UM PROTÓTIPO E ESTUDO SOBRE O DESENVOLVIMENTO DE UM QUADRICÓPTERO

João Paulo Serodio Gonçalves, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

joao_paulosg@outlook.com, dalton@furb.br

Resumo: *Este artigo apresenta o desenvolvimento de um protótipo de quadricóptero elétrico utilizando como base uma plataforma baseada em Arduino. O quadricóptero desenvolvido recebe comandos de um aplicativo iOS, também desenvolvido neste projeto. A comunicação entre o aplicativo e o software que controla o quadricóptero é feita através de uma rede wi-fi utilizando o protocolo UDP e um formato de comunicação próprio. É apresentada a teoria básica de voo de um quadricóptero, seus principais componentes e uma descrição do software utilizado para controlar a aeronave e do aplicativo utilizado para se comunicar com a mesma. O projeto atingiu os objetivos pretendidos, contudo se limitando ao estudo da concepção e escolha do hardware bem como no desenvolvimento dos dois softwares com algumas restrições. O quadricóptero apresentou limitações no quesito de estabilidade e manobrabilidade por não possuir algoritmos específicos para este fim.*

Palavras-chave*: Quadricóptero. Drone. RPAS. Microcontrolador ARM. Arduino. iOS. UDP.*

1 INTRODUÇÃO

Os drones, como são popularmente conhecidos, integram cada vez mais o cotidiano de profissionais, que vão desde a agricultura e logística, chegando à indústria do entretenimento e marketing, além de seu uso para fins de recreação, seja para hobby ou fotografias e filmagens de forma amadora. No meio civil estes equipamentos têm se popularizado cada vez mais desde meados de 2010 (ODRONES, 2015). Este movimento de popularização pode ser observado através da ferramenta Google Trends, onde é possível notar que os termos "drone" e "veículo aéreo não tripulado" começaram a apresentar um ganho significativo nas pesquisas no buscador. Já no meio militar as primeiras pesquisas começaram na década de 1960 (ODRONES, 2015). Atribui-se a invenção do drone como é conhecido hoje ao israelita Abe Karem, engenheiro espacial responsável pelo drone americano mais temido e bem-sucedido, conhecido como Predator (ODRONES, 2015).

Apesar de comumente denominados como drones ou Veículos Aéreos Não Tripulados (VANT), nomenclatura proveniente do termo Unmanned Aerial Vehicle (UAV), segundo o Departamento de Controle do Espaço Aéreo (DECEA), esta denominação é considerada obsoleta na comunidade aeronáutica internacional. O termo adotado tecnicamente pela Organização da Aviação Civil Internacional (OACI) para esse tipo de aeronave é o Remotely Piloted Aircraft System (RPAS), em português Aeronaves Remotamente Pilotadas (BRASIL, 2016).

Olson (2017) classifica os RPASs em três categorias principais: os de asa fixa, os com um único rotor e os de múltiplos rotores. Os de asa fixa normalmente possuem um par de asas rígidas e comportam-se como um avião, sendo assim decolam e pousam na horizontal. Esse tipo de RPAS é muito utilizado para fins militares e mapeamento de grandes áreas, dada a sua capacidade de permanecer no ar por horas e percorrer longas distâncias, dependendo do seu tamanho. Os de rotor único possuem um grande rotor central que funciona como uma grande asa rotativa e um pequeno rotor na cauda para estabilizar e determinar a direção, como em um helicóptero, sendo utilizados normalmente para mapeamento a laser da topografia de terrenos. Os de múltiplos rotores por sua vez são o tipo mais comum de RPAS atualmente. Apesar de serem considerados complexos em seus aspetos técnicos de desenvolvimento e funcionamento, estes equipamentos são simples de serem pilotados, dados os inúmeros sensores e os algoritmos de estabilização de voo empregados. Possuem diferentes configurações, normalmente variando de três até oito rotores. Os de três e quatro rotores geralmente são utilizados para fins de recreação e fotografias aéreas. Já os de seis e oito rotores podem substituir um helicóptero para filmagens aéreas na indústria cinematográfica (OLSON, 2017).

Com uma pesquisa online é possível encontrar inúmeros projetos referentes ao desenvolvimento de drones. D'Andrea (2013) e Massachussetts Institute of Tecnology (MIT) (2011) são alguns exemplos. Com caráter de pesquisa científica, os projetos vão desde quadricópteros relativamente simples, até modelos com aspectos mais complexos de construção (alguns utilizando apenas um rotor) e consequentemente com algoritmos de controle mais elaborados.

O objetivo deste projeto consiste em construir o tipo mais comum de RPAS com múltiplos rotores, um quadricóptero (veículo composto por quatro rotores independentes), cuja plataforma central seja baseada em um microcontrolador compatível com Arduino e que seja controlado através de um aplicativo móvel. Para isso foi realizada uma pesquisa acerca dos componentes necessários para construção do quadricóptero, além da especificação e implementação do software de controle e da montagem do quadricóptero e desenvolvimento de um aplicativo móvel para controle do mesmo. Para tanto tem-se os seguintes objetivos específicos: identificar e selecionar os componentes de

hardware necessários para construção do quadricóptero; controlar o quadricóptero por meio de um software desenvolvido sobre uma plataforma Arduino; comandar o quadricóptero através de um aplicativo móvel iOS.

2 FUNDAMENTAÇÃO TEÓRICA

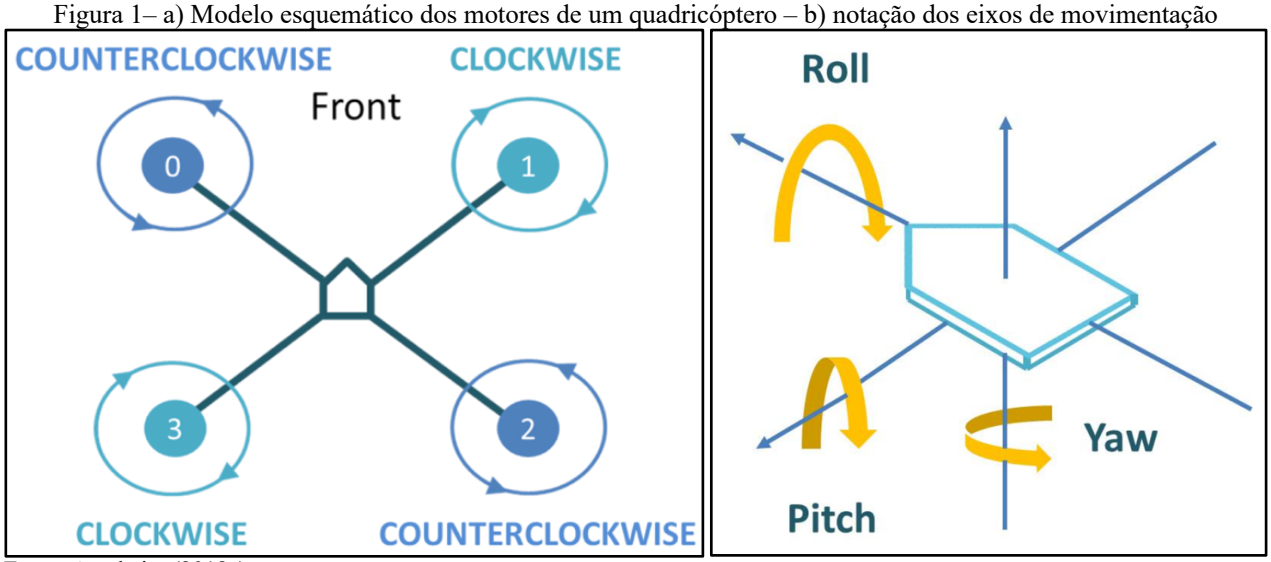
Esta seção apresenta os aspectos da fundamentação teórica utilizados na pesquisa e desenvolvimento deste projeto. A seção 2.1 apresenta os conceitos básicos de voo, estabilidade e manobrabilidade de um quadricóptero. A seção 2.2 apresenta uma descrição dos componentes utilizados na construção do quadricóptero resultante deste projeto. Por fim, na seção 2.3 são apresentados três trabalhos correlacionados ao projeto desenvolvido.

2.1 O VOO DE UM QUADRICÓPTERO

Para que um quadricóptero possa voar é necessária a existência de um diferencial de pressão entre a parte superior e a parte inferior de suas quatro hélices, este diferencial é produzido pela rotação das mesmas. Conforme a velocidade de rotação aumenta, a pressão sobre as hélices diminui, surgindo assim um vetor resultante de força para cima, esta força é denominada empuxo. Conforme o empuxo aumenta, diminui ou iguala-se a força peso, o quadricóptero subirá, descerá ou permanecerá em voo pairado, respectivamente (SILVA, 2011).

Para manter-se o equilíbrio as hélices de um quadricóptero são posicionadas em paralelo, no mesmo plano horizontal e equidistantes do centro de massa (SILVA, 2011). O torque total produzido pelas quatro hélices deve resultar em nulo. Se isso não ocorrer, o equilíbrio e manobrabilidade da aeronave é comprometido. Para se anular o torque um par de hélices deve girar no sentido horário e o outro no sentido anti-horário. A Figura 1a apresenta o modelo esquemático da disposição dos motores de um quadricóptero com a nomenclatura e a direção de rotação dos mesmos. Já Figura 1b apresenta a notação dos eixos no qual uma aeronave pode se movimentar.

Figura 1– a) Modelo esquemático dos motores de um quadricóptero – b) notação dos eixos de movimentação



Fonte: Amebaiot (2018a).

Com o controle independente da velocidade de rotação de cada motor, um quadricóptero pode realizar três manobras básicas: guinada (*yaw,* eixo vertical), arfagem (*pitch*, eixo lateral) e rolagem (*roll*, eixo longitudinal). A execução das manobras ocorre por meio de alterações na velocidade de rotação de seus motores, aumentando a velocidade de um par e diminuindo a de outro. Para que o quadricóptero não tenha sua altitude alterada durante a execução de um comando específico é necessário que o empuxo permaneça constante. Para satisfazer tal condição deve-se aumentar e diminuir a rotação dos pares de motores de maneira proporcional. A seguir são apresentados os conceitos básicos para se efetuar as manobras mencionadas:

a) guinada:
   - sentido horário: aumentar a velocidade dos motores 0 e 2, diminuir a velocidade dos motores 1 e 3,
   - sentido anti-horário: diminuir a velocidade dos motores 0 e 2, aumentar a velocidade dos motores 1 e 3,

b) arfagem:
   - mover para frente: aumentar a velocidade dos motores 2 e 3, diminuir a velocidade dos motores 0 e 1,
   - mover para trás: diminuir a velocidade dos motores 2 e 3, aumentar a velocidade dos motores 0 e 1,

c) rolagem:
   - virar à direita: aumentar a velocidade dos motores 0 e 3, diminuir a velocidade dos motores 1 e 2,
   - virar à esquerda: diminuir a velocidade dos motores 0 e 3, aumentar a velocidade dos motores 1 e 2.

2.2 COMPONENTES DE UM QUADRICÓPTERO

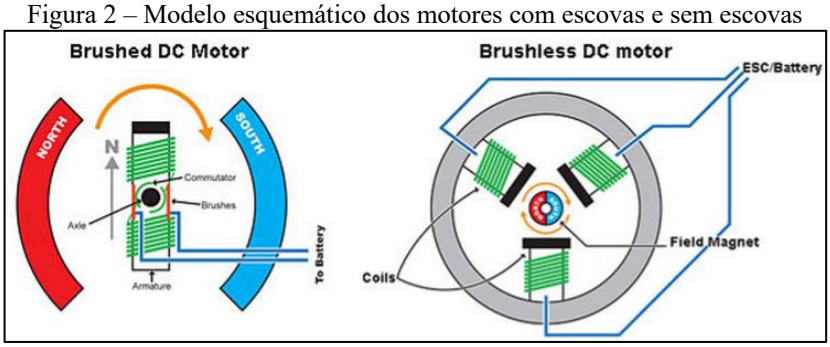
Um quadricóptero elétrico convencional possui seis componentes principais: o chassi, as hélices, os motores, as controladoras eletrônicas de velocidade, a controladora de voo e a bateria (existem outros componentes em um quadricóptero, como o acelerômetro e giroscópio, porém estes não serão abordados neste artigo). A seguir são descritos de forma resumida cada um dos componentes.

O chassi ou *frame* é o componente central de um quadricóptero, ao qual os demais componentes são fixados. Este componente possui uma vasta variedade de tamanhos, pesos e formas para atender diversas necessidades. Entretanto, a grande maioria possui a forma de um X (WANG, 2015). O *frame* pode ser constituído de diferentes materiais, como uma estrutura tubular de fibra de carbono revestida com poliestireno, também conhecida como isopor, e plástico, como é o caso do AR.Drone 2.0 (PARROT, 2018), ou mesmo ligas de titânio e magnésio, como no caso do Phantom 4 Pro (DJI, 2018), que são resistentes e ao mesmo tempo leves.

As hélices afetam diretamente o peso máximo, a velocidade máxima e a velocidade de manobra de um quadricóptero (WANG, 2015). As pás de uma hélice funcionam como as asas de um avião, produzindo a sustentação necessária para este se manter no ar. Por este motivo os quadricópteros são denominados de aeronaves de asa rotativa, como os helicópteros. Conforme uma hélice gira ela também avança como uma rosca de parafuso. A cada volta ela avança uma certa distância, esta é denominada de passo da hélice (HANGAR33, 2015). Pode-se dividir as hélices em duas categorias: as de passo fixo e as de passo variável. As hélices de passo fixo, como o nome sugere, não alteram o passo, fazendo assim com que tenham um único ângulo de ataque (ângulo formado entre a pá da hélice e o vento relativo). Dessa forma, elas possuem desempenho máximo na velocidade e rotação para a qual foram projetadas (HOMA, 2011, pg. 41). Já as de passo variável podem alterar o ângulo de ataque. Assim essas hélices possuem um excelente desempenho em diferentes velocidades e também podem ser utilizadas como freio aerodinâmico, alterando o passo para um ângulo de ataque negativo, também denominado passo reverso, o que além de melhorar a desaceleração de uma rápida aceleração como na decolagem também permite o voo invertido (MIT, 2011).

Com relação aos motores elétricos, existem dois tipos utilizados em quadricópteros: os motores elétricos de Corrente Contínua (CC) com escovas e os motores CC sem escovas, conhecidos como *brushless* em inglês. Os com escovas possuem uma armadura rotativa que funciona como um eletroímã com dois polos. Com um interruptor rotativo revertendo a direção da corrente a cada meio ciclo de rotação, de modo que os polos possam ser repelidos ou atraídos pelos ímãs permanentes conectados ao involucro do motor (BROWN, 2017). Os motores sem escovas por sua vez, não possuem nenhum tipo de contato mecânico entre o rotor (parte diretamente conectada ao eixo) e o estator (parte fixa à carcaça do motor) (SILVA, 2014). Possuem um ímã permanente e mudam a direção de rotação com alterações de polaridade. A Figura 2 apresenta um modelo esquemático simplificado dos dois tipos de motores. À esquerda é possível observar o motor com escovas e à direita o motor sem escovas.

Figura 2 – Modelo esquemático dos motores com escovas e sem escovas



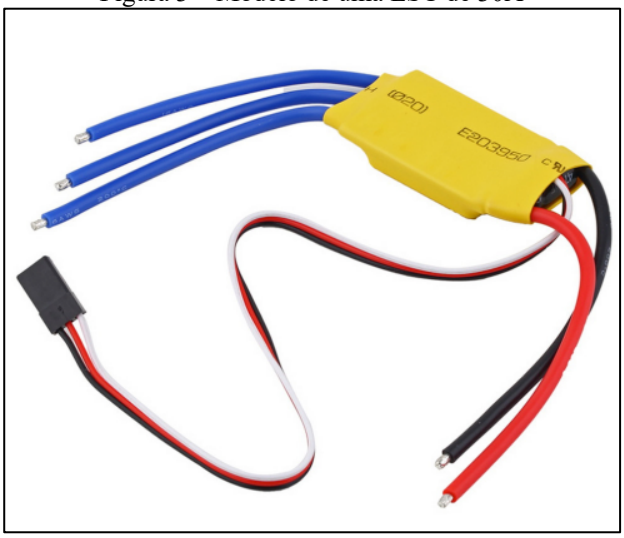
Fonte: MotionControlGuide (2014).

A principal diferença entre os dois tipos de motores é que o motor com escovas exige um cuidado maior na manutenção das escovas, pois se estas estiverem gastas o consumo de energia aumenta e a eficiência do motor diminui. O motor com escovas é então indicado em casos em que o motor não necessite trabalhar por muitas horas (mais de 1.000) e que não seja necessária alta eficiência durante a operação (BROWN, 2017). Já o motor sem escovas é recomendado para longas jornadas de operação e que necessitem de alta eficiência continuamente (BROWN, 2017). Deve-se mencionar, entretanto que os motores sem escovas apresentam um custo mais elevado, pois necessitam de um circuito de acionamento e controle de velocidade dedicado (SILVA, 2014).

Os motores *brushless* são controlados através de um controlador eletrônico dedicado e um mecanismo de feedback de velocidade (BROWN, 2017). Este mecanismo é a controladora eletrônica de velocidade, em inglês Electronic Speed Controller (ESC), responsável por enviar a corrente modulada apropriada para os motores (WANG, 2015). A corrente é gerada por meio de um circuito dedicado que eletronicamente gera uma sequência de acionamento para os conjuntos de Transistores de Efeito de Campo de Semicondutor de Óxido Metálico, em inglês Metal Oxide Semiconductor Field Effect Transistor (MOSFET), que por sua vez acionam as bobinas do motor na sequência correta (SILVA, 2014). Cada motor possui sua respectiva ESC, esta é alimentada pela bateria do quadricóptero e se conecta ao motor por meio de três saídas independentes, além de possuir um conector por onde é recebido o sinal de Modulação de Largura de Pulso,

em inglês Pulse Width Modulation (PWM), de controle (SILVA, 214). A ESC recebe os sinais da controladora de voo e determina quanto de potência deve ser enviada para o respectivo motor a ela conectado. A Figura 3 apresenta um dos modelos mais comuns de ESCs utilizadas em quadricópteros elétricos.

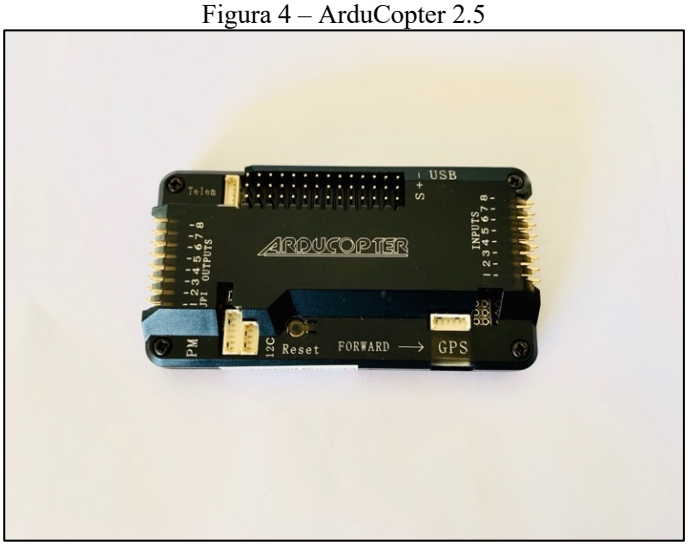
Figura 3 – Modelo de uma ESC de 30A



Fonte: AliExpress (2018a).

A controladora de voo por sua vez é o componente central da aeronave. Ela está encarregada de interpretar os valores obtidos dos sensores que compõem o quadricóptero (caso existam) e receber os comandos externos do usuário. Estes comandos podem vir de um controle remoto via rádio ou mesmo de um dispositivo móvel através de wi-fi. Após efetuada a leitura dos sensores e recebido os comandos, a controladora então calcula e envia os sinais para as ESCs. Existem diversas controladoras de voo disponíveis hoje no mercado que podem ser empregadas em quadricópteros elétricos. Pode-se citar como exemplo a ArduPilot Mega 2.5 que além de controlar RPASs de múltiplos rotores, também pode ser empregada em aeromodelos de helicópteros e de asa fixa (ARDUPILOT, 2018). A ArduPilot Mega 2.5 possui o projeto Open Source no GitHub, nomeado ArduCopter (ARDUCOPTER, 2018) que é o firmware utilizado na controladora. A Figura 4 apresenta o microcontrolador ArduCopter 2.5, uma versão do ArduPilot Mega 2.5 que já possui o firmware específico para controlar drones com múltiplos rotores.

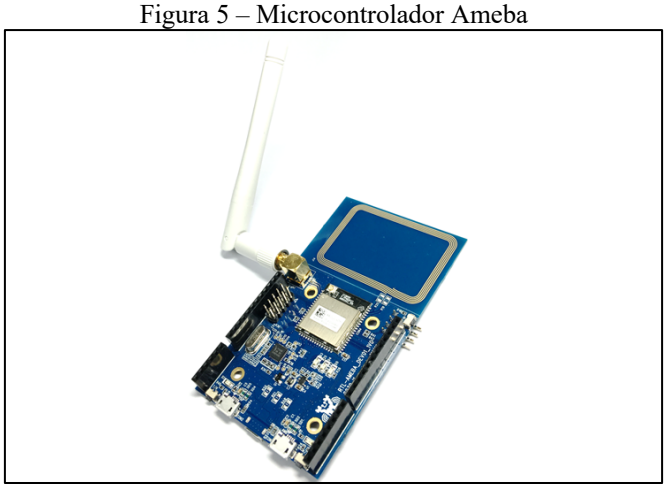
Figura 4 – ArduCopter 2.5



Fonte: arquivo pessoal.

Além dos microcontroladores com a finalidade específica de controlar um RPAS, é possível utilizar também microcontroladores de uso geral para esta finalidade, como por exemplo a Ameba RTL8195 (aqui denominada somente Ameba) da fabricante Realtek (AMEBAIOT, 2018b). A Ameba é um microcontrolador compatível com Arduino, voltada para o desenvolvimento de aplicações com internet das coisas, em inglês Internet Of Things (IOT). Ela possui wi-fi integrado, NFC, 4 pinos com suporte a Modulação de Largura de Pulso, em inglês Pulse Width Modulation (PWM), de um total de 30 GPIOs, entre outras características (ARMMBED, 2018). Esta placa possui ainda um processador de 32bits ARM de 166MHz, 1MB de memoria ROM e 2MB de SDRAM (AMEBAIOT, 2018c). A Figura 5 apresenta o microcontrolador Ameba.

Figura 5 – Microcontrolador Ameba



Fonte: ArmMBED (2018).

2.3 TRABALHOS CORRELATOS

Esta seção apresenta três trabalhos com propósitos correlacionados ao projeto apresentado neste artigo. O primeiro (Quadro 1) trata-se de uma pesquisa acerca do hardware para controle avançado de RPASs do tipo quadricóptero (SILVA, 2014). O segundo (Quadro 2) apresenta os aspectos gerais e análises envolvendo a propulsão elétrica de um quadricóptero (SILVA, 2011). O terceiro (Quadro 3) propõe uma arquitetura de software para o desenvolvimento de quadricópteros (ALVES, 2016).

Quadro 1 – Hardware para controle avançado de veículo aéreo não tripulado do tipo quadricóptero

Referência	Silva (2014)
Objetivos	Apresentar um projeto e desenvolvimento de um sistema de hardware de baixo consumo de energia, para o controle de um quadricóptero. Além disso se propõe a realizar o estudo dos princípios de voo de aeronaves.
Principais funcionalidades	Estudos a cerca do funcionamento de um RPAS e princípios de voo. Além de uma pesquisa sobre: os motores *brushless,* as ESCs, os sensores (sonar, Unidade de Medida Inercial (UMI) e GPS), comunicação sem fio, baterias, controlador de voo e microcontrolador empregados em quadricópteros.
Ferramentas de desenvolvimento	Não foi identificado em função de ser um projeto de estudo.
Resultados e conclusões	Segundo Silva (2014), a utilização dos drones no meio civil está cada vez mais presente, o que resulta na necessidade de pesquisas para se projetar equipamentos confiáveis, precisos e com boa autonomia de voo. Ainda conclui que é um desafio embarcar um sistema totalmente autônomo e robusto em um drone, o que origina a necessidade de se projetar um hardware que ofereça a possibilidade de implementar algoritmos de controle avançados.

Fonte: elaborado pelo autor.

Silva (2014) realizou um levantamento das características dos RPASs e o estudo do seu funcionamento, sua aerodinâmica, princípios de voo e métodos de controle. Além de efetuar a pesquisa de todo o hardware necessário para o funcionamento e automatização do equipamento. Parte da proposta do projeto apresentado neste artigo consiste em elencar e especificar os componentes de hardware necessários para construção de um quadricóptero elétrico, o que representa parte da pesquisa realizada por Silva (2014).

Quadro 2 – Quadricóptero, aspectos gerais e análises da propulsão elétrica

Referência	Silva (2011)
Objetivos	Apresentar uma visão geral da teoria de voo de um quadricóptero desenvolvido no Instituto Federal Fluminense, com a proposta de fazer coleta de dados ambientais (SILVA, 2011).
Principais funcionalidades	O quadricóptero desenvolvido tem como principal finalidade a coleta de dados ambientais como amostras de atmosfera em diversas altitudes e fazer fotografias aéreas de regiões de preservação ambiental.
Ferramentas de desenvolvimento	Não foi identificado em função de ser uma pesquisa e análise da propulsão elétrica.
Resultados e conclusões	Após realizar seus experimentos e efetuar as análises dos mesmos, Silva (2011) definiu as configurações e os propulsores (conjunto motores e hélices) mais adequados as necessidades do projeto do Instituto Federal Fluminense. Silva (2011) ainda conclui que para se obter uma excelente performance é de extrema importância a escolha adequada do sistema propulsor (SILVA, 2011).

Fonte: elaborado pelo autor.

Silva (2011) apresenta equações básicas para efetuar a manobrabilidade de um quadricóptero, mantendo o equilíbrio do mesmo. Considerando por exemplo a necessidade da anulação do torque dos motores para manter o equilíbrio da aeronave. Apesar de todos os testes e cálculos matemáticos realizados por Silva (2011), a escolha de seu trabalho como correlato a este projeto ocorre devido as equações e princípios de controle do quadricóptero apresentados.

Quadro 3 – Proposta de arquitetura de software para drones quadricópteros

Referência	Alves (2016)
Objetivos	Propor uma arquitetura lógica capaz de controlar quadricópteros. Também tem em seu escopo permitir a implementação de camadas especialistas, escalabilidade de recursos e fácil manutenção do software desenvolvido.
Principais funcionalidades	A arquitetura proposta é constituída de 4 elementos: o Equipamento de Comando e Aquisição de Dados (ECAD), a Unidade de Medida Inercial (UMI), a Unidade Central de Processamento (UCP) e os Atuadores.
Ferramentas de desenvolvimento	Não foi identificado em função de ser uma proposta de arquitetura de software.
Resultados e conclusões	Segundo Alves (2016), "durante o funcionamento do sistema o protótipo realizou o voo executando todos os controles de estabilidades programados e foi capaz de checar e interpretar os comandos provenientes do ECAD". Alves (2016, pg. 17) conclui com os resultados de seus testes práticos que a arquitetura proposta permite a implementação de camadas especialistas, escalabilidade de recursos e fácil manutenção do software.

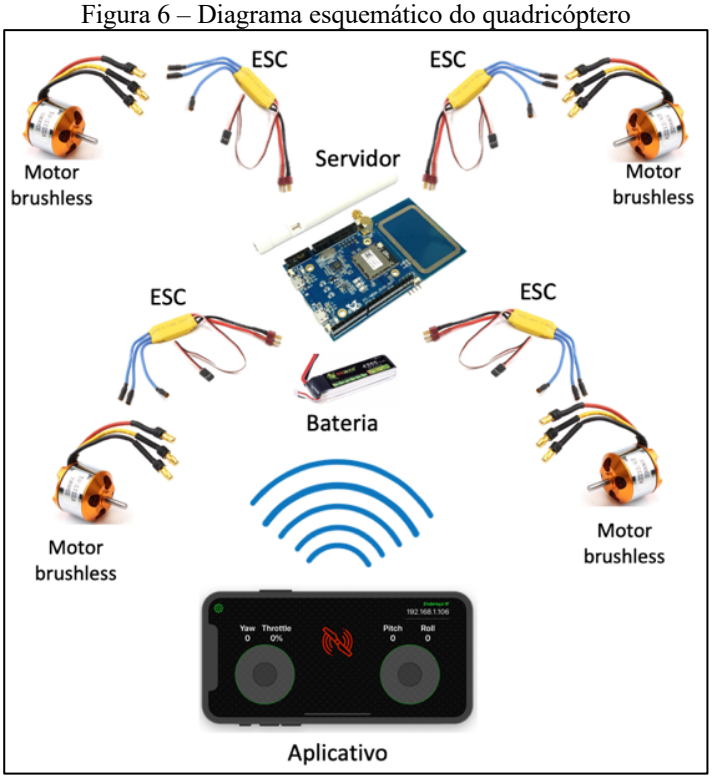
Fonte: elaborado pelo autor.

Alves (2016) projetou uma arquitetura lógica dos componentes de seu quadricóptero e construiu um modelo de baixo custo conforme suas especificações. O trabalho de Alves mostra-se relevante ao projeto aqui apresentado por propor uma arquitetura para o software de controle de um quadricóptero.

3 DESCRIÇÃO DO PROJETO

O projeto divide-se em três elementos: o `Quadricóptero` que representa todos os componentes de hardware utilizados na construção do equipamento; o `Servidor` que contempla o componente de software (desenvolvido em C++) responsável por receber, interpretar e executar os comandos do usuário e então comandar o quadricóptero, e o `Aplicativo` (desenvolvido em Swift) que se comunica com o `Servidor` e age como interface entre o usuário e o quadricóptero. A Figura 6 apresenta um diagrama esquemático simplificado de todo o projeto.

Figura 6 – Diagrama esquemático do quadricóptero



Fonte: elaborado pelo autor.

3.1 O QUADRICÓPTERO

Para a construção do `Quadricóptero` optou-se por utilizar quatro motores *brushless*. O motor selecionado foi o 2212 de 12A. Este motor possui 22mm de diâmetro, por 12mm de comprimento e 1000Kv (RPM/volts), ou seja, para cada volt da bateria o motor entregará 1000 RPM. A ESC selecionada foi uma de 30A, mais que o suficiente para os motores utilizados. Silva (2014) menciona que o recomendável são ESCs com capacidade de no mínimo 20% superior a corrente máxima do motor. A bateria escolhida foi uma 3S Lipo de 11.1v com 4200mAH, que permite aproximadamente 15 minutos de autonomia de voo com os demais componentes escolhidos. Já as hélices selecionadas são feitas de plástico com diâmetro de 254mm, sendo dois pares, um com passo no sentido horário e o outro par com passo no sentido anti-horário. O *frame* (chassi) F-450 de 450mm foi o escolhido para comportar os demais componentes.

Como controladora de voo foi selecionada a Ameba (Ameba RTL8195), isto pois além de possuir inúmeros exemplos no site do fabricante, facilitando assim o desenvolvimento, pretende-se realizar o controle do quadricóptero utilizando um aplicativo móvel através de uma rede wi-fi, aproveitando assim a conectividade wi-fi da Ameba. A controladora é alimentada através de uma bateria extra de 2000mAh por meio de uma conexão USB. A Tabela 1 apresenta a listagem dos componentes utilizados com as respectivas massas e preços (sem considerar os impostos), sendo estes estimados em dólares e pesquisados no site aliexpress.com (ALIEXPRESS, 2018b), com exceção da Ameba que foi pesquisado e adquirido no site sparkfun.com (SPARKFUN, 2018).

Tabela 1 – Descrição dos componentes utilizados no quadricóptero

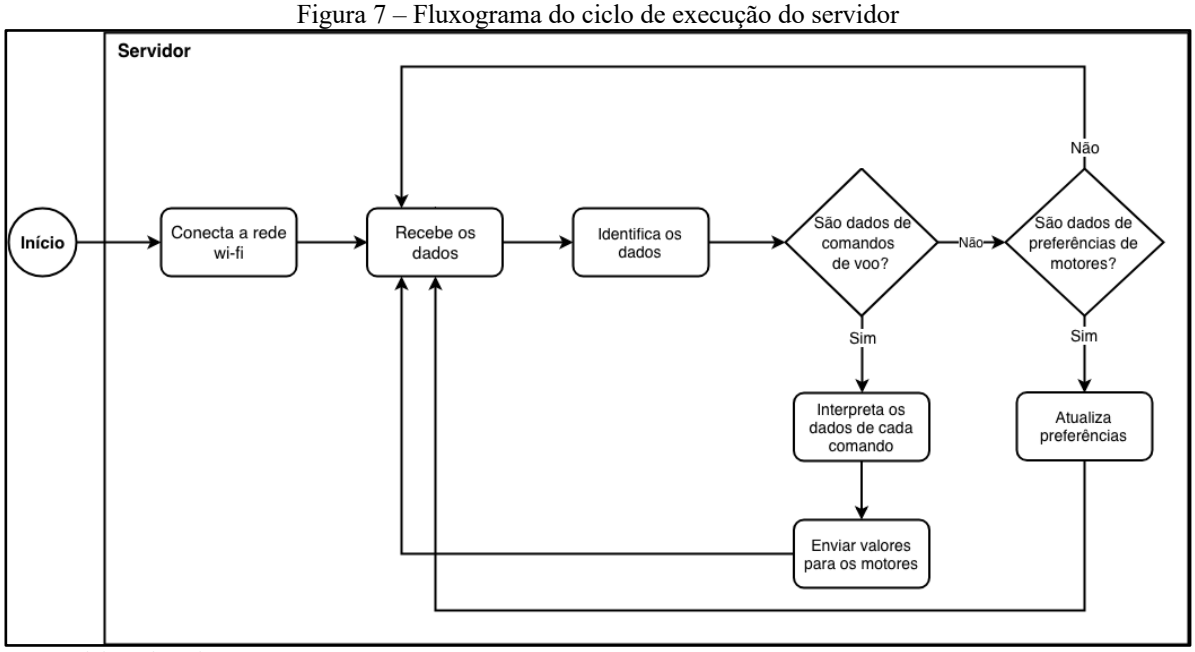
Descrição do equipamento	Quantidade	Preço total (US$) estimado
Microcontrolador Ameba	1	25,00
Motor Brushless – 2212 1000kv	4	24,00
ESC – 30A Simonk Brushless	4	24,00
Hélice – 254mm	4	2,25
Bateria – Lipo 11.1v 4200mah 40c 3s	1	35,00
Frame – F450 Quadcopter Quadro x1	1	12,00
Ameba RTL8195	1	25,00
Bateria extra – 2000mAh	1	8,50

Fonte: elaborado pelo autor.

3.2 O SERVIDOR

O `Servidor` é o software embarcado no microcontrolador Ameba (Ameba RTL8195). Ele é responsável por receber os comandos do usuário, interpretá-los e então aplicá-los sobre o hardware do quadricóptero. O `Servidor` possui três entidades principais: `PegasusServer`, `PegasusProcessor` e `PegasusFlightController`, além de duas entidades específicas para representar os motores: `PegasusMotor` e `PegasusMotorPreferences`. A Figura 7 apresenta o fluxo de execução do `Servidor`. Os próximos parágrafos apresentam um pouco das responsabilidades de cada uma das entidades e o fluxo em mais detalhes. O código fonte completo do projeto do `Servidor` está disponível em: https://github.com/JoaoPauloZ/PegasusServer.

Figura 7 – Fluxograma do ciclo de execução do servidor



Fonte: elaborado pelo autor.

A comunicação entre o Servidor e o Quadricóptero se dá por meio da entidade PegasusMotor. Esta entidade se comunica com os ESCs através da função setSpeed(int speed), onde speed é um número inteiro representando a velocidade desejada, podendo variar entre 0 e 100. O ESC é reconhecido pelo Servidor como sendo um servo motor. Dessa forma os comandos enviados ao ESC precisam ser um ângulo inteiro, variando entre 0º (rotação mínima) e 179º (rotação máxima).

Além da função setSpeed(int speed), a entidade PegasusMotor possui uma propriedade do tipo PegasusMotorPreferences. Esta propriedade especifica os valores para o ângulo mínimo, minAngleESC (predefinido em 65º); ângulo máximo maxAngleESC (predefinido em 179º) e um valor de incremento increaseValue (predefinido em 0). Durante testes iniciais observou-se que para o motor começar a girar efetivamente era necessário um ângulo mínimo de 65º, sendo então este definido como o minAngleESC padrão.

O Quadro 4 apresenta a função setSpeed(int speed). Nele é possível observar que quando a speed for superior a 25 é somado o valor de increaseValue sobre a mesma, antes de realizar a normalização (função map) do valor speed para posteriormente enviá-lo ao ESC.

Quadro 4 – Função setSpeed, responsável por comandar os motores

```
11  void PegasusMotor::setSpeed(int speed) {
12      if (speed >= 0) {
13          if (speed > 25) {
14              speed += preference.increaseValue;
15          }
16          int value = map(speed, 0, 100, preference.minAngleESC, preference.maxAngleESC);
17          esc.write(value);
18      }
19  }
```

Fonte: elaborado pelo autor.

Os dados são recebidos pelo Servidor através da entidade PegasusServer, onde são convertidos de bytes para String. O Quadro 5 apresenta a função listen, executada a cada interação da função loop (padrão do Arduino), nela é possível observar os passos necessários para efetuar esta conversão. Na linha 39 é possível observar que foi definido para o pacote um limite de 255 bytes.

Quadro 5 – Função listen, responsável por receber os pacotes de dados

```
34  void PegasusServer::listen() {
35      // if there's data available, read a packet
36      int packetSize = serverUDP.parsePacket();
37      if (packetSize) {
38          // read the packet into packetBufffer
39          int len = serverUDP.read(packetBuffer, 255);
40          if (len > 0) {
41              packetBuffer[len] = 0;
42          }
43          // convert from char array to String
44          String buffer_str(packetBuffer);
45          // call func PegasusProcessor::process
46          processor.process(buffer_str);
47      }
48  }
```

Fonte: elaborado pelo autor.

Após a obtenção da String com os dados é chamado a função process da entidade PegasusProcessor. Esta encarrega-se de identificar a action e assim definir como os dados devem ser interpretados. Identificando o pacote como sendo de comandos de voo a String é processada e convertida para os respectivos valores inteiros de cada comando. Estes são transferidos para entidade PegasusFlightController, que por sua vez possui a função update, responsável por interpretar os valores e comandar os motores da maneira apropriada.

A função update verifica o valor de cada comando de voo e com base nos princípios descritos na seção 2.1 determina a velocidade de rotação para cada um dos quatro motores, para executar a manobra desejada. O Quadro 6 apresenta parte da função update, onde é possível observar os passos necessários para executar uma manobra de arfagem.

É interessante notar a atribuição de um valor mínimo 5 para os motores que diminuem a rotação. Isto é feito para impedir que o motor pare a rotação completamente, o que provocaria a perda de controle do quadricóptero.

Quadro 6 – Parte da função `update` da entidade `PegasusFlightController`

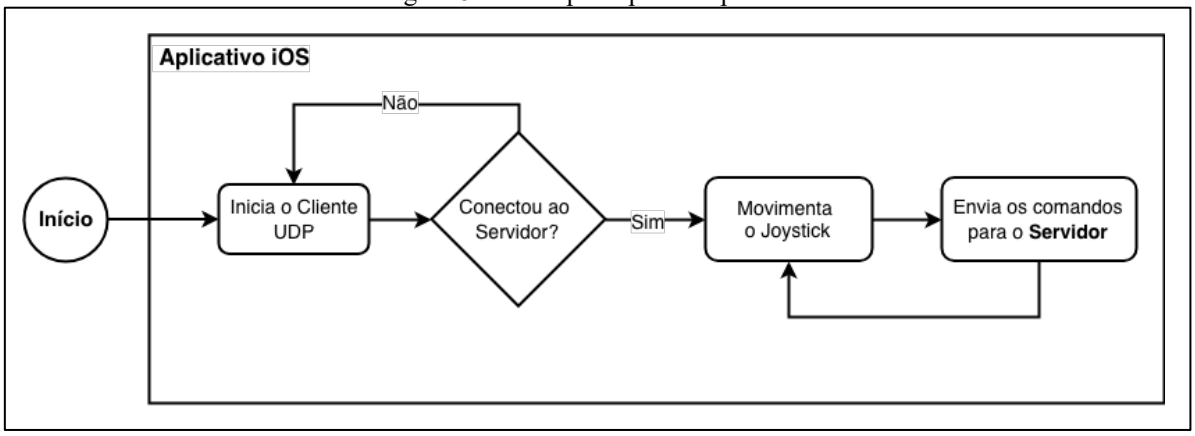
```
31  void PegasusFlightController::update(int throttle, int pitch, int roll, int yaw) {
32
33     int speedMin = 5;
34     int speed0 = throttle;
35     int speed1 = throttle;
36     int speed2 = throttle;
37     int speed3 = throttle;
38
39     if (throttle >= 1 && throttle <= 100) {
40        // [Pitch]
41        // - Move forward: motor 2 and 3 speed up, motor 0 and 1 speed down (pitch > 0)
42        if (pitch > 0) {
43           speed2 += pitch;
44           speed3 += pitch;
45           speed0 -= max(pitch, speedMin);
46           speed1 -= max(pitch, speedMin);
47        }
48
49        // - Move backward: motor 2 and 3 speed down, motor 0 and 1 speed up (pitch < 0)
50        if (pitch < 0) {
51           speed2 -= max(pitch, speedMin);
52           speed3 -= max(pitch, speedMin);
53           speed0 += pitch;
54           speed1 += pitch;
55     ...
89     }
90
91     motor0.setSpeed(speed0);
92     motor1.setSpeed(speed1);
93     motor2.setSpeed(speed2);
94     motor3.setSpeed(speed3);
95
96  }
```

Fonte: elaborado pelo autor.

3.3 O APLICATIVO

Para efetuar a comunicação entre o usuário e o quadricóptero foi desenvolvido um aplicativo para a plataforma iOS. Os elementos fundamentais do aplicativo são os dois *joysticks* utilizados para enviar os comandos ao quadricóptero. Um à esquerda que permite controlar a potência dos quatro motores e executar a manobra de guinada (*yaw,* eixo vertical) e outro à direita, destinado as manobras de arfagem (*pitch*, eixo lateral) e rolagem (*roll*, eixo longitudinal). Na tela principal do aplicativo também se encontra, um botão na parte central que inicia o cliente UPD que envia os pacotes de dados ao servidor, seguindo o padrão de comunicação descrito na seção anterior. Rótulos acima dos *joysticks* indicam os valores atuais de cada eixo. No canto superior direito encontra-se um campo para informar o endereço IP do servidor. O fluxo principal do aplicativo é apresentado na Figura 8, onde é possível observar o processo desde a inicialização do cliente UDP, até o envio dos comandos ao `Servidor`. O código fonte completo do projeto do `Aplicativo` está disponível em: https://github.com/JoaoPauloZ/PegasusApp.

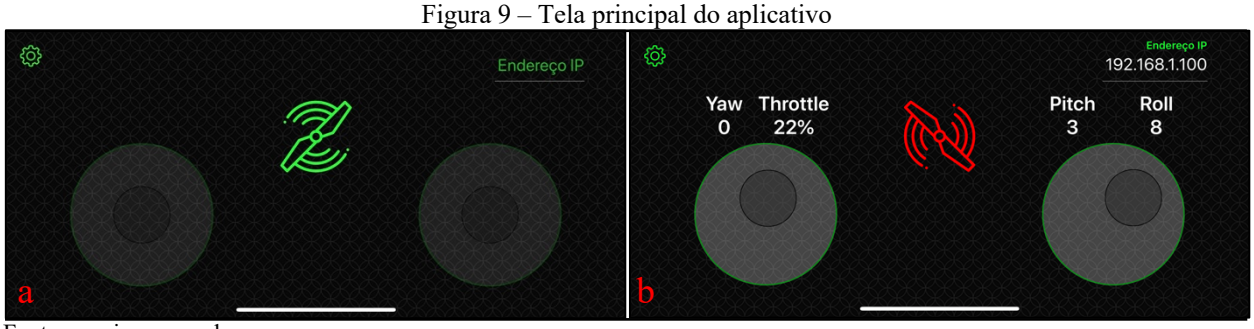
Figura 8 – Fluxo principal do Aplicativo



Fonte: elaborado pelo autor.

A Figura 9 (a) apresenta a tela principal do aplicativo em seu estado inicial, com os *joysticks* bloqueados, rótulos ocultos e o botão de conexão na cor verde, indicando que a conexão com o servidor não foi estabelecida. A Figura 9 (b) apresenta a tela no estado de conexão estabelecida, indicando que o aplicativo está apto para comandar o quadricóptero.

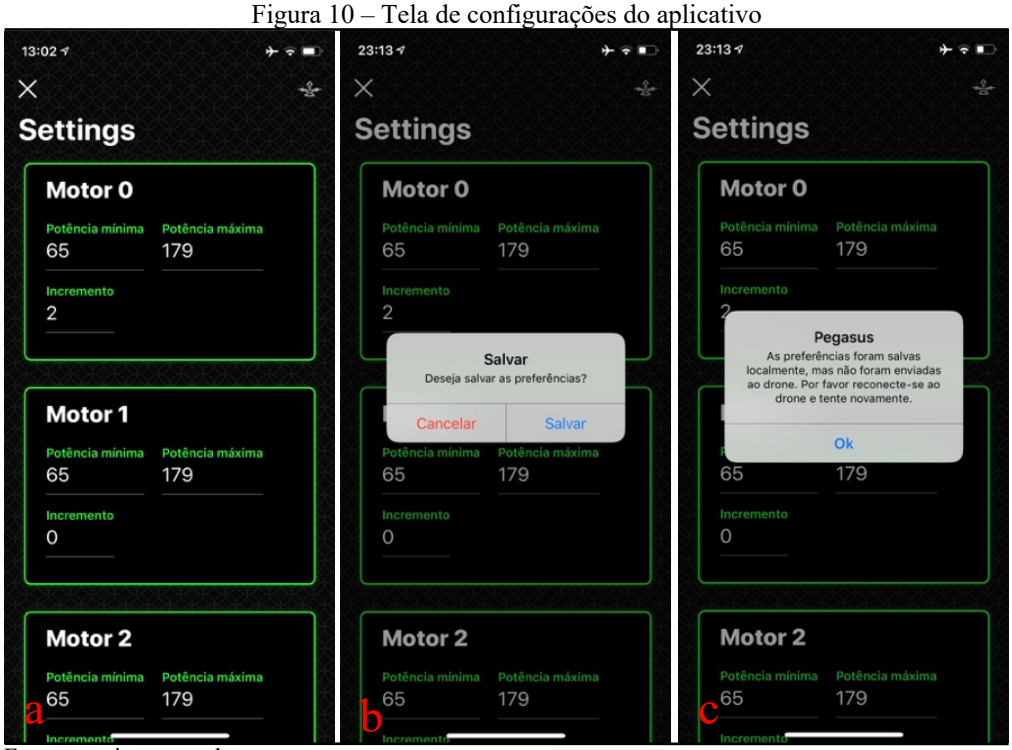
Figura 9 – Tela principal do aplicativo



Fonte: arquivo pessoal.

Para atualizar as preferências dos motores do quadricóptero (ver seção 3.2) foi criado uma tela de configurações. Esta é acessível por meio da engrenagem no canto superior esquerdo da tela principal. Como pode-se observar na Figura 10 (a), nesta tela tem-se os valores das preferências de cada um dos motores e um botão no canto superior direito para salvar as alterações localmente e enviá-las ao `Servidor`. Quando o botão é pressionado é exibido o alerta da Figura 10 (b), se os valores não forem enviados com sucesso é exibido a mensagem da Figura 10 (c).

Figura 10 – Tela de configurações do aplicativo



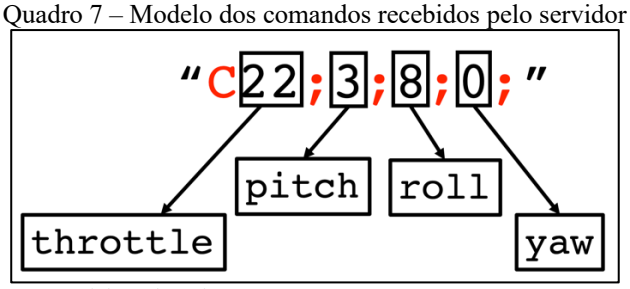
Fonte: arquivo pessoal.

3.4 COMUNICAÇÃO ENTRE O SERVIDOR E O APLICATIVO

A entrada de dados no servidor é feita através de uma rede wi-fi. Os dados são transferidos através do protocolo User Datagram Protocol (UDP). Definiu-se um formato de comunicação mais simplificado que o atual e amplamente utilizado JavaScript Object Notation (JSON). Isso a fim de que o pacote de dados seja menor. Quanto menor a quantidade de dados por pacote, menor é o tempo de envio, logo mais rápido é a resposta aos comandos. A não utilização do formato JSON dispensa o emprego de bibliotecas específicas para trabalhar com o mesmo, dessa forma diminui-se também a complexidade na interpretação dos dados.

O formato definido inicia-se sempre com um caractere de controle `action`. Este indica como os demais caracteres devem ser interpretados. O caractere `C` por exemplo, indica ao Servidor que é um pacote de comandos. Logo na sequência tem-se o valor da potência (*throttle*) a ser empregada nos motores, em seguida separados cada um por um `;` tem-se os valores de arfagem (*pitch*), rolagem (*roll*) e guinada (*yaw*). Os comandos são todos números inteiros variando de 0 a 100. O Quadro 7 apresenta um exemplo do formato de comunicação desenvolvido com os respectivos valores apresentados na Figura 9 (b).

Quadro 7 – Modelo dos comandos recebidos pelo servidor



Fonte: elaborado pelo autor.

A medida que os testes foram avançando observou-se a necessidade de alterar alguns parâmetros dos motores (`PegasusMotorPreferences`) em tempo de execução, assim eliminando a necessidade de recompilar o código do Servidor a cada nova alteração dos valores a fim de melhorar a produtividade durante os testes. Para tanto definiu-se um novo formato de comunicação. O caractere de controle `action` para atualização das preferencias é o `U`. Como no formato dos comandos de voo, a `action` é seguida diretamente por valores inteiros separados por `;`. Sendo estes respectivamente: `increaseValue`, `minAngleESC` e `maxAngleESC` respeitando a ordem dos motores, isto é, começando pelo motor 0 e terminado com o motor 3, sendo o término das preferências de cada motor finalizado pelo caractere `&`. O formato descrito pode ser observado no Quadro 8.

Quadro 8 – Formato de comunicação para alteração das preferências dos motores

```
“UiMt0;minMt0;maxMt0;&iMt1;minMt1;maxMt1;&iMt2;
      minMt2;maxMt2;&iMt3;minMt3;maxMt3;&”

  “U3;66;179;&0;65;179;&1;66;170;&0;77;178;&”
```

Fonte: elaborado pelo autor.

4 RESULTADOS

Esta seção apresenta os resultados, aprendizados, limitações e testes do projeto. Para melhor compreensão e leitura dividiu-se de forma semelhante à seção anterior, com subdivisões específicas para o `Quadricóptero` (componente de hardware), o `Servidor` (software responsável por controlar o `Quadricóptero`) e o `Aplicativo` (software responsável por captar os comandos do usuário e enviar ao `Servidor`).

4.1 O QUADRICÓPTERO

Nos estágios inicias deste projeto esperava-se construir um quadricóptero de baixo custo. Para tanto pesquisou-se diversos modelos de componentes e então foi feita a aquisição de quatro conjuntos de motores com escovas que já possuíam hélices acopladas e eixos para fixação no chassi. Este conjunto era originário do modelo comercial Eachine E58 RC Quadcopter (EACHINE, 2018), tendo os componentes um custo aproximado de 16,00 dólares. Para fixação do conjunto foi utilizado uma placa de acrílico de dimensões de 12×12×2mm. A placa de acrílico foi escolhida por ser leve, resistente, fácil de fazer os furos para a fixação dos conjuntos adquiridos e posteriormente a fixação da Ameba (Ameba RTL8195). A Figura 11 apresenta o modelo inicial do protótipo montado sobre a placa de acrílico.

Figura 11 – Protótipo inicial desenvolvido sobre uma placa de acrílico

Fonte: arquivo pessoal.

A principal dificuldade encontrada nesse protótipo inicial foi conseguir controlar a velocidade dos motores, visto que diferente dos motores *brushless*, os motores com escovas utilizados nesse protótipo não recebem os comandos de controle por meio de ESCs. Dessa forma foi necessário montar um circuito elétrico utilizando um MOSFET para receber os sinais PWM da Ameba e então enviar a potência correspondente para o motor. Entretanto, durante os testes não foi possível controlar a velocidade de forma linear e constante, ocorrendo variações bruscas na potência ou mesmo funcionando apenas com a potência em 0 ou 100%, sem conseguir atingir valores intermediários. O conhecimento escasso do autor em eletrônica fez com que este protótipo fosse descontinuado, pois estava exigindo muito tempo nas fases iniciais do projeto, prejudicando o cronograma.

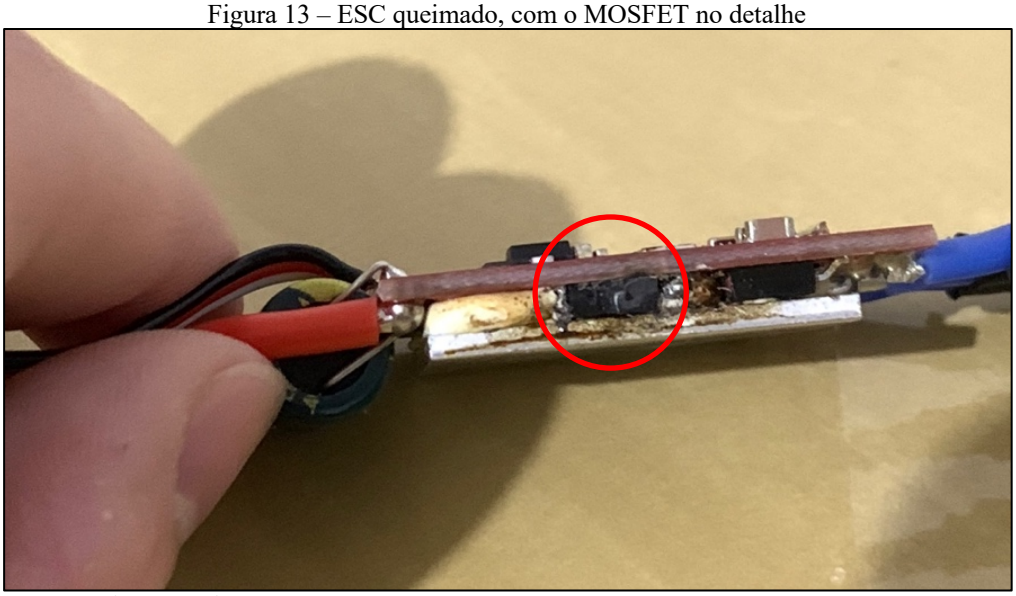
Após os testes insatisfatórios mencionados optou-se então por utilizar quatro motores *brushless* em conjunto com quatro ESCs (especificados na seção 3.1). Todavia o conjunto ficou com um valor mais elevado (aproximadamente 48,00 dólares contra os 16,00 dólares do conjunto anterior). Entretanto o novo conjunto possibilitou o controle adequado dos motores, bastando utilizar a biblioteca `AmebaServo` disponível no Software Development Kit (SDK) do microcontrolador Ameba para se comunicar com os ESCs. Nos primeiros testes desse conjunto foi utilizado um potenciômetro com variação entre 0 e 1023 para se ter um maior controle da velocidade. O valor do potenciômetro era lido pela Ameba que então o normalizava para um ângulo entre 0º e 179º (mencionado na seção 3.2) e então o valor normalizado era enviado para a ESC, que por sua vez girava o motor. Durante estes testes percebeu-se que o motor só começava a girar efetivamente com ângulo superior a aproximadamente 65º, sendo então este definido como o ângulo mínimo (`minAngleESC`) padrão para as preferências dos motores (`PegasusMotorPreferences`). A Figura 12 apresenta o conjunto de motores e ESCs selecionados para a construção do protótipo final.

Figura 12 – Motores e ESCs selecionados

Fonte: arquivo pessoal.

Nos estágios mais avançados, com o quadricóptero já montado, ocorreu um incidente que provocou a queima de um dos ESCs. Após conversar com o professor de eletrônica Jean Strutz (STRUTZ, 2018) chegou-se as possíveis causas, que poderiam ser desde um defeito de fabricação ou mesmo um curto circuito acidental. Entretanto a mais provável, como destacado pelo professor, estaria na sequência dos acontecimentos descritos a seguir: no momento do incidente os ESCs estavam conectados a bateria e ao microcontrolador. Simultaneamente estava sendo enviado uma nova versão do software do `Servidor` para o microcontrolador Ameba (durante os processos de envio do software anteriores os ESCs ficavam desconectados da bateria). Um ESC possui uma série de MOSFETs que em conjunto criam o pulso que gira o motor, porém se duas dessas chaves em série se fecharem simultaneamente o circuito entra em curto. Como o ESC começou a queimar logo após o envio do software é provável que durante a atualização algum pulso tenha sido enviado de forma inapropriada ao ESC, que por sua vez fechou o circuito completamente de duas ou mais chaves e então teve-se o curto circuito que queimou o mesmo. A Figura 13 apresenta o ESC queimado, no detalhe é possível ver o MOSFET onde teve início o curto circuito.

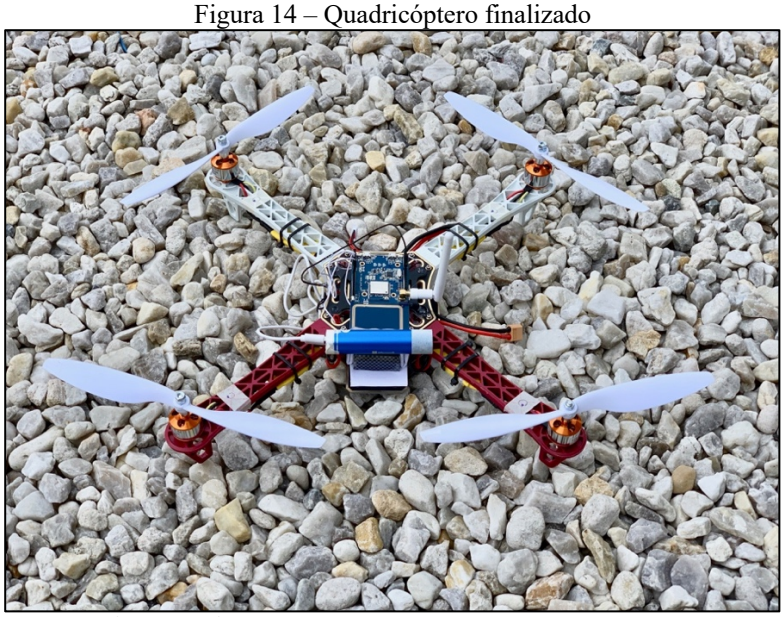
Figura 13 – ESC queimado, com o MOSFET no detalhe



Fonte: arquivo pessoal.

Após o incidente o quadricóptero foi parcialmente desmontado e a ESC danificada substituída. Medidas preventivas foram tomadas, tais como: não enviar mais o software para a Ameba com os ESCs conectados à bateria; periodicamente durante os testes verificar se as ESCs não estão aquecendo em excesso (um pequeno acréscimo na temperatura é normal durante o uso em um componente como esse). A Figura 14 apresenta o quadricóptero finalizado.

Figura 14 – Quadricóptero finalizado



Fonte: arquivo pessoal.

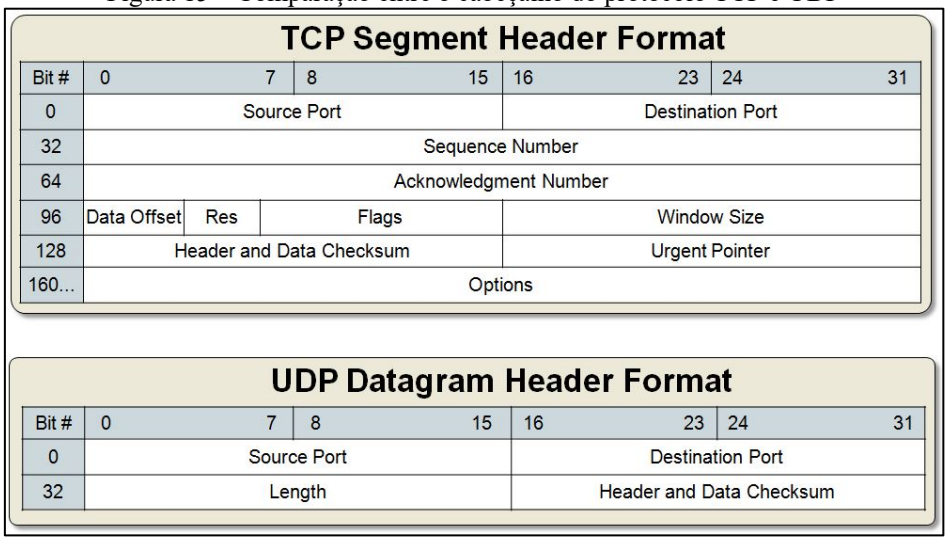
4.2 O SERVIDOR

Inicialmente foi cogitada a utilização do protocolo de comunicação Hypertext Transfer Protocol (HTTP) em conjunto com o formato JSON para a comunicação entre o `Servidor` e o `Quadricóptero`. Entretanto percebeu-se que a extensa quantidade de informações nos metadados do protocolo poderia causar atraso entre o comando do usuário e o efetivo movimento no quadricóptero. Além de diversas características desnecessárias para a finalidade do projeto, como a garantia de entrega dos dados que é dispensável para o controle de um quadricóptero. Já que os pacotes são enviados de forma contínua, a verificação de um pacote perde seu sentido, visto que o próximo pacote (enviado logo na sequência) corrige uma eventual falha no anterior. O formato JSON também foi deixado de lado, pois além de tornar o pacote de dados desnecessariamente grande, também adicionaria uma complexidade extra na interpretação dos dados no `Servidor`. Havendo assim a necessidade de se adicionar uma biblioteca de terceiros para a interpretação do JSON ou uma implementação própria, visto que o SDK da Ameba não possui bibliotecas para este fim.

Descartado o protocolo HTTP e o formato JSON, optou-se então pelo protocolo UDP e um formato próprio para o envio dos comandos apresentado na seção 3.2. O protocolo UDP não possui um cabeçalho extenso e também não garante a entrega dos pacotes. Entretanto permite ao destinatário validar se os dados estão íntegros, assim podendo decidir se os

utiliza ou espera o próximo pacote. A Figura 15 apresenta um comparativo entre o cabeçalho do protocolo Transmission Control Protocol (TCP) que é o protocolo utilizado sob o HTTP e o protocolo UDP. É possível observar como o protocolo TCP requer muito mais informações para garantir a entrega de dados enquanto o UDP possui muito menos informações.

Figura 15 – Comparação entre o cabeçalho do protocolo TCP e UDP



Fonte: Microchip (2018).

Durante os testes com o protótipo foi percebido a necessidade de se ter um valor (`increaseValue`) para incrementar ou decrementar a velocidade dos motores, permitindo assim ajustar possíveis variações na velocidade de cada motor de forma independente. Este valor pode ser alterado com o quadricóptero ligado, por meio da tela de configurações do `Aplicativo`, sem a necessidade de recompilar o software a cada novo teste, otimizando o processo de testes. Entretanto, durante os primeiros testes foi observado que quando aplicado o incremento sobre o valor inicial de rotação (zero) de um motor em específico (`Motor 3` – motor que teve a ESC substituída), este apresentava um comportamento inconsistente. O motor em questão começava a girar um pouco depois que os demais motores, causando assim uma desestabilização no quadricóptero. A solução encontrada para o problema foi simples, porém eficiente. Condicionou-se que o incremento só deve ser aplicado quando a velocidade for superior a 25 (isto é, 25% da potência máxima). A escolha do número 25 como referência deu-se pelo fato do quadricóptero começar a voar por volta dos 30% da potência, ponto em que o valor de `increaseValue` realmente faz diferença sobre o equipamento.

Apesar de adotadas medidas para compensar possíveis inconsistências em nível de hardware, os testes finais com o quadricóptero não foram muito positivos. Mesmo compensado as variações de cada motor e ajustando os componentes individualmente, buscando assim manter a massa do quadricóptero a mais distribuída possível, o mesmo não se manteve estável durante o voo. Não foi possível manter um voo pairado ou executar manobras específicas com a precisão desejada. Embora grande parte das tendências indesejadas ao voo tenham sido corrigidas, seja por ajustes no posicionamento do hardware ou alterações do `increaseValue`, durante a última série de testes ocorreu um acidente. No acidente algumas hélices se danificaram e a Ameba e a bateria extra se desprenderam do chassi do quadricóptero. Os testes então foram interrompidos e não foram mais retomados até a elaboração do presente artigo.

4.3 O APLICATIVO

Como apresentado na seção 2.1, para se manobrar um quadricóptero é necessário incrementar a velocidade de um par de motores e decrementar do outro na mesma proporção. Entretanto nos primeiros testes foi observado que é importante garantir que um certo limite não seja ultrapassado, pois senão a aeronave pode atingir ângulos de inclinação críticos provocando assim a perda de sustentação. Para diminuir as chances de tal situação acontecer em testes futuros, condicionou-se que a atuação dos eixos dos *joysticks* utilizados para efetuar as manobras tenham efeito efetivo apenas a partir do valor 15 e fica a cargo do usuário decidir até que valor ele deseja comandar.

Durante a primeira série de testes notou-se uma certa dificuldade em manter o contato visual com o quadricóptero e simultaneamente observar a indicação dos *joysticks* na tela do `Aplicativo`. Isto porque os *joysticks* possuem uma área de atuação pequena para uma variação de valores muito grande (0 – 100). Uma forma encontrada para diminuir a necessidade de observar a tela do aplicativo a fim de verificar a variação dos valores foi adicionando uma resposta tátil no dispositivo a cada alteração dos valores.

O aplicativo foi testado em dois dispositivos diferentes, em um iPhone XS e um iPad de 5ª geração, ambos rodando iOS 12.1. Durante os testes as respostas do `Aplicativo` foram precisas tanto no iPhone quanto no iPad, não se notou nenhum atraso entre o comando no *joystick* e a respectiva resposta no `Quadricóptero`. Entretanto a utilização do `Aplicativo` no iPhone mostrou-se mais confortável, dado seu tamanho e peso menores.

5 CONCLUSÕES

Diante dos resultados apresentados conclui-se que o projeto atingiu os objetivos pretendidos, contudo se limitando ao estudo da concepção e escolha do hardware bem como no desenvolvimento do `Servidor` e `Aplicativo`, estes com algumas limitações. O protótipo construído se mostrou operacional e atendeu as necessidades do projeto, sendo capaz de decolar utilizando aproximadamente 30% da potência total disponível, evidenciando escolhas adequadas dos componentes de hardware. A escolha da Ameba RTL8195 como microcontrolador também se mostrou apropriada, tendo em vista que esta foi capaz de se comunicar sem grandes dificuldades com os ESCs, além de receber os comandos por meio de sua placa wi-fi integrada. O formato desenvolvido para a comunicação entre o `Aplicativo` e o `Servidor`, unido ao protocolo UDP demonstrou-se rápido e eficiente, suprindo todas as necessidades no quesito de transferência de dados. O `Aplicativo` se demonstrou apto a enviar todos os comandos do usuário ao `Servidor`, além de permitir a alteração das preferências dos motores.

Apesar dos resultados positivos apresentados, os testes realizados também apresentaram limitações do projeto no quesito de estabilidade e manobrabilidade do quadricóptero. Sendo impraticável a realização de um voo pairado ou a execução de movimentos em qualquer um dos três eixos (vertical, lateral e longitudinal) com a precisão adequada. As limitações identificadas podem ser corrigidas com adição de sensores para determinar o posicionamento espacial do quadricóptero e a implementação de um algoritmo dedicado a efetuar a estabilização do mesmo de forma autônoma. Contudo como projeto numa linha de pesquisa relativamente recente, este pode engajar trabalhos futuros a continuar e evoluir na mesma linha de pesquisa. As possíveis extensões propostas para continuar a linha de pesquisa deste projeto são:

a) pesquisar a viabilidade da inclusão de sensores como um acelerômetro, giroscópio ou sonar, para captar os dados do posicionamento do quadricóptero em tempo real;
b) incluir uma camada extra no software do `Servidor` para efetuar o controle de estabilização do quadricóptero de forma autônoma;
c) determinar um limite para a inclinação máxima na execução de manobras no software do `Servidor`;
d) adicionar um sistema de criptografia na comunicação entre o `Aplicativo` e `Servidor`.

REFERÊNCIAS

ALIEXPRESS. **Brinquedos 1 pcs ESC Brushless Motor Speed Controlador RC BEC 30A**. 2018a. Disponível em: <http://pt.aliexpress.com/item/Brinquedos-1-pcs-ESC-Brushless-Motor-Speed-Controlador-RC-BEC-30A-frete-gr-tis-I403/32906288485.html> Acesso em: 29 nov. 2018.

ALIEXPRESS. **AliExpress**. 2018b. Disponível em: <http://aliexpress.com> Acesso em: 29 nov. 2018.

ALVES, Rodrigo R. **Proposta de arquitetura de software para drones quadrotores**. 2016. 27f. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Fundação Universitária Vida Cristã, Pindamonhangaba. Disponível em: <http://www.bibliotecadigital.funvicpinda.org.br:8080/jspui/bitstream/123456789/492/1/RodrigoALVES.pdf>. Acesso em: 29 nov. 2018.

AMEBAIOT. **Introduction to Ameba Quadcopter**. 2018a. Disponível em: <https://www.amebaiot.com/en/ameba-arduino-quadcopter/>. Acesso em: 29 nov. 2018.

AMEBAIOT. **Realtek IoT/Arduino Solution**. 2018b. Disponível em: <https://www.amebaiot.com/en>. Acesso em: 29 nov. 2018.

AMEBAIOT. **Ameba Arduino:** Boards. 2018c. Disponível em: <https://www.amebaiot.com/en/boards/>. Acesso em: 29 nov. 2018.

ARDUCOPTER. **Copter Home.** 2018. Disponível em: <http://ardupilot.org/copter/index.html>. Acesso em: 29 nov. 2018.

ARDUPILOT. **Ardupilot.** 2018. Disponível em: <http://ardupilot.org>. Acesso em: 29 nov. 2018.

ARMMBED. **Realtek RTL8195AM.** 2018. Disponível em: <https://os.mbed.com/platforms/Realtek-RTL8195AM/>. Acesso em: 29 nov. 2018.

BRASIL. Ministério da Defesa. Comando da Aeronáutica. Departamento de Controle do Espaço Aéreo. **Trafego Aéreo:** ICA 100-40: sistemas de aeronaves remotamente pilotadas e o acesso ao espaço aéreo brasileiro. 2015. Disponível em: <https://www.decea.gov.br/static/uploads/2015/12/Instrucao-do-Comando-da-Aeronautica-ICA-100-40.pdf>. Acesso em: 29 nov. 2018.

BROWN, Lisa. **Drone Motors:** Choose the Best Motors for Your Quadcopter. 2017. Disponível em: <https://filmora.wondershare.com/drones/drone-motors.html>. Acesso em: 29 nov. 2018.

D'ANDREA, Raffaello. **O assombroso poder atlético dos quadricópteros**. 2013. Disponível em: <https://www.youtube.com/watch?v=w2itwFJCgFQ>. Acesso em: 29 nov. 2018.

DJI. **Phantom 4 Pro**. 2018. Disponível em: <https://www.dji.com/phantom-4-pro>. Acesso em: 29 nov. 2018.

EACHINE. **Eachine e58 wifi fpv with 2mp wide angle camera high hold mode foldable rc drone quadcopter rtf**. 2018. Disponível em: <https://www.eachine.com/EACHINE-E58-WIFI-FPV-With-2MP-Wide-Angle-Camera-High-Hold-Mode-Foldable-RC-Drone-Quadcopter-RTF-p-1045.html>. Acesso em: 29 nov. 2018.

HANGAR33. **Como as hélices funcionam?**. 2015. Disponível em: <http://blog.hangar33.com.br/como-as-helices-funcionam/>. Acesso em: 29 nov. 2018.

HOMA, Jorge. **Aerodinâmica e teoria de voo:** noções básicas. São Paulo: ASA, 2011. p. 41.

MASSACHUSSETTS INSTITUTE OF TECNOLOGY. Aerospace Controls Laboratory. **MIT ACL - Variable Pitch Quadrotor**. 2011. Disponível em: <https://youtu.be/Vy5Ky50eGJs>. Acesso em: 29 nov. 2018.

MICROCHIP.**TCP vs. UDP**. 2018. Disponível em: <http://microchipdeveloper.com/tcpip:tcp-vs-udp> Acesso em: 29 nov. 2018.

MOTIONCONTROLGUIDE. **Brush vs Brushless DC Motors**. 2014. Disponível em: <http://www.motioncontrolguide.com/learn/tech-tips/motors/subcategory/dc-motors/brush-vs-brushless-dc-motors/>. Acesso em: 29 nov. 2018.

ODRONES. **História dos Drones:** do início aos dias de hoje. 2015. Disponível em: <https://odrones.com.br/historia-dos-drones/>. Acesso em: 29 nov. 2018.

OLSON, Kyler. **The 3 Main Categories of Drones and Their Advantages & Disadvantages**. 2017. Disponível em: <https://botlink.com/blog/the-3-main-categories-of-drones-and-their-advantages-and-disadvantages>. Acesso em: 29 nov. 2018.

PARROT. **Parrot Ar.Drone 2.0 Central Cross**. 2018. Disponível em: <https://www.parrot.com/global/spareparts/drones/parrot-ardrone-20-central-cross#parrot-ardrone-20-central-cross-details> Acesso em: 29 nov. 2018.

SILVA, Leonam Pecly da. Quadricóptero, Aspectos Gerais e Análises da Propulsão Elétrica. In: VIII SEGeT – SIMPÓSIO DE EXCELÊNCIA EM GESTÃO E TECNOLOGIA, 1., 2011, [S.l.]. **Anais...** [s.n.]. Disponível em: <https://www.aedb.br/seget/arquivos/artigos11/39614431.pdf>. Acesso em: 29 nov. 2018.

SILVA, Kleber Lima da. **Hardware para controle avançado de veículo aéreo não tripulado do tipo quadricóptero,** [S.l.], v. 8, n. 1, jul. 2014 Disponível em: <http://www.seer.ufu.br/index.php/horizontecientifico/article/view/24028>. Acesso em: 29 nov. 2018.

SPARKFUN. **Sparkfun**. 2018. Disponível em: <https://www.sparkfun.com> Acesso em: 29 nov. 2018.

STRUTZ, Jean. **Entrevista sobre as possíveis causas do curto circuito e queima do ESC**. Blumenau. 2018. Entrevista feita através de conversação – não publicada.

WANG, David. **Quadcopter Parts:** What are they and what do they do?. 2015. Disponível em: <http://www.quadcopteracademy.com/quadcopter-parts-what-are-they-and-what-do-they-do/>. Acesso em: 29 nov. 2018.