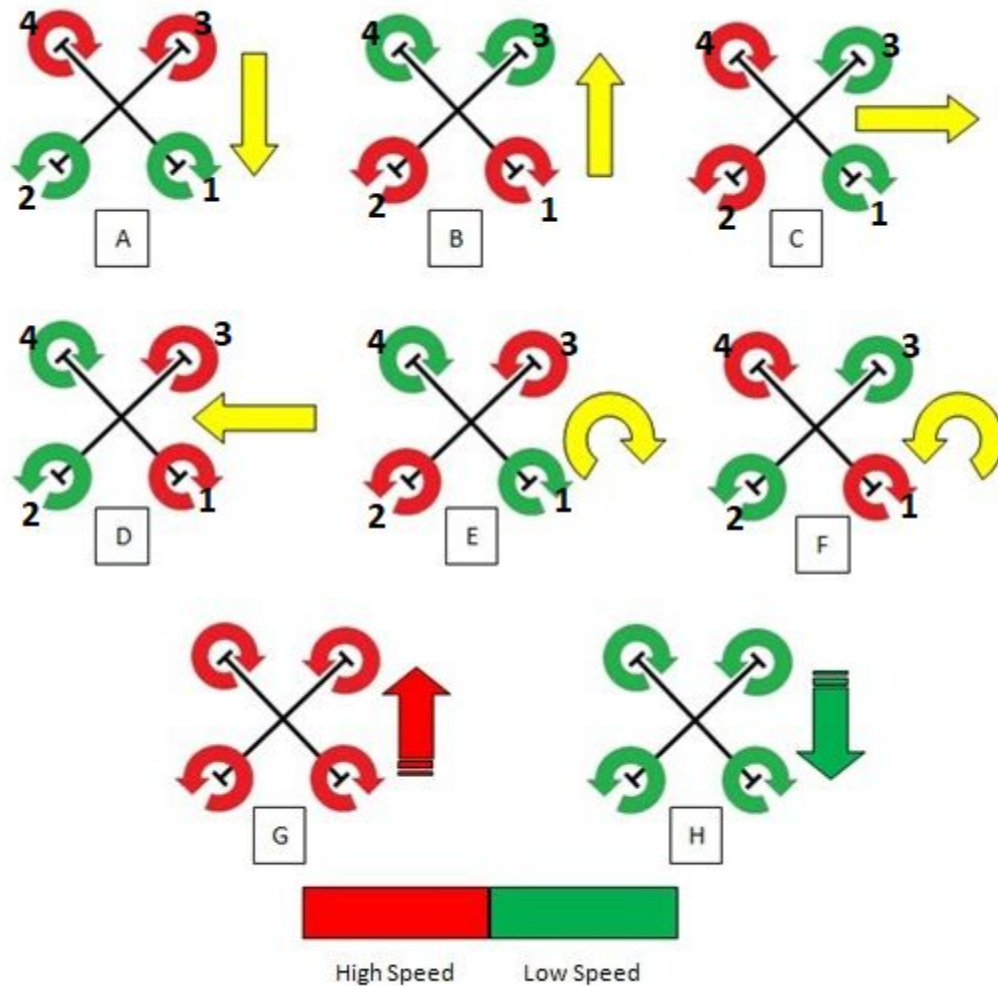


Figura 20 - Identificação dos motores e orientação de Pitch, Roll e Yaw

De forma que:

- A - Pitch negativo
- B - Pitch positivo
- C - Roll negativo
- D - Roll positivo
- E - Yaw negativo
- F - Yaw Positivo
- G - Subida
- H - Descida

Os movimentos de Pitch, Roll e Yaw são termos técnicos usados para a aviação e movimentação de drone, a figura abaixo representa estes eixos baseados na orientação da IMU com os eixos cartesianos.

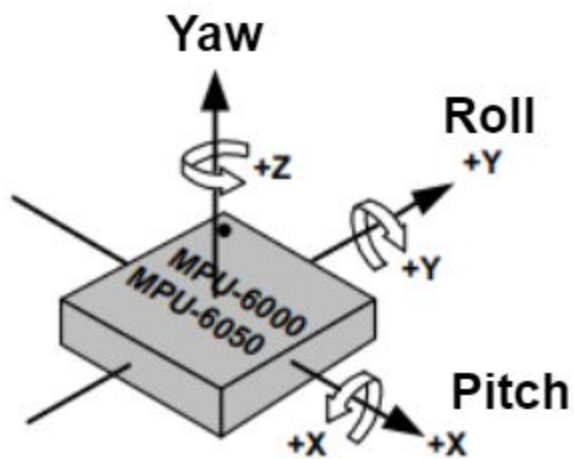


Figura 21 - Eixos de referência MPU-6050; Padrão *Roll*, *Pitch* e *Yaw*.

8. Problemas Identificados

Nesta seção, serão discutidas as dificuldades encontradas pelos alunos ao longo do projeto, separando-os entre problemas resolvidos e não resolvidos.

8.1 Problemas identificados pelos alunos

8.1.1 Resolvidos

1. As 3 baterias adquiridas de LIPO de 3.3V cada, demonstraram muito baixo desempenho, a ponto de inviabilizar os testes devido à rápida descarga de suas células. Fez-se necessário adaptar a interface de alimentação do drone para funcionar com uma fonte regulável em 12V, utilizando para tanto um cabo de alimentação.

8.1.2 Não Resolvidos

1. Cabo de alimentação: A fonte regulável se mostrou um empecilho para permitir os testes de voo e calibração de ganhos de forma adequada, como era financeiramente inviável custear um conjunto de baterias com o desempenho adequado, este problema não foi possível ser contornado.
2. Bancada de testes adequada: No intuito de não usar o próprio drone para os testes, foi um problema que não foi possível de ser contornado. Embora a bancada fabricada possibilitasse alguns testes, muitas não idealizadas eram induzidas no sistema de forma a dificultar a calibração dos ganhos e a visualização efetiva do controle de velocidade. A falta de tempo hábil para solicitar novos componentes e recursos de ferramentas adequados para a confecção da bancada de testes independente impossibilitou resolver esse problema.
3. Calibração Adequada dos Ganhos: Para realizar a calibração de todos os ganhos dos controladores dos eixos, é necessário realizar testes de voo e de resposta ao controle de setpoints, o joystick implementado no teclado por wifi, embora facilitador em alguns pontos, se mostrou pouco eficiente para pilotar o drone, dificultando a calibração dos ganhos com segurança e coerência idealmente seria necessário ter um controle por rádio frequência mas era um custo fora do orçamento previsto. Além disso, realizar todos os testes sem ter o auxílio de uma segunda pessoa para reposicionar o drone, desembolar o fio de alimentação, transformava a tarefa ainda mais difícil de ser atingida.
4. Danificação do Drone: Relacionado intrinsecamente com todas as outras dificuldades não resolvidas, o drone foi danificado durante um dos testes de voos de forma que não seria possível colar ou remendar. O prazo de entrega impossibilitava a fabricação de um frame novo e a remontagem completa do sistema.

9. Autoria e Reconhecimento

Como base para as funções da biblioteca “IMU”, o código de leitura e escrita através do I2C foi baseado na referência [1]. Para entendimento das necessidades de montagem e funcionamento em um escopo macro, foram utilizadas as referências [2] e [3]. Para a confecção da lógica para calibração e funcionamento do controlador PID foram consultadas majoritariamente as referências [4] e [5]. As referências [6] e [7] foram de grande ajuda para compreensão do funcionamento dos ESCs. Além disso, de forma geral, a documentação de utilização das bibliotecas de funções da Arduino IDE foram de grande valia [8].

Houveram, também, consultas para revisar sintaxe e operação de funções padrão de C. Por exemplo, foram feitas consultas em fóruns online como “Stack Overflow”, “Tutorialspoint” e “Geeksforgeeks”.

[1] - Adilson Thomsen, **Tutorial: Acelerômetro MPU6050 com Arduino**. FilipeFlop. Disponível em: <<https://www.filipeflop.com/blog/tutorial-acelerometro-mpu6050-arduino/>> Primeiro acesso em: 09 de out. 2020. 23:00h

[2] - Jack Brown, **HOW TO MAKE AN ARDUINO QUADCOPTER DRONE:STEP-BY-STEP DIY PROJECT**. Mydronelab. Disponível em: <<https://www.mydronelab.com/blog/arduino-quadcopter.html>>. Primeiro acesso em: 09 de out. 2020. 23:00h

[3] - Code & Supply, Rayan Boland, **Embedded Programming for Quadcopters**. Youtube. Disponível em: <<https://youtu.be/CHSYgLfhwUo>>. Primeiro acesso em: 09 de out. 2020. 23:00h

[4] - Electronoobs, **PID brushless motor control tutorial**. Youtube. Disponível em: <<https://youtu.be/AN3yxIBAxTA>>. Primeiro acesso em: 09 de out. 2020. 23:00h

[5] - Joop Brokking, **YMFC-3D part 5 – Quadcopter PID controller and PID tuning**. Youtube. Disponível em: <<https://youtu.be/JBvnB0279-Q>>. Primeiro acesso em: 09 de out. 2020. 23:00h

[7] - How To Mechatronics, **How Brushless Motor and ESC Work and How To Control them using Arduino**. Youtube. Disponível em: <<https://youtu.be/uOQk8SJso6Q>> . Primeiro acesso em: 09 de out. 2020. 23:00h

[6] - Painless360, **RC Basics - Understanding Electronic Speed Controllers (ESC)**. Youtube. Disponível em: <<https://youtu.be/OZNxbxL7cdc>>. Primeiro acesso em: 09 de out. 2020. 23:00h

[8] - Arduino, **Examples from Libraries**, Disponível em <<https://www.arduino.cc/en/Tutorial/LibraryExamples>> Primeiro acesso em: 09 de out. 2020. 23:00h