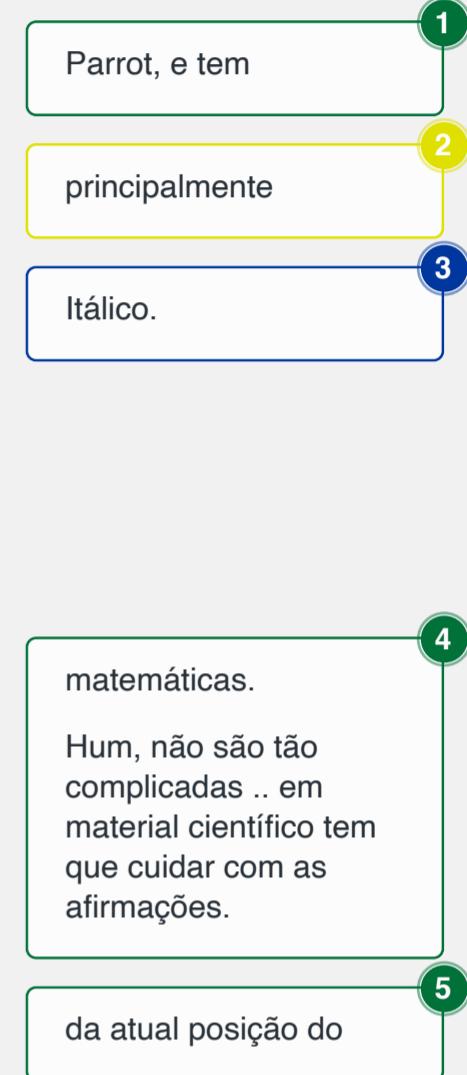


RESUMO

Este trabalho descreve o desenvolvimento de uma arquitetura de voo autônomo aplicada no modelo AR.Drone 2.0 da Parrot, tem como objetivo gerenciar percursos de rotas pré-estabelecidas, visando principal aplicabilidade de vigilância aérea em espaços externos. O desenvolvimento foi realizado utilizando o framework Node.js integrado a bibliotecas NPM como node-ar-drone, ardrone-autonomy e Geolib. Para a seleção do percurso e coleta das coordenadas foi utilizada biblioteca Leaflet e API do Google Maps. Os algoritmos implementados permitem o cadastramento do percurso, acompanhamento das imagens em tempo real e inclui centro de comandos em tela única, habilitado com opções essenciais de usabilidade amigável. As ferramentas e bibliotecas utilizadas se mostraram adequadas, abstraindo milhares de linhas de código e fórmulas matemáticas de alta complexidade. O módulo GPS possui algumas limitações relacionadas a precisão da posição atual do drone, porém ao executar os testes obteve-se êxito em realizar a vigilância de espaços externos, obtendo-se devido ao adequado uso dos recursos técnicos da arquitetura e de hardware do AR.Drone. A arquitetura fora capaz de pilotar autonomamente o drone ao executar os percursos selecionados no mapa, ao final do trajeto trazendo-o de volta a base de origem, finalizando a vigilância.

Palavras-chave: Drone. Ardrone. Parrot. Voo autônomo. GPS. Vigilância. Node. Geolib.



LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

ARM – Advanced RISC Machine

AT – Autonomous 1

AVR – Alf-Egil Bogen e Vegard Wollan 2

BITS – Binary Digit 3 4

CPU – Central Processing Unit

CSS – Cascading Style Sheets

CSV – Comma Separated Values

DDR – Double Data Rate

DSP – Digital Signal Processor

FPS – Frames Per Second

FURB – Universidade Regional de Blumenau 5

GB – Gigabyte 6

GBT – Gigabit 7

GEO – Geographic Location 8

GHZ – Gigahertz 9

GLONASS – Global (Orbiting) Navigation Satellite System 10

GNSS – Global Navigation Satellite System

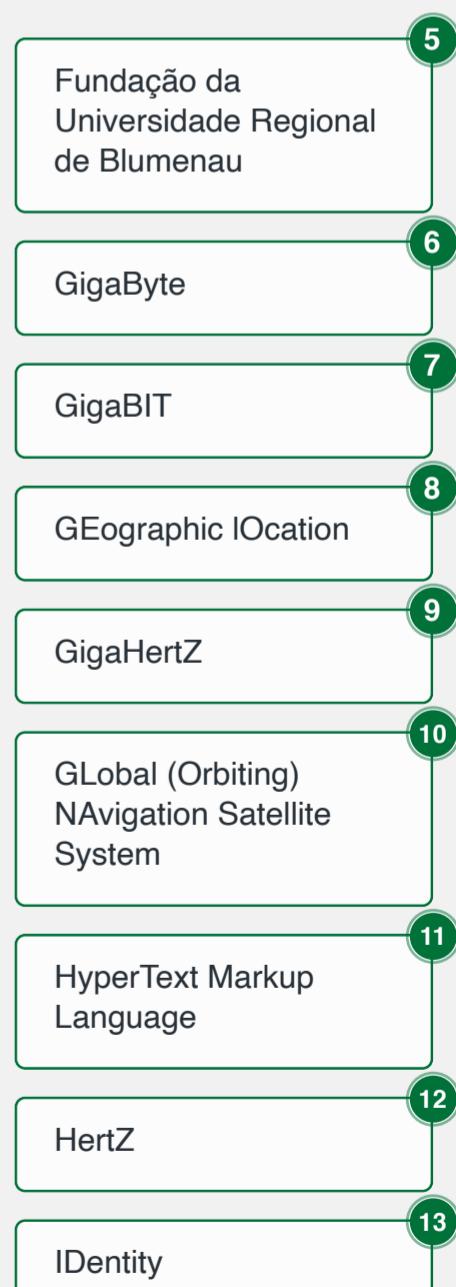
GPS – Global Positioning System

HTML – Hypertext Markup Language 11

HTTP – HyperText Transfer Protocol

HZ – Hertz 12

ID – Identity 13



JPEG – Joint Photographic Experts Group

JS – JavaScript

JSON – Javascript Object Notation 

KB – Kilobyte 

MG - Miligrama 

MHZ – Megahertz 

MIPS – Microprocessor Without Interlocked Pipeline Stages 

MS – Milissegundo 

NPM – Node Package Manager

OBJ – Object File Wavefront 

PA – Precision and Accuracy

QVGA – Quarter Video Graphics Array

RAM – Random Access Memory

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

ROS – Robot Operation System

RPM – Rotação Por Minuto

SDK – Software Development Kit

TCP – Transmission Control Protocol

UC – User Cases

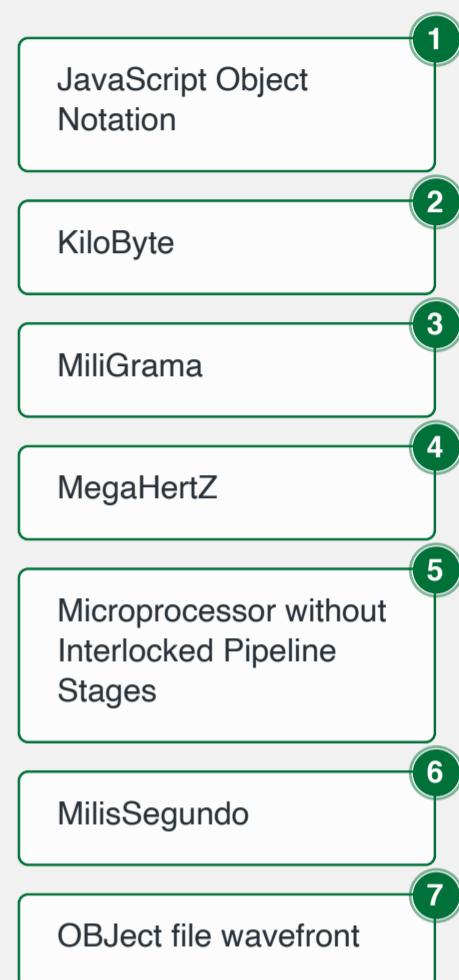
UDP – User Datagram Protocol

UI – User Interface

UML – Unified Modeling Language

USB – Universal Serial Bus

UX – User Experience



1 INTRODUÇÃO

O mundo tecnológico do qual vivemos atualmente se assemelha a uma ficção científica, possuímos tecnologias de ponta que estão aplicadas em diversos campos da ciência. Para muitos o espanto ainda é grande e confuso, pois grande parte dessa tecnologia trouxe mudanças em nossa sociedade, tal como novos modelos de trabalho e novas profissões. Uma dessas tecnologias são os drones¹ que se assemelham a um brinquedo ou hobby de criança grande, porém não se limita apenas a isso, há também inúmeras maneiras dessa tecnologia ser utilizada em prol do apoio em diversas áreas, principalmente com vigilância em áreas abertas e de grande circulação de pessoas (SHIRATSUCHI, 2014).

Parece que o “s” desta palavra está em itálico.

No Brasil, esta máquina é chamada Vant – Veículo Aéreo Não Tripulado) ou “drone” (zangão, na língua inglesa), miniaturas derivadas dos aviões não tripulados produzidos de forma contínua pela indústria bélica há pelo menos 20 anos, principalmente nos Estados Unidos (Shiratsuchi, p.16, 2014):

Segundo Estêvão (2014) a preocupação com a vigilância se tornou comum em todas as sociedades e estados, com grande aumento nos investimentos em monitoração e com a recente chegada de novas tecnologias, trazendo uma nova realidade. Atualmente os Estados Unidos da América possui projetos de investigação para detectar ações criminalistas e de terrorismo, utilizando de sofisticados sistemas de vigilância.

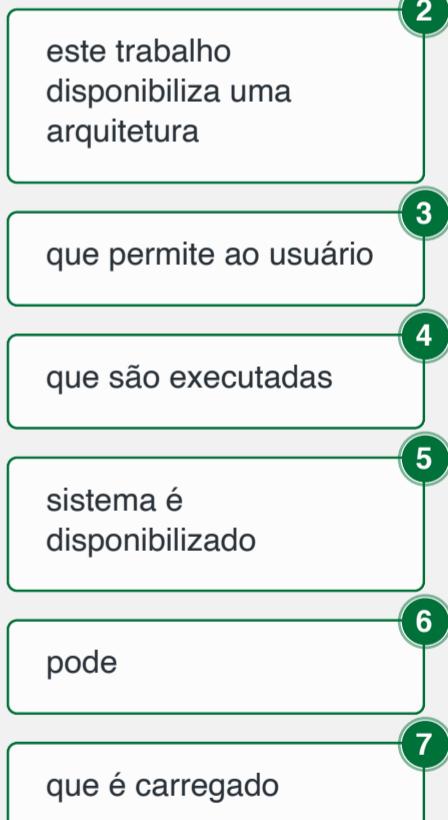
Neste contexto, este trabalho se propõe a disponibilizar uma arquitetura para definir um sistema autônomo de vigilância externa que permitirá ao usuário definir uma base e rotas que serão executadas por um drone. O sistema será disponibilizado numa interface web e o usuário poderá informar a sua base e rota através de pontos que deverão ser percorridos, informando no mapa que deverá ser carregado previamente. A rota será percorrida pelo drone de forma autônoma. Durante o percurso pré-definido, o drone montará um relatório com imagens da rota e o enviará ao sistema.

1.1 OBJETIVOS

Este trabalho tem como objetivo propor uma arquitetura para um sistema de vigilância utilizando drone.

Os objetivos específicos são:

- possuir cadastro de base e rotas para o drone;
- oferecer recursos para percorrer rotas de forma autônoma;
- disponibilizar dados registrados na rota.



2 FUNDAMENTAÇÃO TEÓRICA

As seções desse capítulo estão fundamentadas na seguinte ordem. A seção 2.1 descreve as especificações técnicas e de hardware do AR.Drone 2.0, adaptações necessárias, e sobre seu respectivo módulo externo que habilita o uso de GPS. A seção 2.2 descreve as estruturas da biblioteca NPM do node-ar-drone, assim como a seção 2.3 descreve a estrutura da biblioteca NPM do ardrone-autonomy. A seção 2.4 descreve o framework NPM Geolib para cálculos geográficos. A seção 2.5 é destinada a integração com API do Google Maps usando o framework NPM leaflet-google, pôr fim a seção 2.7 descreve os trabalhos correlatos.

2.1 ESPECIFICAÇÕES TECNICAS E DE HARDWARE AR.DRONE 2.0

O AR.Drone 2.0 consiste em um quadricóptero desenvolvido pela empresa Parrot. O controle deste equipamento pode ser realizado por meio de computadores ou smartphones através de uma conexão WI-FI em modo AD-HOC aberta, e operando por bandas b/g/n (RAHN, 2016, p. 18). Se comparado a outros equipamentos semelhantes, o AR.Drone 2.0 oferece facilidade de compra no mercado, pois tem um custo reduzido (SANTANA; BRANDÃO; SARCINELLI, 2019, p. 2).

Figura 1 - AR.Drone 2.0 Parrot



Fonte: Toman (p. 4, 2017).

Conforme Santana, Brandão e Sarcinelli (2016, p. 2) o AR.Drone 2.0 vem de fábrica com “[...] acelerômetros, giroscópios, magnetômetros, duas câmeras de vídeo e um computador de bordo que gerencia estes sensores e a rede de comunicação sem fio do veículo”.

1 google, e por
Atenção, não tem acento no por

2 Itálico.

3 2019 o u 2016?

4 gerência

2.1.1 Hardware 1

O hardware conta com especificações técnicas acima da média encontradas em drones disponíveis no mercado, sendo elas:

- a) gravação e *streaming* de vídeo em HD;
- b) câmera HD. 720p 30fps (*Frames Per Second*);
- c) lente grande angular: 92 ° na diagonal;
- d) perfil base de codificação H264;
- e) *streaming* de baixa latência;
- f) armazenamento de vídeo em tempo real com o dispositivo conectado na entrada **USB (Universal Serial Bus)**;
- g) foto JPEG (*Joint Photographic Experts Group*);
- h) armazenamento de vídeo em tempo real com Wi-Fi diretamente no seu dispositivo remoto.

Não coloca em subseções ... tem pouco texto dentro destas subseções.

2.1.2 Estrutura →
2.1.3 Eletrônica →
2.1.4 Motores →
2.1.5 Módulo GPS →

O hardware do AR.Drone 2.0 conta

3 entrada Universal Serial Bus (USB);

4 foto Joint Photographic Experts Group (JPEG);

5 Já a estrutura corporal do drone AR.Drone 2.0 conta

6 Remove este item.
Propaganda e não acrescenta informação.

7 carbono: peso

8 Remover.

9 bordo do AR.Drone 2.0 oferece

10 automática. Entre estes recursos se tem:

11 Seguindo a norma da ABNT sempre se descreve por extenso seguido da sigla entre parênteses.

As letras em maiúsculo da sigla devem ficar em maiúsculo no extenso. E somente estas devem ficar em maiúsculo.

Arrumar em todo o texto onde aparecem as siglas.

2.1.2 Estrutura

A estrutura corporal do drone conta com tecnologia mais robusta e resistente, pensado no uso geral tanto em campos de pesquisas quanto em projetos mais ousados. Essa estrutura conta com os seguintes diferenciais:

- a) experimentar seus truques mais ousados não desafiará esse design de ponta feito para durar;
- b) tubos de fibra de carbono: Peso total de 380g (gramas) com casco externo, 420g com casco interno;
- c) peças plásticas de nylon com alto teor de fibra de 30% de carga;
- d) espuma para isolar o centro inercial da vibração dos motores;
- e) casco de fibra de isopor injetado por um molde de metal escovado;
- f) nano-repelente a líquidos em sensores de ultrassom;
- g) totalmente reparável: todas as peças e instruções para reparo disponíveis na internet.

2.1.3 Eletrônica

A tecnologia de **bordo** oferece controle de precisão **extrema** e recursos de estabilização automática.

- a) processador ARM (*Advanced RISC Machine*) Cortex A8 de 1 GHz (*Gigahertz*) e 32 BITS (*Binary Digit*) com DSP (*Digital Signal Processor*) de vídeo de 800 MHz (*Megahertz*) TMS320DMC64x;

11

As letras em maiúsculo da sigla devem ficar em maiúsculo no extenso. E somente estas devem ficar em maiúsculo.

Arrumar em todo o texto onde aparecem as siglas.

- b) Linux 2.6.32;
- c) RAM (*Random Access Memory*) DDR2 (*Double Data Rate*) de 1 Gbit (*Gigabit*) a 200 MHz;
- d) USB 2.0 de alta velocidade para extensões;
- e) Wi-Fi b, g, n;
- f) giroscópio de 3 eixos 2000 ° / segundo de precisão;
- g) acelerômetro de 3 eixos + -50mg (Miligrama) de precisão;
- h) magnetômetro de 3 eixos 6 ° de precisão;
- i) sensor de pressão +/- 10 PA (*Precision and Accuracy*) de precisão;
- j) sensores de ultrassom para medição da altitude do solo;
- k) câmera QVGA (*Quarter Video Graphics Array*) vertical de 60 FPS para medição da velocidade do solo.

2.1.4 Motores

Os motores também chamados de *brushless* são essenciais para que o drone consiga decolar e executar o voo de forma a manter-se estável pairando ao ar, importante levar em consideração o total de potência máxima e mínima suportados ao enviar pulsos elétricos de forma variável.

Os motores suportam variações da quantidade de energia enviadas, com isso há total controle da placa principal chamada de mais ~~mais~~ ³ ² ~~board~~, enviando pulsos controlados aos motores, fazendo com que o drone consiga flutuar e locomover-se ao ar, aumentando e diminuindo a quantidade de energia enviada.

Algumas das características técnicas que envolvem o controle dos motores, além de suas próprias características:

- a) micro rolamento de esferas;
- b) motores *brushless* sem escova. 14.5W (*Watt*) 28.500 RPM (Rotação Por Minuto);
- c) engrenagens *Nylatron* de baixo nível de ruído para redutor de hélice 1 / 8,75;
- d) eixo da hélice em aço temperado;
- e) rolamento em bronze auto lubrificante;
- f) arrasto específico de alta propulsão para grande manobrabilidade;
- g) CPU (*Central Processing Unit*) de 8 MIPS (*Microprocessor Without Interlocked Pipeline Stages*) AVR (Alf-Egil Bogen e Vegard Wollan) por controlador de motor;
- h) parada de emergência controlada por software;

1
ar. É importante também levar .. evitar ter parágrafos longos e com uma só frase ..

2
board. A placa principal envia então

3
Remover.

- i) controlador do motor totalmente reprogramável;
- j) controlador eletrônico do motor resistente à água.

2.1.5 Módulo GPS

Para voar autonomamente para uma determinada coordenada geográfica, é necessário um dispositivo receptor GNSS (*Global Navigation Satellite System*) termo genérico padrão para sistemas de navegação por satélite que fornecem posicionamento geográfico espacial autônomo com cobertura global. Este termo inclui o GPS (*Global Positioning System*), GLONASS (*Global (Orbiting) Navigation Satellite System*), Galileo, Beidou e outros sistemas regionais.

O módulo visto na Figura 2 – Modulo de GPS Ar.Drone original da Parrot é conectado na USB da placa principal, com isso o drone passa a receber e administrar coordenadas geográficas, tal como latitude e longitude.

Figura 2 – Modulo de GPS Ar.Drone



Fonte: Toman (2017).

2.2 BIBLIOTECA NPM NODE-AR-DRONE

O node-ar-drone é uma biblioteca em Node.js desenvolvida com base no SDK (*Software Development Kit*) 2.0 oficial da Parrot, cujo objetivo principal destina-se exclusivamente para controlar o AR.Drone através de um computador conectado na rede WI-FI gerada pelo aparelho, enviando comandos escritos em JavaScript sendo interpretados pelo framework Node.js que envia e recebe pacotes através de conexão UDP (*User Datagram Protocol*) (GEISENDÖRFER, 2014).

A biblioteca conta com API de alto nível que abstrai as funcionalidades disponíveis no SDK (PISKORSKI, 2015), contendo as seguintes características exclusivas:



- a) módulo cliente que disponibiliza uma API de alto nível suportando a grande maioria das funções disponíveis para controlar o drone, de forma um tanto simples na usabilidade e interpretação logica;
- b) possibilidade de criar um pequeno programa autônomo que realiza comandos pré-programados via codificação JavaScript;
- c) disponibilidade de usar os sensores disponíveis no aparelho, de forma fácil bastando chamar métodos específicos, tal como obter altitude;
- d) possibilidade de acessar a câmera vertical e horizontal, salvando fotos e vídeos ou até transmitir em tempo real usando HTTP (*HyperText Transfer Protocol*) web servidor;
- e) acesso direto para enviar comandos UDP de baixo nível, através de métodos basta enviar os dados usando a porta 5556.

2.2.1 Características da API

O framework conta com uma API ampla em quantidades de métodos disponíveis, abstraindo algoritmos que teriam de ser desenvolvidos por completo, as características principais são: 

- a) `ardrone.createClient(opções)`:
 - retorna um novo objeto cliente, opções incluem:
 - ip: o IP (*Internet Protocol*) do drone. O padrão é '192.168.1.1',
 - frameRate: a taxa de quadros do PngEncoder. O padrão é 5,
 - imageSize: o tamanho da imagem produzido por PngEncoder. O padrão é nulo;
- b) `client.createREPL()`:
 - inicia uma interface interativa com todos os métodos cliente disponíveis no escopo ativo. Além disso, o cliente resolve a própria instância;
- c) `client.getPngStream()`:
 - retorna um objeto `PngEncoder` que emite `buffers` individuais de imagem  como eventos de 'dados'. Várias chamadas para esse método retornam o mesmo objeto. O ciclo de vida da conexão (por exemplo, reconectar com erro) é gerenciado pelo cliente;
- d) `client.getVideoStream()`:
 - retorna um objeto `TcpVideoStream` que emite pacotes de TCP (*Transmission Control Protocol*) como eventos de dados. As chamadas para esse método

Remove a sub-seção,
coloca o texto dentro
da seção 2.2

A biblioteca node-ar-drone conta
Fonte courier para
node-ar-drone

disponíveis, e suas
principais
características são:

PNG

1

2

3

4

retornam o mesmo objeto. O ciclo de vida da conexão (por exemplo, reconectar com erro) é gerenciado pelo cliente;

- e) `client.takeoff(retorno de chamada)`:
 - Define **1** o estado interno do voo como *true*, o retorno de chamada é invocado após o drone relatar que está pairando;
- f) `client.land(retorno de chamada)`:
 - Define **2** o estado interno do voo como *false*, o retorno de chamada é invocado após o drone relatar que chegou;
- g) `client.up(velocidade) / client.down(velocidade)`:
 - Faz **3** o drone ganhar ou reduzir a altitude, velocidade pode ser um valor de 0 a 1;
- h) `client.clockwise(velocidade) / client.counterClockwise(velocidade)`:
 - Faz **4** com que o drone gire no sentido horário e anti-horário, a velocidade pode ser um valor de 0 a 1;
- i) `client.front(velocidade) / client.back(velocidade)`:
 - Controla **5** com movimentos para frente e para trás de forma horizontal usando a câmera como ponto de referência, a velocidade pode ser um valor de 0 a 1;
- j) `client.left(velocidade) / client.right(velocidade)`:
 - Controla **6** com movimentos para direita e esquerda de forma horizontal usando a câmera como ponto de referência. A velocidade pode ser um valor de 0 a 1;
- k) `client.stop()`:
 - Define **7** todos os comandos de movimento do drone para 0, tornando-o efetivamente pairado no lugar;
- l) `client.calibrate(dispositivo)`:
 - Pede **8** ao drone para calibrar um dispositivo. Embora o **9** firmware **10** AR.Drone seja compatível apenas com um dispositivo que possa ser calibrado. Esta função também inclui FTRIM;
- m) magnetômetro:
 - O **11** magnetômetro só pode ser calibrado enquanto o drone está voando, e a rotina de calibração faz com que o drone gire no lugar a 360 graus;
- n) FTRIM:
 - O **12** FTRIM redefine essencialmente o Yaw, Pitch e Roll para 0. Tenha muito cuidado ao usar esta função e apenas calibre enquanto estiver em uma superfície plana. Nunca usar enquanto estiver voando;
- o) `client.config(chave, valor, retorno de chamada)`:

Iniciar item com letra minúscula.

Define →
Faz →
Faz →
Controla →
Controla →
Define →
Pede →
O →
O →
Envia →

Itálico.

1

2

- Envia um comando de configuração para o drone.

2.3 BIBLIOTECA ARDRONE-AUTONOMY

A biblioteca `ardrone_autonomy` é uma API de alto nível para o quadricóptero Parrot AR.Drone 1.0 e 2.0. Esta API foi construída com base a versão oficial do AR.Drone SDK 2.0.1. `ardrone_autonomy` é uma bifurcação da biblioteca anterior `node-ar-drone`. Este pacote foi desenvolvido por Laurent Eschenauer (ESCHENAUER, 2014).

2.3.1 Integração de driver com Ar Drone

Frequências de atualização do drone: a frequência de atualização da transmissão de dados do drone depende do parâmetro `navdata_demo`. Quando definido como 1, a frequência de transmissão é definida em 15Hz (*Hertz*), caso contrário, a frequência de transmissão é definida em 200Hz. (`navdata_demo` é um parâmetro numérico que não é booleano, portanto, usa-se (Verdadeiro / Falso) para definir / desabilitar).

Frequências de atualização do driver: O driver pode operar em dois modos: tempo real ou taxa fixa. Quando o parâmetro `realtime_navdata` é definido como *true*, o driver publica qualquer informação recebida instantaneamente. Quando definido como *false*, o driver armazena em cache os dados recebidos primeiro e depois os envia a uma taxa fixa. Essa taxa é configurada através do parâmetro `looprate`. A configuração padrão é: `realtime_navdata = false` (ESCHENAUER, 2014).

2.3.2 API de missões

Este módulo expõe uma API de alto nível para planejar e executar missões, concentrando-se em onde o drone deve ir em vez de seus movimentos de baixo nível, métodos disponibilizados:

a) `mission.createMission()`:

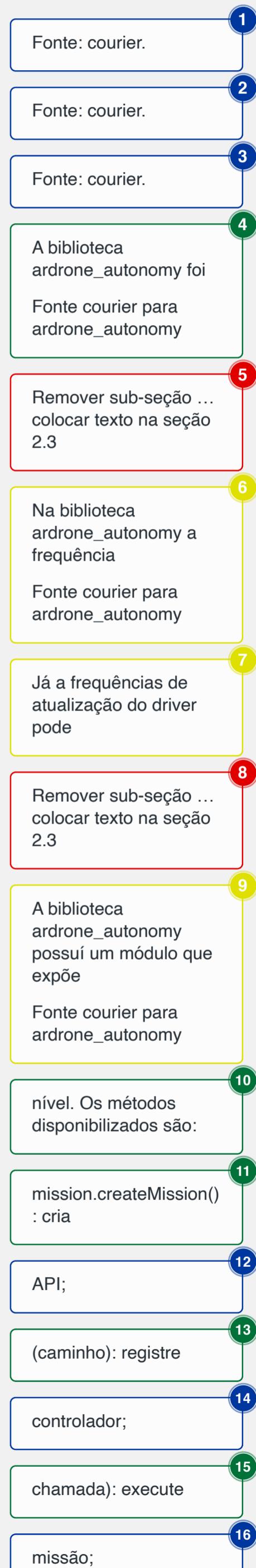
- Cria missão, disponibilizando acesso a todos métodos da API.

b) `mission.log(caminho)`:

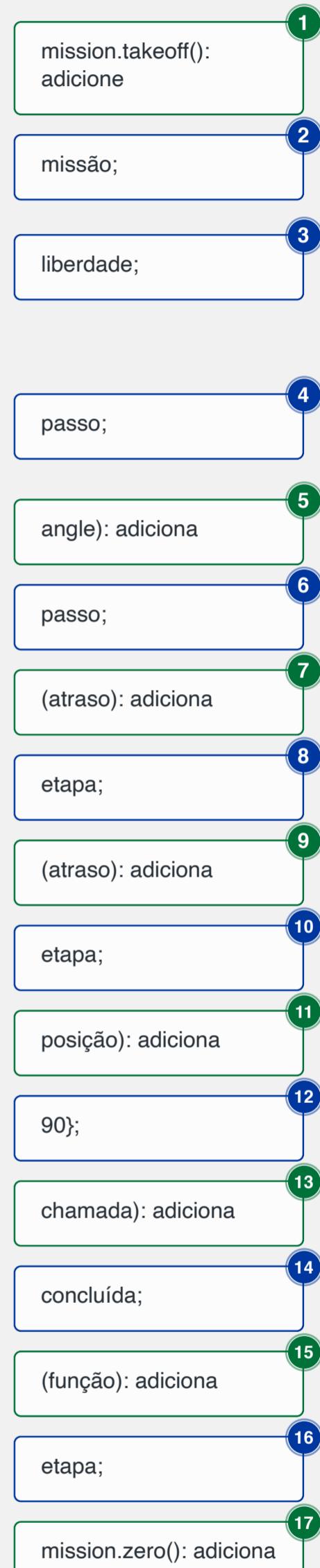
- Registre os dados da missão, no formato CSV (*Comma Separated Values*), no arquivo fornecido. Torna realmente útil depurar / plotar o estado e o comportamento do controlador.

c) `mission.run(retorno de chamada)`:

- Execute a missão. Retorno de chamada da `function(err, resultado)` e será acionado em caso de erro ou no final da missão.



- d) `mission.takeoff()` :
 - Adicione uma etapa de decolagem à missão.
- e) `mission.forward / backward / left / right / up / down (distância)` :
 - Adiciona um passo de movimento à missão. O drone se moverá na direção especificada pela distância (em metros) antes de prosseguir para o próximo passo. O drone também tentará manter todos os outros graus de liberdade.
- f) `mission.altitude(altura)` :
 - Adiciona um passo de altitude à missão. Subirá até a altura especificada antes de prosseguir para o próximo passo.
- g) `mission.cw/ccw(angle)` :
 - Adiciona uma etapa de rotação à missão. Girará pelo ângulo fornecido (em graus) antes de prosseguir para o próximo passo.
- h) `mission.hover(atraso)` :
 - Adiciona uma etapa flutuante à missão. Irá pairar no local pelo atraso especificado em MS (Milissegundos) antes de prosseguir para a próxima etapa.
- i) `mission.wait(atraso)` :
 - Adiciona um passo de espera para a missão. Esperará o atraso especificado (em MS) antes de prosseguir para a próxima etapa.
- j) `mission.go(posição)` :
 - Adiciona um passo de movimento à missão. Irá para a posição especificada antes de prosseguir para a próxima etapa. A posição é um objetivo do controlador, como {x: 0, y: 0, z: 1, yaw: 90}.
- k) `mission.task(função (retorno de chamada))` :
 - Adiciona uma etapa da tarefa à missão. Irá executar a função fornecida antes de prosseguir para a próxima etapa. Um argumento de retorno de chamada é passado para a função, que deve ser chamado quando a tarefa estiver concluída.
- l) `mission.taskSync(função)` :
 - Adiciona uma etapa da tarefa à missão. Irá executar a função fornecida antes de prosseguir para a próxima etapa.
- m) `mission.zero()` :
 - Adiciona um passo de *reset* à missão. Isso definirá a posição / orientação atual como o estado base do filtro Kalman (ou seja, {x: 0, y: 0, yaw: 0}). Se não estiver usando uma etiqueta como sua posição base, convém `zero()` após a decolagem.



2.4 BIBLIOTECA GEOLIB

Biblioteca para fornecer operações geoespaciais básicas como cálculo de distância, conversão de coordenadas decimais em sexagesimal e vice-versa. Atualmente esta biblioteca é 2D, o que significa que a altitude / elevação ainda não é suportada por nenhuma de suas funções.

A biblioteca é escrita em TypeScript. Não é necessário conhecer o TypeScript para usar o Geolib, mas as definições de tipo fornecem informações valiosas sobre o uso geral, parâmetros de entrada.

Valores e formatos suportados, todos os métodos que estão trabalhando com coordenadas aceitam um objeto com uma propriedade lat / latitude e lon / lng / longitude ou uma matriz de coordenadas GEO (*Geographic Location*) JSON (*Javascript Object Notation*), como: [longitude, latitude]. Todos os valores podem estar no formato decimal (53.471) ou sexagesimal (53 ° 21' 16"). Os valores da distância são sempre flutuantes e representam a distância em metros (BIEH, 2020).

2.4.1 Principais funções disponíveis

As principais funções disponíveis na API podem caracterizar-se nas seguintes:

a) getDistance (início, fim, precisão = 1):

- Calcula a distância entre duas coordenadas geográficas. Esta função leva até 3 argumentos. Os dois primeiros argumentos devem ser Geolib *input coordinates* válidos (por exemplo, {latitude: 52.518611, longitude: 13.408056}).
- As coordenadas podem estar no formato sexagesimal ou decimal. O terceiro argumento é precisão (em metros). Por padrão, a precisão é de 1 metro. Se você precisar de um resultado mais preciso, poderá defini-lo como um valor mais baixo, por exemplo 0,01 para precisão de centímetros.
- Você pode configurá-lo mais alto para que o resultado seja arredondado para o próximo valor divisível pela precisão escolhida (por exemplo, 25428 com uma precisão de 100 se torna 25400).

b) getPreciseDistance (start, end [int precision]):

- Calcula a distância entre duas coordenadas geográficas. Esse método é mais preciso que o `getDistance`, especialmente para longas distâncias, mas também é mais lento. Ele está usando a fórmula inversa Vincenty para elipsoides. Ele usa os mesmos argumentos (até 3) que `getDistance (início, fim, precisão = 1)`.

c) getGreatCircleBearing(origin, destination):

1
Remover sub-seção ...
colocar texto na seção
2.4

2
na API da biblioteca
Geolib são:

3
Arruma o formato
destes itens no mesmo
modo dos itens
anteriores.

- Cálculo que obtém o ângulo horizontal entre duas coordenadas considerando o círculo angular norte a sul entre dois pontos conectados na terra, esse conceito chama-se grande círculo.

2.5 API GOOGLE MAPS

A arquitetura que **foi proposta** visava a necessidade de abrir o mapa para visualização e escolha da rota a ser percorrida pelo **drone**, realizei diversos **testes** de implementação com APIs **disponíveis open-source**, porém como validado nos testes, identifiquei como promissora que contemplou as funções necessárias e tempo de resposta ligeiramente melhor que as demais foi a API Google Maps (MAPS, 2020).

O formato escolhido de visualização na abertura do mapa foi por satélite, essa API requer conexão com a internet para abertura inicial do mapa, carregando uma parte do mapa num perímetro de aproximadamente 5km².

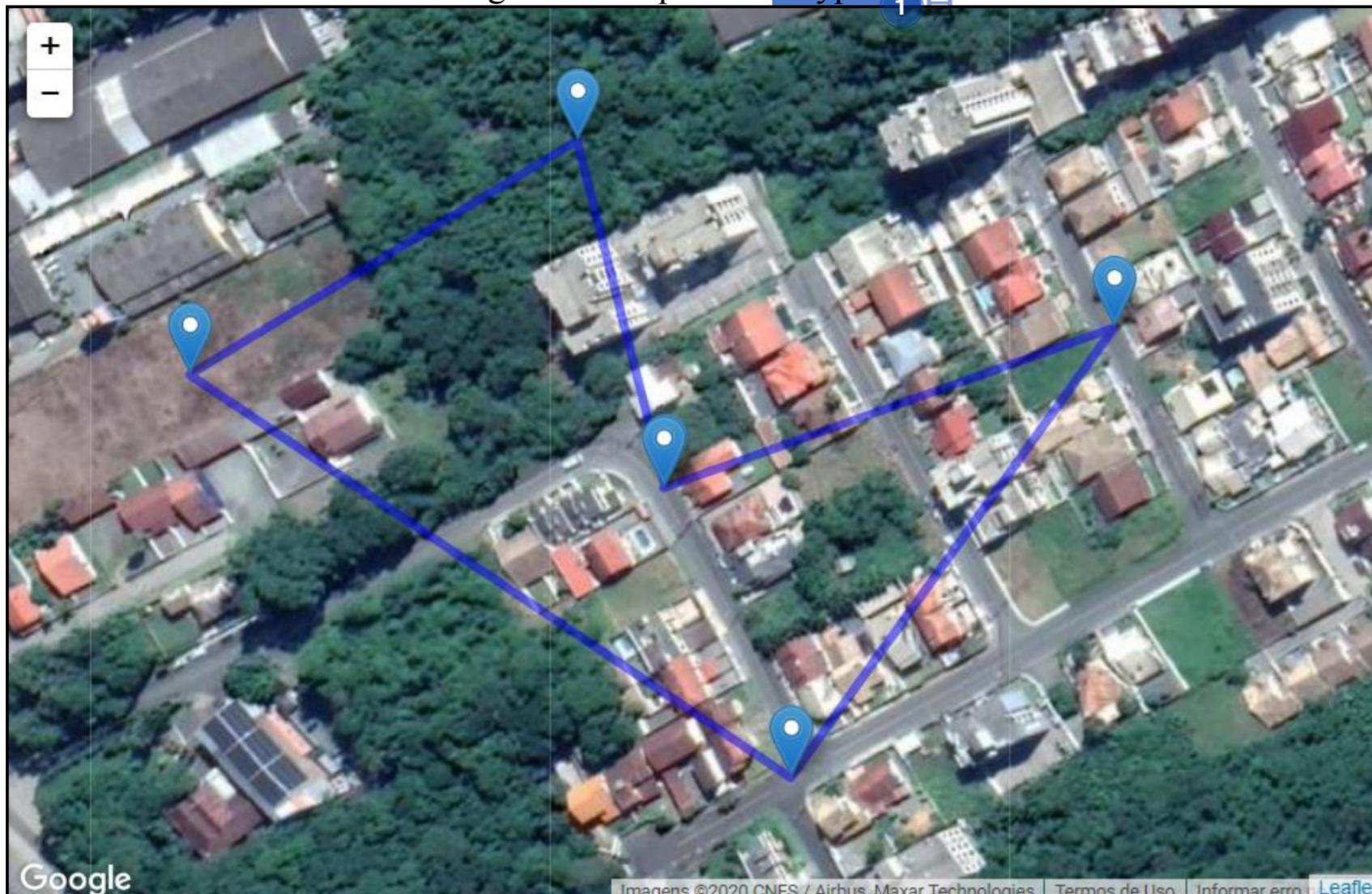
2.5.1 LEAFLET

Para criar as marcações de **waypoints** e traços da rota a ser percorrida como pode ser visto na **Figura 3 - Mapa com waypoints** foi integrado a principal biblioteca JavaScript de código aberto para mapas interativos compatíveis com dispositivos móveis e navegadores. Com apenas 38 KB (*Kilobyte*) de JS (JavaScript), ela possui todos os recursos de mapeamento de forma off-line.

Foi projetado com simplicidade, desempenho e usabilidade em mente. Ele **funciona** de maneira eficiente em todas as principais plataformas móveis e de desktop, pode ser estendido com muitos plugins, possui uma API bonita, fácil de usar e bem documentada e um código-fonte simples e legível. (LIEDMAN, 2020).

- 1 que foi proposta
- 2 Desta forma foi realizado testes
- 3 disponíveis no formato open-source
- 4 Ops, arruma a frase: impessoal, verno no passado, menos exagero nas afirmações.
- 5 Frase muito longa. Só uma frase por parágrafo.
- 6 Remover sub-seção ... texto na seção 2.5
- 7 Itálico.
- 8 Figura 3 foi
- 9 Frase longa. Só uma frase no parágrafo.
- 10 Ela
- 11 Quem ...
- 12 Remover ponto final.

Figura 3 - Mapa com waypoints



Fonte: Elaborado pelo autor (2020).

Itálico.

2.5.1.1 Principais métodos da biblioteca

Os principais métodos da Leaflet para o funcionamento adequado, garantindo uma usabilidade amigável e contendo os recursos necessários são:

- L.map('map').setView([latitude, longitude], 20):
 - Esse método gera um mapa com a visualização inicial através das coordenadas de latitude e longitude passadas como parâmetro, além de atribuir o zoom em 20 metros de altitude;
- L.Google('SATELLITE').addLayer(googleLayer):
 - Esse método adiciona através da API do Google Maps o formato de satélite, ou seja, abre o mapa com visualização satélite;
- L.marker([latitude, longitude], { ícone: objetoÍcone }).addTo(map):
 - Esse método adiciona ao mapa já aberto o ícone desejado, na posição da latitude e longitude passadas no parâmetro.

Remover sub-seção ..
texto na seção 2.5

Arrumar formato dos
itens seguindo exemplo
acima.

2.6 TRABALHOS CORRELATOS

A seguir serão apresentados três trabalhos correlatos. Na seção 2.1 será abordado o trabalho de conclusão de curso de Vanz (2015) que disponibiliza um protótipo de módulo de

2.6.1

seção 2.6.1 aborda

1

2

3

4

5

integração com ROS (*Robot Operation System*). Na seção 2.2  será apresentado o trabalho de conclusão de curso de Rocha (2016) que consiste no sistema móvel multiplataforma para navegação em rotas internas. Para finalizar na seção 2.3  será apresentado o trabalho de Barrow (2014) que procura integrar a navegação e busca autônoma em ambiente interno usando um drone.

2.6.1 Visedu-drone: módulo de integração com ROS

Vanz (2015) tem como objetivo criar um simulador de drone integrado com o *framework* para robótica ROS. O simular estende o VisEdu e foi desenvolvido na linguagem JavaScript utilizando a biblioteca Three.js para abstrair o WebGL e facilitar a manipulação do ambiente virtual. Esse simulador possibilita alterar o comportamento do drone físico simultaneamente, possibilitando ao usuário simular na prática os eventos iguais da realidade. Para controlar o drone físico foi utilizado o driver `ardrone_autonomy`, esse driver efetua a comunicação entre o ROS e o drone. Para disponibilizar a execução através de um navegador web foi utilizado o WebSocket da Rosbridge. Por fim foi constatado que no sistema de navegação implementado, ocorre uma deficiência no tempo de execução ao percorrer o espaço que fora atribuído (VANZ, 2015).

 Na Figura 4 – Arquitetura VisEdu-Drone é possível observar a arquitetura da aplicação, dividida em duas camadas sendo a primeira referente ao que é executado no navegador do usuário, consistindo na interface gráfica onde o usuário interage e visualiza a cena, executando as animações e controlando o drone. Na segunda camada ocorre a comunicação com o AR.Drone Parrot, a camada destacada em verde corresponde à aplicações que rodam no ecossistemas do ROS, sendo subdividida em dois pacotes principais, Rosbridge  e driver  para AR.Drone (VANZ, 2015).



Fonte: courier

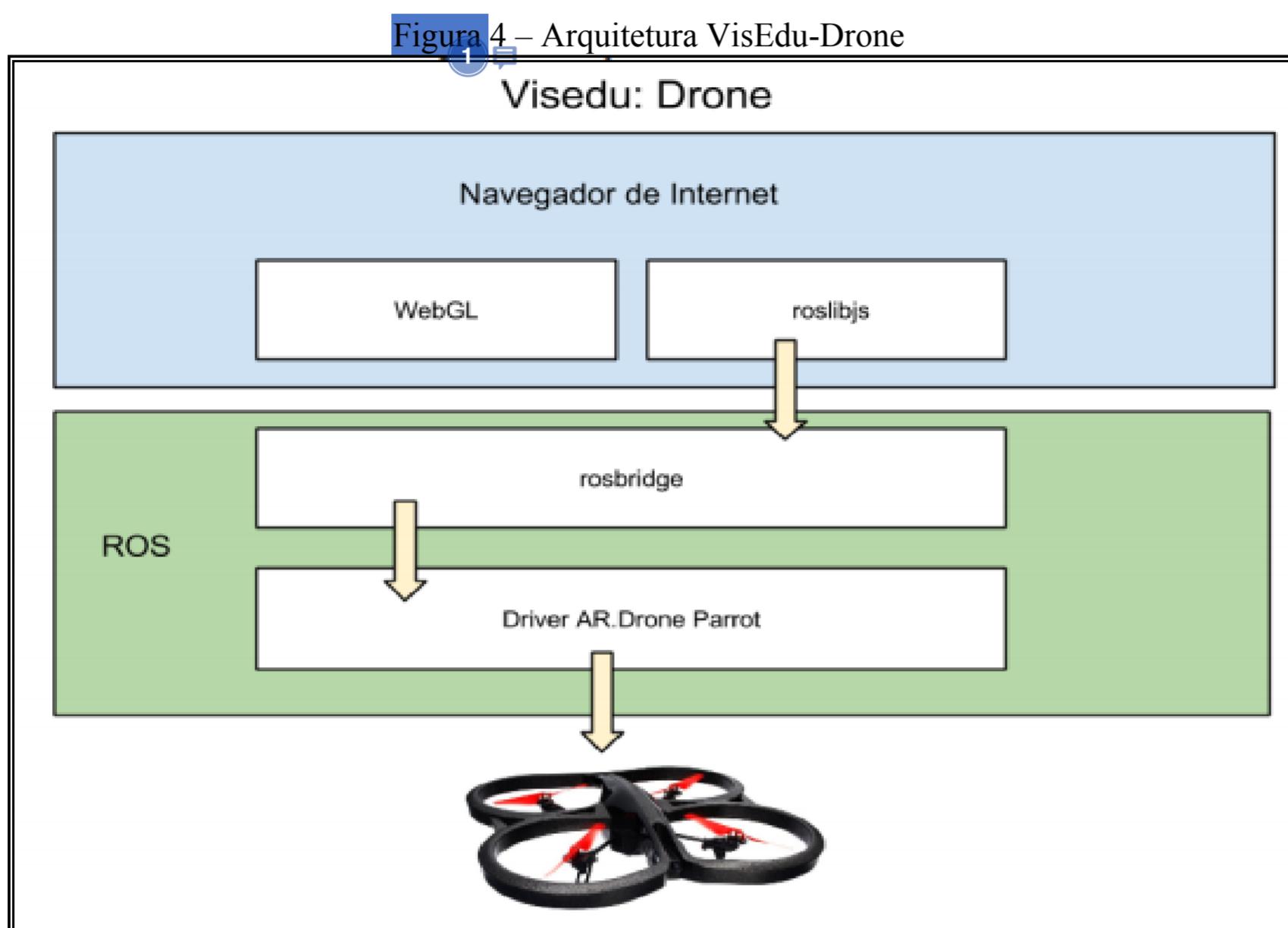
Figura 4 é

Itálico.

5

6

7



Fonte: Vanz (2015).

2.7 FURB MOBILE: SISTEMA MÓVEL MULTIPLATAFORMA PARA NAVEGAÇÃO EM ROTAS INTERNAS

O trabalho de Rocha (2016) trouxe o desenvolvimento de um aplicativo multiplataforma para auxiliar na locomoção dos visitantes pelo campus da FURB (Universidade Regional de Blumenau) em dias do evento Interação FURB. Este aplicativo permite buscar e localizar locais específicos, como laboratórios e salas, por exemplo e foi construído com o framework PhoneGap, utilizando de tecnologias web, como HTML ([Hypertext Markup Language](#)), CSS ([Cascading Style Sheets](#)) e JavaScript, com o auxílio da biblioteca AngularJS (ROCHA, 2016).

A fim de dar suporte as funcionalidades do aplicativo, foi construída uma aplicação servidora que dispõe informações pela web através de uma interface RESTful, além de uma ferramenta para a administração dos mapas e de possíveis rotas pelas partes internas dos blocos do campus. Para a construção de um ambiente gráfico para a apresentação e edição dos mapas, além da apresentação de rotas nestes mapas, foi utilizado a biblioteca Three.js, com isso permitiu a apresentação e importação das plantas baixas dos blocos em arquivos em formato [OBJ](#) ([Object File Wavefront](#)) (ROCHA, 2016).

Diminuir um pouco o tamanho da figura para ver se cabe na página anterior.

Assim evita espaços em branco no texto.

HyperText

Arrumar em todo o texto ... ABNT ... extenso seguido da sigla entre parênteses.

OBJect file wavefront

1

2

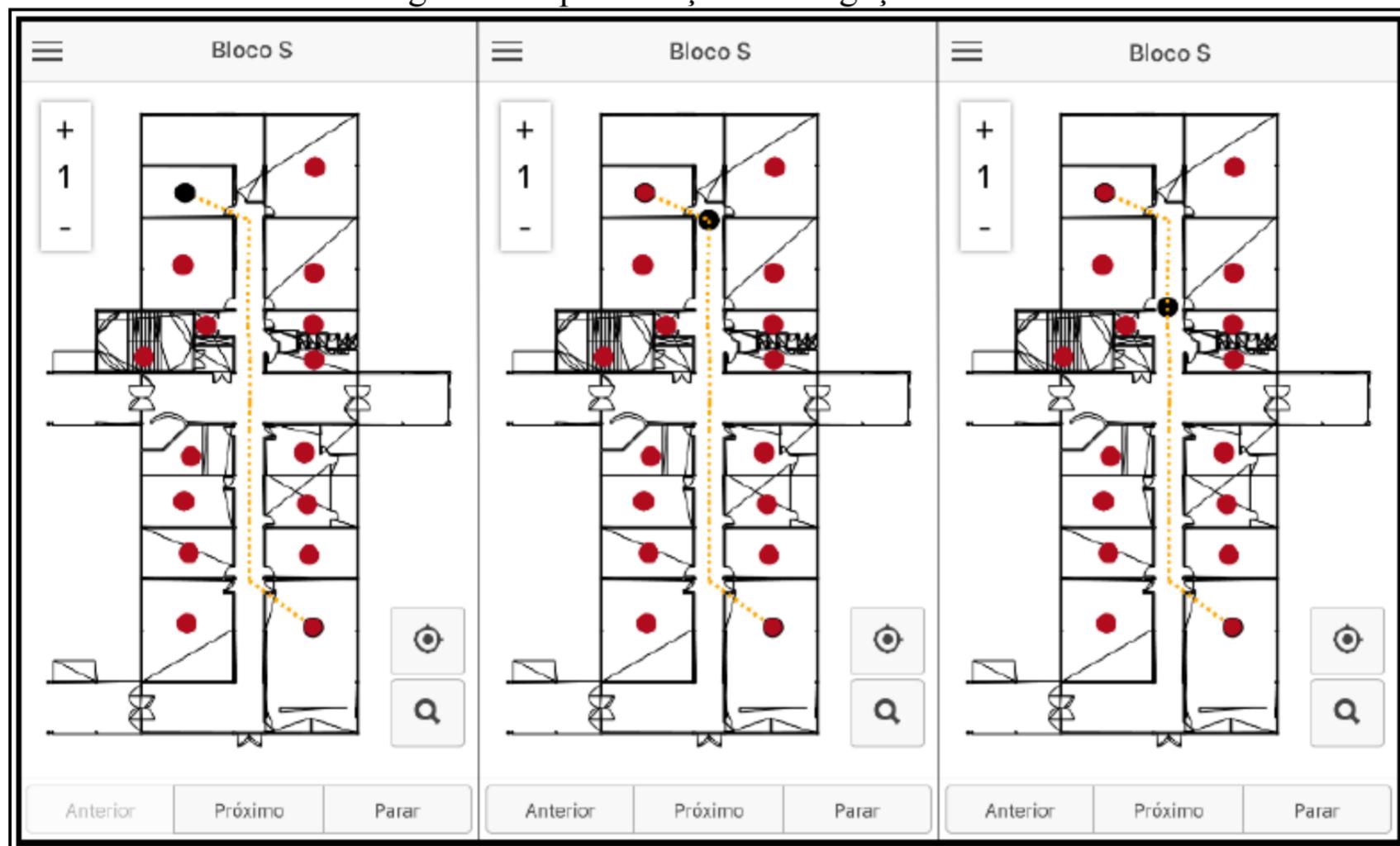
3

4

Na Figura 5 - Apresentação e navegação na rota há a apresentação de uma rota de menor custo entre os pontos de origem e destino que foram selecionadas, sendo esta, apresentada ao usuário do aplicativo em forma de uma linha pontilhada. Na parte inferior da tela do aplicativo, estão as ações de navegação, no qual o usuário poderá navegar na rota, através das ações posterior e anterior. Caso o usuário queira parar a navegação e voltar ao estado de navegação do mapa deve utilizar a ação parar. Se a opção de rota do usuário tenha como destino outro pavimento ou bloco, este será apresentado também em linha pontilhada (ROCHA, 2016).

5 há

Figura 5 - Apresentação e navegação na rota



Fonte: Rocha (2016).

2.8 AUTONOMOUS NAVIGATION AND SEARCH IN AN INDOOR ENVIRONMENT USING AN AR.DRONE

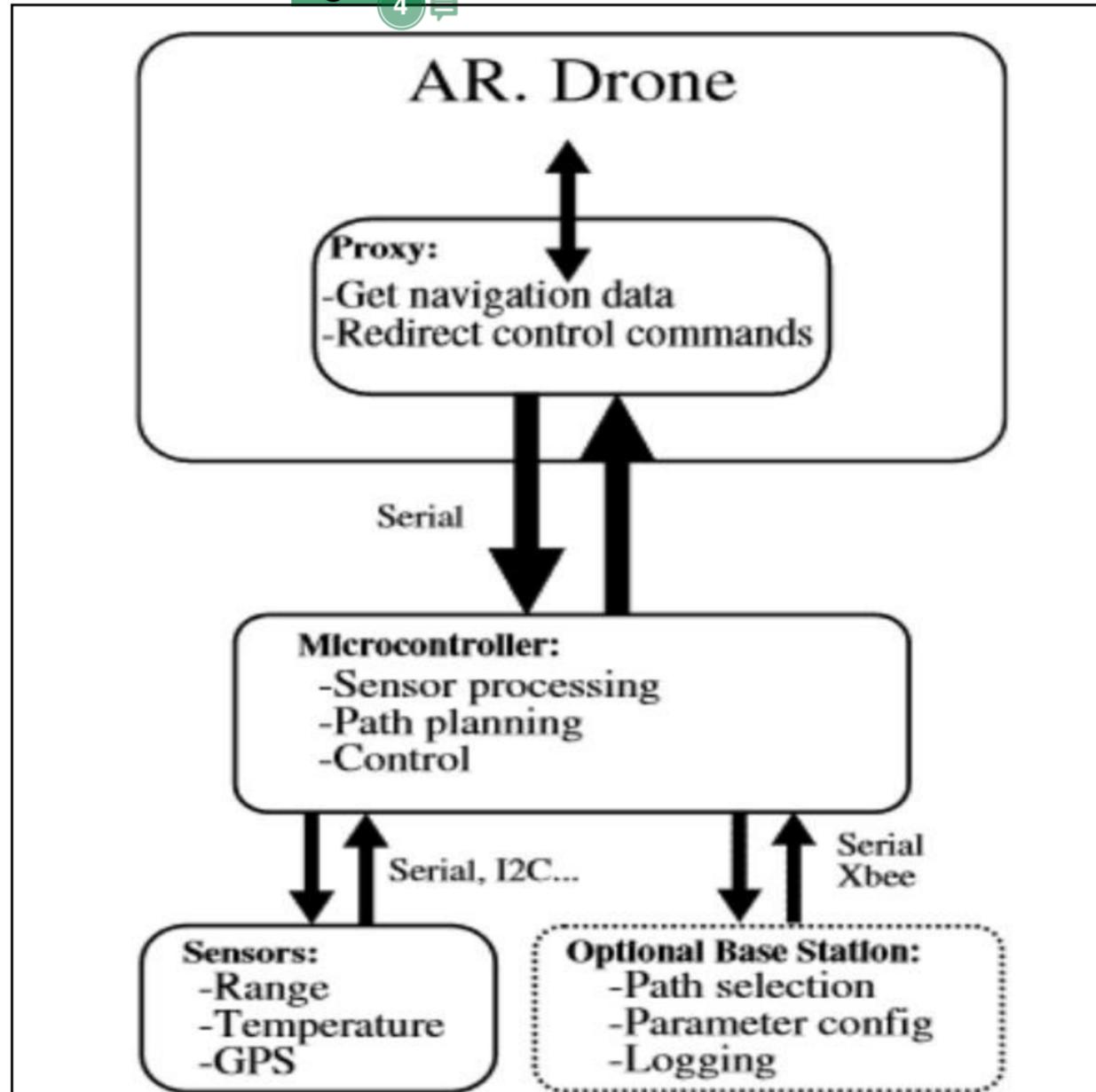
A dissertação de mestrado de Barrow (2014) tem como objetivo entregar um drone autônomo que consegue navegar, buscar e identificar objetos em lugares desconhecidos, excluindo a necessidade de uma pessoa ficar controlando-o. Possui algoritmos de processamento visual que podem ser usados com AR.Drone para identificar objetos e cores, além de processar a rota em tempo real mantendo-se no ar mesmo nos lugares dos quais ainda não foram processados. Ao final de cada processamento o experimento entrega os resultados com possíveis soluções para navegação e busca autônoma.

Na Figura 6 - *Work Breakdown Structure* é apresentada a arquitetura do sistema usada para controlar o drone. O bloco ao meio mostra o micro controlador adaptado ao drone, esse micro

controlador processa os dados dos sensores e traça um caminho, esse processo ocorre antes dos comandos de voo enviados ao drone. Também apresenta a opção de escolher uma base para o drone realizar o log dos dados e descarregá-los através de uma conexão do servidor.

Tradução da Figura 6 - Work Breakdown Structure - Proxy: Obtém os dados de navegação, redireciona os comandos de controle. Micro controlador: Processamento de sensores, planejamento de caminho, controle. Sensores: Distância, temperatura, sistema de posicionamento global (GPS). Estação de base opcional: Seleção de caminho, configurações de parâmetros, log de dados.

Figura 6 - Work Breakdown Structure



Fonte: Barrow (2014).

registro

O QUE É ISTO ???

Arrumar o texto deste parágrafo ... formato.

O texto da figura deve ser “traduzido” na própria figura.

Diminuir um pouco o tamanho desta figura.

1

2

3

4

3 DESENVOLVIMENTO DA ARQUITETURA

Este capítulo apresenta toda estrutura de desenvolvimento da arquitetura, na primeira seção os requisitos funcionais e não funcionais. Na segunda seção os casos de uso e diagramas de atividade usando UML (*Unified Modeling Language*).

A terceira seção detalha toda estrutura usada na implementação da arquitetura, algoritmos aplicados a cálculos de rotas aéreas, arquitetura de conversação entre cliente e serviços, técnicas de transporte de pacotes e mensagens através de redes TCP e UDP, métodos de interpretação da conexão com o hardware do drone.

Por fim na quarta e última seção os resultados e discussões obtidos durante todo o cílico da pesquisa, projeto e trabalho.

3.1 REQUISITOS PRINCIPAIS

Os seguintes requisitos fazem parte da arquitetura:

- a) disponibilizar um sistema web para cadastro de rotas (RF01);
- b) a arquitetura deverá permitir o cadastro de uma base para cada rota (RF02);
- c) a arquitetura deverá gerar um relatório para cada rota a partir das informações obtidas pelo drone (RF03);
- d) disponibilizar recurso de controle ³ porcentagem da carga da bateria do drone (RNF01);
- e) a arquitetura deverá ser desenvolvida em Node.js (RNF02);
- f) a arquitetura deverá possuir integração com a biblioteca NPM Geolib (RNF03);
- g) a arquitetura deverá utilizar cálculos matemáticos para melhorar sua localização (RNF04);
- h) o drone deverá possuir um GPS e gravar as coordenadas da rota (RNF05).

1 UDP e métodos

2 Arrumar.

3 Hum, ficou confuso ...
não tens como ter o
“controle” sobre o
percentual da bateria.

3.2 ESPECIFICAÇÃO

O trabalho segue com a especificação no formato UML, casos de uso exemplificados através de diagramas de atividades, contendo o detalhamento de cada cenário da execução, para elaborar os diagramas foi utilizado o software draw.io para desktop.

4 Frase longa ... arrumar.

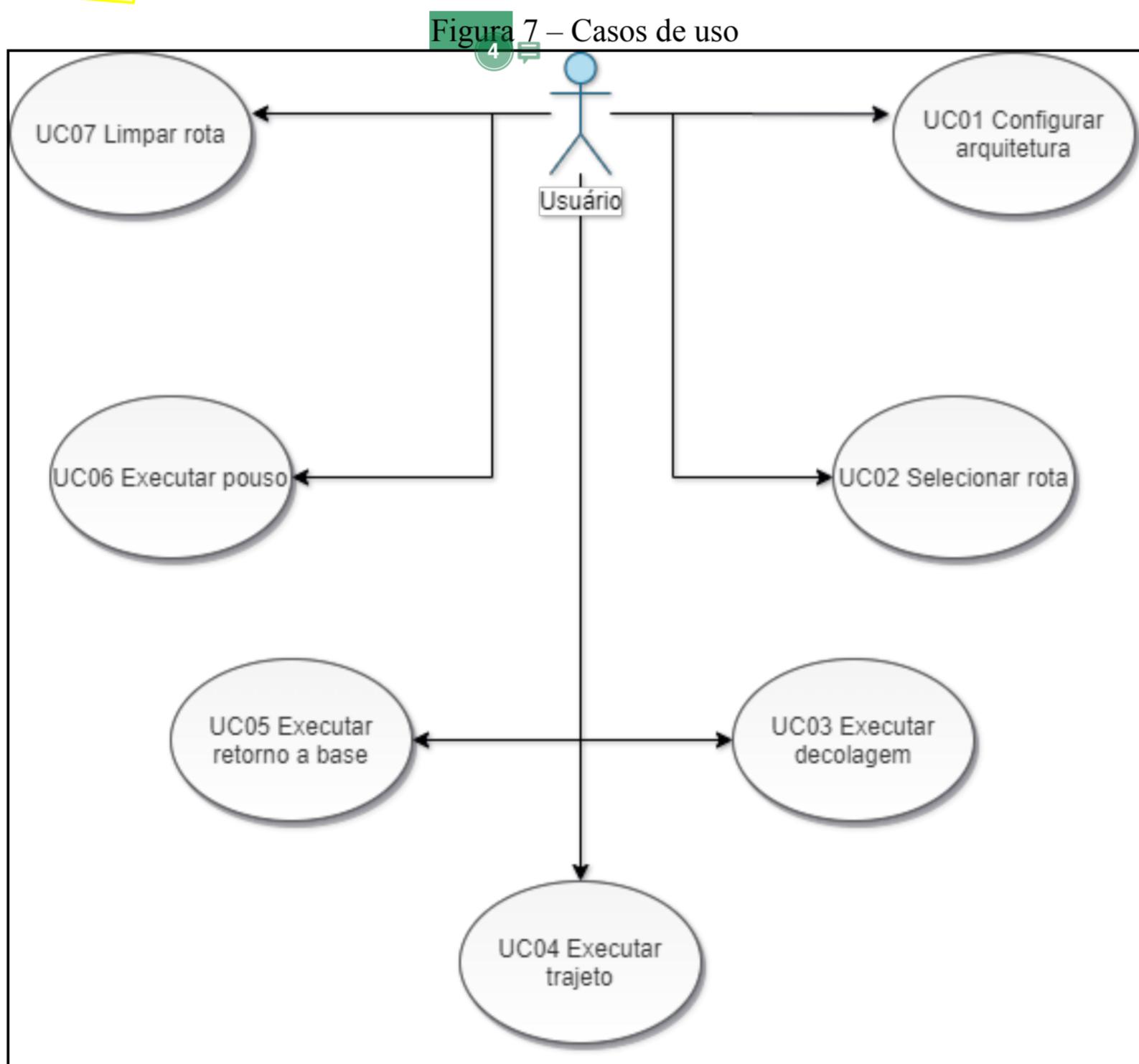
5

5 Evitar espaço em branco no texto.

3.2.1 Casos de uso

Os casos de uso a seguir têm como objetivo exemplificar funcionalidades da arquitetura em alto nível, o ator usuário representará a entidade de interação que utilizará a interface de pilotagem do drone, cada ação está associada ao seu respectivo ator.

Na Figura 7 – Casos de uso será apresentado os através do diagrama de UC (*User Cases*) detalhes das ações que o ator poderá executar através da arquitetura disponibilizada, as etapas serão executadas e controladas através de uma interface web gerada pela arquitetura, desenvolvida respeitando os conceitos mais atuais no quesito UX (*User Experience*) e UI (*User Interface*).



Fonte: Elaborado pelo autor (2020).

- 1 Frase longa ... arrumar.
Só uma frase no parágrafo.
- 2 Arrumar frase.
- 3 Frase longa ... arrumar.
Só uma frase no parágrafo.
- 4 Diminuir um pouco o tamanho da figura.

A seguir será apresentada as descrições de cada UC seguidos de seu respectivo cenário de execução detalhado utilizando diagramas de atividades.

3.2.1.1 Caso de uso: Configurar arquitetura

Inicialmente antes de realizar qualquer missão de voo é preciso configurar a arquitetura através da interface gerada pela **arquitetura, configurações** 1

- marcar se o drone deve virar sua câmera frontal em direção aos pontos selecionados como detalhado no **Quadro 2 – Caso de uso selecionar rota**.
- inserir altitude inicial que o drone deverá subir ao realizar a etapa como descrito no **Quadro 3 – Caso de uso executar decolagem**.
- marcar se deverá realizar a calibração do magnetômetro antes de iniciar o voo.

No **Quadro 1 - Caso de uso configurar** apresenta-se os detalhes da tarefa que define as configurações disponíveis e liberadas ao usuário, logo após é apresentada na **Figura 8 – Atividade de configuração d os detalhes exemplificados através de diagrama de atividades**.

Quadro 1 - Caso de uso configurar arquitetura
UC01. Configurar arquitetura

Descrição	Permitir configurar os parâmetros iniciais da arquitetura.
Requisitos atendidos	RF02, RF01
Pré-condição	Conectado ao drone.
Cenário principal	<ol style="list-style-type: none"> A interface exibe os campos editáveis; O usuário altera os campos; A arquitetura salva os valores de cada campo; O usuário visualiza conferindo os dados salvos na interface.
Fluxo alternativo	No passo 2, o usuário poderá ignorar os campos editáveis e selecionáveis. <ol style="list-style-type: none"> A arquitetura irá atribuir os valores padrões.
Pós-condição	A arquitetura atualiza os valores padrões.

Fonte: Elaborado pelo autor (2020).

Arrumar frase.

1

Arrumar pontuação final dos itens.

2

Frase longa ... arrumar.
Só uma frase no parágrafo.

3

Parei de revisar aqui ...

4

REFERÊNCIAS

- AGUIAR, João Filipe Vieira. **Transferência de Energia sem fios para carregamento de baterias.** 2013. Tese de Doutorado.
- ARRACHEQUESNE, Damien et al. (Ed.). **SOCKET.IO.** 2020. Disponível em: <<https://www.npmjs.com/package/socket.io>>. Acesso em: 16 mai. 2020.
- BARROW, Erik. **Autonomous Navigation and Search in an Indoor Environment Using AR Drone.** 2014. 118 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Coventry University, Blumenau, 2020.
- BIEH, Manuel (Ed.). **GEOLIB.** 2020. Disponível em: <<https://www.npmjs.com/package/geolib>>. Acesso em: 08 jun. 2020.
- BRANDINO, Wandreson Luiz. Apostila TCP/IP. **São Paulo**, p. 4, 1998.
- BRUNO, Fernanda Gloria. **Contra-manual para câmeras inteligentes: vigilância, tecnologia e percepção.** Galáxia. Revista do Programa de Pós-Graduação em Comunicação e Semiótica. ISSN 1982-2553, n. 24, 2012.
- BRUNO, Fernanda. **Estética do flagrante: controle e prazer nos dispositivos de vigilância contemporâneos.** Revista Cinética, 2008.
- CARTER, Carl. Great circle distances. 2002.
- DE FARIA, Rodrigo Ribeiro; COSTA, Marledo Egidio. A inserção dos veículos aéreos não tripuláveis (drones) como tecnologia de monitoramento no combate ao dano ambiental. **Revista Ordem Pública**, v. 8, n. 1, p. 81-103, 2015.
- ESCHENAUER, Laurent (Ed.). **ARDRONE-AUTONOMY.** 2014. Disponível em: <<https://www.npmjs.com/package/ardrone-autonomy>>. Acesso em: 03 abr. 2020.
- ESTÊVÃO, Tiago Vaz. O Novo Paradigma da Vigilância na Sociedade Contemporânea- "Who Watches the Watchers". **Observatorio (OBS*)**, v. 8, n. 2, p. 155-169, 2014.
- GEISENDÖRFER, Felix et al. (Ed.). **AR-DRONE.** 2014. Disponível em: <<https://www.npmjs.com/package/ar-drone>>. Acesso em: 03 abr. 2020.
- HAHN, Evan. Express in Action: Writing, building, and testing Node.js applications. Manning Publications 2016.
- LIEDMAN, Per et al. (Ed.). **LEAFLET.** 2020. Disponível em: <<https://www.npmjs.com/package/leaflet>>. Acesso em: 11 mai. 2020.
- Maps JavaScript API (Ed.). **GOOGLE MAPS.** 2020. Disponível em: <<https://developers.google.com/maps/documentation/javascript/tutorial>>. Acesso em: 12 jun. 2020.
- MONKEN, Maurício; BARCELLOS, Christovam. Vigilância em saúde e território utilizado: possibilidades teóricas e metodológicas. **Cadernos de Saúde Pública**, v. 21, p. 898-906, 2005.
- PISKORSKI, Stephane et al. (Ed.). **AR.DRONE DEVELOPER GUIDE.** 2012. Disponível em: <<https://forum.developer.parrot.com/t/ar-drone-2-0-sdk-for-android-on-windows/465>>. Acesso em: 03 mar. 2020.

Este trabalho é de Blumenau??

RAHN, RAFAEL RONALDO. **ESTABELECIMENTO DE ROTAS PARA AR. DRONE UTILIZANDO DELPHI 10 SEATTLE.** 2016. 73 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2020.

ROCHA, Marcus Otávio. **FURB-MÓBILE:** Sistema Móvel Multiplataforma para Navegação Em Rotas Internas. 2016. 61 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2020.

SANTANA, Lucas Vago; BRANDAO, Alexandre Santos; SARCINELLI FILHO, Mario. **SISTEMA PARA ESTIMACAO E CONTROLE DA POSICAO 3D DE UM QUADRICOTOR EM AMBIENTES INTERNOS.** 2016. 61 f. TCC (Pós-Graduação) - Curso de engenharia elétrica, Centro de Centro Tecnológico, Universidade Federal do Espírito Santo.

SCHAEFER (Blumenau). Assessora de Comunicação (Ed.). **Superávit da Oktoberfest é usado para compra de drone para a PM de Blumenau.** 2018. Disponível em: <<http://oktoberfestblumenau.com.br/noticias/superavit-da-oktoberfest-e-usado-para-compra-de-drone-para-a-pm-de-blumenau/>>. Acesso em: 13 abr. 2020.

SHIRATSUCHI, L. S. O avanço dos drones. **Embrapa Agrossilvipastoril-Artigo de divulgação na mídia (INFOTECA-E),** 2014.

TOMAN, David (Ed.). **PARROT AR.DRONE 2.0 REVIEW.** 2017. Disponível em: <<https://www.ign.com/articles/2017/04/17/parrot-ardrone-20-review>>. Acesso em: 15 abr. 2020.

VANZ, José Guilherme. **VISEDU-DRONE: Módulo de Integração com Robot Operating System.** 2015. 89 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2020.

WEISSHUHN, Bernhard et al. (Ed.). **DRONESTREAM.** 2013. Disponível em: <<https://www.npmjs.com/package/dronestream>>. Acesso em: 15 abr. 2020.

WILSON, Douglas et al. (Ed.). **EXPRESS.** 2019. Disponível em: <<https://www.npmjs.com/package/express>>. Acesso em: 16 mai. 2020.