UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

JOSÉ FELIPPE VEDOVATTO LOUREIRO

SISTEMA AUTÔNOMO PARA POUSO E RECARGA DA BATERIA DE UM ARDRONE 2.0

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2017

JOSÉ FELIPPE VEDOVATTO LOUREIRO

SISTEMA AUTONOMO PARA POUSO E RECARGA DA BATERIA DE UM ARDRONE 2.0

Trabalho de Conclusão de Curso como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação, do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Robison Brito

PATO BRANCO

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

TERMO DE APROVAÇÃO

Às 15 horas e 30 minutos do dia 06 de julho de 2017, na sala V003, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Robison Cris Brito(orientador), Pablo Gauterio Cavalcanti e Vinicius Pegorini para avaliar o trabalho de conclusão de curso com o título **Sistema Autônomo para Pouso e Recarga da Bateria de um Ardrone 2.0**, do aluno **Jose Felippe Vedovatto Loureiro**, matrícula 1261576, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____
Prof. Robison Cris Brito
Orientador (UTFPR)

_____
Prof. Pablo Gauterio Cavalcanti
(UTFPR)

_____
Prof. Vinicius Pegorini
(UTFPR)

_____
Profa. Beatriz Terezinha Borsoi
Coordenador de TCC

_____
Prof. Pablo Gauterio Cavalcanti
Coordenador do Curso de
Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

LOUREIRO, José Felippe Vedovatto. Sistema autônomo para pouso e recarga da bateria de um ArDrone 2.0. 2017. 62f. Trabalho de Conclusão de Curso de bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Um drone é um tipo de Veículo Aéreo não Tripulado (VANT) que na maioria das vezes possui quatro hélices. Estes podem ser utilizados em diversas aplicações entre elas se locomover por locais de difícil acesso para veículos terrestres. Além da praticidade dos drones em relação a outros veículos aéreos, o seu custo é relativamente menor se comparado com veículos de grande porte, o que os torna atraentes para atividades variadas. Ainda oferece segurança em relação a eventos perigosos, como acidentes e/ou áreas de risco, por não precisar de piloto. Em um sistema com voo autônomo surge a preocupação com o pouso e o carregamento automático de sua bateria, que não costuma durar mais do que alguns minutos. Utilizando dispositivos embarcados, como sua câmera e módulo receptor de sinal geoestacionário (GPS), é possível implementar funções para otimizar seu voo e carregamento. Com o objetivo de apresentar uma solução para tal problema, este trabalho traz um modelo que utiliza reconhecimento de imagem para permitir o pouso do drone em um sistema autônomo. Esta rotina de pouso guiada pela imagem gerada pelo drone, torna o processo de pouso e recarga autônomos, sem intervenção humana.

Palavras-chave: Drone. Reconhecimento de imagem. Pouso.

ABSTRACT

LOUREIRO, José Felippe Vedovatto. Autonomous system for landing and recharge of the batteries of an AR.Drone 2.0. 2017. 62f. Trabalho de Conclusão de Curso de bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

A drone is a type of Unmaned Aerial Vehicles (UAV) that most of the times can have four propellers. They can be used in many applications, one of those is to move through places of difficult access. Besides drone´s practicity over other aerial vehicles, its price is way lower compared to large vehicles, which turns them attractive to many activities. Also it offers safety in dangerous situations, like fires or accidents, as it doesn´t need an onboard pilot. In a system with autonomous flight the concern with its landing and recharging of the batteries, which doesn´t last more than a few minutes, arises. Using onboard devices, like it´s cameras and GPS receiver modules, it is possible to implement functions to optimize its capabilities. With the goal to present a solution to such problem, this essay proposes a model which utilizes image recognition to allow a drone to land in an autonomous system. This landing routine based on its image turns flight and landing into autonomous processes, without human intervention.

Keywords: Drone. Image recognition. Landing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGO

LISTA DE SIGLAS E ABREVIATURAS

API	*Application Programing Interface*
CSS	*Cascading Style Sheets*
FAB	Força Aérea Brasileira
GPS	*Global Positioning System*
HAMMER	*Heterogeneous Autonomous Mobile Maritime Expeditionary Robots*
HTML	*Hypertext Markup Language*
JPEG	*Joint Photographic Experts Group*
JSON	*JavaScript Object Notation*
Li-Po	*Lithium-Polymer Battery*
NAVSTAR	(*Navigation System Timing And Ranging*)
NPM	*Node Package Manager*
OpenCV	*Open Source Computer Vision Library*
PDI	Processamento Digital de Imagens
ROS	(*Robot Operating System*)
SVA	Sistema de Visão Artificial
TCC	Trabalho de conclusão de curso
UTFPR	Universidade Tecnológica Federal do Paraná
UAV	*Unmaned Aerial Vehicle*
VTNT	Veículo Terrestres não Tripulado
VANT	Veículos Aéreo não Tripulado
VSNT	Veículo de Superfície não Tripulado
VSbNT	Veículo subaquático não Tripulado

SUMÁRIO

1 INTRODUÇÃO

Com os avanços tecnológicos na área da eletrônica, os veículos aéreos não tripulados (VANTs) tornaram-se cada vez mais acessíveis, seja pelo seu custo cada vez menor, seja pelas opções existentes no mercado. Com a ajuda da computação e programas especializados é possível automatizar seu voo, não sendo mais necessário o seu controle remoto. Dentre os tipos de VANTs encontram-se os drones, que são de menor porte e mais simples. O nome drone origina-se da similaridade do barulho do equipamento com o de um zangão (em inglês *drone*). Neste trabalho, será projetada uma solução para duas das maiores limitações relacionadas ao voo autônomo de drones - o pouso e o carregamento da bateria, sendo hoje necessária a intervenção humana para tais atividades.

1.1 CONSIDERAÇÕES INICIAIS

As utilidades possíveis para o uso de aeronaves não tripuladas vêm sendo cada vez mais abrangentes devido aos avanços industriais. O objetivo é tornar cada vez menores os custos e riscos em operações que envolvem voos não tripulados. Hoje, são várias as áreas que usufruem desta tecnologia, como obtenção de imagens aéreas para mapeamento e sensoriamento, auxílio no resgate de pessoas, monitoramento remoto, logística, e por hobby.

O grupo americano de pesquisa Teal estima que a produção mundial de Veículos Aéreos não Tripulados terá um aumento de $2.6 bilhões de dólares em 2016 para $10.9 bilhões de dólares em 2025, implicando em uma taxa de crescimento anual de 15.4%. Durante a próxima década o mercado totalizará uma arrecadação de $65 bilhões de dólares (PHILIP FINNEGAN et al., 2013).

Varella (2016) afirma que o mercado de VANTs civis já movimentou $5,2 bilhões de dólares no mundo inteiro, e em dez anos, a cifra deve chegar aos $11 bilhões de dólares. A reportagem ainda apresenta diversos exemplos de aplicações que empresas brasileiras estão implementando, como controle de mudas que foram plantadas de maneira incorreta em plantações de eucalipto, monitoramento de áreas de preservação ambiental e até mesmo serviço de entrega de encomendas.

No meio civil, algumas das demandas existentes são monitoração ambiental (poluição, clima ou aplicações científicas), acompanhamento de incêndios florestais, controle de fronteiras, combate ao tráfego de drogas, vigilância aérea, mapeamento, monitoramento de tráfego, agricultura de precisão, ajuda humanitária e buscas e salvamentos (BASTIANELLI

ET AL, 2012). Existe ainda uma demanda muito grande para uso de VANTs no meio militar, seja para reconhecimento, vigilância, avaliação de danos após um ataque, como mecanismo de contingência ou para a comunicação.

Do ponto de vista operacional, um drone é um equipamento que precisa ser manuseado com cuidado devido as suas hélices que giram a altas rotações e seu peso, que pode variar de algumas gramas à vários quilogramas ,para homologação de uso, os VANTs foram divididos em três categorias: até 2,5Kg, de 2,5 Kg a 25Kg e acima de 25Kg (FAB, 2015). Suas hélices giram em alta rotação, o que pode ocasionar sérios ferimentos em caso de acidente. Este fato dificultou por muitos anos sua homologação para uso comercial pela Força Aérea Brasileira (FAB). Em todas as categorias, para homologação, o voo autônomo só pode ocorrer em áreas não habitadas (zona rural e campos, por exemplo).

Com os avanços da engenharia de controle e ciência dos materiais, foi possível desenvolver pequenos VANTs quadrimotores, também conhecidos como microdrones, que podem ser equipados com câmeras, módulo GPS e outros sensores, e operados por uma estação de controle no solo. Na maioria dos casos o envio de comandos é realizado por radiofrequência, tornando o alcance do dispositivo relativamente curto, dependendo da potência do transmissor.

Um quadrimotor que tem ganhado o mercado ultimamente por seu preço reduzido e facilidade de utilização é o Parrot ArDrone 2.0. Produzido na França e vendido mundialmente este drone é bem equipado, com um processador de 32 bits Linux permite o controle do voo pelo smartphone, realiza o monitoramento de seus e sensores e contém sistema de segurança. O Wi-Fi tem alcance de 50 metros e sua bateria possibilita voos de aproximadamente 12 minutos. Pode ser equipado com o módulo FreeFlight - GPS (*Global Positioning System*) para habilitar o voo por GPS, mas somente dentro do alcance do sinal Wi-Fi do drone. Por este motivo, este drone costuma ser utilizado em pesquisas científicas, como apresentado por Lioulemes et al (2014), Aitken et al (2016) e Duarte (2017).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um protótipo de uma base que permita o pouso e a recarga da bateria de um drone de forma autônoma, sem a intervenção humana.

1.2.2 Objetivos Específicos

- Montar protótipo de base para carregamento e o reconhecimento de imagem.
- Implementar o reconhecimento da base na imagem gerada pela câmera do drone e o voo por GPS.
- Efetuar o pouso através de uma rotina de controle de pouso.

1.3 JUSTIFICATIVA

A grande desvantagem do uso de drones ainda é a limitação da autonomia do voo nos veículos à bateria. Drones quadrimotores tem autonomia de voo de 10 a 30 minutos, porém, os drones de baixo custo dificilmente possuem autonomia maior do que 10 minutos, ainda mais quando controlado remotamente, pois existe a necessidade de uma comunicação sem fio constante. Além disso, após o término da bateria, o processo de recarga é lento e necessita de supervisão humana.

Drones vem sendo utilizados em vários cenários diferentes, como no monitoramento de gado de corte e leite, no controle de fronteira, sistemas de emissões de alerta, empresas de segurança, serviços de entrega, entre outros. Em casos onde a área de ação é demasiada grande, o uso de drones quadrimotores torna-se inviável, tanto pela limitação da bateria quanto pelo controle que deve ser remoto.

Alguns trabalhos ainda ressaltam a importância dos VANTs na agricultura de precisão. George, Tiwari e Yadav (2013) afirmam que os VANTs fornecem a melhor plataforma para avaliar a produção agrícola, sendo rápidos e eficientes se comparados às técnicas terrestres tradicionais, permitindo a leitura de dados em praticamente todos os tipos de terreno. Para Brandão, Martins e Soneguetti (2015), os VANTs são dispositivos versáteis que podem ser usados nas mais variadas aplicações agrícolas, como o monitoramento automático de plantios, monitoramento de canais de água e irrigação, detecção e controle de pestes, e até mesmo na pecuária, com a contagem de animais no pasto, entre outras aplicações.

Segundo Bastos (2015), os drones têm ganhado espaço, entre outras áreas, na agricultura e na pecuária. Sua versatilidade vale o investimento, já que podem desempenhar diversas funções em regiões agrícolas e possui custo relativamente baixo, que

varia de acordo com modelo e tecnologias embarcadas.

A eficiência de um drone está associada ao desempenho do seu hardware e de aplicativos de controle. Desta maneira deve-se aproveitar ao máximo o uso de sua bateria, controlando sua necessidade de carregamento. A utilização de câmeras e outros dispositivos variam de acordo com a atividade desejada, entretanto, deve-se garantir a sua capacidade de continuar em funcionamento. Consequentemente, o desenvolvimento de pesquisa e testes nessa área é relevante.

Em um Trabalho de conclusão de curso realizado por dois ex-alunos da UTFPR, Elias Ricardo Vieira e Gustavo da Silva Lopes, foi implementado um sistema para automação do voo de um drone AR Drone 2.0. O sistema, a partir de determinada posição geográfica de destino, controla o voo do drone de forma a garantir a ida e a volta do mesmo (VIEIRA; LOPES, 2016). Entretanto o drone, no momento do retorno, deve ter supervisão humana para pousar e a realizar a devida recarga de sua bateria.

O presente trabalho visa a construção de um protótipo de uma base que, após a confirmação da posição pelo GPS, deverá ser reconhecida pelo drone utilizando-se da imagem, da implementação da rotina de pouso e da realização da subsequente recarga do drone.

1.4 ESTRUTURA DO TRABALHO

O trabalho será divido em cinco capítulos. O primeiro apresenta as considerações iniciais com o contexto sobre o qual o protótipo deve ser desenvolvido, os seus objetivos e a justificativa. O segundo capítulo apresenta o referencial teórico apresentando conceitos sobre drones, GPS e o processamento de imagens. No terceiro capítulo serão detalhadas as ferramentas e as tecnologias utilizada na modelagem e implementação do protótipo. Por último, a conclusão e considerações finais sobre trabalho.

2 REFERENCIAL TEÓRICO

Neste capítulo encontram-se o embasamento teórico sobre Veículos Aéreos Não Tripulados, assim como as duas principais áreas pertinentes ao trabalho, GPS e processamento de imagens.

2.1 VEÍCULOS AÉREOS NÃO TRIPULADOS

Veículos Aéreos Não Tripulados (VANT), do inglês *Unmanned Aerial Vehicles* (UAV), são sistemas aéreos que podem ser remotamente controlados, com propósitos militares ou civis. Também conhecidos como Drones, e usualmente equipados com câmeras, podem contar com diversos outros dispositivos, como módulo GPS, mísseis em casos militares ou boias em casos de salvamento(SOUTHWORTH, 2012).

Por meio dos avanços na engenharia de controle tornou-se possível desenvolver VANTs quadrotores de porte pequeno, que podem ser microdrones equipados com câmeras, sensores e outros dispositivos capazes de serem operados por uma estação de controle em solo (QUARITSCH et al., 2008). A principio foram desenvolvidos para fins militares, mas passaram a ser utilizados em aplicações civis. Dentre as aplicações destacam-se a obtenção de imagens aéreas convencionais, em tempo real, imagens em infravermelho e modelos estereoscópicos para fotogrametria. Alguns drones ainda possibilitam a adição de diferentes dispositivos para auxiliarem suas tarefas, como no caso de módulos GPS.

Uma das configurações possíveis de um VANT é o quadrotor ou quadricóptero (SANTOS; LÓPEZ; MORATA, 2010). Eles podem assumir diferentes configurações dependendo da disposição de seus rotores. Nela existem quatro conjuntos motor/hélice de mesma dimensão, fixados cada um no extremo de dois eixos ortogonais um ao outro. A Figura 1 traz a imagem de um quadricóptero.

Figura 1 – Exemplo de quadricóptero.
Fonte: Costa (2008, p. 10).

Os movimentos de um quadrotor estão associados a variação de rotação de cada um dos rotores. Os motores são dispostos em pares de contrarrotação, de modo que, o torque de reação gerada a partir do primeiro par de motores é exatamente o oposto da reação de torque a partir do segundo par de motores, que estão girando no sentido oposto (COSTA, 2008). A posição do quadrotor é descrita por três coordenadas (c1, c2, c3) do centro de massa em relação ao plano de referência terra.

Sua atitude absoluta é descrita por três ângulos de Euler (ω, θ, Φ). Estes três ângulos são chamados respectivamente ângulo de guinada ($-\pi \leq \omega < \pi$) que é o movimento de rotação sobre seu eixo y, representado na Figura 2, ângulo de inclinação ($-\pi/2 < \theta < \pi/2$) que é o movimento de rotação sobre seu eixo x e ângulo de rolagem ($-\pi/2 < \Phi < \pi/2$) sobre seu eixo z. Estes ângulos permitem os movimentos do quadrotor.

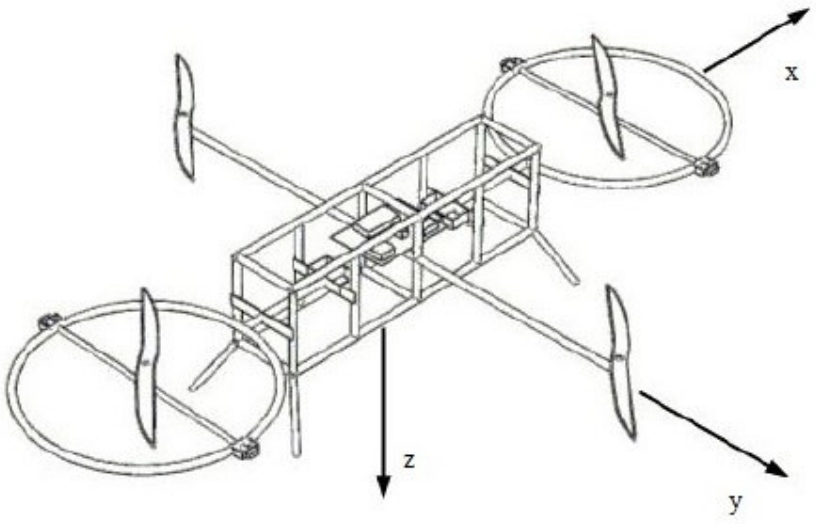

Figura 2 – Eixos de um drone.
Fonte: Adaptado de Costa (2008).

O controle de movimento da aeronave pode ser realizado variando-se a velocidade relativa de cada rotor, para assim alterar o empuxo e o torque produzido por cada um. Aplicando igual pressão para todos os rotores o drone paira ou ajusta sua altitude, mostrado na Figura 3(a), onde todas as setas possuem o mesmo tamanho (mesma força). Para ajustar sua guinada é necessário aplicar um diferencial de rotação, aplicando pressão em um dos pares de rotores, como mostra a Figura 3(b). Por fim, para movimenta-lo em certa direção, aplica-se mais pressão a um rotor de um dos pares e diminui-se a pressão do rotor oposto, como na Figura 3(c).

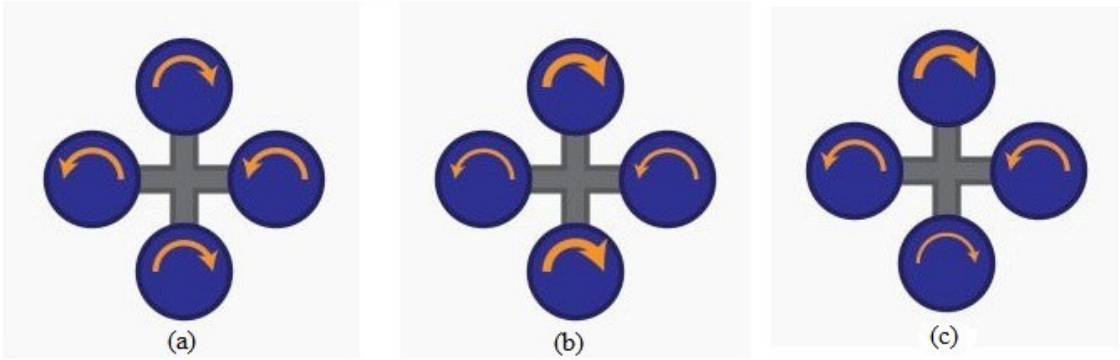

Figura 3 – Controle de movimento
Fonte própria.

O núcleo do sistema de navegação é composto de sensores inerciais de baixo custo, que são continuamente ajudados pelo GPS, bússola magnética e informações da altura barométrica (PEGORARO, 2013). Nos casos nos quais drones são controlados por radiofrequência, estes também possuem módulo receptor e microcontrolador, que realiza os comandos. As imagens capturadas podem ser enviadas em tempo real por meio de radiofrequência para o controlador, ou ser armazenadas no próprio drone em dispositivos de armazenamento não volátil.

O Parrot AR.Drone baseia-se em um quadrotor, no qual quatro motores sem escova são usados para controlar as quatro hélices fixas. Juntos, eles constituem o par de geradores de impulso variáveis. Cada motor é montado em pés de plástico PA66 altamente resistentes carregando a placa de controle sem escova. Cada placa de controle usa seu próprio micro controlador ATMEGA8L de 8 bits e sistema de segurança para desligar o motor no caso de um obstáculo entrar no caminho da hélice. Os quatro pares são conectados usando quatro tubos de fibra de carbono e uma cruz central reforçada com plástico. A estrutura de carbono suporta a cesta principal, onde é transportada a bateria de lítio-polímero (Li-Po) que pesa 80g. As 3 células da bateria Li-Po fornecem 11,1 V e 1000 mAh. A capacidade da bateria possibilita um voo de 10 a 15 minutos de duração. A estrutura principal é fixada em cima de uma espuma, que filtra as vibrações dos motores. Duas proteções podem ser conectados ao sistema. A primeira é uma capa removível que pode ser usado para evitar que as hélices sejam danificadas em caso de incidente, a segunda não fornece proteção para as hélices e é indicado para voos em ambientes abertos.

A placa mãe incorpora um processador Parrot P6 (32bits ARM9-core, 468 MHz), um chip Wi-Fi, uma câmera orientada verticalmente e um conector para a câmera frontal. O sistema operacional em tempo real é baseado em Linux. O processador P6 também é responsável pela aquisição do fluxo de dados das duas câmeras. Para maximizar a qualidade da visão imersiva a câmera frontal usa uma lente diagonal de grande ângulo de 93 graus. A saída é um sinal VGA de resolução 640x480, com um *framerate* de 15 quadros por segundo. A câmera vertical é usada nos algoritmos de navegação, para medir a velocidade do veículo. Ela consiste em uma câmera de lente diagonal com 64 graus, produzindo dados a um *framerate* de 60 quadros por segundo. Um conector USB está incluído para permitir a conexão de complementos como GPS e laser, entre outros.

A placa de navegação usa um microcontrolador PIC de 16 bits que funciona a 40 MHz e serve como uma interface com os sensores. Esses sensores são um acelerômetro de 3 eixos, o giroscópio de 2 eixos, giroscópio vertical de 1 eixo e 2 sensores de ultrassom. Os sensores

de ultrassom são usados para a estimativa de altitude e deslocamentos verticais do VANT. Eles também pode ser utilizados para determinar a profundidade do cenário pela câmera vertical. Os sensores atuam na faixa de 40 kHz de frequência e sua distancia limite de funcionamento é de 6 m. A figura 4 ilustra os componentes de um ArDrone 2.0.

Figura 4 – Componentes do ArDrone 2.0.
Fonte: https://www.ifixit.com/Device/Parrot_AR.Drone.

2.2 GPS

A implementação do programa NAVSTAR, GPS (*Navigation System Timing And Ranging, Global Positioning System*) iniciou-se em 1973. O NAVSTAR é sistema de radio navegação por satélites desenvolvido e controlado pelo Departamento de Defesa dos Estados Unidos da América com o objetivo de ser o principal sistema de navegação das forças armadas americanas. Em 1979 foram lançados quatro satélites, sob a responsabilidade do Departamento de Defesa dos Estados Unidos, Divisão Espacial. Inicialmente projetado para uso militar tornou-se fundamental para diversas áreas de pesquisa e desenvolvimento (HUERTA; MANGIATERRA; NOGUERA, 2005).

GPS é um sistema que tem como objetivo determinar as coordenadas espaciais de acordo com um sistema de referencia mundial. Os pontos podem estar em qualquer lugar do planeta, estáticos ou em movimento, e em qualquer momento que se deseja. Para obtenção das coordenadas o sistema se baseia em, no mínimo, quatro satélites com coordenadas conhecidas. Por meio do cálculo dos sinais emitidos pelos satélites é possível obter as distancias referentes a cada um, fazendo assim, a triangulação da posição. A Figura 5 demonstra este funcionamento.

A geocodificação é o processo computacional de transformação de uma descrição de endereço postal para um local na superfície terrestre (representação espacial em coordenadas numéricas). A geocodificação inversa, por outro lado, converte as coordenadas geográficas inseridas em uma descrição de um local, geralmente o nome de um local ou endereço postal.

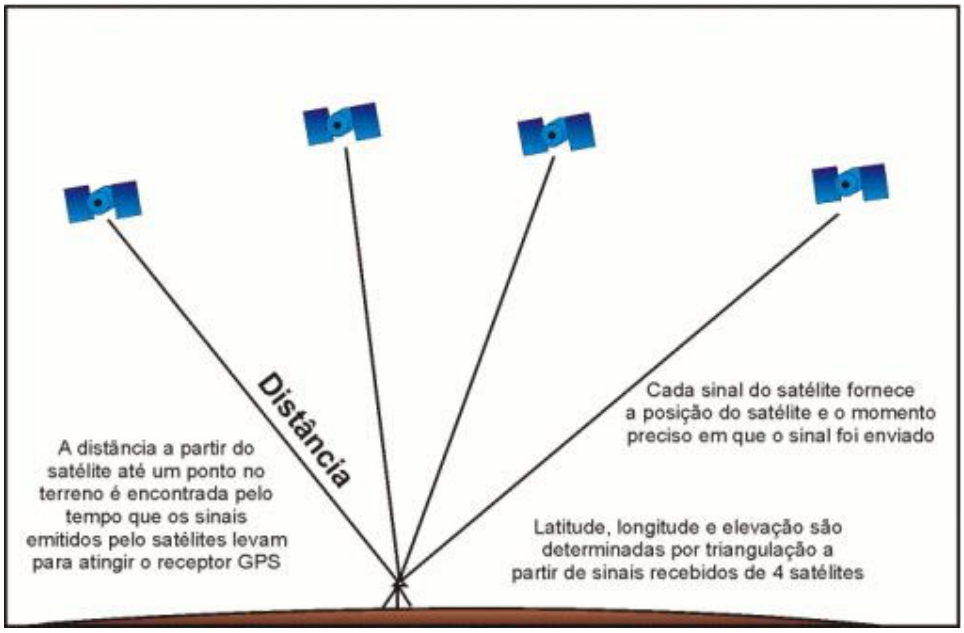

Figura 5 – Funcionamento GPS.
Fonte própria.

2.3 PROCESSAMENTO DE IMAGEM

O Processamento Digital de Imagens (PDI) não é algo trivial, pois engloba um conjunto de tarefas interconectadas (QUEIROZ; GOMES, 2001). O primeiro passo é a captura de uma imagem, que nada mais é do que a absorção da iluminação refletida pelo ambiente por meio de um sistema de aquisição. Após a captura, é necessário aplicar algum método de digitalização sobre esta, para que possa ser representada de forma apropriada para tratamento computacional. Imagens podem ser representadas em duas ou mais dimensões.

Uma imagem monocromática pode ser considerada uma matriz de tamanho *m* por *n*, na qual *m* e *n* são coordenadas espaciais e o valor da célula matricial é proporcional à intensidade luminosa (brilho ou nível de cinza) no ponto considerado. Uma imagem não monocromática, por sua vez, contém três valores associados a célula matricial, que correspondem a intensidade de verde, vermelho e azul naquele ponto. Como computadores só trabalham com informações discretas, é necessário representar imagens como arranjos bidimensionais de pontos. Na Figura 6 tem-se uma representação matricial de uma imagem.

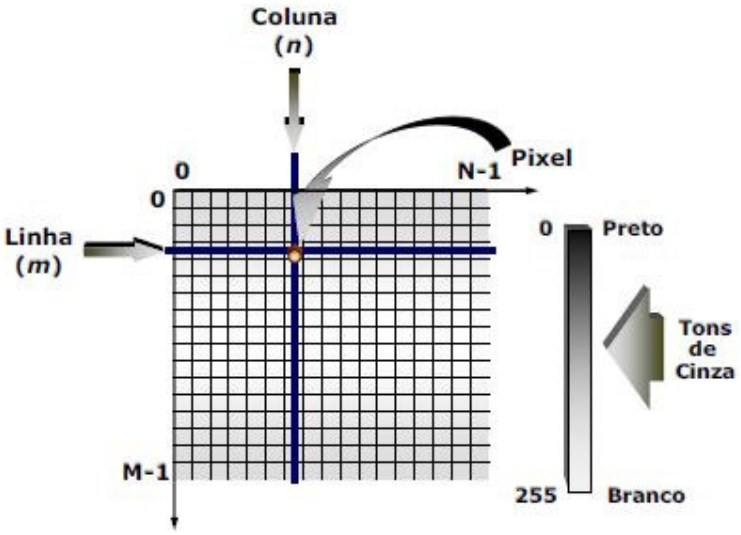

Figura 6 – Representação de uma imagem
Fonte: Queiroz e Gomes (2001, p. 8).

Filho e Neto (1999, p. 9) definem um Sistema de Visão Artificial (SVA) como "um sistema computadorizado capaz de adquirir, processar e interpretar imagens correspondentes a cenas reais". A Figura 7 traz um diagrama de blocos de um SVA onde os blocos são etapas do processamento e interpretação da imagem. A visão computacional é de fato um SVA.

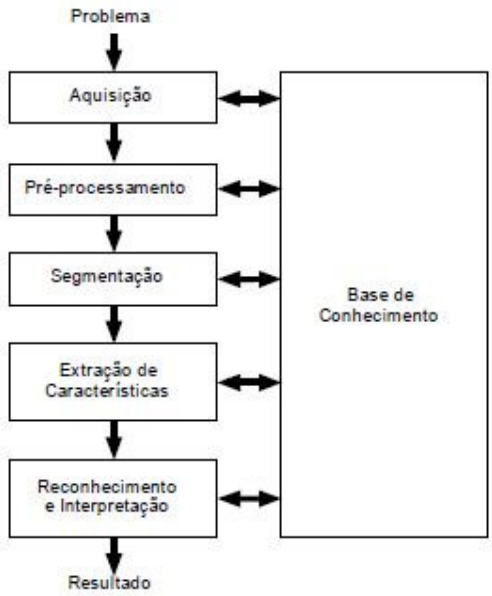

Figura 7 – Um SVA e suas etapas.
Fonte: Filho e Neto (1999, p. 9).

Após a aquisição é necessário preparar a imagem na etapa de pré-processamento. O objetivo do pré-processamento é aprimorar a qualidade da imagem para as etapas subsequentes. As técnicas de pré-processamento envolvem duas categorias principais: métodos que operam no domínio espacial e os que operam no domínio da frequência (ESQUEF; ALBUQUERQUE; ALBUQUERQUE, 2003). As técnicas de processamento no domínio da frequência são filtros que agem sobre o espectro da imagem, enquanto que no domínio espacial são baseadas em filtros que manipulam a imagem. Para realçar determinadas características de uma imagem é comum combinar vários métodos que sejam baseados nestas duas categorias.

Com a imagem tratada é possível realizar a segmentação. A segmentação varia muito em sua metodologia, devendo sempre ser adequada às características e necessidades de cada problema específico, levando em conta os objetivos que se desejam alcançar. A tarefa básica da etapa de segmentação é a de dividir uma imagem em suas unidades significativas, ou seja, nos objetos de interesse que a compõem. Esta tarefa, apesar de simples de descrever, é uma das mais críticas para o sistema (FILHO; NETO, 1999).

Após a divisão de uma imagem nas suas partes de interesse é necessário o reconhecimento das informações. Esta parte do processamento, tendo como apoio uma base de conhecimento previamente estabelecida, vai classificar os objetos a partir de informações encontradas na imagem (ALBUQUERQUE; ALBUQUERQUE, 2000). Esta etapa é

normalmente aplicada após uma fase de segmentação e extração de características da imagem. A extração de características identifica e calcula parâmetros nos objetos segmentados. Um exemplo de parâmetro pertinente a um objeto é o seu perímetro ou sua área.

Todas as tarefas das etapas descritas acima são baseadas na existência de um conhecimento sobre o problema a ser resolvido, uma base de conhecimento. Idealmente, esta base de conhecimento deveria não somente guiar o funcionamento de cada etapa, mas também permitir a realimentação entre elas. Esta integração entre as várias etapas através da base de conhecimento ainda é um objetivo difícil de alcançar. Na área de reconhecimento de objetos, destacam-se os algoritmos e técnicas baseadas em redes neurais e classificadores bayesianos (QUEIROZ; GOMES, 2001).

2.4 ESTADO DA ARTE

Existem diversos drones no mercado com aplicabilidades variadas, podendo ser destinados ao uso recreativo, didático ou militar. Em seguida são destacados projetos que estão contextualizados neste projeto.

Hui et al (2013), traz uma abordagem de cooperação entre um veículo aéreo não tripulado (VANT) e um veículo terrestre não tripulado (VTNT) baseada na visão monocular para o pouso do VANT. Utilizando rastreamento para marcar um alvo no VTNT, o VANT pode acompanhar e aterrisar de forma autônoma no VTNT em movimento. Os laços de controle baseados em controladores PID são empregados para realizar dois níveis de controle: controle de estabilização e controle de posição. O controle de estabilização é executado em uma placa de microcontrolador de bordo com o auxílio de uma unidade de medição de inércia. As imagens capturadas pela câmera a bordo são transmitidas para um controle central terrestre, este presente em um laptop por meio de canais sem fio. A posição relativa do VANT para o veículo terrestre é estimada a partir das imagens recebidas, e não por meio de GPS.

Assim a posição atual estimada do VTNT é enviada para o VANT via Wi-Fi. A abordagem proposta baseada em visão para detectar e localizar o alvo, assim como a altura do VANT, é robusta quando se tem um fundo de imagem muito diversificado. Experimentos práticos mostram que o VANT pode monitorar de forma autônoma o VTNT e realizar o pouso no alvo em movimento. Como as posições lineares absolutas (x, y, z) não podem ser obtidas a partir do microcontrolador de bordo, a navegação com base em imagem é fundamental para voos autônomos em ambientes sem acesso ao GPS. Uma câmera monocular de baixo custo é montada no centro geométrico do VANT, com sua lente para baixo. Um círculo com um raio

de 20 cm foi anexado na pista sobre o VTNT para servir como alvo. A Figura 8 apresenta o drone utilizado no trabalho em questão.

Figura 8 – Dispositivo utilizado no projeto.
Fonte: Hui et al (2013).

Barresi e Allasia (2013) exploram a movimentação de veículos aéreos não tripulados (VANTs) dentro de um aeroporto. Este é baseado no vídeo da câmera frontal fornecido pelo VANT para o piloto remoto, as informações de GPS também são utilizadas. O GPS, porém, introduz um erro sistemático no sistema, o que não permite a completa rodagem do equipamento pela pista de rolagem e o estacionamento automático. O projeto desenvolvido pelos autores apresenta uma nova abordagem onde o processamento de imagem permite uma retificação do sinal do GPS, fazendo uso do reconhecimento automático de sinais e marcas de aeroportos. Os trajetos de decolagem bem como os sinais de localização da pista de rodagem são identificados e correlacionados com a informação do GPS. Várias abordagens haviam feito uso da transformação de Hough para encontrar a pista para aterrissagem ou decolagem, sem cooperação entre a posição e os sistemas ópticos. O objetivo do trabalho é criar uma rotina que automatize o pouso e estacionamento com base nas imagens e no GPS.

Em outro projeto, Cocchioni et al (2014) abordam a cooperação entre dispositivos não tripulados realizando missões em áreas *indoor* / *outdoor*. Neste os autores focam na interação entre Veículos Terrestres não Tripulados (VTNTs) e Veículos Aéreos não Tripulados (VANTs) para ampliar a autonomia de voo de um VANT, por meio de uma plataforma de aterrisagem e recarga. A VTNT atua como uma estação de recarga e hospeda o VANT durante a transição *indoor* / *outdoor* e vice-versa. A plataforma foi projetada com o

objetivo de obter uma aterrisagem robusta. A sincronização e a coordenação da cooperação são gerenciadas por uma estação de controle.

Esta estação foi desenvolvida usando uma ferramenta de software com base na integração de Stateflow, geração automática de código C e ROS (*Robot Operating System*). Todos os componentes de software do VANT, VTNT e da própria estação foram desenvolvidos usando ROS. Os resultados obtidos mostram que o VANT é capaz de pousar sobre a VTNT com alta precisão (<5 cm para os eixos x e y) graças a um algoritmo visual de estimativa de posição. O sistema também funciona na presença de ventos de até 20-25 km/h. A Figura 9 traz o *layout* da estação.

Figura 9 – *Layout* da estação.
Fonte: Cocchioni et al (2014).

Ainda no cenário de interação entre veículos, Min-Fan et al (2014), apresentam um método de cooperação entre dois dispositivos móveis aéreos, um em alta altitude e outro em baixa altitude, para autonomizar a navegação e o pouso.

No pouso autônomo baseado em visão computacional, a precisão do sinal de GPS e a eficiência do algoritmo de rastreamento e detecção de alvo afetam o desempenho do sistema de pouso autônomo. Desta maneira, utilizando a visão abrangente e a alta flexibilidade de um dispositivo móvel aéreo de alta altitude, é possível controlar um dispositivo móvel aéreo de baixa altitude, para que este execute os procedimentos de pouso corretamente. O controlador

de voo consegue rastrear o alvo e controlar o dispositivo em tempo real. Isto é possível por meio de um sistema de controle de alto nível, utilizando lógica *fuzzy* e redes neurais para calcular o posicionamento e realizar as manobras do dispositivo de baixa altitude. A Figura 10 representa um modelo do sistema.

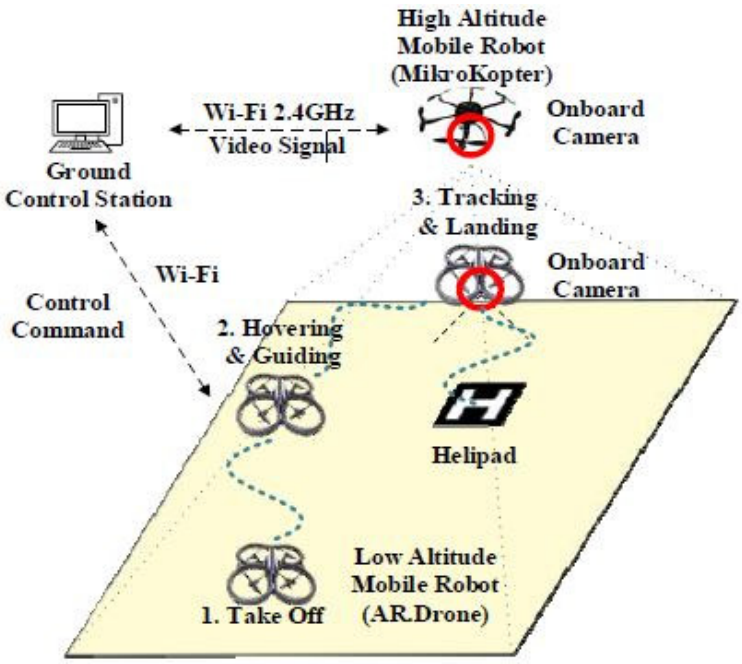

Figura 10 – Modelo de cooperação entre dois dispositivos móveis aéreos.
Fonte: Min-Fan et al (2014).

Maini e Sujit (2015) afirmam que a vigilância aérea e o mapeamento são as principais áreas para o desenvolvimento de aplicações que envolvam veículos aéreos não tripulados (VANTs) pequenos. Quando se trata do mapeamento de áreas de grande porte, por exemplo, podemos ter missões que durem um tempo mais prolongado. No entanto, a duração da missão é limitada pela capacidade de combustível do VANT. Isso exige o reabastecimento do VANT para o sucesso e a conclusão da missão. Os pontos de interesse da área a ser mapeada podem não estar acessíveis a partir de uma única estação de reabastecimento e, portanto, são necessárias múltiplas estações de reabastecimento. Além disso, as estações de reabastecimento não podem ser colocadas em qualquer lugar da região, devido a restrições do terreno. Por fim, as rotas das missões dos VANTS devem ser planejadas de acordo com a disposição das estações de reabastecimento.

Portanto, o planejamento das rotas dos VANTS e a disposição das estações de reabastecimento são problemas que andam juntos, e obter soluções ótimas torna-se difícil. Os

autores desenvolveram uma estratégia gananciosa de coordenação entre a estação de reabastecimento móvel e o VANT, com a ajuda de uma plataforma de simulação criada com auxilio do software matemático Matlab para testar e validar a estratégia. Os testes de campo foram conduzidos utilizando um quad-rotor.

Uma combinação bem sucedida de sistemas autônomos não tripulados heterogêneos para futuras missões em alto mar está se tornando uma realidade. Djapic, Prijic e Bogart (2015) são responsáveis pelo projeto *Heterogeneous Autonomous Mobile Maritime Expeditionary Robots* (HAMMER), que tem por objetivo integrar operacionalmente uma plataforma de superfície autônoma com três tipos diferentes de veículos não tripulados, sendo eles aéreo, de superfície e submarino. O sistema HAMMER consiste em vários veículos marítimos trabalhando em conjunto, dentre eles o veículo de superfície não tripulado (VSNT) atua como nó central e principal mecanismo de transporte, pode ser usado para transportar os veículos aéreos não tripulados (VANT) e veículos subaquáticos não tripulados (VSbNT). O sistema foi projetado para ser modular e pode facilmente ser ampliado se necessário, tem autonomia para navegar várias centenas de milhas.

Para a implementação do VSNT foi utilizado um catamarã de 16 pés (área de aterrisagem de 70x40 polegadas). Utilizando o veiculo de superfície para base de operações de um VANT, é possível garantir a interoperabilidade, coordenação e cooperação de robôs marinhos móveis autônomos em ambientes onde o acesso á GPS é negado. As três principais áreas de pesquisa relacionadas ao projeto são processamento de imagem, estimativa de estado e controle cooperativo autônomo. Devido aos desafios do ambiente marítimo, tanto software quanto hardware utilizados devem ser estudados para fornecer o nível de flexibilidade e recursos necessários para se obter uma aterrissagem eficiente e robusta. A comunicação também deve ser confiável, sendo desenvolvida para permitir o compartilhamento eficiente de dados e mensagens de controle entre a superfície e as plataformas aéreas.

3 MATERIAIS E MÉTODO

3.1 MATERIAIS

No quadro 1 estão os materiais que serão utilizados para o desenvolvimento do projeto.

Ferramentas	Versão	Disponível em	Aplicação no projeto
ArDrone	2.0	http://www.parrot.com/usa/products/ardrone-2/	Drone utilizado no projeto
Ar-Drone	0.3.3	http://.github.io/felixge	Biblioteca para auxiliar no controle autônomo do drone
Angular.JS		https://angularjs.org/	Framework para desenvolvimento do projeto
Bower	1.3.9	https://bower.io/	Gerenciador de pacotes
Express	4.13.1	http://expressjs.com/pt-br/	Framework para desenvolvimento web com Node.JS
JSON		http://www.json.org/json-pt.html	Formato de arquivo para transferência de dados entre servidor e cliente
Node.JS	4.4.1	https://nodejs.org/en/	Compilador JavaScript
NPM	2.4.20	https://www.npmjs.com/	Gerenciador de bibliotecas para Node.JS
OpenCV	2.4.13	http://opencv.org/	Biblioteca para visão computacional
Parrot Flight Recorder		https://www.parrot.com/fr/accessoires/drones/flight-recorder-pour-parrot-ardrone-20	Modulo GPS para ArDrone fabricado pela empresa Parrot
Visual Paradigm	Community Edition	https://www.visual-paradigm.com/	Desenvolvimento de diagramas

Quadro 1 – Ferramentas utilizadas

A seguir, um resumo sobre cada ferramento utilizada:

- **ArDrone**: este é um drone fabricado pela empresa francesa Parrot, tem estrutura de fibra de carbono, pesando apenas 380 gramas, com a autonomia da bateria de aproximadamente 12 minutos. Seu controle é feito por meio de um aplicativo disponibilizado pela empresa, o qual possui versão para dispositivos móveis ou computadores *desktop* e utiliza Wi-Fi para a comunicação. O ArDrone possui 4 motores de 14,5W.
- **Ar-Drone**: é uma biblioteca para Node.JS, que auxilia no voo do ArDrone. Por meio de funções desta, é possível determinar o caminho que o drone deve seguir, por exemplo.
- **Angular.JS:** é um *framework* MVC *client-side* que trabalha com tecnologias já conhecidas como *HyperText Markup Language* (HTML), *Cascading Style*

Sheet (CSS) e JavaScript. Será utilizado para implementar o cliente do sistema.

- **Bower**: é um gerenciador de pacotes para *web* que realiza grande parte das tarefas que seriam realizadas manualmente, como por exemplo, gerência das dependências da aplicação. Utilizado pelo servidor para gerenciar as dependências.
- **Express:** possibilita desenvolver uma aplicação *web* ou uma *Application Programming Interface (*API). Express é um *framework* web para Node.JS. Será utilizado para a criação da interface *web*.
- ***JavaScript Object Notation***: De acordo com o site oficial, JSON é uma formatação leve para intercâmbio de dados. É baseado em JavaScript e utilizado tanto pelo servidor quanto pelo cliente, para troca de dados (http://www.json.org/json-pt.html).
- **Node.JS:** é um interpretador JavaScript para trabalhar com o servidor. É altamente escalável e pode ser programando diretamente com diversos protocolos de rede e Internet, ou utilizar bibliotecas que acessam diversos recursos do sistema operacional. O Node.JS é orientado a eventos de entrada e saída. É usado em conjunto com o Express para desenvolver o servidor.
- **Node Package Manager:** NPM é o gerenciador de pacotes do Node.JS. Ele se tornou o gerenciador padrão e foi integrado ao instalador do Node.JS. Faz parte do desenvolvimento da comunicação servidor-cliente. Fornece amplo suporte a comunidade desenvolvedora.
- **OpenCV**: é uma biblioteca aberta grátis para o uso acadêmico e comercial. Possui interfaces C++, C, Python e Java e suporta Windows, Linux, Mac OS, iOS e Android. O OpenCV foi projetado para eficiência computacional e com forte foco em aplicações em tempo real. Escrito em C/C++ otimizado, a biblioteca pode aproveitar o processamento multi-core. O uso da biblioteca abrange diversas áreas, incluindo arte interativa, inspeção de minas, montagem de mapas na web ou através de robótica avançada. Embora a biblioteca OpenCV não seja desenvolvida para JavaScrip existem adaptações criadas para torná-la disponível. Dentre estas a utilizada neste trabalho é opencv-peterbraden.
- **Parrot Flight Recorder:** é um módulo GPS fabricado pela mesma empresa que fabrica o drone. O módulo ainda possui função de retorno, que ao ser acionada,

conduz o drone ao local de onde decolou.

3.2 MÉTODO

Este trabalho surge da necessidade de um sistema existente e para garantir compatibilidade os elementos utilizados são predefinidos. O drone Ar Drone 2.0 foi escolhido para o projeto original por existir uma API de código aberto disponibilizada para desenvolvedores. É necessário então um servidor para acessar o drone e fazer o processamento das rotinas e uma interface gráfica para exibir o resultado do tratamento de imagem.

Para o desenvolvimento do servidor foi escolhida a linguagem Node.JS como servidor e Angular.JS para o *frontend*. O motivo para a escolha dessas ferramentas é pela existência de uma biblioteca para auxiliar na automação do voo do ArDrone disponível para Node.JS. Como Node.JS e Angular.JS tem grande compatibilidade com JSON e ambas utilizam JavaScript para desenvolvimento, foi criado um servidor que envia informações via JSON ao cliente, assim como o cliente, envia informações ao servidor utilizando JSON. O cliente e os acesso às informações foram com as tecnologias HTML e Angular.JS, pois são linguagens amplamente utilizadas na internet e são altamente compatíveis.

Os testes realizados foram os de compatibilidade e comunicação entre servidor e o Drone. Foram testadas as interfaces de captação e de processamento de dados por meio de testes manuais, além de serem verificados os dados obrigatórios e sua integridade. O reconhecimento de imagem foi testado com diferentes imagens para garantir o funcionamento do algoritmo.

O OpenCV é uma biblioteca de visão computacional e aprendizagem de máquina de código aberto. O OpenCV foi construído para fornecer uma infraestrutura comum para aplicações com visão computacional e para acelerar a percepção das máquina nos produtos comerciais. Sendo um produto com licença aberta, o OpenCV facilita a utilização e modificação do código pelas empresas.

A biblioteca possui mais de 2500 algoritmos otimizados, que inclui um conjunto abrangente de algoritmos de visão computacional e de aprendizagem de máquina. Esses algoritmos podem ser usados para detectar e reconhecer rostos, identificar objetos, rastrear objetos em movimento, extrair modelos de objetos 3D, entre outros.

4 RESULTADO

4.1 ESCOPO DO SISTEMA

O trabalho tem por objetivo implementar a automação do pouso e recarregamento de um ArDrone 2.0, controlando sua aproximação por meio do GPS e seu pouso por meio da imagem gerada pela câmera. Assim que o drone estiver em contato com a base, o descarregamento é iniciado. A arquitetura é apresentada na Figura 11. A comunicação entre o servidor e o Drone é feita por meio de conexão Wi-Fi, fornecido pelo próprio Drone. O servidor é responsável pela comunicação, persistência de dados e envio de informações.

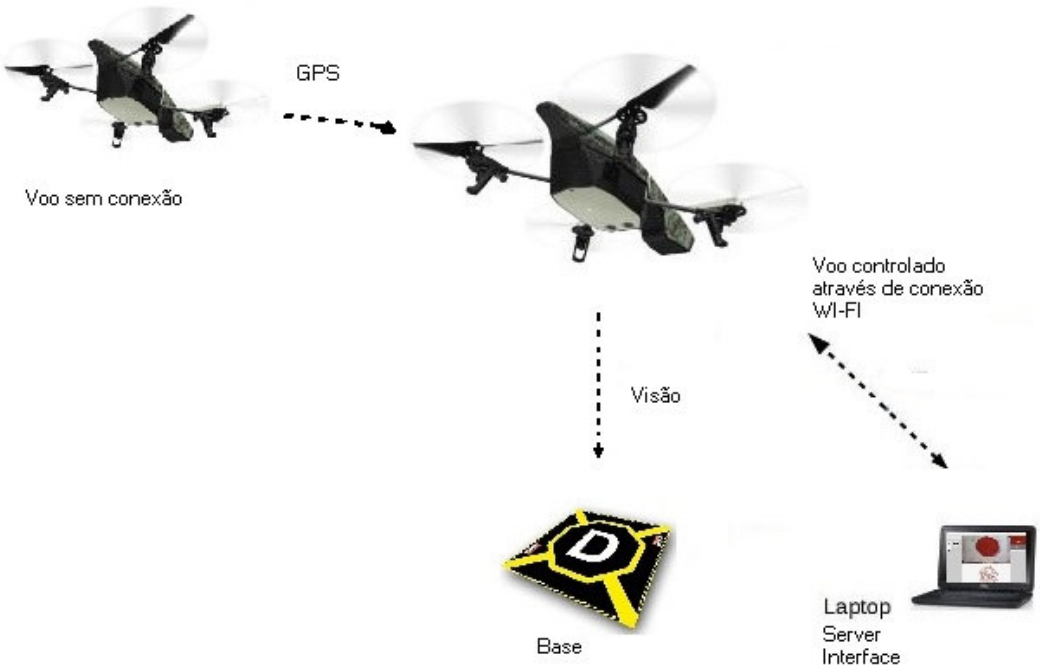

Figura 11 – Representação do sistema.
Fonte: https://www.ifixit.com/Device/Parrot_AR.Drone.

Abaixo temos um resumo do sistema em geral:

- **GPS**: o drone ao receber sinal de bateria baixa deve se direcionar a base mais próxima, isso é feito utilizando um sinal GPS. Ao atingir o destino o drone deve então iniciar a rotina de pouso. O módulo utilizado é o Parrot AR.Drone 2.0 Flight Recorder.
- **Visão**: o drone fornece a imagem para o servidor que deve fazer o

processamento da imagem e enviar a ação de controle necessária. A imagem gerada pelo drone é no formato JPEG, com dimensões de 640 x 360 pixels.

- **Base**: a base deve ser registrada no sistema com as coordenadas geográficas onde ela for instalada. É indicado que a base não seja instalada em local com muita sombra, visto que a diferença no brilho da imagem pode afetar o reconhecimento.
- **Wi-Fi:** drone e servidor comunicam-se através de conexão wi-fi. A conexão tem um limite de distância de aproximadamente 50m. Essa conexão não é segura, podendo ser alvo de ataques.
- **Servidor:** o servidor é uma interface web e é o responsável pelo reconhecimento de imagem e envio das ações de controle do drone. É desenvolvido utilizando Express e Node.JS e implementado em uma máquina Linux.

4.2 MODELAGEM DO SISTEMA

O objetivo da aplicação é auxiliar o drone no seu pouso quando houver a necessidade de carregamento de suas baterias. Para isso o drone deve, no momento em que detectar sinal de bateria baixa, se direcionar a coordenada geoestacionária da base mais próxima. Ao atingir seu alvo o reconhecimento de imagem é acionado e procura pela base começa. Após o reconhecimento o drone deve alinhar-se com a base e efetuar o pouso. O carregamento acontece quando os pés do drone entram em contato com a base metálica dos cones.

O módulo GPS do equipamento tem precisão de 2 metros, ou seja, dado um determinado ponto geoestacionário, o drone o reconhecerá estando a 2 metros ou menos do ponto (D), demonstrado na Figura 12. O ângulo da câmera inferior é de 63 graus (β). Tendo em mãos estas duas informações e utilizando trigonometria é possível descobrir a altura (h) que o drone deve voar para obter visão da base, que é de, no mínimo, 3,5 metros.

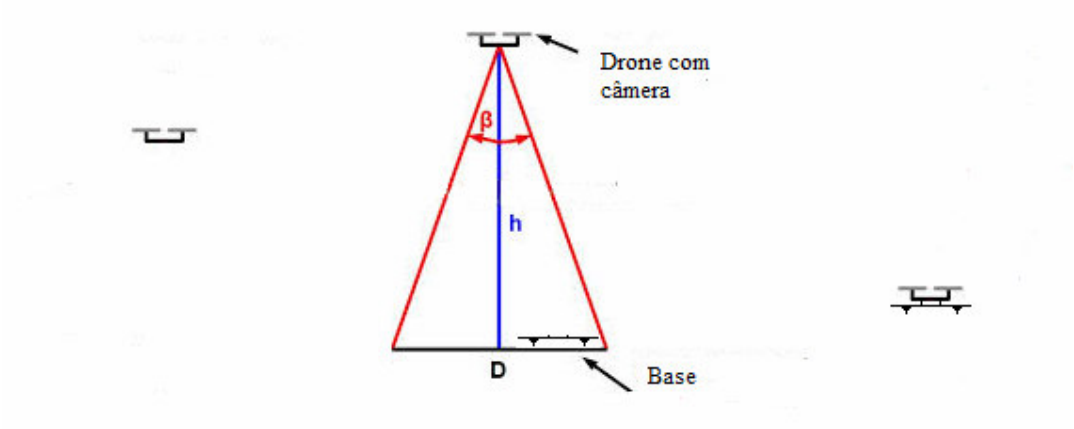

Figura 12 – Demonstração do processo de pouso.
Fonte própria.

Para o pouso do drone é necessária uma base que seja facilmente reconhecida e que facilite o pouso do mesmo. Desta forma, a base foi projetada como uma circunferência, com sinais para identificar a direção que o drone deve pousar (a frente do drone esteja voltada para o lado correspondente na base). A Figura 13 é apresentado o projeto da base.

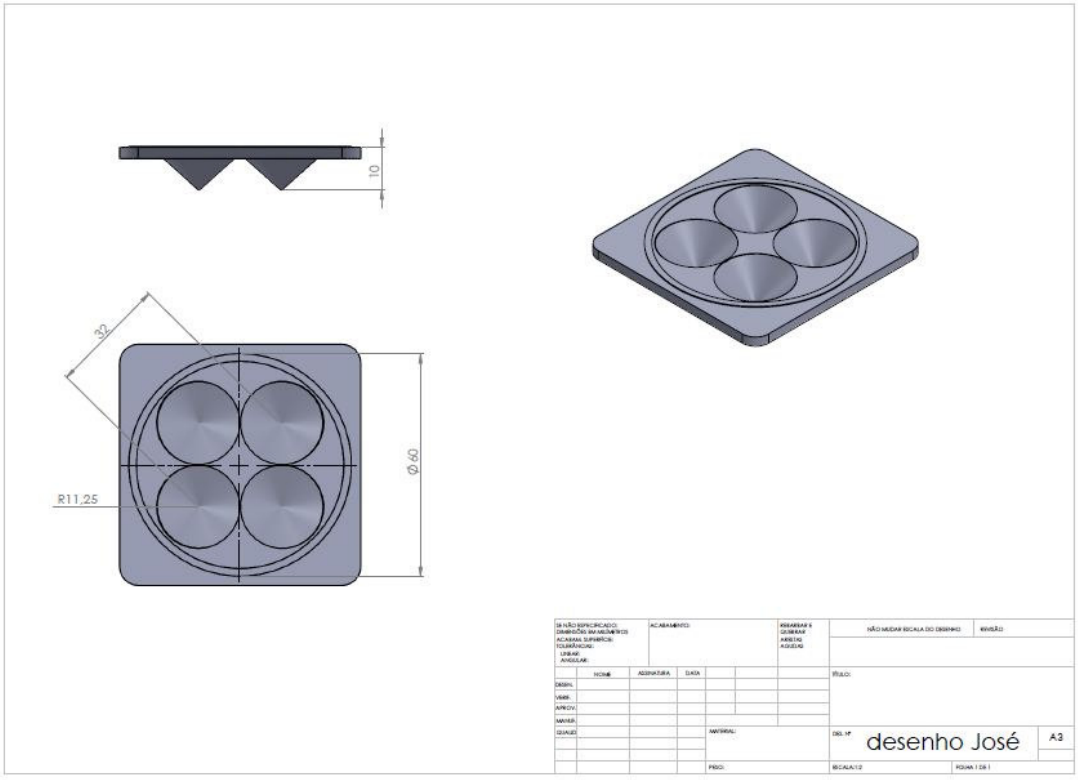

Figura 13 – Projeto da base.
Fonte própria.

Objetos geométricos são relativamente fáceis de serem identificados, devido a suas características específicas, como perímetro, área e raio. Mesmo baseando-se na imagem a precisão necessária para que haja o contato entre os pólos deve ser alta. Desta maneira a base possui 4 orifícios em forma de cone invertido, permitindo que quando o drone pousa, mesmo com um pequeno erro de posicionamento, o contato conduzindo por gravidade a posição correta para o carregamento. Assim, cada pé do AR.Drone se encaixará em um destes orifícios dos cones invertidos, conforme Figura 14. O cone possui 9 cm de raio e 12 cm de altura.

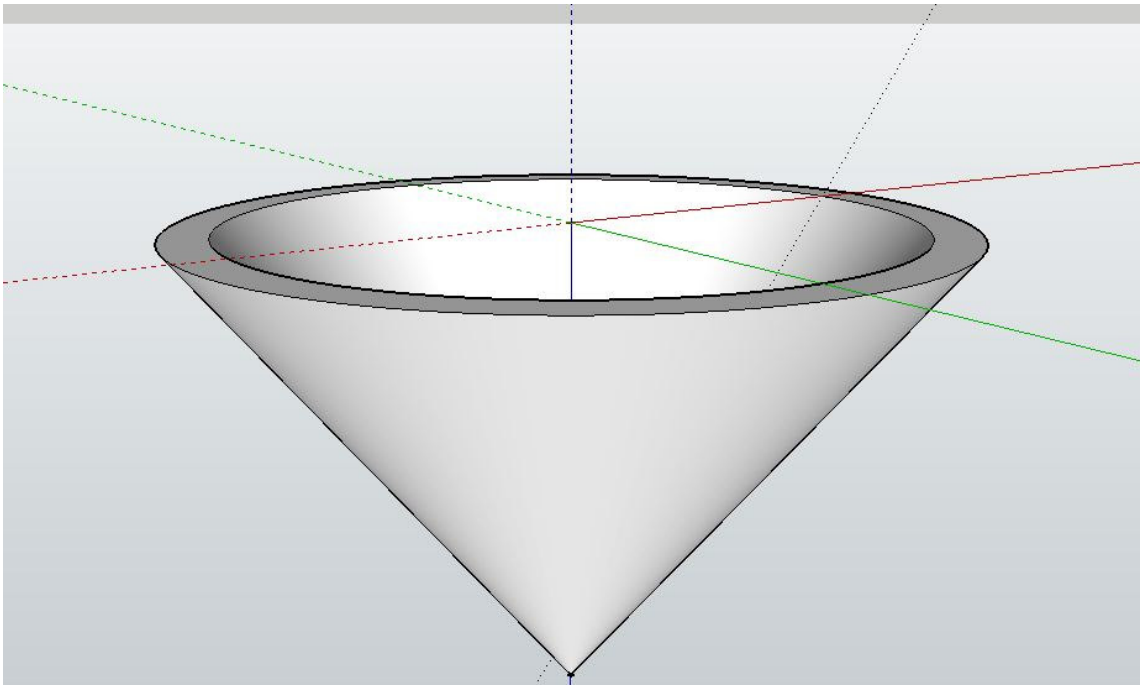

Figura 14 – Projeto do cone.
Fonte própria.

O carregamento da bateria do drone foi projetado por meio de um adaptador que deve ser ligado a uma fonte de energia. Para conectar o carregador da bateria foi desenvolvida uma adaptação, que inclui o conector do carregador e um cabo para estender o acesso a energia e é conectada a base. Os pés do drone foram adaptados com pontas metálicas, conectadas as três células da bateria, para que no momento do pouso, fechem contato com o ambiente de carregamento. A ponta do cone contém uma base metálica para garantir o contato com os polos da bateria. A Figura 15 demonstra as adaptações.

Figura 15 – Representação da extensão e dos contatos metálicos.
Fonte própria.

Por motivos de compatibilidade com o ambiente de controle do ArDrone, o servidor foi desenvolvido em JavaScript, utilizando Node.JS e Angular.JS. Estas ferramentas facilitam o desenvolvimento da atividade devido a biblioteca Ar-Drone, disponível para Node.JS, que contém varias funções para manipulação do drone.

Para a aplicação *web,* se faz necessário um servidor *web* para as trocas de mensagens e comunicação com o cliente. Esta aplicação web resume-se a uma tela para exibir a imagem recebida do drone, assim como a imagem resultado do tratamento, e alguns botões para controlar o drone. Esta aplicação também é responsável por enviar os comandos de voo, fazer o tratamento de imagem e reconhecer a base de pouso do Drone. Estas informações são enviadas do servidor para o Drone a partir da comunicação Wi-Fi. O drone pode ser considerado um roteador Wi-Fi, que aceita conexões de diversos dispositivos. A Figura 16 demonstra a comunicação entre drone e servidor.

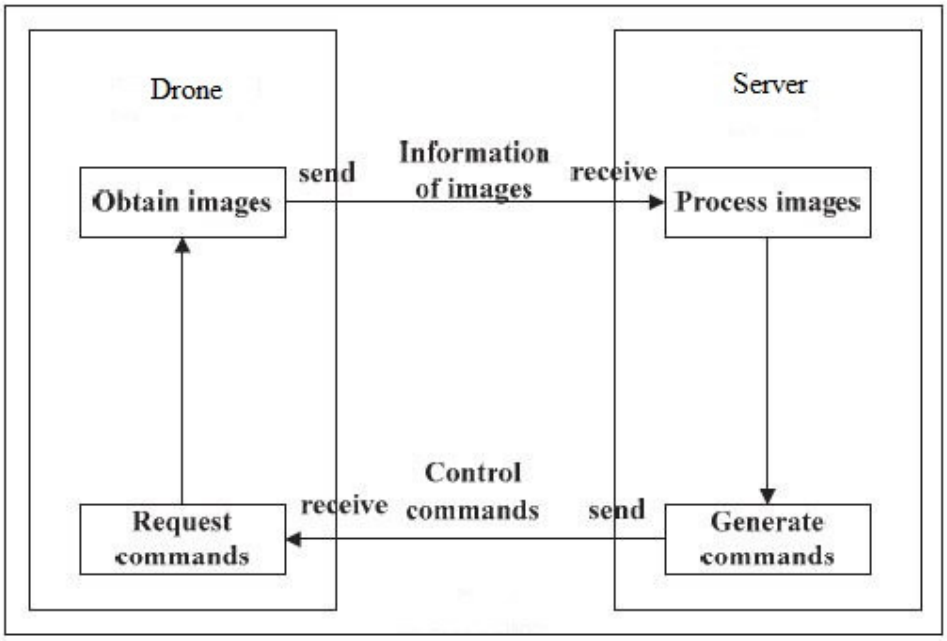

Figura 16 – Representação da comunicação entre drone e servidor.
Fonte própria.

Como Node.JS e Angular.JS tem grande compatibilidade com JSON e ambas utilizam JavaScript para desenvolvimento, cliente e servidor foram criados de maneira que se comuniquem via JSON. O cliente e os acessos às informações foram codificados utilizando as tecnologias HTML e Angular.JS, pois são linguagens amplamente utilizadas na internet e são compatíveis umas com as outras. O drone captura suas imagens em formato JPEG e o tratamento destas é feito pelo servidor utilizando a biblioteca aberta OpenCV.

Para identificar a base de recarga foi definido um algoritmo baseado em detecção de cores. A Figura 17 apresenta o fluxo do algoritmo.

Figura 17 –Fluxograma de detecção de cores.
Fonte própria.

O espaço de cores HSV é composto por 3 matrizes, matriz de matizes, de saturação e de valores. No OpenCV, o intervalo de valores para os campos das matrizes são, respectivamente, 0-360, 0-255 e 0-255. Matiz representa a cor, saturação representa a quantidade com que a respectiva cor é misturada com branco e valor representa a quantidade com que a cor é misturada com preto (Szeliski, 2010). Desta maneira, quando transforma-se a imagem para o espaço HSV, uma cor acaba sendo definida por um único parâmetro, ao invés de três como é no RGB, facilitando assim o processamento.

A cor vermelha foi escolhida para ser utilizada como cor de referencia para a aproximação do drone na base, e tem valores entre 170-180, 160-255, 60-255, respectivamente. Aqui, a matiz é definida exclusivamente para essa distribuição de cores. Já saturação e valor podem variar de acordo com a condição de iluminação do ambiente. Desta maneira é necessário realizar duas buscas, uma para o limite inferior dos valores e outra para

o limite superior. Por fim, somando-se as duas imagens tem-se uma terceira imagem como resultado, que contém somente os pixels desejados.

Assim é possível retirar informações relevantes da imagem resultado. Como a imagem contém somente os pixels no qual a cor vermelha foi reconhecida, utiliza-se uma função para descobrir o centro da área presente. Com isso têm-se os valores de X e Y necessários para controlar a centralização do drone em relação à base.

4.3 IMPLEMENTAÇÃO

4.3.1 Aplicação

A implementação do módulo controlador da interface foi baseada no conceito de *Single Page Application* (SPA), que é uma aplicação web ou site que se encaixa em uma única página da web, utilizando JavaScript e Angular.JS. Para simplificar o desenvolvimento uma página modelo foi utilizada chamada “index.html”, que se encontra na Listagem 2 presente nos apêndices. Essa página contém os itens de controle do sistema e também faz a importação dos *scripts* JavaScript utilizados e arquivos CSS. O servidor foi construído em Node.JS, o *frontend* utilizando Angular.JS e a interface da página foi feita com HTML.

A aplicação consiste de uma única pagina, contendo um medidor de bateria, três botões, um para decolar, um para pousar e o ultimo para trocar a câmera que esta sendo utilizada, dois campos de imagem, uma exibindo a imagem original recebida do drone e a outra a imagem já tratada, e por ultimo uma interface que permite a escolha da cor a ser procurada. A Figura 18 exibe a tela da aplicação.

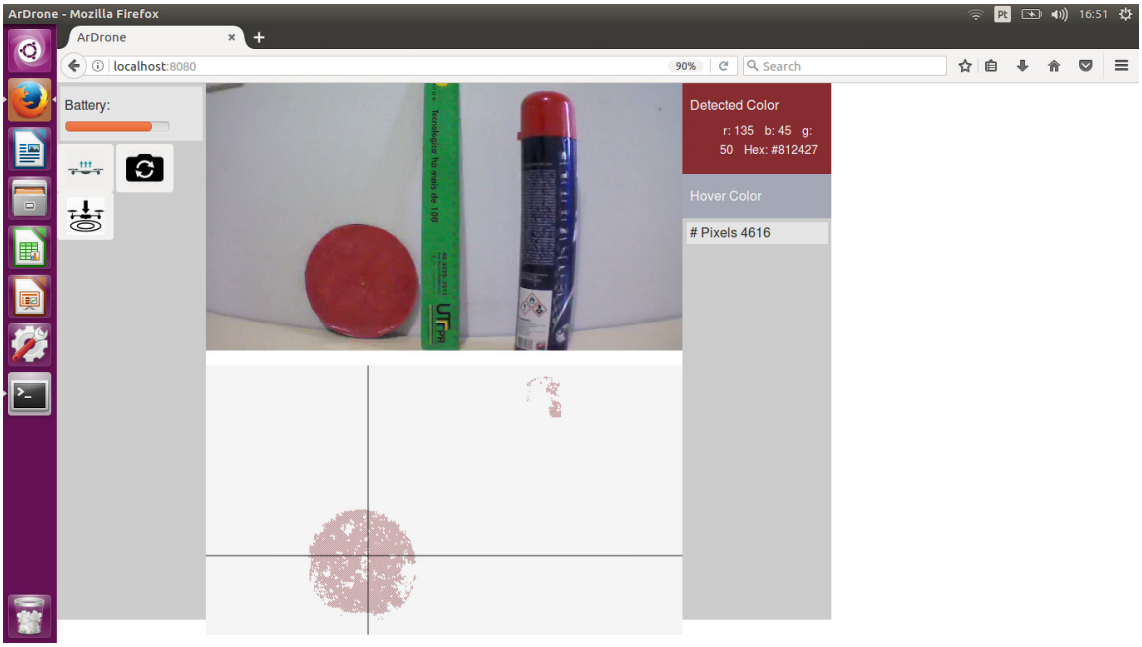

Figura 18 –Tela inicial da aplicação.
Fonte própria.

O código fonte da aplicação é dividida em dois arquivos JavaScript, o app.js e o main.js. O app.js consiste no arquivo de inicialização do sistema, presente nos apêndices como Listagem 3. Ele é responsável por fazer a configuração inicial do drone, iniciar o servidor e controlar o envio de comandos ao drone. A biblioteca Ar-Drone, implementada no servidor faz o envio dos comandos necessários para que o drone realize a ação selecionada. Esta biblioteca não tem suporte para voo por GPS e portanto o voo não será implementado. Os dados de GPS porém podem ser obtidos do drone, para confirmar a posição correta. Esses comandos são enviados via Wi-Fi, fornecido pelo próprio drone no qual o notebook deve estar conectado. O servidor faz uso de Web Sockets para se conectar ao drone, que é uma tecnologia de comunicação cliente-servidor bidirecional em tempo real.

A parte de código referente às funções e o controlador da página HTML são encontrados no main.js. Inicialmente tem-se o controlador da aplicação que foi instanciada no arquivo HTML. Este controlador tem por finalidade incrementar a funcionalidade do Angular.js, recebe como parâmetro o escopo da página e executa uma função em *closure*. O *closure* é uma forma de funções internas (funções definidas dentro de outras funções) acessarem as variáveis da função externa. O main.js encontra-se nos apêndices, na Listagem 4.

O controlador inicia salvando informações que são enviadas pelo NavData do ArDrone. O NavData é responsável por enviar todas as informações dos sensores e câmeras

do drone. O controlador entra então em um laço, onde as funções de leitura e tratamento são chamadas.

Após o laço principal encontram-se as funções necessárias para realizar o tratamento e o controle de voo. Temos as funções de followBottom e followFront, que decidem a ação a ser tomada dependendo de dois parâmetros, X e Y, que são os valores do centro da área reconhecida no tratamento de imagem. A função de detecção de cor será comentada mais adiante. As funções adicionais auxiliam na atualização das telas e variáveis do sistema.

Temos em seguida a função WsClient que é responsável por enviar os comandos de voo para o servidor. Os protótipos de funções e por fim os *listeners*, responsáveis por atualizar a interface da aplicação.

O arquivo package.json é o ponto de partida de qualquer projeto NodeJS. Ele é responsável por descrever o projeto, informar as *engines* (versão do node e do npm), url do repositório, versão do projeto, dependências de produção e de desenvolvimento dentre outras coisas. O nome e a versão juntos formam um identificador que se supõe ser completamente exclusivo e é apresentado na Listagem 5 dos apêndices.

4.3.2 Reconhecimento de Imagem

A função responsável pelo tratamento de imagem utiliza algoritmos baseados em funções de OpenCV, visto que o projeto OpenCV não é desenvolvido para linguagem JavaScript. Isto acaba prolongando o desenvolvimento do algoritmo em geral, visto que não existe documentação. A Listagem 1 traz o algoritmo utilizado no projeto.

```
function detectColor(){

        b = frameBuffer;
        count = 0;
        var xSum = 0;
        var ySum = 0;

        lower_hsv_threshold = [0, 100, 100];
        upper_hsv_threshold = [10, 255, 255];
        //threshold cor vermelha
        lower_hsv_threshold2 = [165, 100, 100];
        upper_hsv_threshold2 = [179, 255, 255];

        cv.readImage(ns.getImageData(b), function (err, img) {
            if (err) throw err;

            const width = img.width();
            const height = img.height();

            let imgG = img.copy();
```

```
            let imgH = img.copy();
            let imgH2 = img.copy();

            //imgH.medianBlur(3);
            //imgH2.medianBlur(3);

            //HSV
            imgH.convertHSVscale();
            imgH2.convertHSVscale();
            imgH.inRange(lower_hsv_threshold, upper_hsv_threshold);
            imgH2.inRange(lower_hsv_threshold2, upper_hsv_threshold2);

            imgH.save('h.png');
            imgH2.save('h2.png');

            var result = new cv.Matrix(img.width(), img.height());
            result.addWeighted(imgH, 0.7, imgH2, 0.9);
            result.save('frameHSV.png');

            //contours
            var big = new cv.Matrix(width, height);
            big = result.copy();
            var im_canny = result.copy();
            im_canny.canny(lowThresh, highThresh);
            im_canny.dilate(nIters);

            var contours = im_canny.findContours();
            const lineType = 8;
            const maxLevel = 0;
            const thickness = 1;

            for(i = 0; i < contours.size(); i++) {
                  if(contours.area(i) > maxArea) {
                        var moments = contours.moments(i);
                        var cgx = Math.round(moments.m10 / moments.m00);
                        var cgy = Math.round(moments.m01 / moments.m00);
                        big.drawContour(contours,  i,  GREEN,  thickness,
lineType, maxLevel, [0, 0]);
                        big.line([cgx - 5, cgy], [cgx + 5, cgy], RED);
                        big.line([cgx, cgy - 5], [cgx, cgy + 5], RED);
                  }
            }
            centerX = cgx;
            centerY = cgy;
            detected = {x: cgx, y: cgy};

            big.save('frameCON.png');
            if (width < 1 || height < 1) {
                  throw new Error('Image has no size');
            }
}
```

Listagem 1 – detectColor().

Esta função inicia lendo a imagem do drone, que é recebida no momento da chamada da função. Duas cópias são feitas da imagem original para iniciar o tratamento. O primeiro passo é transformar as imagens para o espaço de cores HSV. Após isso a função

img.inRange() do OpenCV procura por todos os pixels na imagem que estejam dentro do limite estipulado para a cor desejada. O motivo de criarmos duas cópias é que uma ira ser tratada no limite superior do estipulado e a outra no limite inferior. Somam-se os resultados e temos a imagem resultado. Para identificarmos o centro da área detectada é utilizada a função img.findContours(), que procura na imagem pelos contornos presentes. Como na imagem resultado temos somente a área desejada, pois a função de contorno elimina possíveis ruídos, as coordenadas do centro da área serão a referencia de controle para o voo, considerando o quão próximo as coordenadas estão do centro da imagem. Assume-se que o centro da imagem gerada pelo drone é seu centro de rotação. O resultado de um teste de reconhecimento de cor vermelha é presente na Imagem 19.

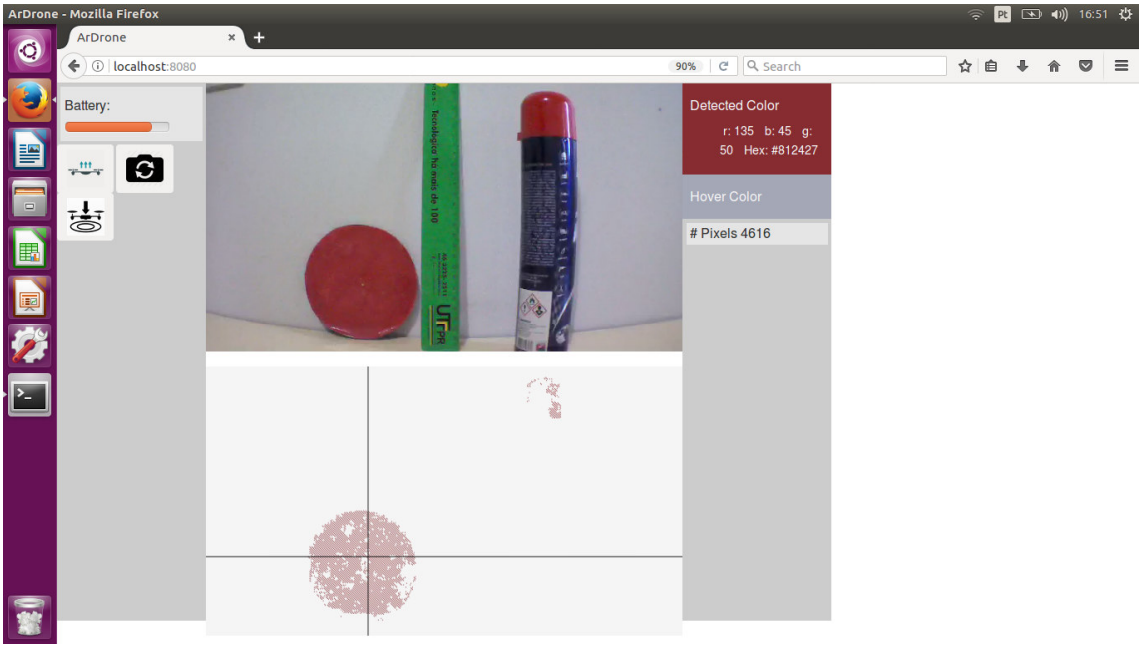

Figura 19 – Resultado de reconhecimento da cor vermelha.
Fonte própria.

O algoritmo apresenta resultados satisfatórios, reconhecendo a cor vermelha dentre outras testadas. Um grande problema encontrado é a intensidade da iluminação sobre os objetos em foco. Muitas vezes uma sombra ou diferença de iluminação em um objeto ocasiona a não identificação de cores que deveriam ser encontradas. A qualidade da imagem gerada pela câmera é inferior á câmera frontal e apresenta resultados inferiores. Entre as Figuras 20 e 21 podemos ver a diferença da imagem entre as câmeras.

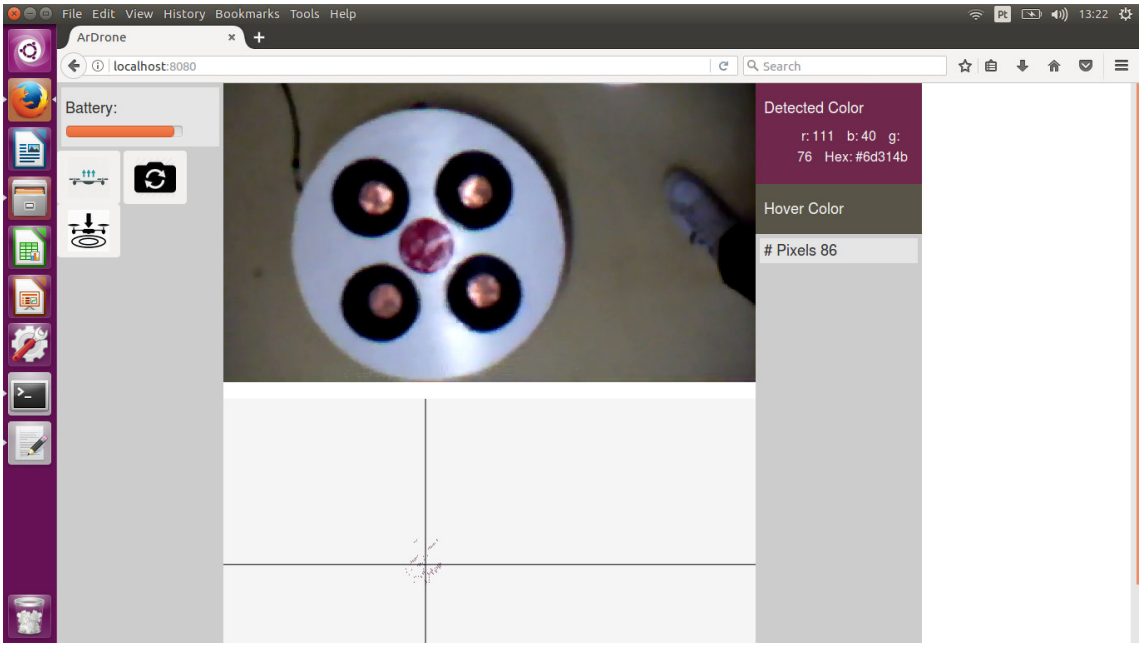

Figura 20 – Imagem da câmera inferior.
Fonte própria.

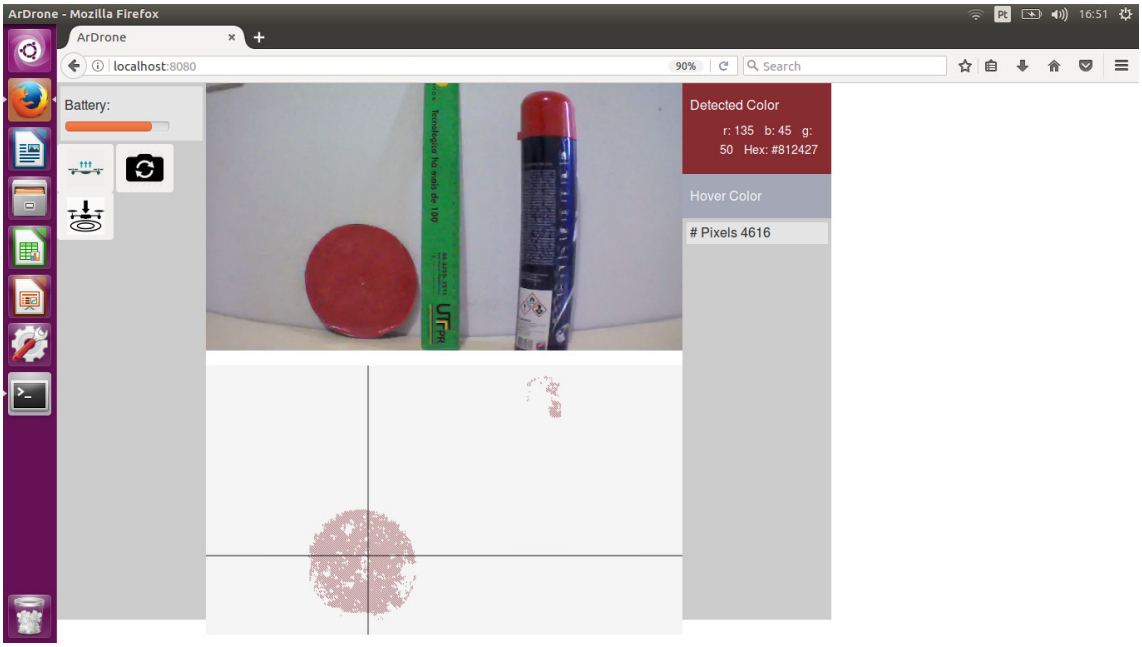

Figura 21 – Imagem da câmera superior.
Fonte própria.

Para motivos de teste, o algoritmo foi realizado de tal maneira que o drone não realize o pouso no momento em que encontra o centro do alvo. Isto permite que o drone siga a referência e comprove o funcionamento da rotina. Na Imagem 22 temos o reconhecimento da referência na imagem,.

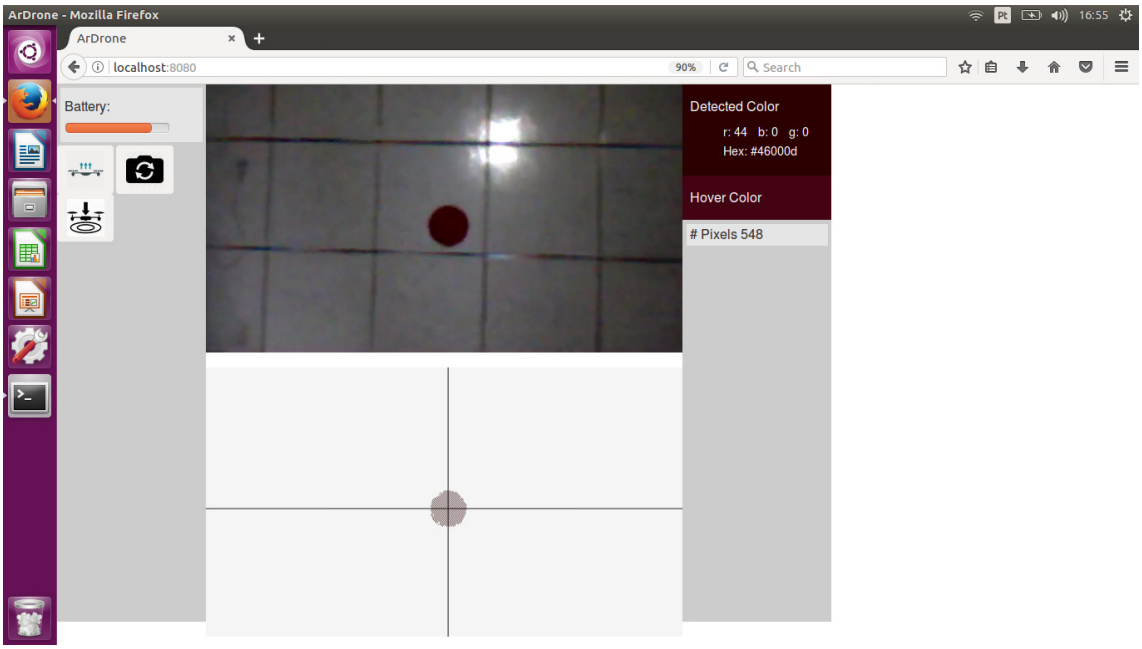

Figura 22 – Drone detectando a referência.
Fonte própria.

A Imagem 23 traz o resultado do controle de voo realizado pelo servidor.

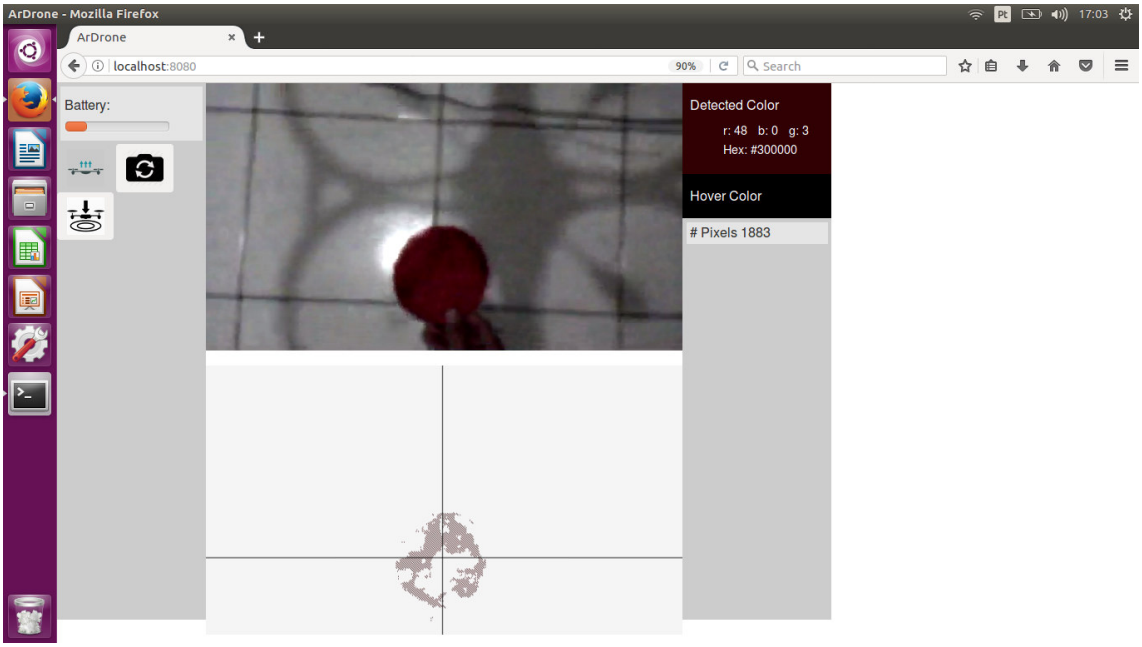

Figura 23 – Drone seguindo a referência.
Fonte própria.

4.3.3 Base

A base foi confeccionada utilizando uma chapa de madeira MDF, tendo 60 centímetros de circunferência, na qual foram inseridos quatro orifícios circulares de 11,5 centímetros de raio. Estes orifícios servem de acomodação para os 4 cones projetados, que foram fabricados em impressora 3D. Os conectores internos aos cones foram feitos utilizando placas de fenolite de formato circular, e foram soldadas aos cabos que ligam ao carregador. A base e os cones são mostrados na Figura 24.

Figura 24 – Base projetada.
Fonte própria.

O carregador da bateria foi conectado aos contatos na base através de um cabo adaptado. Esta adaptação consiste em um conector de bateria de celular, que é do mesmo modelo da conexão externa. O complemento do par de conectores do carregador de bateria de celular foi utilizado para fabricar os polos internos que ficam em contato com a bateria. A Figura 25 traz a disposição dos conectores.

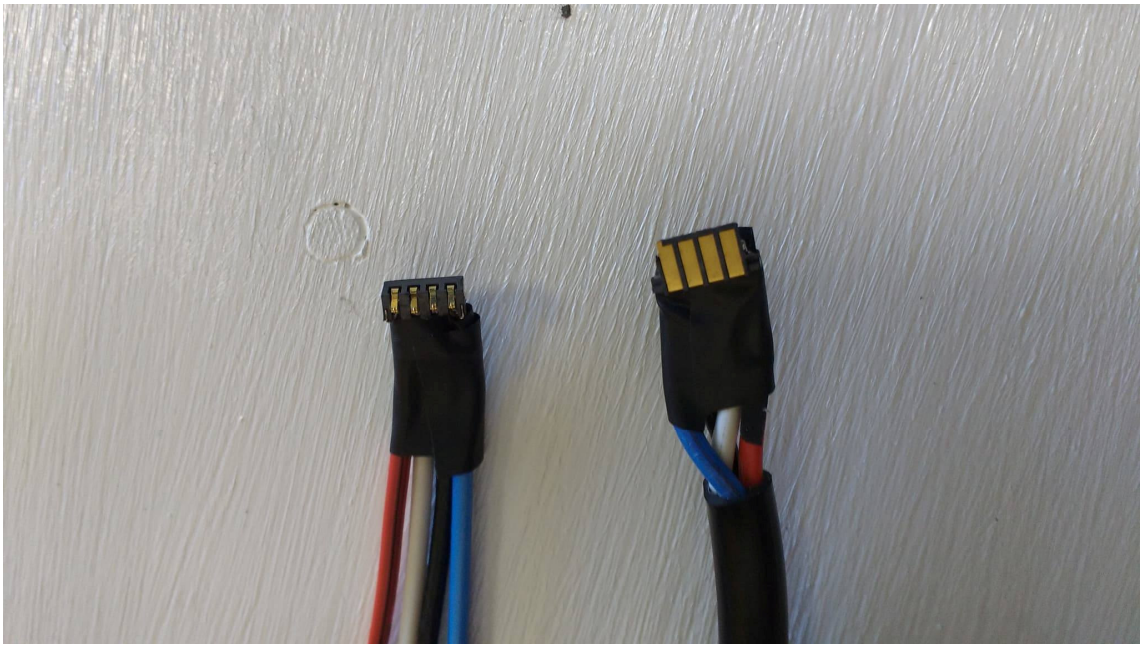

Figura 25 – Conectores adaptados.
Fonte própria.

Com o avanço do desenvolvimento do projeto alguns problemas foram surgindo e que devem ser tratados. A preocupação com a segurança do drone e daqueles que trabalham com ele sempre deve ser levada em consideração. Segundo o guia do usuário do ArDrone 2.0, o equipamento de maneira alguma deve ter a bateria carregada enquanto a mesma estiver alimento o circuito interno do microcontrolador. Desta maneira foi impedida a realização do carregamento da maneira como havia sido planejada.

Este é um problema que pode ser resolvido com a confecção de um circuito especifico para cortar a alimentação do drone no momento em que o mesmo pousa. Um circuito acionador com opto-acoplador para garantir a isolação dos componentes, pode ser projetado para ser embarcado junto à bateria interna do drone, de maneira que no momento do pouso, um sinal recebido da base acione o triac e inverta sua lógica de ação, cortando a alimentação do controlador. A imagem 26 representa o circuito descrito.

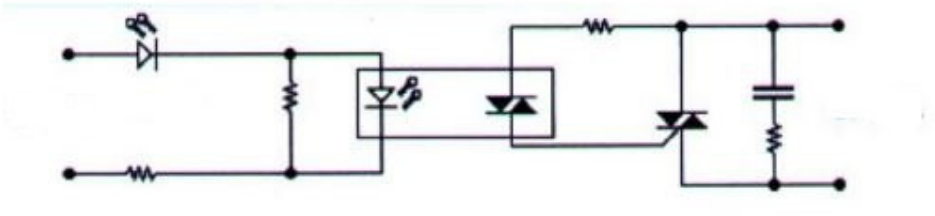

Figura 26 –Circuito acionador com opto-acoplador.
Fonte: http://padteleletronica.blogspot.com.br/2013/10/rele-de-estado-solido.html.

5 CONCLUSÃO

A proposta do projeto parte da necessidade de um sistema já existente, este desenvolvido por Vieira e Lópes (2016). Sabe-se que os drones vêm sendo utilizados mais e mais tanto por empresas quanto por usuários anônimos, ocasionando o aumento do interesse geral do produto. Dentre as limitações existentes em um drone, destacam-se a baixa autonomia de voo, assim como pouso e recarga com a necessidade da intervenção humana.

Muitos dos elementos utilizados no projeto foram pré-definidos com base no sistema já existente. Isso não modifica o fato da necessidade de um estudo aprofundado para definir o escopo e o desenvolvimento do trabalho. Com a ajuda de materiais especializados, trabalhos já desenvolvidos e livros publicados é possível estipular uma rota de ação para atingir os resultados esperados e de maneira técnica.

A utilização da linguagem Node.JS para o desenvolvimento da aplicação apresenta a primeira dificuldade, sendo inicialmente desconhecida ao autor. Como a linguagem possui uma biblioteca especifica para auxiliar o controle do ArDrone torna-se de grande valia para o projeto. Da mesma maneira, a linguagem JavaScript é fundamental para o desenvolvimento da aplicação *web*. O sistema só é possível graças aos milhares de desenvolvedores ao redor do mundo que trabalham junto e desenvolvem bibliotecas, muitas vezes simples, mas que facilitam muito a construção da aplicação e a compatibilidade de dispositivos. O projeto mostra que é possível atingir os objetivos desejados, sendo a aplicação *web* e a interface gráfica simples.

Para a manufatura dos cones da base foi utilizada uma impressora 3D, o que nos possibilita projetar peças da maneira mais adequada para o projeto. Os testes de carregamento não foram conduzidos devido ao risco de dano ao equipamento e a necessidade de se projetar um circuito especifico para o sistema. A base foi construída com MDF e pintada para facilitar o desenvolvimento do reconhecimento de imagem, já que sabemos exatamente o que devemos procurar na imagem gerada pelo drone.

O projeto mostra-se viável e com grandes possibilidades. Com a ajuda da biblioteca OpenCV é possível montar diversas aplicações baseadas em reconhecimento de imagem. O problema maior é com a qualidade da imagem da câmera inferior do drone, sendo esta de baixa qualidade o que muitas vezes dificulta o processamento.

REFERÊNCIAS

AITKEN, Jonathan M.; MCAREE, Owen; VERES, Sandor M. Symbiotic relationship between robots – a ROS ARDrone/YouBot library. Department of Automatic Control and Systems Engineering. Belfast, United Kingdom, September, 2016.

ALBUQUERQUE, Márcio P.; ALBUQUERQUE, Marcelo P. Processamento de Imagens: Métodos e Análises. Rio de Janeiro, Centro Brasileiro de Pesquisas Físicas, 2000.

BASTOS, Teresa Raquel. 15 usos de drones na agricultura e na pecuária. Disponível em: <http://revistagloborural.globo.com/Noticias/Pesquisa-e-Tecnologia/noticia/2015/05/15-usos-de-drones-na-agricultura-e-na-pecuaria.html> Acesso em: 25 ago. 2016.

BASTIANELLI, G.; SALAMON, D.; Schisano, A.; Iacobacci, A. Agent-based simulation of collaborative unmanned satellite vehicles. IEEE First AESS European Conference on Satellite Telecommunications (ESTEL), 2012.

BARRESI, Federico Francesco; ALLASIA, Walter. Airport Markings Recognition for Automatic Taxiing. Università degli Studi di Torino, 2013.

BRANDÃO, A. S. Martins, F. N. Soneguetti, H. B. A vision-based line following strategy for an autonomous UAV. 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2015.

COCCHIONI, Franceso; PIERFELICE, Valerio; BENINI, Alessandro; MANCINI, Adriano; FRONTONI, Emanuele; ZINGARETTI, Primo; IPPOLITI, Gianluca; LONGHI, Sauro. Unmanned Ground and Aerial Vehicles in Extended Range Indoor and Outdoor Missions. 2014 International Conference on Unmanned Aircraft Systems (ICUAS), 2014.

COSTA, S. E. A. P. Controlo e Simulação de um Quadrirotor convencional. Dissertação para obtenção do Grau de Mestre em Engenharia Aeroespacial. Instituto Superior Técnico, Universidade Técnica de Lisboa, 2008.

DJAPIC, Vladimir; PRIJIC, Christopher; BOGART, Frank. Autonomous Takeoff & Landing of Small UAS from the USV. California, USA. 2015.

DUARTE, Rafael Mendes. Low Cost Brain Comuter Interface Systems for AR.Drone Control. Dissertação para defesa de Mestrado da UFSC. Universidade Federal de Santa Catarina. Maio, 2017.

ESQUEF, Israel A.; ALBUQUERQUE, Márcio P.; ALBUQUERQUE, Marcelo P. de A. Processamento Digital de Imagens. Rio de Janeiro, Centro Brasileiro de Pesquisas Físicas, 2003.

FAB, Força Aérea Brasileira, Disponível em: http://www.fab.mil.br/noticias/mostra/23937/ESPAÇO%20AÉREO%20-%20Comando%20da%20Aeronáutica%20publica%20nova%20legislação%20sobre%20aeronaves%20remotamente%20pilotadas. Acessado em 13/10/2016.

FAHLSTROM, Paul; GLEASON, Thomas. Introduction to UAV Systems. Chichester, United Kingdom: John Wiley & Sons, 2012.

FILHO, Ogê M.; NETO, Hugo V. Processamento Digital de Imagens. Rio de Janeiro, Brasport, 1999.

GEORGE, E.A.; TIWARI, G.; YADAV, R.N.; PETERS, E.; SADANA, S. UAV systems for parameter identification in agriculture. Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS), 2013 IEEE.

HUERTA, Eduardo; MANGIATERRA, Aldo; NOGUERA, Gustavo. GPS: posicionamento satelital. Rosario, UNR Editora, Universidad Nacional de Rosario, 2005.

HUI, Cheng; YOUSHENG, Chen; XIAOKUN, Li; SHING, Wong Wing. Autonomous Takeoff, Tracking and Landing of a UAV on a Moving UGV Using Onboard Monocular Vision. 32nd Chinese Control Conference. 2013.

LIOULEMES, Alexandros; GALATAS, Georgios; METSIS, Vangelis; MARIOTTINI, Gian Luca; MAKEDON, Fillia. Safety Challenges in using AR.Drone to collaborate with humans in indoor environments. Department of Computer Science and Engineering – University of Texas at Arlington. 2014.

MAINI, Parikshit; SUJIT, P. B. On Cooperation Between a Fuel Constrained UAV and a Refueling UGV for Large Scale Mapping Aplications. International Conferenceon of Unmanned Aircraft Systems. 2015.

MIN-FAN, Ricky Lee; SHUN-FENG, Su; JIE-WEI, Eric Yeah; HUSAN-MING, Huang; CHEN, Jonathan. Autonomous landing System for Aerial Mobile Robot Cooperation. SCIS & ISIS Kitakyushu, Japan, 2014.

PEGORARO, Antoninho J. Estudo do potencial de um veículo aéreo não tripulado/quadrotor, como plataforma na obtenção de dados cadastrais. Florianópolis, Universidade Federal de Santa Catarina, 2013.

PHILIP FINNEGAN et al., "World Unmanned Aerial Vehicle Systems, Market Profile and Forecast 2013", Teal Group Research, June. 2013. Diponível em: < http://tealgroup.com/index.php/about-teal-group-corporation/press-releases/129-teal-group-predicts-worldwide-civil-uas-production-will-total-65-billion-in-its-2016-uas-market-profile-and-forecast>. Acesso em: 25 ago. 2016.

QUARITSCH, M.; STOJANOVSKI, E.; BETTSTETTER, C.; FRIEDRICH, G.; HELLWAGNER, H.; RINNER, B. Collaborative microdrones: Applications and Research Challenges. Turim, Itália, 2008.

QUEIROZ, José E. R. de; GOMES, Herman M. Introdução ao Processamento Digital de Imagens. Revista RITA, UFCG, 2001.

RONCOLATO, Murilo. Mercado de drones cresce sem lei no Brasil e indústria nacional fica para trás. Diponível em: <http://blogs.estadao.com.br/link/mercado-de-drones-cresce-sem-lei-no-pais-e-industria-nacional-fica-para-tras/>. Acesso em: 25 ago. 2016.

SANTOS, Matilde; LÓPEZ, Victoria; MORATA, Franciso. Intelligent fuzzy controller of a quadrotor. IEEE, 2010, p. 141-146.

SOUTHWORTH, Matt. **Drones**. Friends Committee on National Legislation. 2012. Disponível em: <http:/ /www.fcnl.org>. Acesso em: 03 nov. 2016.

SZELISKI, Richard. Computer Vision, Algorithms and Applications. Setembro, 2010. Disponivel em: http://szeliski.org/Book/. Acesso em: 24 out. 2016.

VARELLA, João. Os drones invadem os negócios. Disponivel em: <http://www.istoedinheiro.com.br/noticias/mercado-digital/20140124/drones-invadem-negocios/146050.shtml> Acessado em: 15 out. 2016.

VIEIRA, Elias R.; LOPES, Gustavo H. da S. Aplicação para automação de voo de drone. UTFPR, 2016.

APÊNDICES

O arquivo index.html é responsável por criar a interface da aplicação utilizando HTML5. Temos a declaração do controlador, dos botões, da interface gráfica para escolha da cor, *display* para a carga da bateria e os canvas para mostrar as imagens.

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>ArDrone</title>
    <link                rel="stylesheet"                type="text/css"
href="./bower_components/css/bootstrap.min.css">
    <link rel="stylesheet" type="text/css" href="./src/css/index.css">

</head>
<body ng-app="app" ng-controller="controller"> //definição do controlador

<div id="wrapper">
<div id="leftWidgets" class="widget-container left">

    <div class="widget">
    <h4>Battery:</h4>//batteria
    <progress   id="battery"   max="100"   value={{battery}}   >Battery
Meter</progress>

    </div>
    <div id="switchCamera">
        <button         id="flight"         class="btn"><img         src
="./src/images/drone_take_off.png" height="50" width="50"></button>
        <button     class="btn"     ng-click="switchCamera()"><img     src
="./src/images/camera-switch-128.png" height="50" width="50"></button>
    </div>
      <div id="flightControl">
        <button         id="landing"         class="btn"><img         src
="./src/images/landing.png" height="50" width="50"></button>
        //botoes
    </div>

</div>

<div id="streams" class="left">
    <div id="video" style="width: 640px; height: 360px"></div>
    <canvas id="testCanvas" style="width: 640px; height: 360px"></canvas>
    <br>
    <canvas   id="track"   class=""   style="width:   640px;   height:
360px"></canvas>
</div>//widget de cor
    <div  id="rightWidgets"  ng-click="setTargetRadius()"  class="widget-
container left">
        <div id="pickedColor">
        <h4>Detected Color</h4>
        <ul class="color-info">
            <li id="rVal"></li>
            <li id="gVal"></li>
            <li id="bVal"></li>
            <li id="hexVal"></li>
```

```
        </ul>

        </div>
        <div id="preview">
        <h4>Hover Color</h4>
       </div>
        <h4 class="widget" id="pixelCount"></h4>

        </div>

    //local dos arquivos da aplicacao
    <script             src="./bower_components/jquery/dist/jquery.min.js"
type="text/javascript" charset="utf-8"></script>
    <script                src="./bower_components/angular/angular.min.js"
type="text/javascript" charset="utf-8"></script>
    <script                 src="./bower_components/css/bootstrap.min.css"
type="text/javascript" charset="utf-8"></script>
<script            src="//localhost:5555/dronestream/nodecopter-client.js"
type="text/javascript" charset="utf-8"></script>
<script src="./src/main.js" type="text/javascript"></script>
</div>

</body>
</html>
```

Listagem 2 – index.html.

O arquivo app.js é responsável pela inicialização do servidor e comunicação com o drone. Utiliza linguagem JavaScript.

```
//configura drone para mandar navdata

var navdataOptions = (
    navdataOptionMask(arDroneConstants.options.DEMO)
  | navdataOptionMask(arDroneConstants.options.VISION_DETECT)
  | navdataOptionMask(arDroneConstants.options.MAGNETO)
  | navdataOptionMask(arDroneConstants.options.WIFI)
  | navdataOptionMask(arDroneConstants.options.ZIMMU_3000)
  | navdataOptionMask(arDroneConstants.options.ALTITUDE)
);

drone.config('general:navdata_demo', true);
drone.config('general:navdata_options', navdataOptions);

//cria registros para as pastas da aplicacao

var staticDir = 'src',
    check = new RegExp('^/' + staticDir, 'i'),
    check2 = new RegExp('^/bower_components', 'i'),
    check3 = new RegExp('^/node_modules', 'i'),
    dist = ".";

//cria server
var server = http.createServer(function(req, res) {
    fs.createReadStream(__dirname + "/index.html").pipe(res);
});

oldHandlers = server.listeners('request').splice(0);
```

```
server.removeAllListeners('request');

server.on('request', function (req, res) {
    var i = 0;
    if (handler(req, res)) {
        return;
    }

    for (; i < oldHandlers.length; ++i) {
        oldHandlers[i].call(server, req, res);
    }
});

function handler(req, res, next) {
    var path, read;
    if     (!check.test(req.url)     &&     !check2.test(req.url)     &&
!check3.test(req.url)) {
        return false;
    }
    path = dist + req.url;
    console.log('checking static path: %s', path);
    read = fs.createReadStream(path);

    read.pipe(res);
    read.on('error', function (e) {
        console.log('Stream error: %s', e.message);
    });

    return true;
}
//responsabel por criar um link entre o app.js e o main.js
// do cliente do ardrone

var wsServer = new ws.Server({server: server});
wsServer.on('connection', function(conn) {
    function send(msg) {
        conn.send(JSON.stringify(msg));
    }
    var cameraMode = 0;
    conn.on('message', function(msg) {
        try {
            msg = JSON.parse(msg);
        } catch (err) {
            console.log('err: '+err+': '+msg);
        }
        var kind = msg.shift();
        switch (kind) {
            case 'on':
                var event = msg.shift();
                drone.on(event, function(data) {
                    send(['on', event, data]);
                });
                break;
            case 'takeoff':
                drone.takeoff(function() {
                    send(['takeoff']);
                });
                break;
            case 'land':
                drone.land(function() {
                    send(['land']);
```

```
                });
                break;
            case 'right':
                drone.right(msg[0]);
                break;
            case 'up':
            drone.up(msg[0]);
            break;
            case 'clockwise':
                drone.clockwise(msg[0]);
                break;
            case 'front':
                drone.front(msg[0]);
                break;
            case 'stop':
                drone.stop();
                break;
            case 'camera':
                if(cameraMode==0) {
                    drone.config('video:video_channel', 3);
                    cameraMode=3;
                } else {
                    drone.config('video:video_channel', 0);
                    cameraMode=0;
                }

                break;
            default:
                console.log('unknown msg: '+kind);
                break;
        }
    });
});

droneStream.listen(5555);
server.listen(8080);
```

Listagem 3 – app.js.

O arquivo main.js é o responsável pela logica de controle da aplicação fornecendo as funções de reconhecimento de imagem e de controle de voo.

```
app.controller('controller', ['$scope', function ($scope) {
      var tempNavdata;
    client.on('navdata', function loginNavData(navdata){
        if(navdata != null && navdata.demo != null) {
            //coleta dados do drone
            $scope.battery = navdata.demo.batteryPercentage;
            $scope.altitude = navdata.demo.altitudeMeters;
            $scope.latitude = navdata.demo.latitude;
            $scope.longitude = navdata.demo.longitude;
            tempNavdata = navdata;
            $('#battery').attr('value', navdata.demo.batteryPercentage);
        }
    });
    $scope.maxDiff = 100;
    $scope.accuracy = 3;
```

```
    $scope.fps = 200;
    $scope.battery;
    $scope.altitude;
    $scope.altitudeTarget = 1;

    setState('ground');

    var y;
    var x;
    //main loop
    $scope.mainLoop = function(){
            //Limpa a tela
            clearInterval(interval);
            ctx.clearRect(0, 0, w, h);
            //Detecta cor
            detectColor();

            //Atualiza a tela

            updateUIText();
            drawCrossHair(detected.x,detected.y);

            var xVal = (detected.x - w / 2) / (w / 2);
            var yVal = (detected.y - h/2) / (h/2);

            var radi = getRadius(detected.x, detected.y);

            var radidiff = radi-$scope.targetRadius;

            //rotina de controle

            if (state === "follow" && !isNaN(xVal) && !isNaN(yVal)) {
                if (camera_mode == CameraModes.FRONT_FOLLOW) {
                    followFront(xVal,yVal,radi,radidiff);

                } else if(camera_mode == CameraModes.BOTTOM_FOLLOW){
                    followBottom(xVal,yVal);
                }
                else {
                    client.stop();
                }
            } else {
                client.stop();
            }
        interval = setInterval($scope.mainLoop, $scope.fps);
    }

    var interval = setInterval($scope.mainLoop, $scope.fps);

    $scope.targetRadius = 0;
    $scope.setTargetRadius = function() {
        $scope.targetRadius = getRadius(detected.x, detected.y);
    }
//funcao para trocar camera
    $scope.switchCamera = function() {

        client.camera();
        if(camera_mode == CameraModes.FRONT_FOLLOW){
            camera_mode = CameraModes.BOTTOM_FOLLOW;
        }
        else if(camera_mode == CameraModes.BOTTOM_FOLLOW){
```

```
            camera_mode = CameraModes.FRONT_FOLLOW;
        }
            console.log(camera_mode);
    }
//funcao de deteccao
      function detectColor(){

              b = frameBuffer;
              count = 0;
              var xSum = 0;
              var ySum = 0;

              lower_hsv_threshold = [0, 100, 100]
              upper_hsv_threshold = [10, 255, 255]
              //threshold cor vermelha
              lower_hsv_threshold2 = [165, 100, 100]
              upper_hsv_threshold2 = [179, 255, 255]

              cv.readImage(ns.getImageData(b), function (err, img) {
                  if (err) throw err;

                  const width = img.width();
                  const height = img.height();

                  let imgG = img.copy();
                  let imgH = img.copy();
                  let imgH2 = img.copy();

                  //imgH.medianBlur(3);
                  //imgH2.medianBlur(3);

                  //HSV
                  imgH.convertHSVscale();
                  imgH2.convertHSVscale();
                  imgH.inRange(lower_hsv_threshold, upper_hsv_threshold);
                  imgH2.inRange(lower_hsv_threshold2,
     upper_hsv_threshold2);

                  imgH.save('h.png');
                  imgH2.save('h2.png');

                  var result = new cv.Matrix(img.width(), img.height());
                  result.addWeighted(imgH, 0.7, imgH2, 0.9);
                  result.save('frameHSV.png');

                  //contours
                  var big = new cv.Matrix(width, height);
                  big = result.copy();
                  var im_canny = result.copy();
                  im_canny.canny(lowThresh, highThresh);
                  im_canny.dilate(nIters);

                  var contours = im_canny.findContours();
                  const lineType = 8;
                  const maxLevel = 0;
                  const thickness = 1;

                  for(i = 0; i < contours.size(); i++) {
                        if(contours.area(i) > maxArea) {
                              var moments = contours.moments(i);
                              var    cgx    =    Math.round(moments.m10    /
```

```
      moments.m00);
                              var    cgy    =    Math.round(moments.m01    /
      moments.m00);
                              big.drawContour(contours,      i,      GREEN,
      thickness, lineType, maxLevel, [0, 0]);
                              big.line([cgx  -  5,  cgy],  [cgx  +  5,  cgy],
      RED);
                              big.line([cgx,  cgy  -  5],  [cgx,  cgy  +  5],
      RED);
                        }
                  }
                  centerX = cgx;
                  centerY = cgy;
                  detected = {x: cgx, y: cgy};

                  big.save('frameCON.png');
                  if (width < 1 || height < 1) {
                        throw new Error('Image has no size');
                  }
      }
      //funcoes de seguimento
    function followBottom(xVal,yVal){
        client.right(xVal/6);
        client.front(-yVal/6);
        console.log($scope.altitude);
        if($scope.altitude < $scope.altitudeTarget) {
            client.up(.05);
        }
        else {
            client.up(-.05);
        }

    }
    function followFront(xVal, yVal, radi, radidiff){
        client.clockwise(xVal / 4);
        client.up(-yVal / 6);
        if(radi > 10) {
            if (radidiff < 0) {
                client.front(.05);
            }
            else if(radidiff > 0) {
                client.front(-.05);
            }
        } else{
            client.stop();
        }
    }



    function drawCrossHair(detctX, detctY){
        ctx.beginPath();
        ctx.moveTo(0, detctY);
        ctx.lineTo(640, detctY);
        ctx.moveTo(detctX, 0);
        ctx.lineTo(detctX, 360);
        ctx.strokeStyle = "black";
        ctx.stroke();
        ctx.closePath();
    }
```

```
    function updateUIText(){
        var pixelColor = "rgb(" + pickedColor[0] + ", " + pickedColor[1]
+ ", " + pickedColor[2] + ")";
        $('#pickedColor').css('background-color', pixelColor);
        $('#rVal').html("r: " + pickedColor[0]);
        $('#gVal').html("b: " + pickedColor[1]);
        $('#bVal').html("g: " + pickedColor[2]);
        $('#targetRadius').html("radius: " + $scope.targetRadius);
        lastCount = count;
    }

    function getRadius(xCenter, yCenter){
        var s = frameBuffer;
        var xDis = Math.abs(w-xCenter);
        ns.getImageData(s, xCenter, h-yCenter, xDis, 1);
        var farthestXRight = 0;

        for(var i=0; i < (xDis*4);i+=4){
            var  isMatch  =  (Math.abs(s[i]  -  pickedColor[0])  /  255  <
maxDiff
            && Math.abs(s[i+1] - pickedColor[1]) / 255 < maxDiff
            && Math.abs(s[i+2] - pickedColor[2]) / 255 < maxDiff);
            if(isMatch){
                farthestXRight = i/4;
            }
        }
        return farthestXRight;
    }
    var flightButton = document.getElementById('flight');
    flightButton.addEventListener('click', function () {

            setState('takeoff');
            client.takeoff(function () {
                  setState('follow');
            });

    });

      var landingButton = document.getElementById('landing');
    landingButton.addEventListener('click', function () {

        setState('land');
        client.land(function () {
            setState('ground');
        });
    });

}]);

function setState(val) {
    console.log('new state: ' + val);
    this.state = val;
}

function WsClient() {
    this._conn = null;
    this._connected = false;
    this._queue = [];
    this._listeners = {};
    this._takeoffCbs = [];
    this._landCbs = [];
```

```
    var self = this;
    self._conn = new WebSocket('ws://' + window.location.host);
    self._conn.onopen = function () {
        self._connected = true;
        self._queue.forEach(function (msg) {
            self._conn.send(msg);
        });
        self._queue = [];

        self._conn.onmessage = function (msg) {
            try {
                msg = JSON.parse(msg.data);
            } catch (err) {
                console.error(err);
                return;
            }
            var kind = msg.shift();
            switch (kind) {
                case 'takeoff':
                    self._takeoffCbs.forEach(function (cb) {
                        cb();
                    });
                    self._takeoffCbs = [];
                    break;
                case 'land':
                    self._landCbs.forEach(function (cb) {
                        cb();
                    });
                    self._landCbs = [];
                    break;
                case 'on':
                    var event = msg.shift();
                    self._listeners[event].forEach(function (cb) {
                        cb.apply(self, msg);
                    });
                    break;
                default:
                    console.error('unknown message: ' + kind);
            }
        };
    };

}

WsClient.prototype._connect = function () {
    var self = this;
    self._conn = new WebSocket('ws://' + window.location.host);
    self._conn.onopen = function () {
        self._connected = true;
        self._queue.forEach(function (msg) {
            self._conn.send(msg);
        });
        self._queue = [];

        self._conn.onmessage = function (msg) {
            try {
                msg = JSON.parse(msg.data);
            } catch (err) {
                console.error(err);
                return;
```

```
            }
            var kind = msg.shift();
            switch (kind) {
                case 'takeoff':
                    self._takeoffCbs.forEach(function (cb) {
                        cb();
                    });
                    self._takeoffCbs = [];
                    break;
                case 'land':
                    self._landCbs.forEach(function (cb) {
                        cb();
                    });
                    self._landCbs = [];
                    break;
                case 'on':
                    var event = msg.shift();
                    self._listeners[event].forEach(function (cb) {
                        cb.apply(self, msg);
                    });
                    break;
                default:
                    console.error('unknown message: ' + kind);
            }
        };
    };

};

WsClient.prototype._send = function (msg) {
    msg = JSON.stringify(msg);
    if (!this._connected) {
        this._queue.push(msg);
        return;
    }
    this._conn.send(msg);
};

WsClient.prototype.on = function (event, cb) {
    var cbs = this._listeners[event] = this._listeners[event] || [];
    cbs.push(cb);
    if (cbs.length === 1) {
        this._send(['on', event]);
    }
};

WsClient.prototype.takeoff = function (cb) {
    this._send(['takeoff']);
    if (cb) {
        this._takeoffCbs.push(cb);
    }
};

WsClient.prototype.land = function (cb) {
    this._send(['land']);
    if (cb) {
        this._landCbs.push(cb);
    }
};

WsClient.prototype.right = function (val) {
```

```
    this._send(['right', val]);
};
WsClient.prototype.clockwise = function (val) {
    this._send(['clockwise', val]);
};

WsClient.prototype.up = function (val) {
    this._send(['up', val]);
};

WsClient.prototype.front = function (val) {
    this._send(['front', val]);
};

WsClient.prototype.stop = function () {
    this._send(['stop']);
};

WsClient.prototype.camera = function () {
    this._send(['camera']);
};

$(function () {

    $('#testCanvas').hide();

    //calculate offset for clicking and hovering on canvas
    var leftOffset = $('.widget-container').width();


    $('#video').mousemove(function (e) { // mouse move handler

        var canvasX = Math.floor(e.pageX - leftOffset);
        var canvasY = Math.floor(e.pageY);

        ns.getImageData(c, canvasX, h - canvasY, 1, 1);

        var pixelColor = "rgb(" + c[0] + ", " + c[1] + ", " + c[2] + ")";
        $('#preview').css('background-color', pixelColor);
    });

    $('#video').click(function (e) { // mouse click handler

        var canvasX = Math.floor(e.pageX - leftOffset);
        var canvasY = Math.floor(e.pageY);

        ns.getImageData(c, canvasX, h - canvasY, 1, 1);

        //change detection color
        pickedColor[0] = c[0];
        pickedColor[1] = c[1];
        pickedColor[2] = c[2];

        var pixelColor = "rgb(" + pickedColor[0] + ", " + pickedColor[1]
+ ", " + pickedColor[2] + ")";
        $('#pickedColor').css('background-color', pixelColor);

        //color info
        $('#rVal').html("r" + c[0]);
        $('#gVal').html("b" + c[1]);
        $('#bVal').html("g" + c[2]);
```

```
        $('#rgbVal').val(c[0] + ',' + c[1] + ',' + c[2]);
        $('#rgbaVal').val(c[0] + ',' + c[1] + ',' + c[2] + ',' + c[3]);
        var dColor = c[2] + 256 * c[1] + 65536 * c[0];
        $('#hexVal').html('Hex: #' + dColor.toString(16));
    });

    setInterval(function updateUIPixelCount() {
        $('#pixelCount').html("# Pixels "+lastCount);
    }, 300);

});
```

Listagem 4 – main.js.

O arquivo package.json lista as dependências que a aplicação necessita para funcionar.

```
{
  "name": "drone",
  "version": "1.0.0",
  "description": "Drone nodejs",
  "main": "index.js",
  "private": true,
  "scripts": {
    "test": "test"
  },
  "author": "JoseFVL",
  "license": "ISC",
  "dependencies": {
    "ar-drone": "^0.3.3",
    "send": "^0.3.0",
    "ws": "^0.4.31",
    "opencv": "^6.0.0",
    "dronestream": "git://github.com/felixge/node-dronestream.git#soccer"
  },
  "devDependencies": {
    "express": "^4.15.2"
  }
}
```

Listagem 5 – package.json.