**UNIVERSIDADE FEDERAL DE SANTA CATARINA**

**CENTRO TECNOLÓGICO**

**DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS**

**Acadêmico**

Título: subtítulo (se houver)

Florianópolis

201X

**Acadêmico**

Título: subtítulo (se houver)

Relatório submetido à Universidade Federal de Santa Catarina como requisito para a aprovação na disciplina **DAS 5511: Projeto de Fim de Curso** do curso de Graduação em Engenharia de Controle e Automação.

Orientador(a): Prof. \_\_\_\_\_\_

Co-orientador(a) (se houver): Prof. \_\_\_\_\_\_\_\_\_

Florianópolis

201X

**Acadêmico**

Título: subtítulo (se houver)

Esta monografia foi julgada no contexto da disciplina DAS5511: Projeto de Fim de Curso e aprovada na sua forma final pelo Curso de Engenharia de Controle e Automação.

Florianópolis, \_\_\_\_\_\_ de \_\_\_\_\_\_\_\_\_\_ de \_\_\_\_\_

**Banca Examinadora:**

<nome do orientador na empresa/instituto>

Orientador na Empresa

Nome da Empresa

Prof. <nome do orientador na UFSC>

Orientador no Curso

Universidade Federal de Santa Catarina

Prof. <nome do professor avaliador>

Avaliador

Universidade Federal de Santa Catarina

<nome aluno 1>

Debatedor

Universidade Federal de Santa Catarina

<nome aluno 2>

Debatedor

Universidade Federal de Santa Catarina

***Dedicatória*** (Opcional)

**Agradecimentos** [ou **Acknowledgements**, se em inglês] (opcional)

Essa seção não possui uma estrutura pré-definida.

O mais usual é separar em parágrafos. Mencionar apenas pessoas, instituições e outros que tenham tido importância para a realização do trabalho.

**Epígrafe (Opcional – NBR10520). Elemento opcional, no qual o autor apresenta uma citação, seguida de indicação de autoria, relacionada à matéria tratada no corpo do trabalho.**

ResumO

Apresenta as informações principais do documento (Descrição geral da empresa (natureza, mercado, processos, etc.), problema-foco atacado no PFC, o que foi feito, principais resultados atingidos, etc.). Deve conter entre 100 e 500 palavras (NBR6028/2003) em um único parágrafo. Se o documento for escrito em outra língua que não o Português, então é necessário fazer um Resumo **Estendido** em Português, ao invés deste resumo enxuto.

**Palavras-chave**: No mínimo 3 (três) e separadas por ponto (.)

**Abstract**

Resumo em língua inglesa.

**Key-words**: No mínimo 3 (três) e separadas por ponto (.)

**LISTA DE ILUSTRAÇÕES (OPCIONAL)**

|  |  |
| --- | --- |
| Figura 01 – xxxxxxxx ....................................................................................... | xx |
| Figura 02 – xxxxxxxxx ..................................................................................... | xx |
|  |  |
|  |  |
|  |  |

**LISTA DE TABELAS (OPCIONAL)**

|  |  |
| --- | --- |
| Tabela 01 – xxxxxxxx ....................................................................................... | xx |
| Tabela 02 – xxxxxxxxx ..................................................................................... | xx |
|  |  |
|  |  |
|  |  |

**LISTA DE ABREVIATURAS E SIGLAS (oPCIONAL)**

IBGE - Instituto Brasileiro de Geografia e Estatística

UFSC - Universidade Federal de Santa Catarina

BU - Biblioteca Universitária

VANT - Veículo Aéreo Não Tripulado

**LISTA DE SÍMBOLOS (oPCIONAL)**

dab - Distância euclidiana

**Sumário**

|  |  |  |
| --- | --- | --- |
| **1** | **INTRODUÇÃO ........................................................................................** | **XX** |
| **1.1** | **Justificativa ...........................................................................................** | **XX** |
| 1.1.1 | Objetivo Geral ......................................................................................... | XX |
| 1.1.2 | Objetivo Específico ................................................................................. | XX |
| **2** | **DESENVOLVIMENTO ............................................................................** | **XX** |
| **3** | **CONSIDERAÇÕES FINAIS E PERSPECTIVAS ...................................** | **XX** |
|  | **REFERÊNCIAS ......................................................................................** | **XX** |
|  | **APÊNDICE A – XXXXXX ........................................................................** | **XX** |
|  | **ANEXO A – XXXXXXX ...........................................................................** | **XX** |

O modelo acima é apenas uma ideia de representação, mas é possível utilizar a ferramenta “sumário” disponível no word, por exemplo. NBR 6027/2012.

# Introdução

## Definição do assunto

De acordo com o Censo Demográfico 2011 o setor elétrico brasileiro é responsável pela geração, transmissão e distribuição de energia elétrica para 97,8 % dos 57.324.167 lares brasileiros [1]. Sua estrutura reflete da continentalidade inerente ao território da nação e seu desenvolvimento e avanço são historicamente entrelaçados ao progresso socioeconômico brasileiro. O setor é composto por agentes que gerenciam suas três áreas chave: geração, transmissão e distribuição.

Os agentes de geração produzem a energia que é transportada para regiões próximas aos centros urbanos pelos agentes de distribuição e finalmente entregues ao consumidor final pelos agentes de distribuição [2]. Este intercâmbio de energia é possível devido a extensa malha de conexão, Sistema Integrado Nacional (SIN), com mais de 100 mil quilômetros de comprimento [2] [3].

No ano de 2017, apesar da queda de 1,2 %, o consumo de energia elétrica no Brasil alcançou 467 TWh, colocando o Brasil entre os 10 maiores consumidores do mundo [4]. De acordo com o estudo e projeções realizadas pelo Empresa de Pesquisa Energética (EPE) para 2027 o consumo de energia no Brasil está estimado para 744 TWh [5].

Com um modelo de mercado aberto a iniciativa privada, através de leilões de energia desde 2004, são regidos pela Câmera de Comercialização de Energia Elétrica (CCEE) os leilões de energia, onde os participantes, devem optar pratica do menor preço. Concessionárias de energia são comumente os compradores e os agentes responsáveis pela distribuição de energia proveniente de localidades próximas a centros urbanos, até a residência do consumidor final ou de clientes de maior demanda [6].

Considerando a importância da energia elétrica no desenvolvimento do país e um aumento previsto nos anos que seguem, seu funcionamento contínuo e integridade são portando, fatores de preocupação estatal que na pratica toma forma através de ações regulatórias de agencias tal como a Agencia Nacional de Energia Elétrica (ANEEL). Para assegurar a qualidade da distribuição de energia pode-se destacar o uso pela ANEEL de dois indicadores o DEC (Duração Equivalente de |Interrupção por Unidade Consumidores), e o FEC (Frequência Equivalente de Interrupção por Unidade Consumidora), que medem a duração e frequência de interrupções de energia respectivamente [7].

Como parte integrante do sistema de distribuição, as concessionárias são responsáveis pela redução da tensão das linhas de transmissão para níveis residenciais entre 127 e 220 V. Esta conversão ocorre em subestações que são instalações que agrupam o conjunto de equipamentos necessários a execução de sua função. Sua manutenção e continuo melhoramento são de responsabilidade das concessionarias e as faltas e prejuízos gerados ao consumidor decorrente do seu mal funcionamento, são monitorados e fiscalizados pela ANEEL através de análise de indicadores.

O Brasil conta com mais de 63 concessionárias de energia elétrica [7], e dentre elas a Companhia de Eletricidade Paranaense (COPEL) resolve por investir em um projeto de pesquisa de caráter experimental para o aumento da confiabilidade dos planos de manutenção preditiva dos equipamentos que compõe suas subestações. Substituindo o processo manual de coleta de fotos térmicas de equipamentos para posterior análise de danos, por um sistema de geração e execução de rotas de inspeção adaptativas guiadas por um Veículo Aéreo Não Tripulado (VANT), a COPEL espera obter uma melhoria nos seus indicadores DEC e FEC, redução de gastos com deslocamento de equipes de inspeção, melhoria no planejamento dos planos de manutenção e aumento da segurança dos colaboradores em operações de campo em subestação.

## Contextualização de desenvolvimento

O projeto está sendo executado peça Fundação Centros de Referencias em Tecnologias Inovadores (Fundação CERTI) por intermédio de proposta realizada a chamada aberta em forma de edital para Projeto e Desenvolvimento publicada pela concessionária de energia COPEL.

O projeto proposto consiste em duas frentes de desenvolvimento: a primeira sobre o desenvolvimento de um sistema fixo de detecção de falhas em equipamentos de subestação através do uso de câmeras térmicas e o segundo, no qual este projeto é baseado, o desenvolvimento de um módulo experimental para o uso de um VANT como o agente que adicionará mobilidade para inspeção das falhas.

## Justificativa

A implementação desta solução permite a o monitoramento de baixo custo de equipamentos de uma subestação e a possiblidade de modificação e avaliação de novas rotas de inspeção caso novos equipamentos ou requisitos sejam alterados/adicionados pelo cliente.

Em alternativa a inspeção manual, onde o operador está na subestação controlando o VANT através de um controle remoto, a solução proposta também visa aumentar a segurança diminuindo a exposição do trabalhador aos riscos de vida, como por exemplo, proximidade a linhas energizadas.

Devido a falhas em equipamentos a COPEL gastou cerca de R$ 85 milhões em multas devido a análise de indicadores como DEC e FEC. Com os planos de manutenção melhorados pela repetitividade dos resultados obtidos por análises de fotos tiradas durante a execução das rotas adaptativas os indicadores DEC e FEC tendem diminuir.

O contratante possui cerca de 183 subestações e o planejamento de manutenção demanda profissionais dedicados a tirar fotos dos equipamentos em cada uma delas. O custo para cada hora de trabalho é de R$ 83 e a duração em média de uma inspeção é cerca de 4 horas contando o deslocamento. Por motivos de segurança dois profissionais são necessários o que gera um custo de R$ 121.500,00. A redução esperada para inspeção é de 50 %, incorrendo em uma economia de R$ 60.750,00.

## Objetivo Geral

Desenvolver um sistema para controle de rotas autônomas e adaptativas para VANT

## Objetivos Específicos

* Desenho da arquitetura conceitual do sistema
* Mapeamento das alternativas de implementação dos módulos do sistema
* Implementação de um ambiente virtual com VANT simulado na estação de comando
* Implementação do módulo de path planning no ambiente de virtual na estação de comando
* Implementação do módulo de coleta de pontos e geração de trajetórias na estação de comando
* Implementação do módulo de execução de trajetória e controle adaptativo na estação de comando
* Analisar o funcionamento da execução de trajetórias e controle adaptativo do VANT em ambiente de simulação na estação de comando
* Implementação do ambiente de simulação no VANT
* Implementação da comunicação entre VANT e estação do comando
* Analisar o funcionamento do controle adaptativo no ambiente de simulação no VANT
* Analisar o funcionamento do controle adaptativo no VANT em aplicação real
* Analise sobre melhorias em VANT utilizado

## Metodologia

A metodologia de desenvolvimento a ser utilizada para este projeto segue modelo utilizado pela CERTI, denominado Sistema de Desenvolvimento de Soluções (SDS). Este modelo é aplicável de forma abrangente a projetos de desenvolvimento de sistemas e produtos.

O SDS divide um projeto em quatro possíveis fases, apresentadas a seguir:

- Fase 0. Mapeamento e elaboração do conceito básico da solução: esta etapa tem por objetivo levantar requisitos e mapear soluções e cenários de uso, para desenvolvimento de um conceito de produto e definição da arquitetura a ser desenvolvida.

- Fase 1. Desenvolvimento da solução conceitual: esta etapa tem por objetivo caracterizar a arquitetura detalhada da solução escolhida, agregando informações detalhadas sobre o conceito e modelo, potenciais provedores, mercado e demais fatores envolvidos. O objetivo é desdobrar o conceito de solução até o nível que permita ser desenvolvido um produto ou solução.

- Fase 2. Desenvolvimento da solução detalhada: esta etapa tem por objetivo a elaboração do projeto de engenharia detalhado da solução, incluindo hardware, software, aspectos de usabilidade, bem como materiais e processo básico para sua fabricação. Nesta etapa são realizadas atividades de design e construção de produto.

- Fase 3. Fabricação em escala e testes: esta etapa tem por objetivo a fabricação em escala do produto.

## Alternativas de desenvolvimento

O uso de VANTs para inspeção no setor elétrico ainda está na sua infância, porém seu horizonte de aplicação é promissor para área de geração, transmissão e distribuição de energia, alcançando, segundo estudos da PwC, um mercado de até $ 9.46 bilhões [8]. Refletindo esta tendência, projetos com o uso de VANT principalmente para verificação de linhas de transmissão, usinas eólicas e painéis solares lideram as primeiras tentativas de aproveitamento do uso de VANTS.

Segundo Rangel, Kienitz e Brandao [9] os métodos tradicionais de inspeção de linhas de transmissão consistem no sobrevoo da linha com o uso de um helicóptero ou da patrulha com veículos terrestres no qual uma equipe acompanha a linha de transmissão do solo, segundo o estudo ambos os métodos apresentam problemas. O uso de helicóptero aproxima o operador da linha viva e sua performace depende principalmente das condições climáticas e de terreno onde as linhas se encontram, expondo os operadores a um grande risco de vida. A patrulha terrestre é limitada, pois grande parte das linhas de transmissão ficam em regiões de difícil acesso.

O trabalho consiste na utilização de um VANT equipado com câmeras de vídeo e dispositivos de telemetria e controle para o monitoramento de linhas de transmissão de alta tensão. A operação do VANT é realizada em uma base de comando equipada com laptop, dispositivos receptores de frames de vídeo, telemetria, joystick, dispositivos de visualização e softwares específicos. O software de navegação desenvolvido, atualiza a interface em tempo real com informações de voo, latitude, longitude e imagens de câmera em tempo real. Baseado nas informações coletadas o software permite a criação de rotas em três dimensões posteriormente usadas para inspeção das linhas. A interface do sistema é realizada diretamente em na tela de um laptop ou utilizado um capacete equipado com monitores LCD.

Ainda sobre inspeção de linhas de transmissão Deng, Wang, Huang, Tan e Liu [10] exploram as vantagens no aumento de eficiência no uso de múltiplos VANTs e um modelo de comunicação multiplataforma para inspeção de linhas de transmissão. Os vários VANTS servem de fonte de imagens de curta e longa distância da linha que são transmitidas em real time para uma estação de controle para o sistema de navegação e para um escritório onde são realizadas a análise das imagens.

Para a operação de inspeção é proposto a união das capacidades de diferentes tipos de VANTs como os de asa fixa que entregam tempos longos de voo porém com imagens não tão detalhas quanto a VANTs multirotores, que obtém imagens mais detalhadas devido a flexibilidade de controle que permite pairar próximo a pontos de interesse, mas com duração de bateria limitada à curtos intervalos de tempo.

É proposto também um novo modelo de comunicação entre os diferentes tipos de VANT e a estação de comando, aumentando a capacidade de comunicação a longa distância, permitindo a inspeção de linhas de transmissão mais longas. Segundo levantamento, o limite para controle e comunicação com VANT multirotores é limitado a linha de visão, que na pratica está entre 100 m e 1 km enquanto que os de asa fixa costumam não permitir a comunicação em realtime, neste último a configuração da missão costuma ser possível apenas antes do voo. A proposta utiliza um VANT que permanece conectado a alimentação e instalado em um ponto intermediário da rota de inspeção fornecendo um link de comunicação para todos os outros VANTs durante a inspeção. Esta técnica é utilizada quando os VANTs fogem a linha de visão, o que pode ser frequente em terrenos irregulares.

O trabalho ainda propõe uma solução para transmissão das imagens de inspeção para os escritórios especialistas. Em contrapartida a prática anterior de download das imagens em cartões de memória, transporte e finalmente analise em escritórios, localizados a centenas ou milhares de quilômetros do local de inspeção, as imagens de alta qualidade são transmitidas via satélite, sendo possível o envio simultâneo de 4 canais de vídeos de alta definição.

Frente a crescente implementação de plantas fotovoltaicas ao redor do mundo Oliveira, Aghaei, Madukanya, Nascimento e Rüther avaliam a aplicabilidade comercial do uso simultâneo de técnicas de inspeção de painéis fotovoltaicos por meio de termografia infravermelha (IRT) e o uso de VANT (aIRT) para cobrir grandes áreas referentes aos parques fotovoltaicos [11].

Seu estudo expande o atual método de coleta de imagens realizado em câmeras infravermelhas com auxílio de sistemas de elevação. Este método consome grandes intervalos de tempo e é intensivo em relação ao trabalho exigido para a coleta de imagens, portanto costuma ser praticado em setores do parque selecionados de forma aleatória. No caso citado, o estudo foi motivado por um evento chamado tsunami meteorológica que comprometeu o funcionamento de células fotovoltaicas de um parque inteiro, a coleta de dados manual tornaria o processo de verificação e posterior manutenção extremamente laborioso.

Na solução foram utilizados uma câmera termográfica acoplada a um VANT DJI Phantom 3 Professional. O VANT é controlado remotamente e transmite as imagens para estação de comando via uma antena Cloverleaf de 5.8 GHz para posterior análise por meio de técnicas de termografia.

## Próximos capítulos

Este trabalho está divido em 7 capítulos, dos quais o conteúdo é descrito a seguir. Este capítulo de introdução define o assunto, objetivos, justificativas e traz um estudo sobre o desenvolvimento de trabalhos relacionados

O capítulo 2 descreve a empresa e o departamento onde este trabalho foi realizado e apresenta um pequeno histórico de colaboração do autor para com a instituição.

No capítulo 3 são descritos os conceitos e embasamento teórico sobre determinados tópicos necessários para a compreensão das tecnologias utilizadas ao longo do desenvolvimento deste trabalho.

No capítulo 4 são apresentados os requisitos da solução, modelos de concepção conceitual da solução e justificativas sobre a decisão sobre o modelo conceitual escolhido.

No capítulo 5 são exibidos os diagramas de desenvolvimento de software, os códigos fonte dos módulos desenvolvidos e as interfaces de comando do operador.

No capítulo 6 é composto por uma análise crítica dos resultados da execução do caso de uso principal e a especificação de um novo VANT baseada em aspectos de melhoria para uso em projetos futuros de aplicação similar.

No capítulo 7 se conclui sobre os resultados alcançados, se analisa os limites da aplicação e se propõe a continuidade do desenvolvimento em forma de desenvolvimento de novos trabalhos na área.

# A empresa

A seguir são descritos detalhes sobre o escopo de atuação da fundação na qual este trabalho foi desenvolvido. No decorrer da descrição ênfase é dada ao laboratório onde especificamente o trabalho foi realizado. Em seguida um breve histórico da atuação do autor para com a Fundação CERTI é apresentado.

## A fundação CERTI

Como instituição de Ciência, Tecnologia e Inovação, a Fundação CERTI (Fundação Centros de Referência em Tecnologias Inovadoras) nasceu direcionada para a pesquisa tecnológica aplicada, num contexto em que o Brasil demandava saltos de qualidade e desenvolvimento de know how próprio e inovador especialmente no campo da informática e das tecnologias de ponta, incluindo particularmente a automação industrial [12].

A CERTI, em seu histórico de 35 anos de existência, atua em prol do desenvolvimento econômico, sustentabilidade, inclusão social e educação, promovendo a inserção estratégica de tecnologias voltadas para a melhoria da qualidade de vida da sociedade brasileira. Para tanto, atua como entidade realizadora de ações de Inovação, desenvolvendo produtos, processos industriais, software, soluções de serviços e novos modelos de negócios.

A abrangência dos trabalhos compreende desde planos de negócio de novos produtos e tecnologias ao processo de lançamento no mercado, passando por prototipagem, teste, certificação, fabricação de lotes piloto, desenvolvimento de fornecedores e demais etapas da criação de soluções completas e qualificadas no mercado, atuando em diversos setores, entre eles: Energia Sustentável, Economia Verde, Tecnologia da Informação e Comunicação, Convergência Digital e Desenvolvimento de produtos, Metrologia e Automação.

Atualmente a CERTI é composta por diversos centros que operam diferente áreas de competência, tais centros são conhecidos como Centros de Referência. São eles:

* CDM - Centro de Convergência Digital e Mecatrônica
* CMI - Centro de Metrologia e Instrumentação
* CPC - Centro de Produção Cooperada
* CEI - Centro de Empreendedorismo Inovador
* CELTA - Centro Empresarial para Laboração de Tecnologias Avançadas
* CEV - Centro de Economia Verde
* CES - Centro de Energia Sustentável

O Centro de Metrologia e Instrumentação (CMI), onde este trabalho foi desenvolvido, é composto por dois conjuntos laboratoriais referentes aas áreas de instrumentação e metrologia. Este trabalho foi desenvolvido no laboratório de instrumentação do CMI, que oferece serviços como:

* Desenvolvimento de sistemas de medição e bancadas de teste: desenvolvimento de produtos voltados para o controle de qualidade e de suporte a engenharia de produtos
* Serviços de planejamento e execução de testes especiais: com foco em placas eletrônicas é realizado o planejamento, execução e análise de testes funcionais para produtos mecatrônicos
* Desenvolvimento de soluções para monitoramento de ambientes: integração de sensores de variáveis ambientais como: velocidade do vento, temperatura, umidade do ar, pressão atmosférica, radiação solar e nível para instalação de sistemas de monitoramento em regiões remotas.

Como clientes, somente o laboratório possui clientes como Celesc, Copel, Engie, Embraer e Petrobras. Na linha do problema que se desenvolve neste trabalho, o laboratório atua em outro projeto da Petrobras que propões a utilização de VANT para detecção de bora em tanques de armazenamento de petróleo bruto.

## O colaborador

O autor atua na Fundação CERTI desde o ano de 2011, quando ingressou à Fundação CERTI como estagiário para atuar como na área de calibração de instrumentos de Forca, Pressão e Massa no extinto LFPM (Laboratório de Força Pressão e Massa). Em 2012, é contratado como técnico metrologista atuando em procedimentos de calibração a ajudando nas atividades práticas de cursos ofertados pela Fundação CERTI.

Em 2013 devido ao ingresso no curso de Engenharia de Controle e Automação, o autor passa a integrar o laboratório de instrumentação do CMI, onde até presente data participa do planejamento e desenvolvimento dos sistemas de monitoramento de placas eletrônicas, desenvolvimento de bancadas de teste e sistemas de medição, assim como de outros serviços citados na secção anterior.

Em 2017 o autor é contratado como pesquisador técnico passando a integrar funções mais intimas a área de instrumentação e testes.

# Fundamentação

Neste capítulo são descritos os conceitos de necessários para a compressão do conteúdo envolvido nas atividades de desenvolvimento do sistema de geração de rotas adaptativas.

## ROS e Gazebo

O ROS (Robot Operating System) é um sistema operacional para aplicações no mundo robótica, que reúne ferramentas, bibliotecas e convenções que tem por objetivo dar suporte ao desenvolvimento de aplicações de alta complexidade e robustez. O propósito do seu desenvolvimento é baseado no desenvolvimento colaborativo entre laboratórios ou desenvolvedores entusiastas, no qual a *expertise* dos colaboradores é disseminada em uma aplicação de nível mundial e de alta confiança [13].

Gazebo é o software de simulação que acompanha o ROS, servindo como suporte a criação de um ambiente de teste, com simulação de física, sensores e robôs.

A seguir são descritos os conceitos básicos de funcionalidades do ROS e do Gazebo utilizados ao longo do desenvolvimento do trabalho.

### Nodos

Nodo são programas que realizam operações modulares. Em aplicações ROS, onde a modularidade é característica, é normal encontrar muitos nodos trabalhando em conjunto compreendendo a solução final. Cada nodo opera funções que dependem de informações e dados provindos de diferentes programas, para isso comunicam entre si através de serviços, tópicos ou *actionlibs*. Por exemplo em uma aplicação de navegação um robô móvel é possível que um nodo seja responsável pelo controle das rodas, controle de posição, leitura do sensor laser, detecção de obstáculos, do sistema de localização, e com a interface de usuário [15].

Até o presente momento nodos podem ser escritos em Java, C++ e Python. Com esta funcionalidade fabricantes de sensores, ou desenvolvedores de algoritmos podem integrar a funcionalidade de seus desenvolvimentos utilizando as API´s (*Application Programming Interface*) fornecidas nas linguagens acima mencionadas. Neste trabalho utilizou-se uma solução composta por programas em Python e Java. Para ser distribuído um nodo precisa pertencer a um pacote.

### Pacotes

Um pacote é um conjunto de nodos, bibliotecas, *softwares* proprietários, arquivos de configurações, tipos de dados, tabelas e tudo que é essencial para o funcionamento de uma determinada função. A ideia é de que o que estiver reunido e denominado sob um determinado pacote, realize funcionalidades similares, que exista uma hierarquia de suporte a uma funcionalidade superior entre seus integrantes [16].

A estrutura de um pacote é padronizada, e é necessário que durante o desenvolvimento da aplicação se siga com atenção a estrutura determinada pelo *framework* ROS, para que o código seja compilado utilizando a ferramenta catkin\_make.

Exemplos de pacotes são:

* *Moveit*: pacote utilizado para *path planning* de braços robóticos e robôs móveis.
* *Depthimage\_to\_scan*: pacote utilizado para conversão de nuvem de pontos em tipo de dados *scan*.
* *Navigation*: pacote utilizado para navegação especializada em robótica móvel.
* Geonav\_transform: pacote utilizado na conversão de coordenadas geográficas para coordenadas cartesianas.

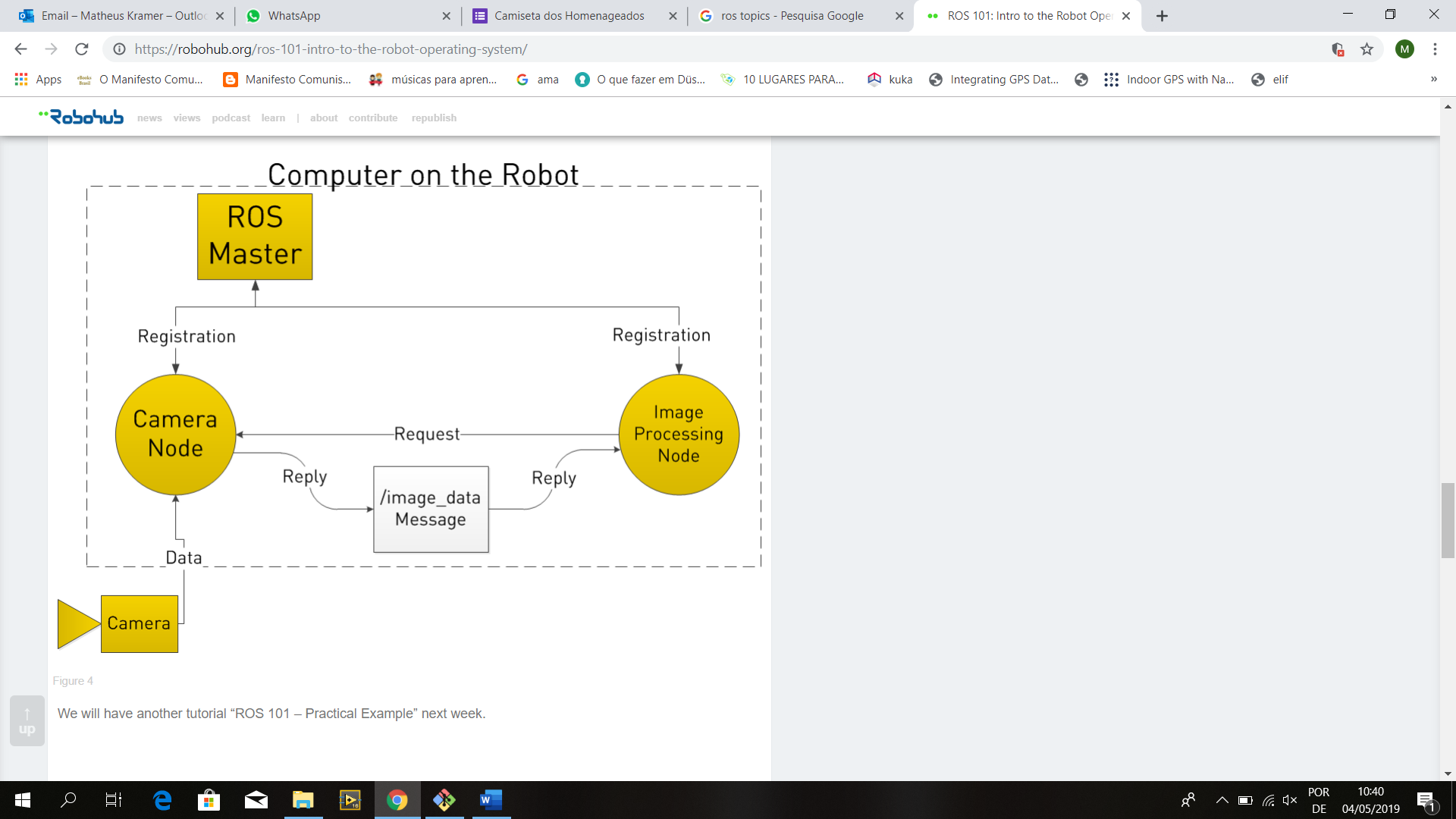
A comunidade ROS encoraja o desenvolvimento e modificação dos códigos fornecidos em pacotes. Esta prática intensifica o uso e publicação e revisão de bugs e correções. Alguns destes pacotes são extensivamente testados e com robustez comprovada para aplicações em robótica.

### ROS Master

O ROS master é o programa *core* que realiza a comunicação entre as diferentes funcionalidades que permeiam a solução. ROS Master pode ser pensando como um comunicador. Todos os nodos que rodam na camada ROS registram-se no master para poder operar, no ato de registro os nodos armazenam em uma tabela, localizada no master, suas intenções de leitura e escrita [14].

O master atua como um comunicar/distribuidor que no instante de recebimento de um dado o distribui para quem o está requisitando. Em um exemplo, podemos citar a conversão de nuvem de pontos para o tipo *scan*, nesta conversão pode-se utilizar um nodo fornecedor da nuvem de pontos, um driver que realiza leitura dados de uma câmera especifica, enquanto outro nodo realiza a conversão da nuvem de pontos para o tipo de dados *scan*, este pode então ser utilizado para aplicações de mapeamento e detecção de obstáculos. Abaixo a exibe Figura 1 os registro de dois nodos com o ROS master

Figura - Registro com ROS Master



Fonte: https://robohub.org/ros-101-intro-to-the-robot-operating-system/

### Tópicos

Tópicos são a forma mais simples de troca de mensagem disponível em ROS, é conhecida como comunicação 1-para-muitos. Funcionam como *buses* virtuais que desacoplam o produtor do consumidor. Os nodos ao registrar suas intenções de escrita (*publishers*) e leitura (*subscribers*), denominam um nome dado ao espaço de memória onde deverão ser publicadas determinadas informações. Podemos entender este espaço de memória com um tópico [17].

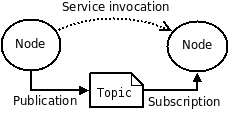
O ROS *master* gerencia as publicações enviando os dados aos nodos que a solicitam quando um outro nodo fornece o dado. Os tópicos são um mecanismo chave de comunicação em ROS, pois promovem a modularidade entre os nodos. Utilizando um tópico o nodo que consume um dado não tem como saber “quem” está enviando dados para a realização da função, assim como o nodo que publica o dado não conhece onde está sendo realizada a aplicação de seus dados. O formato de mensagens utilizado nos tópicos é definido em arquivos de tipos nos pacotes

### Serviços

Serviços é a forma de comunicação 1-para-1 mais simples disponível no ROS. Em alternativa à dinâmica dos tópicos, os serviços entregam a funcionalidade de pergunta e resposta. Esta propriedade é utilizada quando necessária a confirmação de execução de algum procedimento. O formato de mensagens utilizado nos serviços são definidos em arquivos de tipos nos pacotes [18].

Na sua inicialização, um nodo registra no *master* quais são os serviços que fornece, caso forneça algum. Os serviços dependem da aplicação a qual o nodo pertence. Em pacotes de SLAM *(Simultaneos Localization and Mapping)* é comum haver um nodo fornecendo um serviço de reconfiguração do mapa, este serviço quando chamado reseta os parâmetros da aplicação promovendo uma nova localização do robô em um mapa de obstáculos. Na Figura 2 é exibido dois nodos trocando mensagens por serviços e tópicos.

Figura - Serviços e Tópicos



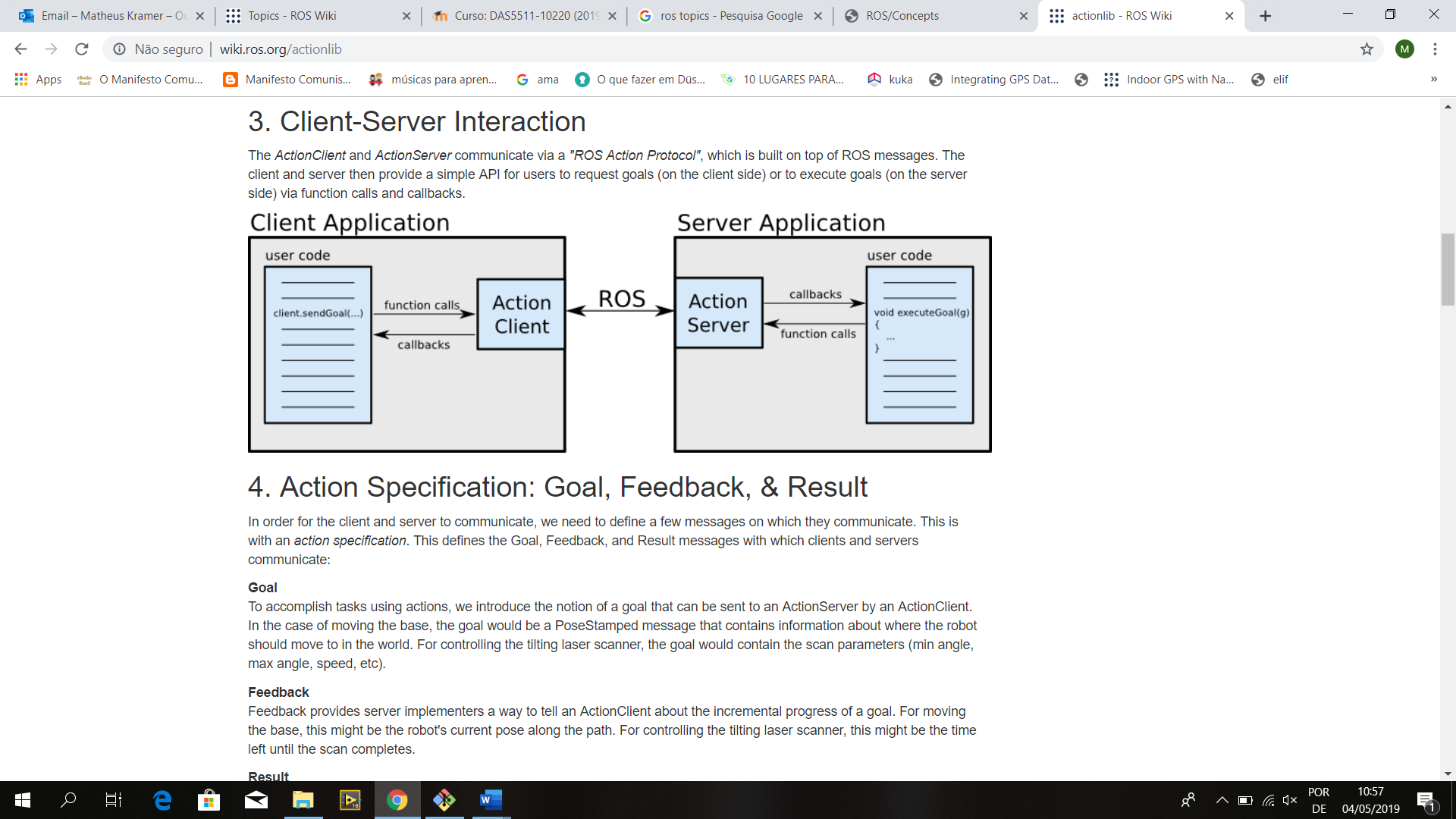
Fonte: http://library.isr.ist.utl.pt/docs/roswiki/ROS(2f)Concepts.html

### *Actionlibs*

*Actionlib* é uma forma de serviço mais complexa utilizada em ROS. Possui as mesmas caraterísticas que motivam a necessidade do uso de serviços, porem entregam um acompanhamento de estado necessário para aplicações críticas. Seu funcionamento é no formato cliente-servidor.

O *actionlib server* pode ser compreendido como uma interface. Neste interface um *actionlib client* pode realizar um requisição (*goal*) como uma posição para um robô móvel. Durante a execução da operação com dados de entrada fornecidos pelo cliente, o servidor mantém o cliente atualizado utilizando mensagens (*feedback*), no caso do robô móvel esta mensagem poderia ser a atual posição do robô ou se há um obstáculo ou não. Ao término da execução, um resultado é entregue através de mensagem (*result*), este resultado inclui informações referentes ao sucesso da requisição do cliente, no caso acima poderia ser se o robô alcançou a posição ou foi impossibilitado devido a alguma condição de obstáculo instransponível [19]. A Figura 3 a seguir, apresenta a interface entre cliente e servidor usando *actionlib*.

Figura - Interface de Actionlib



Fonte: http://wiki.ros.org/actionlib

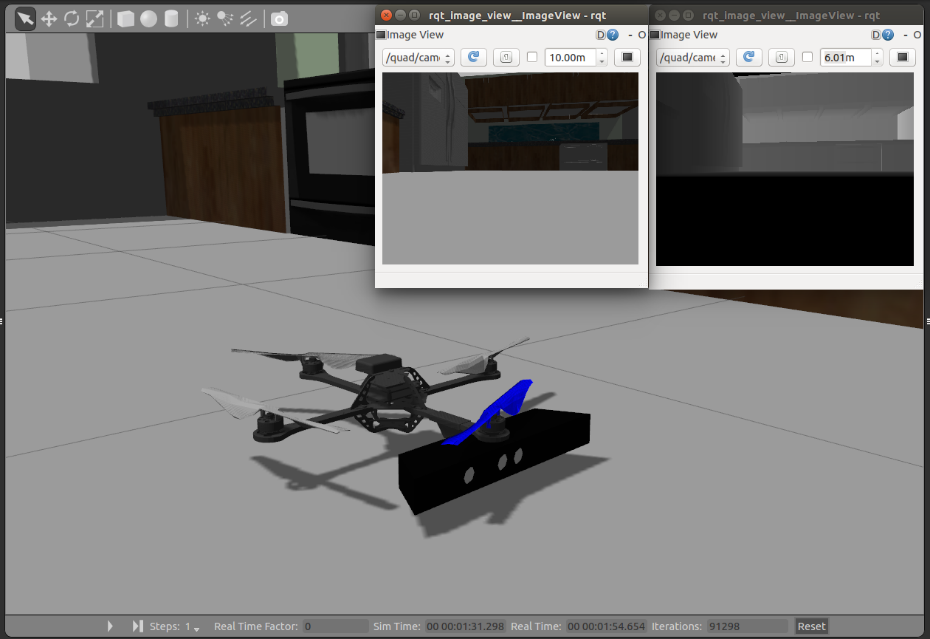
Como acima citado, *actionlibs* são comumente fornecidos em aplicações que envolvem movimento de partes físicas de robôs, pois promovem a possiblidade de parada caso alguma condição perigosa seja detectada.

### Gazebo

O Gazebo é uma ferramenta de simulação especializada para robótica. Através do seu uso é possível realizar a avaliação de algoritmos de *path planning*, navegação, execução de rotas, treino de algoritmos de inteligência artificial e testes de múltiplos robôs em ambientes *indoor* e *outdoor* [20]*.*

Através de *plugins* é possível gerar dados simulados como coordenadas de GPS (*Global Positioning System*), iGPS (*indoor* GPS), *feedbacks* de *encoders*, variação térmica de objetos, entradas digitais dentre outros. Alguns *plugins* estão disponíveis em pacotes ROS. A Figura 4 apresenta um VANT no ambiente Gazebo, uma câmera Kinect está acoplada ao VANT.

Figura - Simulação no Gazebo



Fonte: https://www.wilselby.com/research/ros-integration/model-dynamics-sensors/

Seu funcionamento é realizado através de dois programas principiais, um responsável por realizar os cálculos referentes a simulação física e outro por exibir a simulação ao usuário. O funcionamento da simulação é separado da visualização, podendo o usuário optar por não mostrar a simulação caso necessário economizar processamento.

Arquivos de objetos 3D podem ser importados no Gazebo desde que possuam o formato COLLADA. Esta funcionalidade é bastante proveitosa, do ponto de vista que os robôs podem ser testados em seus ambientes de produção virtuais antes mesmo de serem comprados ou construídos.

### Outro módulos ou funcionalidades

A seguir são descritos de maneira sucinta algumas funcionalidades de algumas nomenclaturas utilizadas neste trabalho:

* *Launch file:* arquivo XML que serve para iniciar nodos e configurar parametros no ROS.
* Inserção de pacotes: Neste trabalho existem 2 tipos de pacotes, aqueles instalados a partir do comando “*apt-get”* do Ubuntu e outros pacotes compilados do código fonte.

## *Sampled-based Motion planning e ROS*

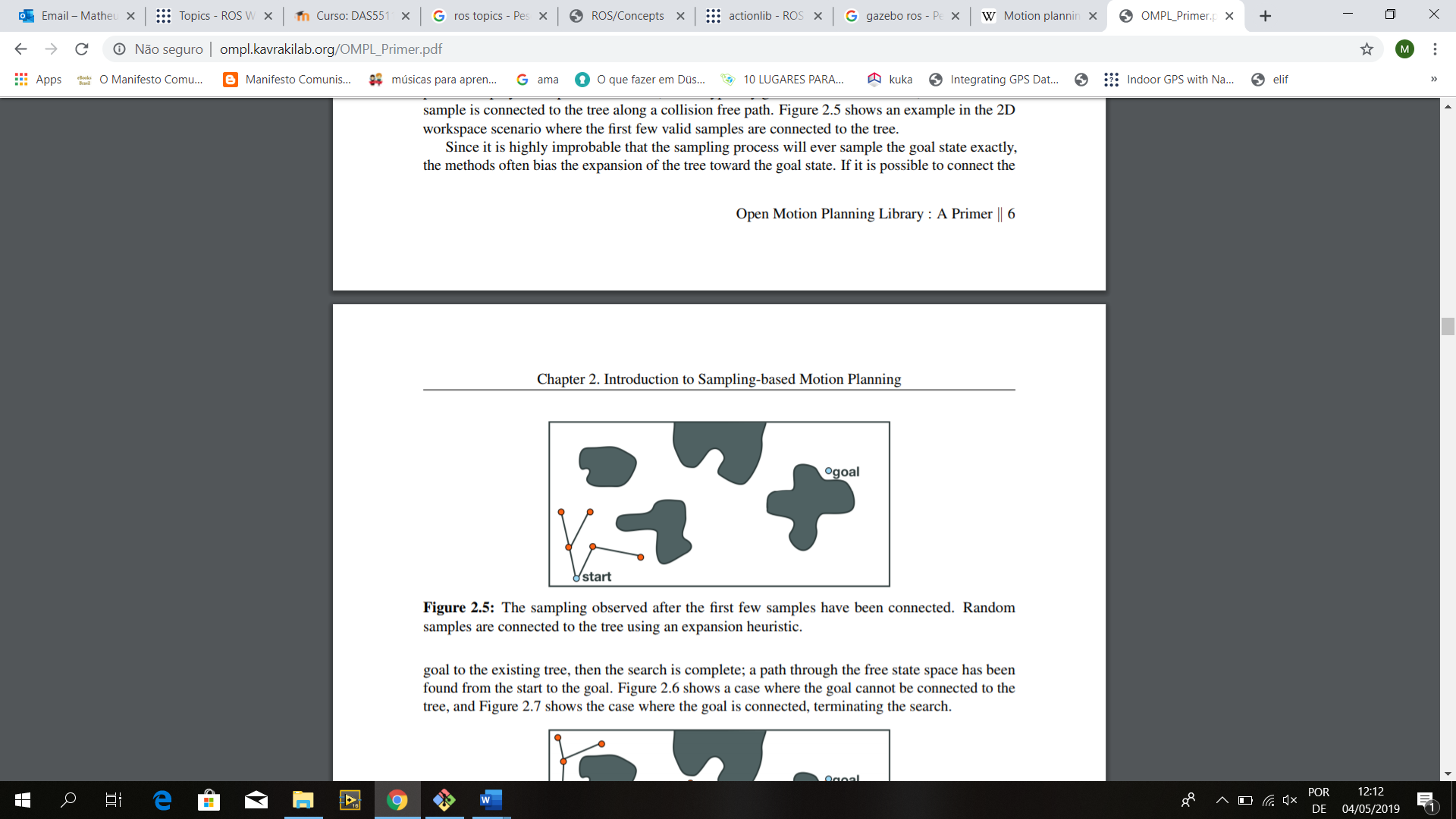
Para exemplificar os conceitos sobre *sampled-based motion planning* consideremos um problema clássico de no qual um objeto 3D precisa realizar um deslocamento de um ponto A para um ponto B ao mesmo tempo que existem obstáculos em sua vizinha que devem ser evitados. O possui 6 graus de liberdade, 3 para movimento nos planos coordenados (x, y, z) e 3 para representar a rotação em torno dos eixos (roll, pitch, yaw). Para o entendimento desta técnica de *motion planning* os seguintes conceitos são fundamentais:

* *Workspace*: Espaço físico onde o objeto pode operar
* *State space*: Representa todas as possíveis configurações para o objeto dentro do *workspace*, um único ponto no espaço de estados representa um estado
* *Free State Space*: parcela do *state space* que representa estados livres de obstáculos
* *Path*: Conjunto de estados que representam uma solução para o problema de deslocamento de um ponto inicial para um ponto final obedecendo a restrições

Ao contrário dos algoritmos tradicionais as técnicas de *sampled-based motion planning* não realizam uma análise contínua de todo o *state space.* A análise contínua, quando envolve elevado número de graus de liberdade, que numa aplicação comum de robótico podem ser 6, tende a consumir muito tempo de processamento devido ao número de possíveis soluções e complexidade referentes ao tamanho do *workspace* [21, p. 4].

O *Sampled-based Motion planning* pode ser implementado de formas diferentes, aqui será abordado os algoritmos baseados em expansão de árvore de estados. Estes algoritmos operam a partir de um estado inicial dado (posição inicial do objeto), realizam a expansão de uma árvore de estados, e para cada novo estado uma verificação de restrições é realizada, caso o caminho entre o estado antigo e novo obedeça às restrições, o estado é retido, caso contrário é descartado. Na Figura 5 abaixo, pode-se observar o estado inicial, final e 2 ramos de expansão a partir do estado inicial.

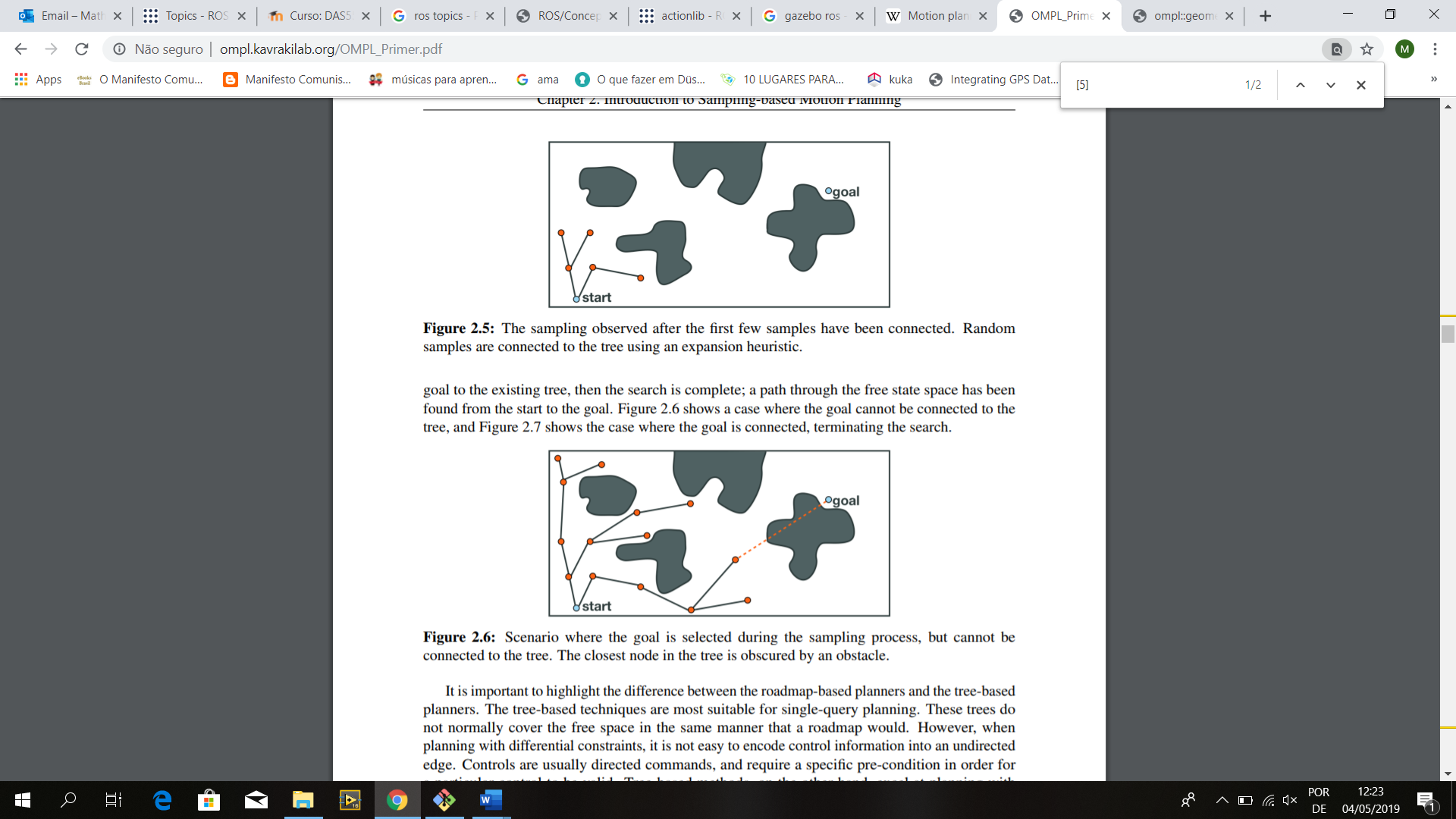
Figura 5 – Motion planning: State space inicial



Fonte: *Open Motion Planning Library: A primer*

Na Figura 6 pode-se observar uma condição em que o caminho definido entre o estado gerado e o anterior não obedece às restrições de colisão, neste caso o algoritmo descarta este estado e realiza uma nova tentativa de geração de estados.

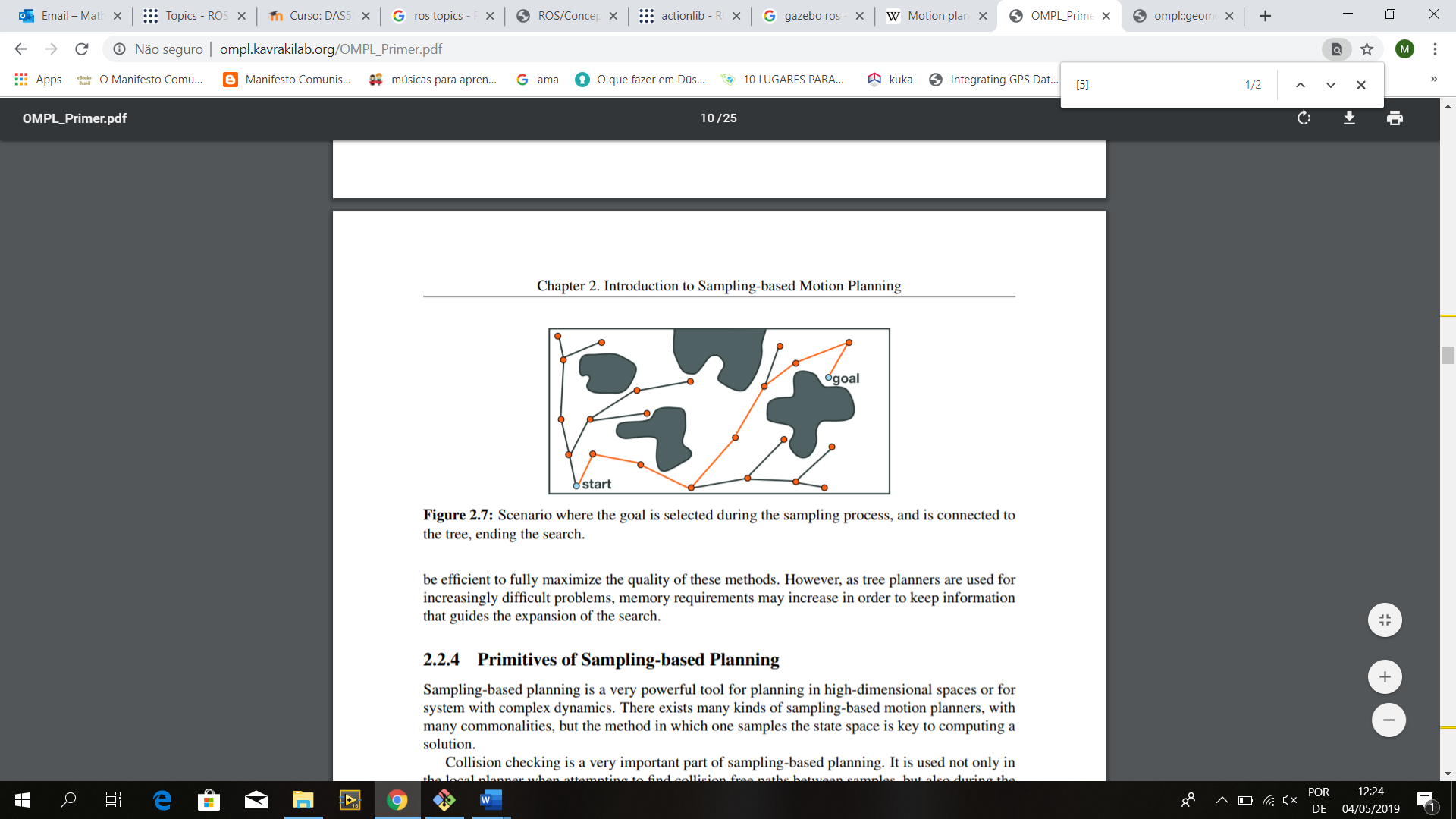
Figura - Motion planning: estado descartado



Fonte: *Open Motion Planning Library: A primer*

Na Figura 7 pode-se observar que um caminho foi encontrado até o estado desejado. Observa-se também que ramificações sem resultado são geradas, porém esta técnica de *motion planning* requer menos memória que as tradicionais por não precisar de uma representação explicita do espaço de estados no decorrer de suas interações [21, p. 7].

Figura 7 - Motion planning: goal state alcançado



Fonte: *Open Motion Planning Library: A primer*

A forma como a árvore é expandida e quantas ramificações possui depende do algoritmo utilizado, normalmente o algoritmo possui o nome da heurística utilizada na determinação de novos estados [21, p. 6].

O pacote *MoveIt* disponível para ROS possui a biblioteca OMPL (*Open Motion Planning Library*) que possui a implementação de diferentes algoritmos que implementam a técnica de *sampled-based motion planning* [22].

## *Octomap framework*

### Motivação do Octomap

## 

É comum as aplicações na área de robótica utilizarem representações volumétricas do espaço onde executam suas funções. Um robô móvel autônomo quando executando função de navegação, precisa conhecer o ambiente onde tarefa é executada, assim como um braço manipulador quando executando movimentos em sua célula de trabalho. Muitas das aplicações em robótica requerem três aspectos em relação as características de uma representação 3D:

* Representação probabilística: as medições utilizadas para criação de um modelo 3D são oriundas de sensores, e estes possuem uma incerteza associada aos seus resultados. Um modelo 3D robusto deve levar em consideração as incertezas de múltiplas medições ou até mesmo de mais de uma fonte de dados para estimar a real localização dos objetos na cena.
* Áreas não mapeadas: a representação de áreas não mapeadas é importante para que o robô não utilize áreas desconhecidas para estabelecer rotas, caso contrário uma rota em ambiente desconhecido poderia levar à colisão.
* Eficiência: a velocidade de acesso e memória ocupada pelos mapas deve ser reduzido ao máximo para que o tempo de acesso seja reduzido e que a aplicação não fique restrita por requerer no robô, uma elevada capacidade de armazenamento.

Alguns dos métodos de mapeamento 3D costumam apresentam problemas em algum desses aspectos, modelos criados diretamente a partir de nuvem de pontos ocupam bastante espaço de memória e em sua maioria não permitem representar áreas não descobertas.

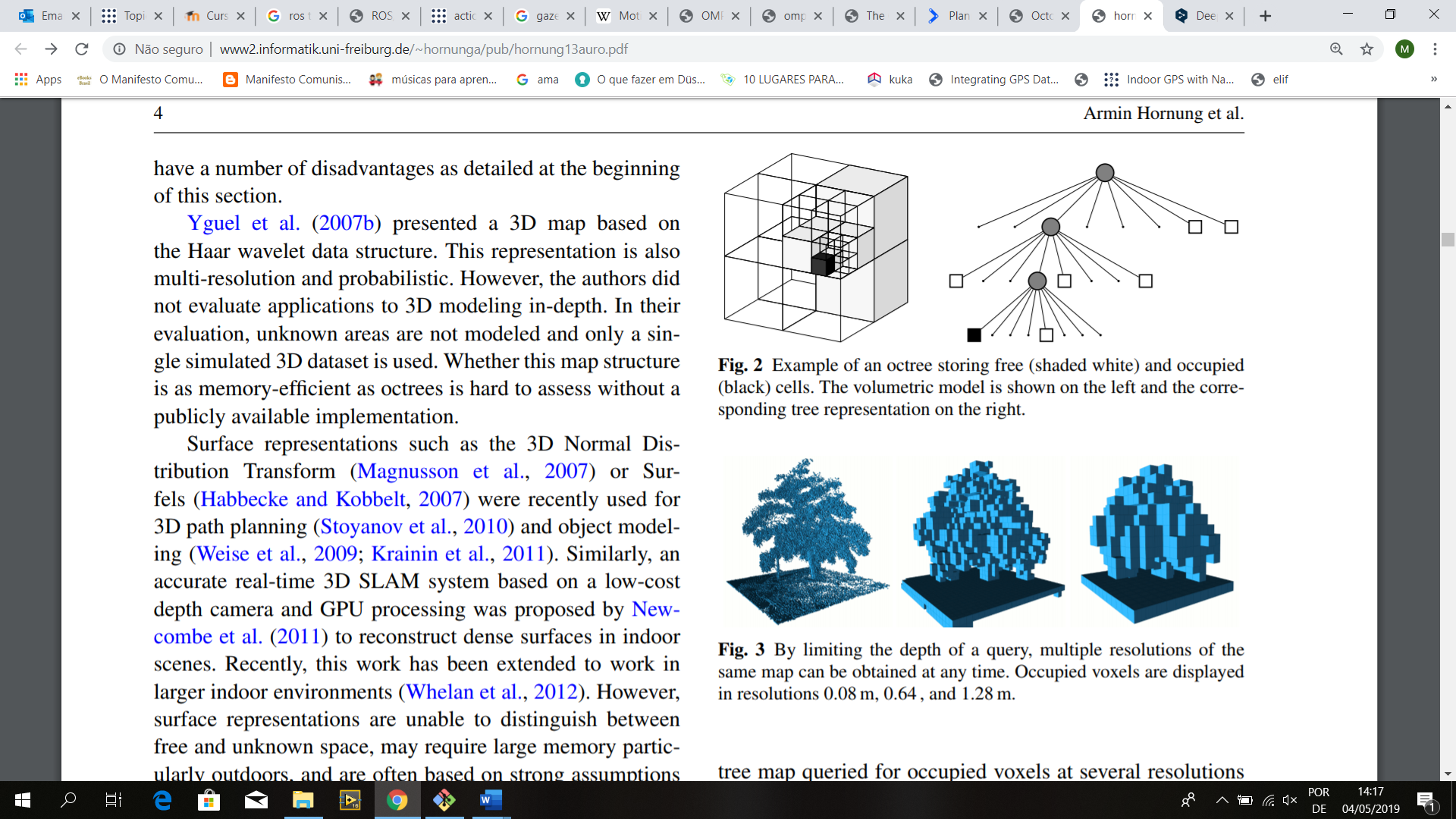
Octomap é um *framework* que apresenta uma proposta para manter as vantagens dos métodos tradicionais incluindo os três aspectos discutidos anteriormente, este é baseado na representação volumétrica do espaço. O *framework* está disponível através de uma biblioteca em C++ e também disponível diretamente para ROS através de um pacote [23].

### Representação do Octomap

A representação de um Octomap é baseada em Octotree. Octotree é uma estrutura hierárquica do tipo árvore que representa subdivisões de um espaço 3D. Cada nó da árvore representa o espaço contido em um volume cúbico, os nodos são conhecidos como voxel.

A expansão da árvore acontece a partir da expansão do primeiro voxel em 8 subdivisões de igual volume. Os nodos gerados são novamente subdivididos em 8 partes e este processo acontece recursivamente até que um voxel mínimo seja alcançado. O tamanho deste voxel mínimo representa a resolução do Octomap. Na Figura 8 pode-se ver um exemplo de Octotree com a indicação sobre estado ocupado (volume em preto) [23].

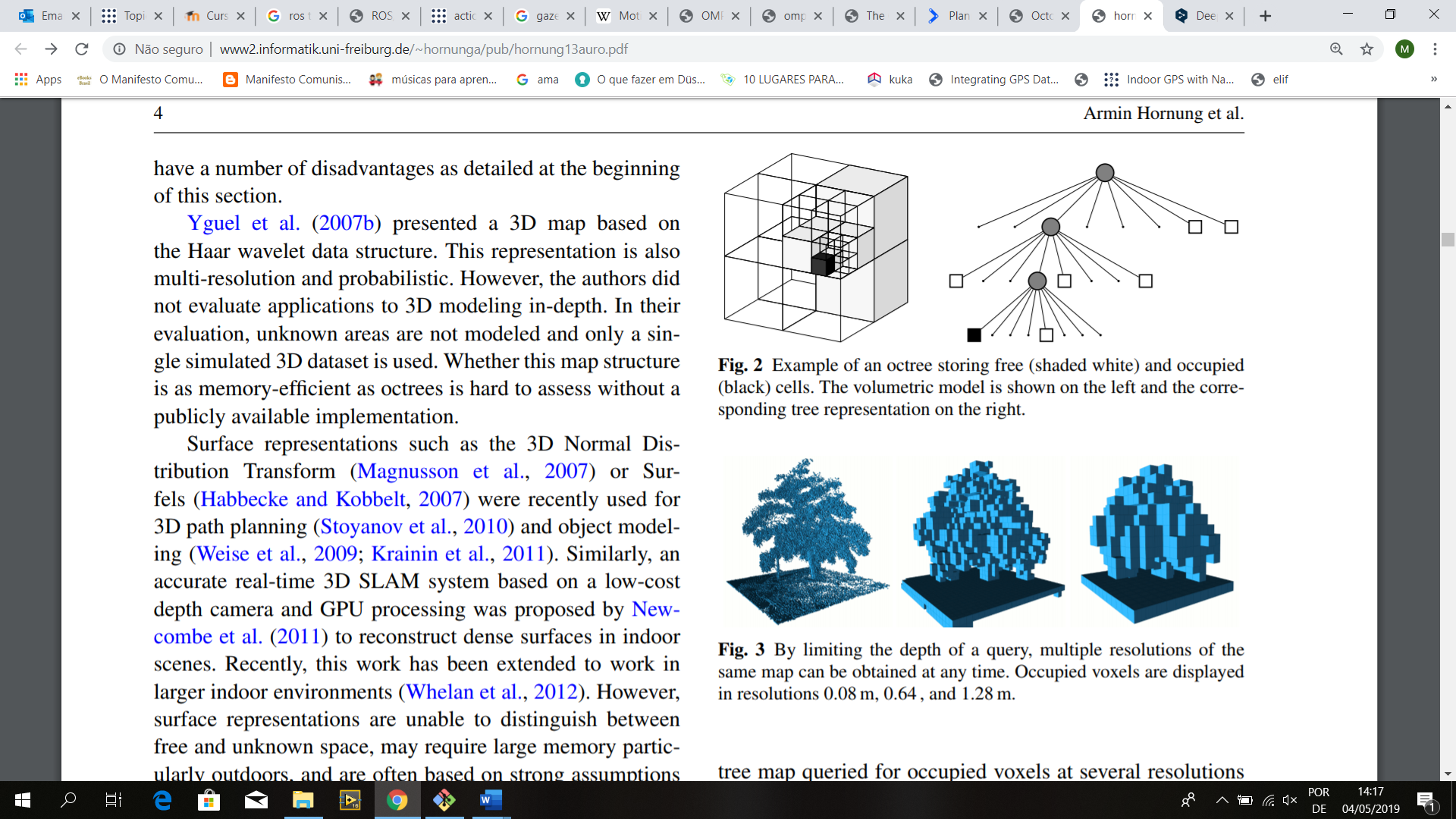
Figura - Expansão de uma Octotree



Fonte: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees

Limitando a profundidade da expansão da árvore é possível obter representações de um objeto em diferentes resoluções. Um exemplo de Octotree é apresentado na Figura 9, abaixo é possível visualizar o mesmo objeto sendo representado em diferentes resoluções (0.08 m, 0.64 e 1.28 m respectivamente).

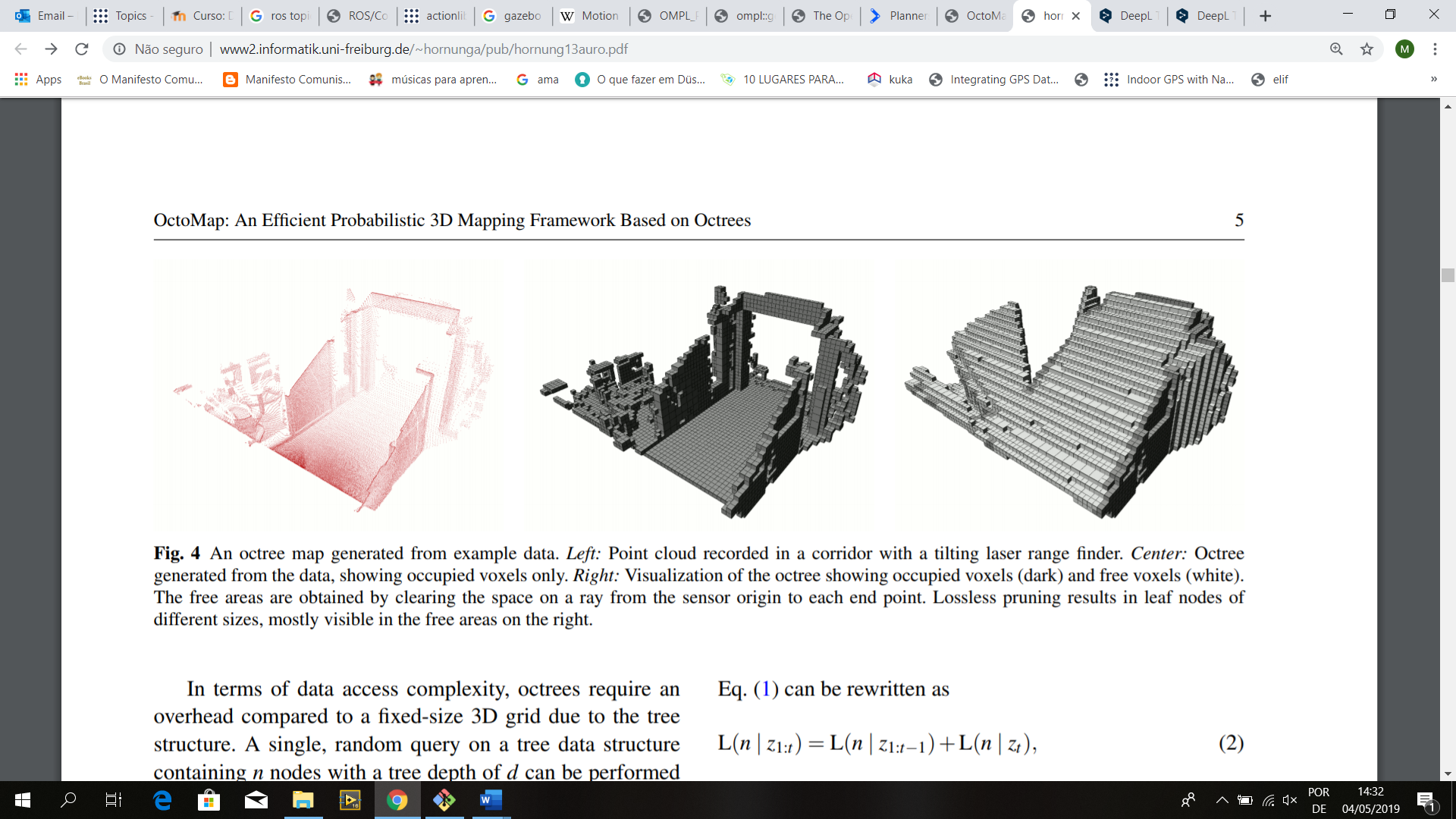
Figura - Representação de resoluções diferentes de um Octomap



Fonte: Fonte: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees

O Octotree apresenta representação das áreas ocupadas e livres. Na Figura 10 é possível observar uma representação de uma escadaria. Os dados brutos são de uma câmera que fornece dados de nuvem de pontos.

Figura - Áreas ocupadas e livres de um Octomap



Fonte: OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees

Ainda na Figura 10 podemos observar os dados brutos (esq.) a área ocupada (centro) e a área livre e ocupada (em preto e a dir.).

## Sistema de referência global e projeções de mapa

A seguir serão detalhados aspectos ligados a forma de representar a posição de um objeto sobre a superfície da terra e como representar esta posição em um mapa 2D, conivente para visualização e criação de rotas.

* + 1. Sistema de referência geométrico

Para aplicações de que requerem a geolocalização é comum a utilização das coordenadas fornecidas pelo sistema GPS ou o GLONASS (*Globalnaya navigatsionnaya sputnikovaya sistema)*. Frequentemente utilizado, o sistema GPS é mantido pelo governo americano, utiliza uma série de padrões e parâmetros para representar a posição de um objeto sobre a superfície da terra. A essa série de características, regras e parâmetros damos o nome de sistema de coordenadas geográficas.

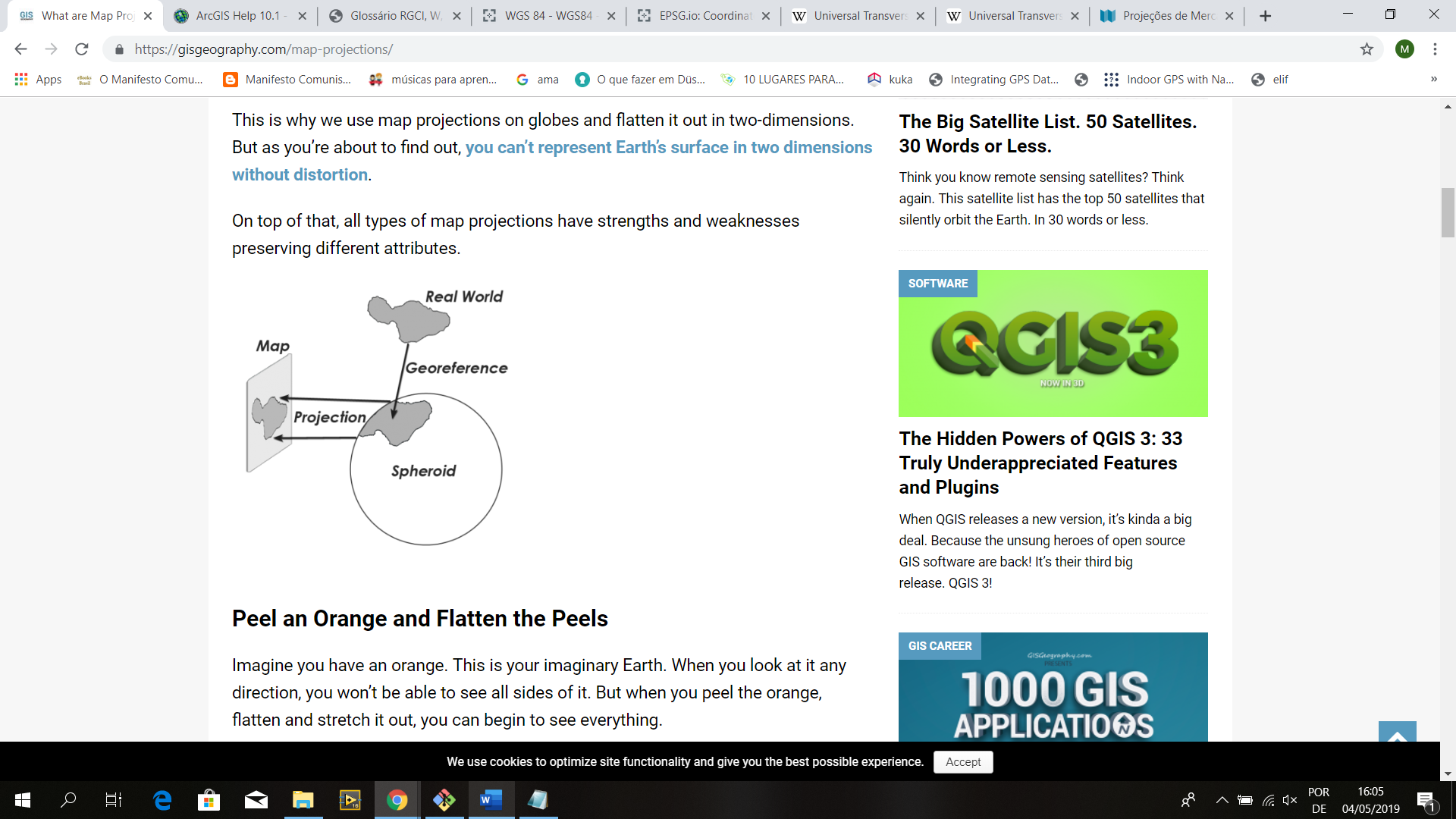
Um sistema de coordenadas geográficas utiliza a superfície de uma esfera tridimensional para estabelecer localização sobre a superfície da terra. Inclui em suas definições unidade angular de medida, um meridiano primário, e um ponto de referência baseado em um esferoide [24]. Um ponto é referenciado utilizando ângulos chamadas de latitude e longitude, que são medidos a partir do centro da terra até o ponto de interesse.

O sistema de referência geográfica utilizado pelo GPS chama-se WGS84 (World Global System 1984), foi desenvolvido pelo departamento de defeso dos EUA [25] e possui as seguintes características [26]:

* Faixa de latitude: -90 a 90
* Faixa de longitude: -180 a 180
* Meridiano primário: Greenwich
* Coordenação centrais: intercessão com equador (0,0)
  + 1. Projeção de mapas

Para representar de forma mais conveniente a superfície da terra, como para avaliar distancias, são utilizadas técnicas de projeção para de uma representação em duas dimensões [27]. Na Figura 11 podemos ver a representação de uma projeção genérica.

Figura - Mapa projetado



Fonte: What are Map Projections?

Como é impossível representar perfeitamente uma superfície 3D em uma 2D [27] sempre haverá distorções, porem existem milhares de técnicas de projeção, cada qual com suas vantagens e desvantagens. Exemplos desta projeção são a Universal Transversa de Mercator (UTM), projeção de Mercator e projeção de Peters.

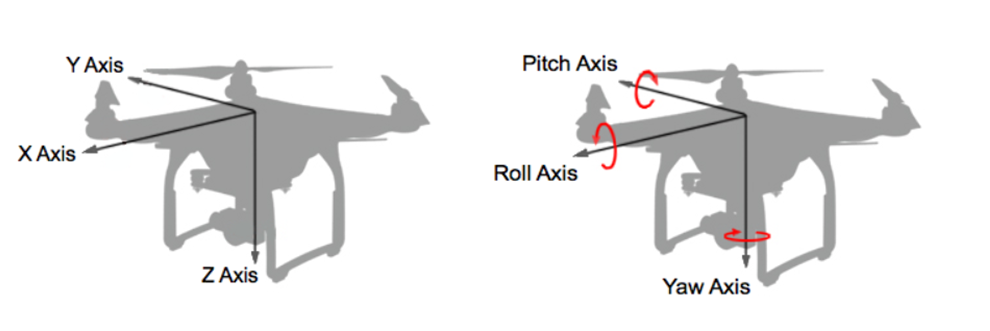
## Dinâmica de voo

Para controle de um VANT conceitos sobre o funcionamento destes equipamentos são necessários para entender seu funcionamento manual ou autônomo [28]. Estes conceitos são referentes a família de VANT oferecidos pela DJI

### Sistema de coordenadas do corpo:

Sistema coordenado centrado no centro e massa do equipamento. O sentido positivo do eixo X é direcionado para frente do equipamento, o eixo Y a direita e Z para baixo. A rotação é descrita sob os mesmos eixos utilizando a regra da mão direita. Rotação em X, Y e Z são representadas respectivamente por Roll, Pitch e Yaw. A Figura 12 abaixo traz a representação da translação e rotação.

Figura - Translação e rotação com DJI



Fonte: DJI Flight Control

### Sistema de coordenada global

Uma convenção popular em aplicações com VANT é a convenção NED (North-East-Down). Esta convenção alinha as direções X, Y e Z com as direções Norte, Leste e para baixo. No começo fica pouco intuitivo a convenção com Z para baixo, porém o sistema fica consistente uma vez que segue a regra da mão direita

### Orientação e voo

Para controlar o VANT são utilizados as combinações ângulos roll, pitch, yaw.

* Pitch: O ângulo de pitch determina se o VANT está deslocando para frente ou para traz. Para frente os motores de trás gira mais rápido, enquanto que para trás os motores da frente giram mais rapidamente.
* Roll: O ângulo de roll determina como o VANT gira em torno do eixo X. Para inclinar a direita, os motores da esquerda gira mais rápido enquanto que para girar a esquerda, os motores da direita giram mais rapidamente.
* Yaw: O ângulo yaw determina como o VANT gira em torno do eixo Z. Em aplicações com VANT metade dos motores giram em um sentido e a outra metade em outro sentido. Para girar no sentido horário os motores que assim estão configurados aceleram e para girar no sentido anti-horário os motores neste sentido giram mais rapidamente.

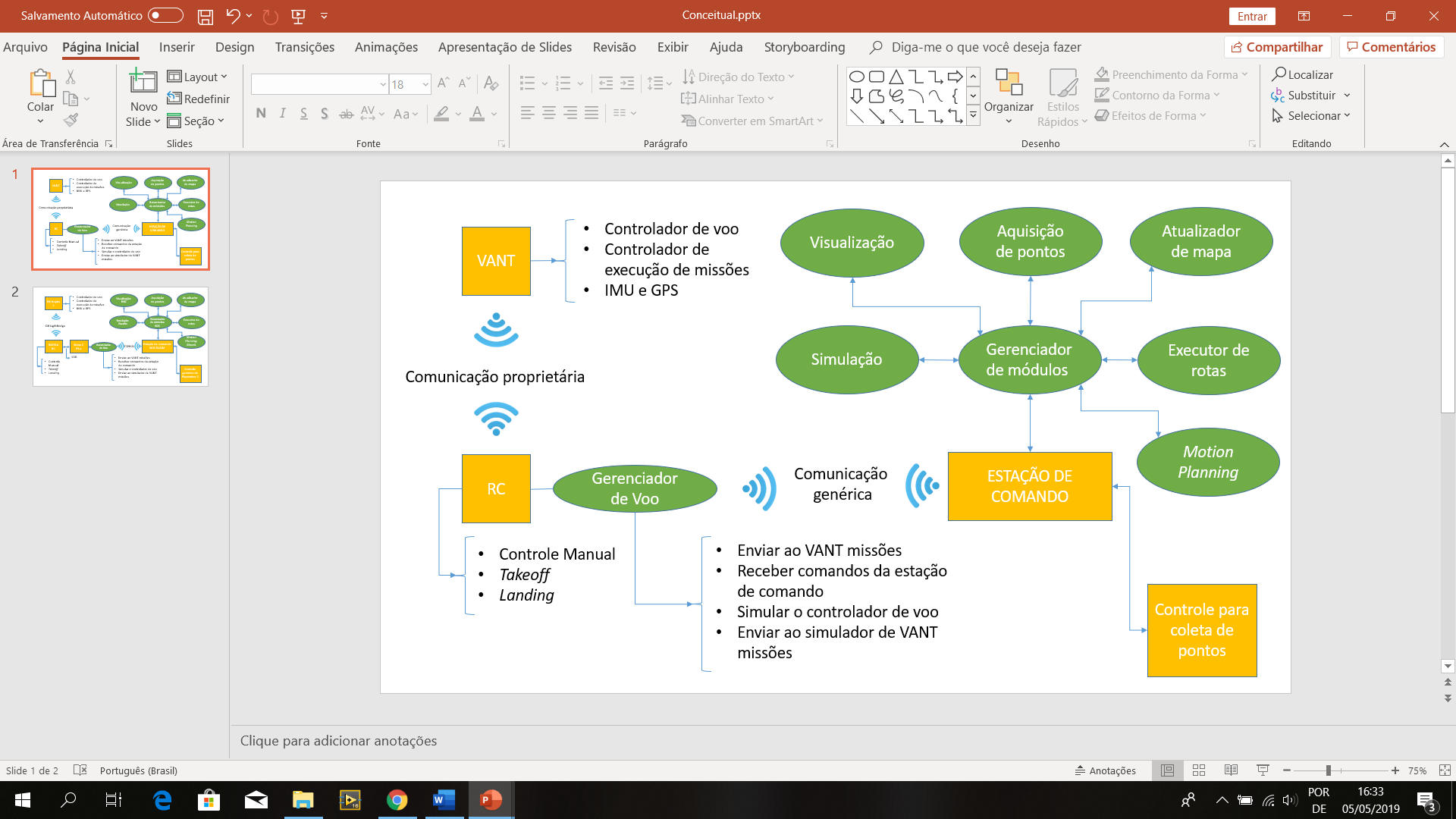
# Modelos

Este capítulo contém as descrições do modelo conceitual e posteriormente do modelo de tecnologias. No modelo conceitual, são apresentadas as funções básicas de cada módulo da solução, aquilo que cada componente deve desempenhar para que o sistema funcione de forma integrada. Em seguida é apresentado um modelo, aqui chamado de modelo de tecnologias, onde são descritas as tecnologias utilizadas para atender as premissas básicas do modelo conceitual, assim com as suas justificativas de uso.

## Modelo conceitual

Na Figura 13 abaixo podemos observar uma proposta para a arquitetura conceitual de todo sistema.

Figura - Modelo Conceitual



Fonte: Arquivo pessoal

Este modelo contém todos os sistemas necessários à execução do planejamento e execução de rotas adaptativas. Os principais módulos são descritos abaixo:

### VANT

O VANT é o veículo aéreo que será utilizado. Para tanto deve incluir em suas funcionalidades:

* Controlador de voo: deve incluir funções de alto nível para controle através de um controle manual utilizando um controlador de rádio ou posições enviadas para o VANT.
* Geração e controle alto nível de missões: possibilidade de estabelecer um conjunto de coordenadas geográficas (latitude, longitude e altura) que devem ser executas em sequência
* GPS: deve ser possível obter a posição geográfica do VANT durante voo

### Controlador de rádio

O VANT deve vir acompanhado de um controlador de rádio ou sua integração com à um deve ser possível. A principais funções a serem desempenhadas pelo controlador de rádio são:

* Controle manual: deve ser possível através do controle controlar a translação e a orientação do VANT.
* Takeoff: através do controle deve ser possível enviar o comando para que o VANT estabeleça posição padrão de takeoff, ficando a alguma distância do solo. Esta distância varia de fabricante para fabricante, sendo possível sua alteração em algumas ocasiões
* Land: deve ser através de um botão de ação rápido no controle pousar o VANT independente da posição onde estiver. Este comando deve colocar o VANT em solo.
* Comunicação com VANT: a comunicação com o VANT deve ser a proprietárias caso se utilize um controlador nativo. Caso contrário deve se seguir as instruções do fabricante para integração

### Gerenciador de voo

O gerenciador de voo é um software que deverá tratar as mensagens de controle que são geradas na estação de comando. Funciona como uma ponte entre o sistema gerador de rotas e o controle do VANT. Deve realizar as seguintes funções:

* Receber comandos da estação de comando: Deve receber as mensagens de comando da estação fixa. Deve ser capaz de executar os procedimentos referentes as mensagens. Estas mensagens são mensagens para criação, upload, e execução de rotas.
* Enviar ao VANT missões: Mediante comando da estação, o gerenciador de voo deve ser capaz de enviar os dados ao VANT.
* Simular o controlador de voo: Um simulador de voo deve ser estabelecido para que se possa executar testes de execução de rotas.
* Enviar ao simulador de VANT missões: Mediante comando da estação, o gerenciador de voo deve ser capaz de enviar os dados ao VANT.
* A comunicação com a estação de comando: Esta comunicação deve ser estabelecida preferencialmente utilizando algum protocolo de comunicação com garantia de recebimento de mensagens.

### Estação de comando

A estação de comando é o principal módulo da solução. A estação deve manter a aplicação de geração de rotas funcional. Para isso deve possuir capacidade de *hardware* e *software* suficientes para implementar suas funcionalidades. Suas principais funcionalidades são:

* Visualização: a estação de comando deve apresentar a visualização do atual estado do VANT e exibir o mapa 3D em que se está navegando com o VANT.
* *Motion Planning:* a estação deve possuir um *software* planejador de rotas que baseado em um mapa de colisão estabeleça rotas que evitem a colisão com as áreas de alta probabilidade de colisão do mapa.
* Geração de rotas adaptativas: a estação deve possuir um *software* que fornece a geração de rotas, com possibilidade de realizar variações de rotas mediante recebimento de variáveis externas
* Executor de rotas execução de rotas: a estação deve possuir um software capaz de controlar o envio de pontos para o gerenciador de voos. Pela natureza do controle adaptativa, a estação determina, sob resultado de outros processos, qual rota deverá enviar ao gerenciador.
* Simulação: a estação deve possuir um *software* ser capaz de simular o VANT em seu ambiente de operação para que se possa realizar testes e avalições de rotas geradas.
* Gerenciador de módulos: a estação deve implementar um gerenciador de módulos que integra os diferentes softwares que

consistem no módulo da estação.

### Controle para coleta de pontos

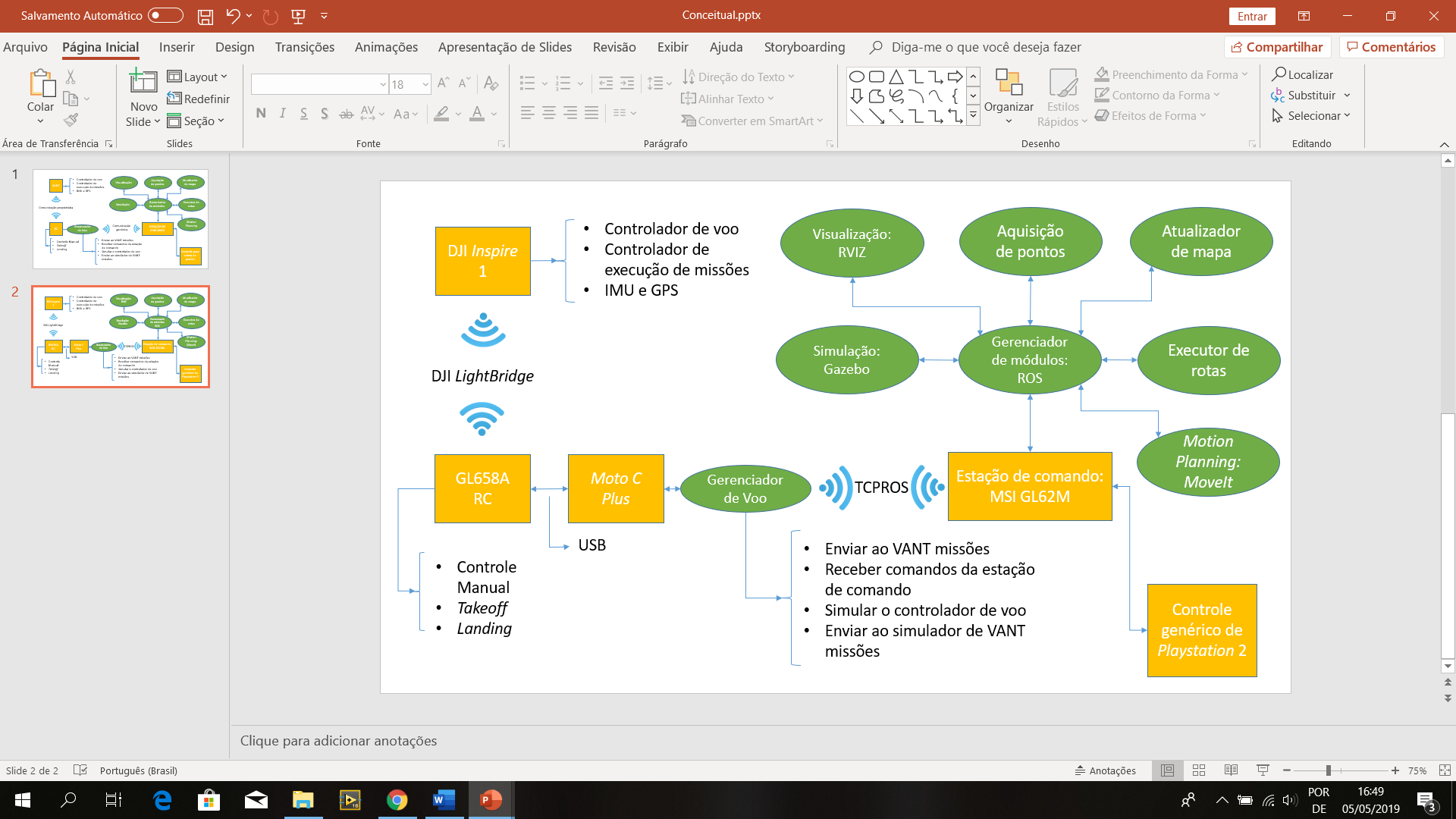
O controle para coleta de pontos deve ser implementado para que se possa controlar o VANT dentro do ambiente simulado, permitindo a coleta de pontos em diferentes posições para posterior cálculo de rotas. Suas principais funções são:

* Controle de orientação e translação: através do uso do controle deve ser possível deslocar o VANT dentro do ambiente virtual alterando a posição cartesiana e a orientação em volta do eixo Z (yaw).
* Comando para coleta de pontos: um botão deve estar disponível para que se possa selecionar a posição atual como ponto para posterior cálculo de rota
* Comando para armazenar pontos: após a coleta dos pontos um botão deve estar disponível para enviar o comando para salvar os pontos em um arquivo
* Comando para ativar modo automático: deve estar disponível um comando para mudar o VANT para o modo automático (autônomo)
* Comando para ativar modo manual: deve estar disponível um comando para acionar o modo manual.

## Modelo de tecnologias

Na Figura 14 abaixo apresenta uma proposta para a arquitetura de tecnologias e como comunicam entre si para atender as demandas do modelo conceitual proposto anteriormente.

Figura - Modelo de tecnologias



Fonte: Arquivo pessoal

Os principais módulos são descritos abaixo:

### DJI Inspire 1, GL658A e LightBridge

Como opção à função de VANT e controlador de rádio, este trabalho foi desenvolvido com o uso do VANT DJI *Inspire 1* e o controlador de rádio que vem de fábrica com o multicoptero, o GL658A.

A opção pelo DJI *Inspire 1* e o controlador de rádio GL658Aé justificada principalmente pela sua imediata disponibilidade e por atender a todos os requisitos descritos na secção 4.1.1 e 4.1.2

O protocolo LightBridge é um protocolo proprietário da DJI e permite além da comunicado de longo alcance (5km), o download simultâneo de imagens da câmera integrada ao VANT [29]. A obrigatoriedade de seu uso atende aos requisitos necessários para comunicação e facilita o desenvolvimento uma vez que não é necessário desenvolver um protocolo de comunicação próprio para controlar o VANT.

* + 1. ROS

A escolha do ROS como gerenciador de módulos do sistema é principalmente justificada pela disponibilidade de pacotes, ambiente de desenvolvimento colaborativo e pelo tipo de licença de uso.

Aliado a variedade de pacotes disponíveis, mais de 3000 [30], o desenvolvimento de ferramentas para ROS conta com uma comunidade bastante ativa. A comunidade conta com mais de 10.000 usuários distribuídos em sites de fórum e suporte, a wiki do ROS recebe cerca de 30 atualizações diárias [30].

Para fins comerciais o ROS é interessante pois, sua licença de uso é do tipo BSD (*Berkeley Software Distribution*), que é bastante permissiva para o reuso de código em aplicações comerciais de código fechado [30].

### Estação de comando MSI FL6258M 7REX + Linux

Como *hardware,* a estação de comando é implementada em um notebook MSI GL62M 7RE com sistema operacional Ubuntu 16.04 instalado. A escolha pelo computador é justificada pelo poder de processamento (Intel i7 H *series*), placa gráfica dedicada (GTX 1050Ti), capacidade de armazenamento (1 TB) e pela disponibilidade imediata. A escolha pelo sistema operacional Linux realizado devido a recomendações dos desenvolvedores quanto a confiabilidade oriunda de testes realizados em plataforma Linux [31]

### Moto C Plus, Gerenciador de Voo e comunicação com Estação de Comando

Para acessar as funções de mais baixo nível do DJI, é necessário utilizar o DJI *Mobile* SDK (*Software Development Kit*) que fornece duas bibliotecas, uma para *Android* e outrapara *iOS* [32]*.* Devido a imediata disponibilidade do telefone Moto C *Plus,* pertencente ao autor, este aparelho foi usado como *hardware* do gerenciador de voo.

A comunicação entre o Moto C Plus e o DJI é estabelecida unicamente através de conexão USB [33]. E devido ao sistema no aparelho ser o *Android Nougat 7.0* a biblioteca de que se utilizou para o *software* do gerenciador de voo foi a disponível para *Android*.

Para comunicação entre o gerenciador de voo e a estação de comando, foi utilizada uma biblioteca para Java que implementa a comunicação com ROS em aplicações distribuídas. Esta biblioteca foi utilizada porque na estação de comando o sistema gerenciador de pacotes é o ROS e é desejável aproveitar os padrões de comunicação garantidos pelo *framework* para evitar o desenvolvimento de um protocolo próprio de comunicação para apenas troca de mensagens.

Para realizar a comunicação esta biblioteca utiliza o TCPROS. O TCPROS é uma camada de transporte para mensagens e serviços que estabelece comunicações padrão TCP *socket* para realizar o transporte das mensagens [34]

### Aquisição de pontos, atualizador de mapa, executor de rotas e controle USB genérico

Os módulos de *software* que desempenham as funções de aquisição de pontos, atualizador de mapa e executor de rotas foram implementados em Python devido a disponibilidade de API para desenvolvimento para ROS. A escolha por Python se deve maior disponibilidade de funções de alto nível e pela facilidade de desenvolvimento e teste a opção em C++.

A escolha pelo controle USB genérico é justificada pela disponibilidade imediata e pelo controle disponível atender as demandas de botões para realizar as funções de coleta, armazenamento de arqueios, troca de modos e os controles analógicos para realização de controle de translação e orientação do VANT virtual.

### RViz, Gazebo e *MoveIt*

A escolha para visualização do ambiente 3D e do VANT foi o visualizador RViz. Este módulo acompanha a instalação da versão completa do ROS e possui confiável integração para visualização de variáveis, sensores e câmeras A escolha pelo Gazebo é devido a sua integração com ROS, vir acompanhado da instalação da versão completa e por possuir os plugins para simulação de GPS, colisão e aceitar modelos STL e COLLADA conforme descritos em 3.1.7.

O uso do pacote *MoveIt* é justificado pelo fato do mesmo possuir integração com a biblioteca OMPL, que conforme descrito em 3.2 apresenta técnicas com custo reduzido de processamento para estabelecer um rota livre de colisões. O uso do *MoveIt* também é justificado por aceitar mapas colisões originados com o protocolo Octomap descrito em 3.3. O uso do Octomap é justificado pelo seu compacto armazenamento, cálculo probabilístico de localização e por ser possível gerar um mapa a partir de um modelo 3D [35].

### RViz, Gazebo e *MoveIt*

# Desenvolvimento

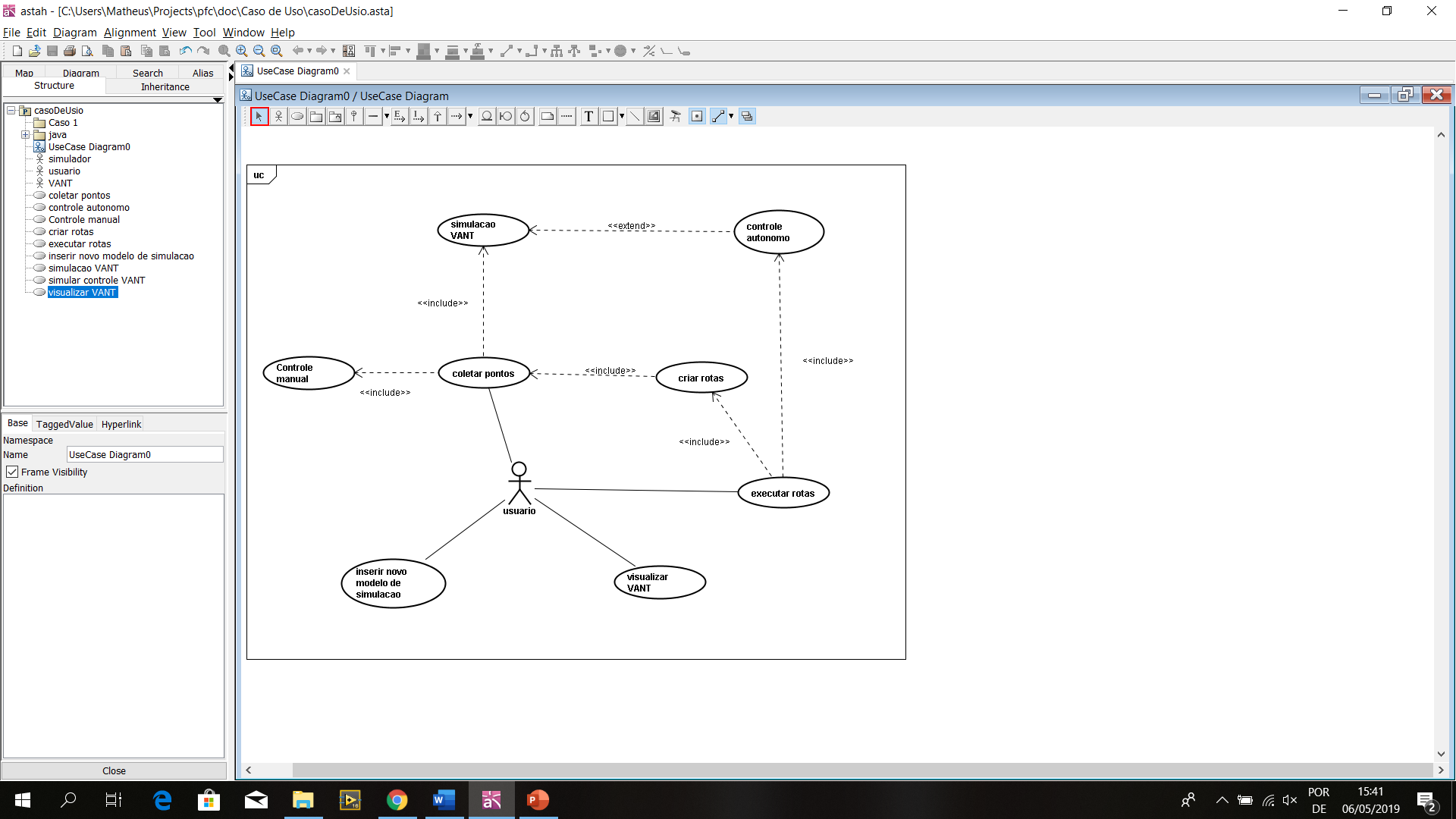
Este capítulo aborda o desenvolvimento de software referente ao sistema proposto neste trabalho. A descrição da implementação está dividida em 5 seções cada qual descrevendo os resultados de parte da metodologia utilizada para a execução do plano de desenvolvimento.

A metodologia utilizada para o desenvolvimento de software pode ser descrita como uma metodologia ágil que de forma abrangente envolveu a aquisição de requisitos, proposta de arquitetura e por fim iterações de desenvolvimento e teste com constante alteração nos requisitos da aplicação.

## Casos de uso

Como forma de captação de requisitos utilizou-se de um diagrama de casos de uso para análise das principais funções que o sistema deve desempenhar. Abaixo na Figura 1, observa-se no diagrama de casos de uso, as principais funcionalidades que o sistema na visão do usuário.

Figura - Diagrama de caso de uso



Fonte: Arquivo pessoal

A seguir a descrição dos casos de uso principais:

* Visualizar VANT: visualização do estado atual do VANT e do ambiente em que se encontra
* Inserir novo modelo de simulação: o usuário deseja inserir um novo modelo 3D que representa para o sistema o ambiente que o VANT irá navegar
* Coletar pontos: o usuário navega com o VANT no ambiente virtual, através do caso de uso controle manual e simulação VANT, e seleciona pontos para gerar rotas.
* Executar rotas: o usuário seleciona um arquivo originado a partir do caso de uso coleta de pontos. Com este arquivo rotas são criadas, através do caso de uso criar rotas, e enviadas para o controlador de rotas que controla a execução das trajetórias no VANT que opera de forma autônoma, caso de uso controle autônomo
* Controle autônomo: este caso ainda pode estender sua funcionalidade para rodar em ambiente simulado

## Requisitos

Para levantamento de requisitos além da modelagem do caso de uso, utilizou-se dados coletados de algumas reuniões de acompanhamento. Devido ao estágio inicial do projeto, e pelo fato de estar associado ao um módulo experimental da solução, requisitos originados pelo cliente não puderam em grande parte compor a Tabela 1 que segue abaixo

Tabela - Requisitos de software

|  |  |
| --- | --- |
| **F1. Coletar posições** | |
| Descrição: O sistema deve possuir um módulo de coleta de pontos | |
| Requisitos não-funcionais | |
| Identificação | Restrição |
| F1.1 | A coleta de pontos deve ser realizada em ambiente simulado |
| F1.2 | O deslocamento do VANT para coleta deve ser realizado através de controle manual |
| F1.3 | O controle utilizado para coleta de pontos deve ser um controle USB genérico com no mínimo 3 botões de funções disponíveis |
| F1.4 | O módulo de coleta de pontos deve possuir interface com ROS |
| F1.5 | O módulo de coleta de pontos deve gerar uma lista de posição e orientação dos pontos coletados |
| F1.6 | O módulo de coleta de pontos deve implementar a função de seleção de ponto, salvar arquivo e zerar pontos |
| F1.7 | O módulo de coleta de pontos deve gerar arquivos com nomes sequenciais |
| **F2. Criar rotas de inspeção** | |
| Descrição: O usuário do sistema deve ser capaz de criar rotas de inspeção | |
| Requisitos não-funcionais | |
| Identificação | Restrição |
| F2.1 | A rota deve ter pelo menos 2 pontos |
| F2.2 | A rota não deve colidir com obstáculos |
| F2.3 | A rota deve manter distância mínima de 1 metro de qualquer objeto |
| F2.4 | A rota deve ser criada com base em um mapa 3D inserido pelo usuário |
| F2.5 | A rota deve ser criada com base em lista de posições coletadas |
| F2.6 | A rota deve ser gerada por um algoritmo de *sampled-motion planning* em no máximo 1 segundo |
| F2.7 | O algoritmo de *sampled-motion planning* deve possuir parâmetro configurável para distância máxima |
| F2.8 | O algoritmo de *sampled-motion planning deve* possuir parâmetro configurável área de interesse |
| F2.9 | O algoritmo de *sampled-motion planning* deve gerar pontos o mais próximo possível |
| F2.10 | O algoritmo de *sampled-motion* planning deve usar um Octomap como fonte de dados de colisão |
| F2.11 | A inserção de novos Octomap deve ser feita de forma manual |
| F2.12 | O Octomap pode ser atualizados durante execução |
| F2.13 | A rota deve ser criada utilizando lógica de nível |
| F2.14 | A rota deve ser composta por elementos com dados de posição, orientação e nível |
| F2.15 | Nível é um número inteiro de 1 à infinito |
| **F3. Executar de rotas de inspeção** | |
| Descrição: O usuário do sistema deve ser capaz de executar rotas de inspeção | |
| Requisitos não-funcionais | |
| Identificação | Restrição |
| F3.1 | A execução da rotas quando em aplicações com VANT real deve ser realizada em ambiente simulado e real ao mesmo tempo |
| F3.2 | A execução da rotas pode ser realizada somente em ambiente simulado |
| F3.3 | A execução da rotas deve ser constituída de módulo gerenciador de voo e módulo executor de rotas |
| F3.4 | Os módulos gerenciador de voo e executor de rotas devem possuir interface com ROS |
| F3.5 | Os módulos gerenciador de voo e executor de rotas devem comunicar-se somente via ROS (TCPROS) |
| F3.6 | Os módulos gerenciador de voo e executor de rotas devem ser implementados em sistemas terrestres |
| F3.7 | O módulo gerenciador de voo deve possuir conexão com a internet |
| F3.8 | O módulo gerenciador de voo deve possuir uma interface de criações de missões |
| F3.9 | O módulo gerenciador de voo deve possuir uma interface de execução de missões |
| F3.10 | O módulo gerenciador de voo deve possuir uma interface de leitura de posição e orientação |
| F3.11 | O módulo executor de rotas deve executar rotas com base em arquivos de rota |
| F3.12 | O módulo executor de rotas deve converter os pontos da rota para coordenadas geométricas WGS84 (latitude, longitude e altura) |
| F3.13 | O módulo executor de rotas deve filtrar as rotas para que tenham pelo menos 0.5 m de distância entre um ponto e outro |
| F3.14 | O módulo executor de rotas deve aguardar recebimento de sinal de rota concluída vindo do módulo gerenciador para continuar a enviar segmentos da rota global |
| F3.15 | O módulo executor de rotas deve executar as rotas por lógica de nível |
| F3.16 | O módulo executor de rotas deve executar em sequência rotas de mesmo nível |
| F3.17 | O módulo executor de rotas deve aguardar sinal de decisão quando há alteração de nível de rotas |
| **F4. Simular ambiente com VANT** | |
| Descrição: O sistema deve ser capaz de gerar um ambiente virtualizado com VANT inserido | |
| Requisitos não-funcionais | |
| Identificação | Restrição |
| F4.1 | A simulação do ambiente com VANT deve permitir a aplicação de forças nos objetos |
| F4.2 | A simulação do ambiente com VANT deve permitir a inserção de modelos STL e COLLADA |
| F4.3 | A simulação do ambiente com VANT deve incluir um VANT om câmera RGB e point cloud |
| F4.4 | A simulação do ambiente com VANT deve prover física e de sensores |
| F4.5 | A simulação do ambiente com VANT deve prover interface de controle de translação e rotação |
| F4.6 | A simulação do ambiente deve ser mantida por um módulo atualizador de mapa |
| F4.7 | O módulo atualizador de mapa deve manter uma cena com um Octomap e parâmetros de colisão |
| **F5. Simular controlador de voo do VANT** | |
| Descrição: O sistema deve ser capaz de gerar uma simulação de um ambiente com um VANT inserido | |
| Requisitos não-funcionais | |
| Identificação | Restrição |
| F5.1 | A simulação do controlador de voo deve permitir interface de controle de translação e rotação |
| F5.2 | A simulação do controlador de voo deve permitir leitura de posição e orientação e envio |
| F5.3 | A simulação do controlador de voo deve permitir configurar um volume de simulação do VANT |
| **F6. Visualização do VANT** |  |
| Descrição: O sistema deve ser capaz de gerar um ambiente virtualizado com VANT inserido | |
| Requisitos não-funcionais | |
| Identificação | Restrição |
| F6.1 | A visualização do VANT deve permitir visão de primeira pessoa no VANT |
| F6.2 | A visualização do VANT deve permitir visão do Octomap |
| F6.3 | A visualização do VANT deve permitir realizar *sampled-motion planning* entre ponto inicial e ponto de interesse |
| F6.4 | A visualização do VANT deve permitir escolha de diferentes tipos de algoritmos de *sampled-motion planning* |
| F6.5 | A visualização do VANT deve permitir a visualização da rota planejada |

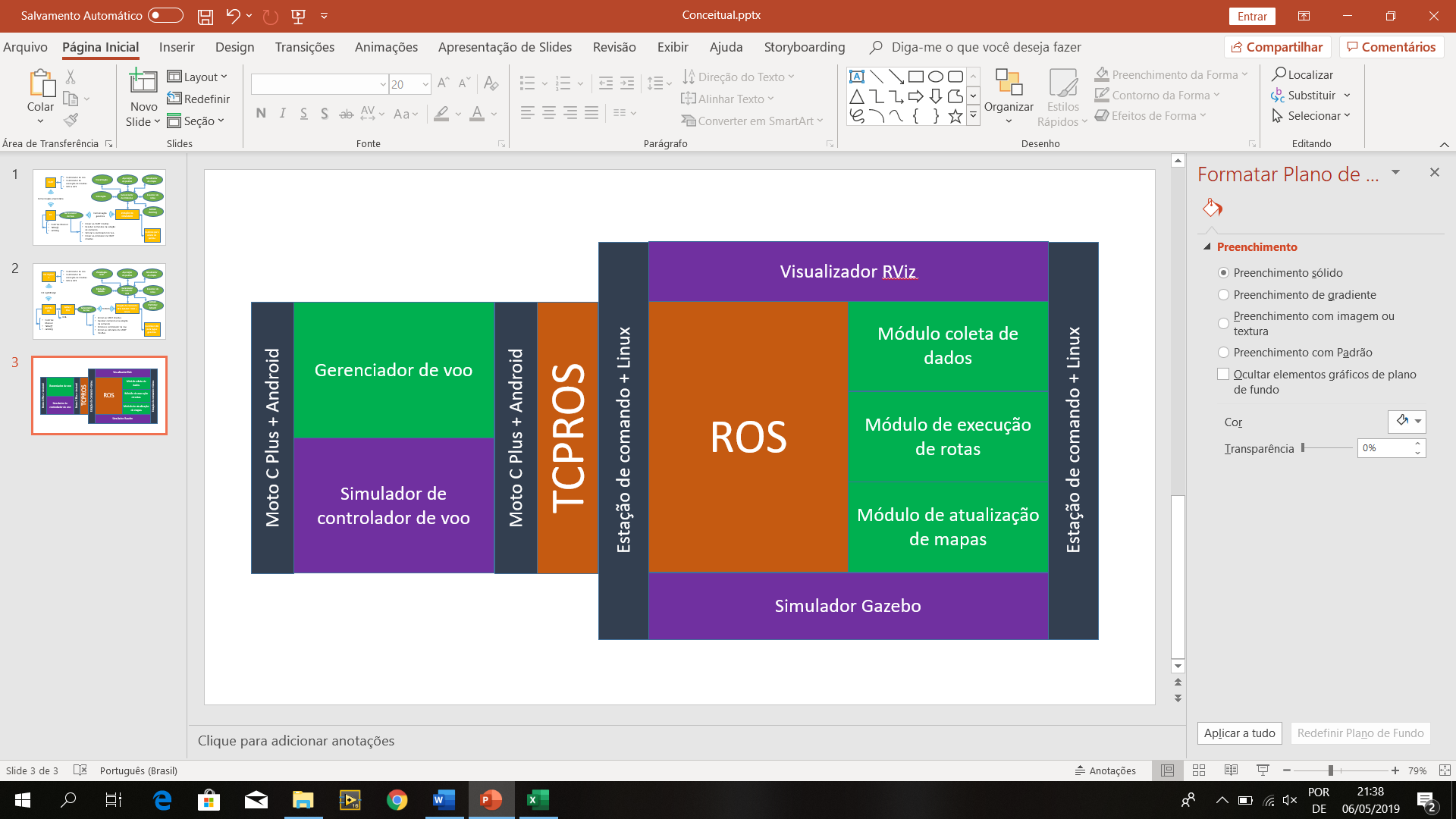
Fonte: Arquivo pessoal

Os requisitos aqui apresentados foram determinados durante o desenvolvimento como básicos para se estabelecer um sistema de navegação. Podemos destacar aqui por exemplo o requisito de distância mínima de obstáculos.

## Arquitetura

Segundo os requisitos descritos acima uma arquitetura de software foi realizada afim de se atendar as demandas anteriormente expostas. A Figura 16 traz a visão de arquitetura de software:

Figura - Arquitetura de software



Fonte: Arquivo pessoal

Os itens em azul escuro são os sistemas físicos que contem os módulos descritos nos retângulos interiores. A comunicação entre os dois sistemas físicos é realizada via TCPROS. Os sistemas físicos implementam:

* Celular Moto C Plus: Sistema operacional Android com módulos de gerenciador de voo e simulador de controlador de voo da DJI.
* Estação de comando: Sistema operacional Linux Ubuntu 16.04, com ROS, módulo de coleta de pontos, módulo de execução de rotas, módulo de atualização de mapas, simulador Gazebo e visualizador RViz.

A arquitetura acima foi realizada pensando no desenvolvimento de um sistema que possui uma espécie de cópia digital, onde em um primeiro momento simulações podem ser realizadas para se obter as melhores rotas e parâmetros para posteriormente executar as rotas no VANT real de forma sincronizada com sua virtualização. Desta forma é possível o acampamento em tempo real do estado do VANT em um ambiente virtualizado.

## Implementação

A implementação do sistema foi realizada sob os módulos especificados na arquitetura. Nas seções que seguem, serão descritos os e as tecnologias utilizadas na implementação dos módulos.

### Visualizar VANT + simulação

Para a implementação da simulação e visualização foi utilizado um pacote de para ROS com integração com simulação em Gazebo, que orginalmente fora desenvolvido para execução de mapeamento com base em Octomap e posterior navegação em ambiente virtual utilizando um VANT genérico com uma câmera Kinect. Este modelo foi desenvolvido por Will Selby e atende às especificações descritas na seção de requisitos 5.2 [36].

Posteriormente como será descrito, houveram modificações no pacote original, como por exemplo, o modelo original, que foi substituído por um de uma subestação. Na Figura 17 pode-se observar os nodos (losangos) e tópicos e serviços (retângulos) que constituem o pacote utilizado.

Figura - Visualização e simulação do VANT: Nodos, tópicos e serviços

Uma imagem contendo mapa, texto

Descrição gerada automaticamente

Fonte: Arquivo pessoal

A compressão das interfaces de alguns dos nodos é essencial para o entendimento da implementação do trabalho. A seguir serão descritos os principais módulos e suas interfaces:

#### move\_group:

O node move\_group é pertencente ao pacote *MoveIt*. É o principal nodo deste pacote, sendo a central de planejamento de rotas do sistema desenvolvido. A este nodo são realizadas as requisições de planejamento de rotas, que são respondidas em tópicos de resultado.

* Subscrições
  + planning\_scene: tópico onde dados de percepção do ambiente são mantidos. Contém dados brutos que definem o Octomap, localização do Octomap e distância mínima das juntas do VANT ao mapa. Atualização desse tópico é realizada por nodos de percepção.
  + joint\_states: tópico onde os valores das juntas que o MoveIt controla deve ser publicado por outros nodos.
* Serviços
  + move\_group/action\_topics: através desse serviço é possível solicitar o planejamento de rota entre um ponto e outro. Por meio de protocolo específico, é possível selecionar o algorítmico utilizado e a área de permitida para determinação da rota.
  + multi\_dof\_joint\_trajectory\_action/action\_topics: através desse serviço é possível realizar a execução de rotas no VANT simulado. Este serviço precisa de valores de entrada, que são as rotas produzidas com o serviço anterior.

#### *action\_controller*:

Este nodo realiza o controle da execução dos resultados do nodo *move\_group*. Este nodo adquire todas as posições de uma rota planejada e controla a sua execução. A interface é descrita abaixo:

* Publicações:
  + cmd\_3dnav: este tópico possui uma missão completa publicada por este controlador. Posteriormente este tópico será subscrito pelo controlador de missões que executará cada ponto da missão
* Subscrições
  + multi\_dof\_joint\_trajectory\_action/action\_topics: Dentre os action topics fornecidos pelo nodo *MoveIt* está multi\_dof\_joint\_trajectory\_action/result que ao término de um planejamento de rota bem sucedido publica as rotas de inspecao. O *action\_controller* redireciona as trajetórias para o tópico cmd\_3dnav

#### n\_rviz:

Este nodo implementa a interface gráfica de usuário. O RViz permite uma flexibilidade elevada em relação aos dados que podem ser visualizados, é possível selecionar tópicos de acordo com o teste que está sendo realizado. Os principais tópicos são o *planning\_scene* que contém a definição do Octomap e das trajetórias que no mapa podem ser realizadas e os tópicos de câmera, que permitem a visualização em primeira pessoa do ambiente virtual. Abaixo a figura uuu do RViz reproduzindo o mapa da solução original, com visualização do planning\_scene e visão de primeira pessoa:

Mapa da cozinha uuu

Os principais serviço vistos no nodo n\_rviz são o planejamento de rota (*plan*) e execução de rota (*execute*), conforme visto na barra de ferramentas da figura uuu. Após o planejamento é possível observar uma linha de sucessão de estados que representa a trajetória proposta pelo algoritmo escolhido. A figura iiii mostra um pequeno de trecho de planejamento.

Figura iiii

#### gazebo:

O nodo gazebo fornece a simulação do VANT e o modelo do ambiente. Publica e subscreve em tópicos informações de sensores para que o controle do VANT seja possível. A seguir as principais publicações e subscrições do nodo gazebo

* Publicações:
  + quad/camera\_/depth/points: este tópico possui os dados de *point cloud* gerados pelo sensor Kinect simulado. Estes dados se desejável servem para atualizacao do Octomap durante a inspeção.
  + quad/ground\_truth/odometry: este tópico possui informações de iGPS (*indoor GPS*)para alimentar o controlador de posição do VANT.
  + /tf: este tópico possui a transformada entre o sistema global e a base do VANT. Este dado é necessário para que o nodo move\_group conheça a posição atual do VANT.
* Subscrições
  + quad/command/motor\_speed: este tópico possui as 4 velocidades que são configuradas aos motores do VANT simulado no gazebo. As velocidades são resultado do controlador de orientação
  + Quad/imu: este tópico possui os dados referentes as acelerações do VANT simulado nos três eixos coordenados. O controlador de orientação é alimentado com os dados deste tópico.

#### quad/joy\_node:

Este nodo realiza a interface entre a porta USB da estação de comando e o *framework* ROS. É realizado a transferência dos comandos mecânicos gerados no controle *USB* para um tópico na camada ROS. Sua ação é dada orientada a eventos do usuário do controle, somente publica novas mensagens no ROS quando o usuário pressiona o controle. Abaixo a descrição da principal publicação deste nodo:

* Publicações
  + quad\_joy: este tópico contém valores de entrada de botos binários e analógicos do controle físico.

#### quad/quad\_joystick\_interface

Este nodo realiza a interpretação dos botões pressionados transferindo-os para entrada de controle do VANT. Suas funções são a troca entre modos manual e autônomo, publicar as acelerações e impulsos sobre eixos coordenados, no caso de manual. Suas principais publicações são descritas abaixo:

* Publicações:
  + quad/command/trajectory: neste tópico são publicados os comandos de mudanças de estado. Os possíveis estados de controle do VANT são manual e automático
  + quad/command/roll\_pitch\_yawrate\_thrust: neste tópico são publicados os valores de aceleração nos eixos coordenados quando o modo de controle do VANT é manual. É realizada validação e quando necessário, arredondamentos para valores máximos de aceleração sobre os eixos coordenados caso ultrapassem os limites.

#### quad/waypoint\_node

Este nodo é o controlador de missão do VANT. Este controlador checa se novas posições são publicadas no tópico cmd\_3dnav e dispara eventos de controle de cada posição da missão. Realiza o controle de missa do VANT. Sua interface é descrita abaixo:

* Publicações
  + quad/command/waypoint: este tópico possui dados de um ponto da missão.
* Subscrições
  + /cmd\_3dnav: este tópico possui os *setpoints* da missão publicados pelo *action\_controller*
  + quad/command/trajectory: este tópico possui dados relacionados ao modo como o VANT deve operar, manual ou autônomo
  + quad/ground\_thruth: este tópico possui dados sobre a localização do VANT no ambiente simulado. Este dado é utilizado pelo controlador para checar se o ponto da missão foi alcançado

#### quad/position\_controller\_node:

Este nodo é o controlador de posição do VANT. Sua interface é descrita abaixo:

* Publicações
  + quad/command/roll\_pitch\_yawrate\_thrust: quando em modo manual este tópico receberá dados deste controlador. Os dados são a aceleração nos eixos coordenados e o impulso.
* Subscrições
  + quad/command/waypoint: este tópico possui o *setpoint* que deve ser controlado no VANT
  + quad/ground\_truth/odometry: este tópico possui os dados de localização do VANT no ambiente simulado

#### quad/atitude\_controller\_node:

Este nodo é o controlador de orientação do VANT. Este controlador controla os ângulos roll, pitch e yaw através da modificação das velocidades dos motores no VANT simulado. É utilizado em ambos os modos de funcionamento, manual ou automático. Sua interface é descrita abaixo:

* Publicações
  + quad/command/motor\_speed: neste tópico são publicadas as velocidades dos motores. A lógica de como a posição e orientação do VANT pode ser modificada através da alteração da velocidade é descrita na seção 3.5
* Subscrições
  + quad/imu: este tópico possui dados de aceleração do VANT medidos pelos sensores simulados do Gazebo. É utilizado pelo controlador para controlar os ângulos nos eixos.
  + quad/command/roll\_pitch\_yawrate\_thrust este tópico possui as *setpoints de* angulo que devem ser controladas pelo nodo.

### Instalação do ROS

O ROS possui diferentes distribuições. O objetivo de versionar o ROS é garantir a validação e confiabilidade de uma coleção de pacotes desenvolvidos para uma determinada versão após correções da comunidade. Uma nova versão do ROS é lançada sempre que uma nova versão do Ubuntu é lançada [37]

O critério para decisão de qual versão instalar foi tomado com base no pacote utilizado par a simulação do ambiente virtual e do VANT descrito na seção 5.4.1. Esta decisão foi tomada com prioridade a este pacote pois este apresenta muitos sub pacotes que são essenciais para o funcionamento da lógica de *motion planning* e de movimentação do VANT virtual. O pacote foi originalmente implementado utilizando a versão Kinetic Kame do ROS, portanto esta foi a distribuição escolhida.

A instalação da versão completa ROS Kinetic é bem genérica e dispensável de descrição de detalhes. Foi realizada conforme instruções do site do ROS.org (<http://wiki.ros.org/kinetic/Installation/Ubuntu>).

* + 1. Inserir modelo de simulação e mapa de colisão.

Esta seção trata da inserção do modelo 3D pelo usuário no ambiente de simulação e na conversão do modelo 3D para um mapa de colisão no formato *Octomap*. Com base no mapa de colisão, os nodos do *MoveIt* realizam o planejamento com base nas informações de áreas livres e ocupadas que o mapa de colisão possui.

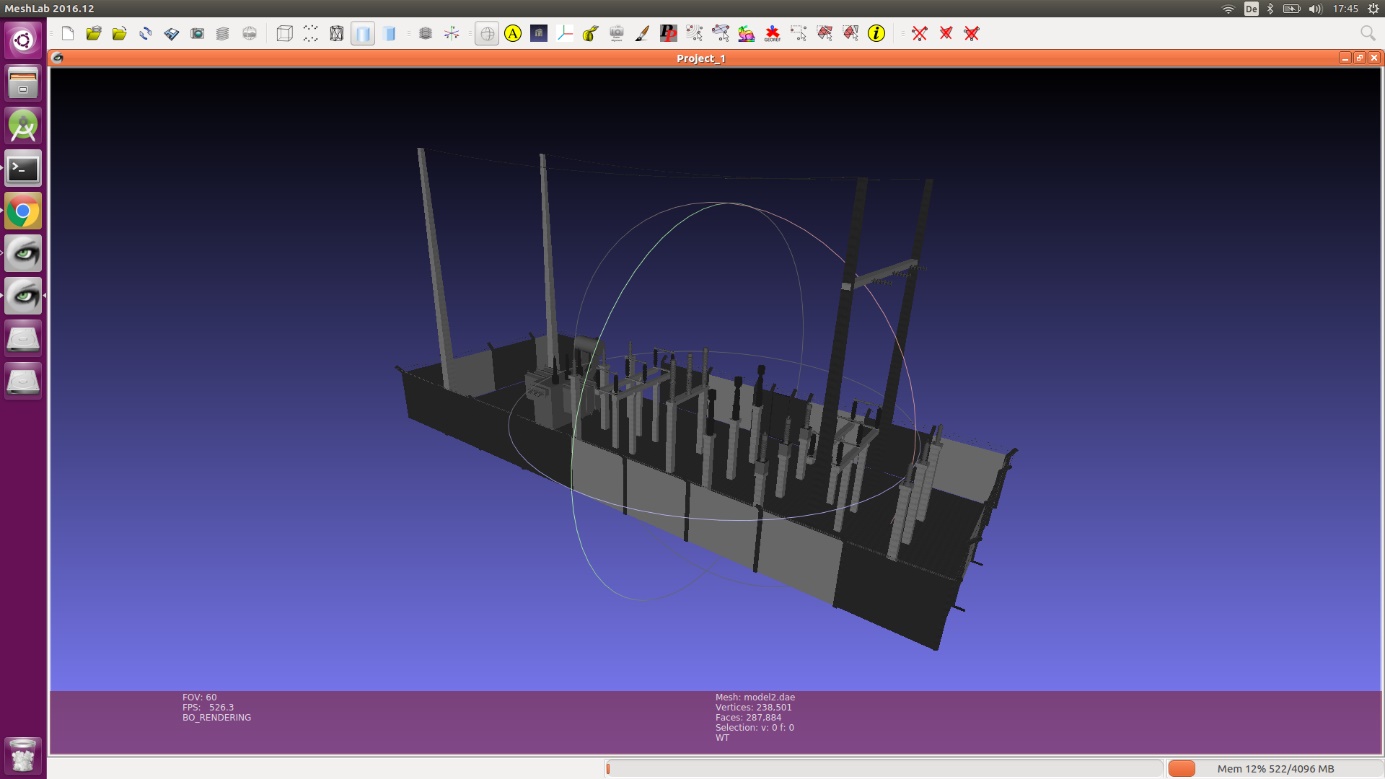
Para o trabalho em desenvolvimento optou-se por não automatizar o processo de conversão de modelo 3D para o mapa de obstáculos. A justifica é fundada no fato de que esse processo mesmo em ambientes virtuais tende a ocorrer com rara frequência. Em aplicações reais a conversão do modelo 3D do ambiente poderia servir como um primeiro mapa do ambiente, porém devido a dinâmica e mudança nos ambientes é recomendada a atualizacao constante através sensores de *point cloud*.

A seguir serão descritos os procedimentos utilizados na conversão de um modelo 3D de subestação para o arquivo que representa o *Octomap* deste modelo.

* Encontrar modelo

O modelo de subestação foi encontrado no site [https://3dwarehouse.sketchup.com](https://3dwarehouse.sketchup.com/). Neste site, modelos 3D gratuitos podem ser encontrados. Na busca pelo modelo optou-se por um modelo que contivesse isoladores, transformadores e cabos. O formato para inserção no Gazebo (simulação) é preferencialmente COLLADA, portanto optou-se por um modelo neste formato. O modelo é representado no software MeshLab (<http://www.meshlab.net/>) na Figura 18.

Figura - Modelo 3D de simulação



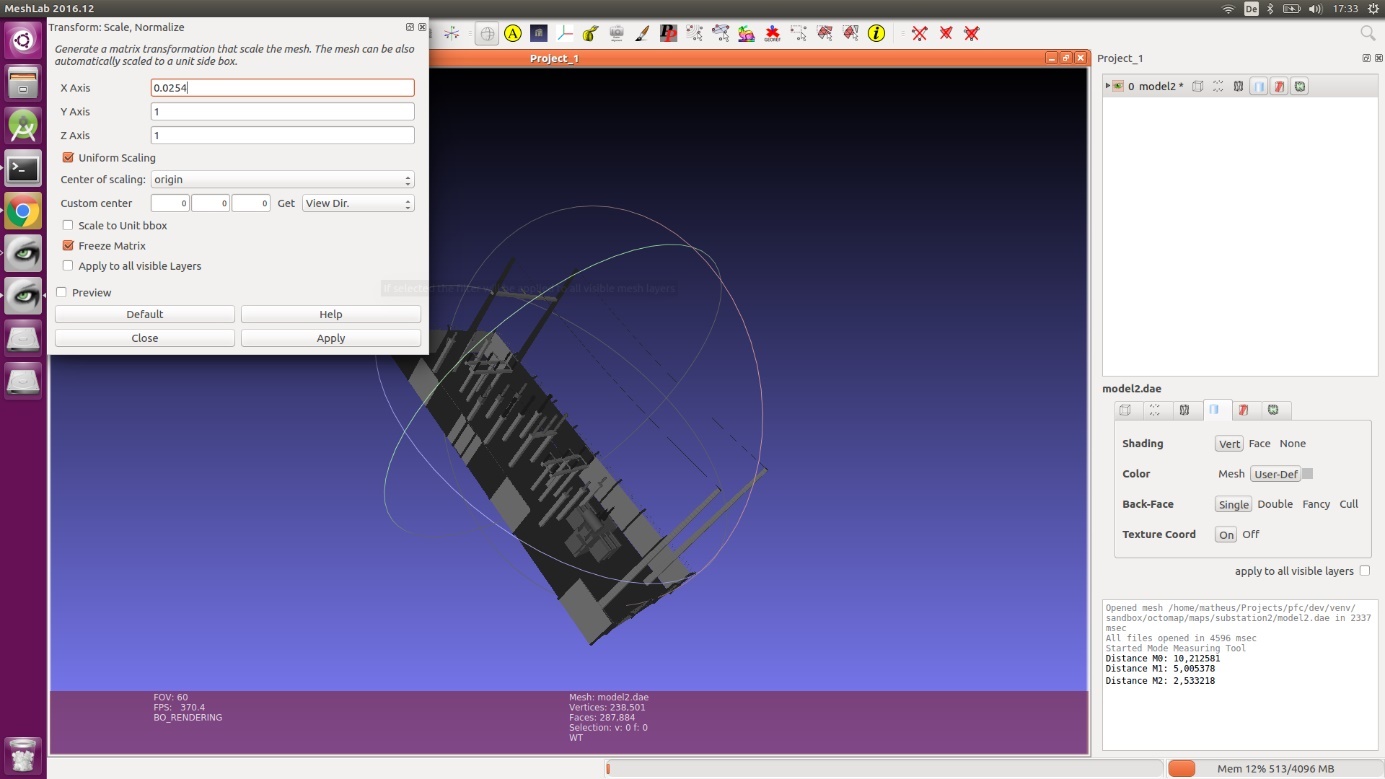
Fonte: arquivo pessoal

* Converter para STL

Para realizar a conversão do modelo COLLADA para Octomap é necessário converter o modelo para um formato intermediário, neste caso .STL. A razão desta conversão é que o algoritmo utilizado para conversão do modelo 3D para Octomap aceita preferencialmente como entrada .STL. Conversão foi realizada no software MeshLab.

Maioria dos modelos no formato COLLADA disponíveis no [https://3dwarehouse.sketchup.com](https://3dwarehouse.sketchup.com/) estão originalmente configurados com unidade de polegadas. O software MeshLab espera que os arquivos estejam com unidade configurada em metros, portanto é necessário realizar um ajuste de escala neste caso. Para realizar modificação todas as dimensões foram escaladas usando o fato de multiplicação de 0.0254. Na Figura 19 o processo de alteração de escala é exibido:

Figura - Alteração de escala no Modelo 3D



Fonte: Arquivo pessoal

Um vez obtido o mapa na escala correta, o assistente de exportação é utilizado para converter o modelo para o formato .STL

* Converter para formato Octomap

O procedimento para converter um modelo 3D STL para um formato *Octomap* com árvore hierárquica de voxels consiste em dois processos, transformação do modelo 3D em um *grid* de voxels e posterior transformação *grid*  em uma árvore hierárquica.

Para transformar o modelo 3D em um grid de *voxels* foi utilizado o programa binvox. O binvox cria um *grid b*inário de volxels [35]. O binvox é encontrado no site do desenvolvedor (<http://www.patrickmin.com/binvox/>). O resultado será um arquivo .binvox

Em seguida este *grid* de *voxels* é transformado em uma árvore hierárquica de voxels através do programa binvox2bt Figura 20 - Octomap construído. Este programa vem como parte do pacote *Octomap* (<http://wiki.ros.org/octomap>) para ROS. A árvore gerada representa as regiões livres e ocupadas, que são usadas como entrada para algoritmos de *motion planning* como os contidos no pacote *MoveIt* que serão utilizados neste trabalho. Para detalhes de como a árvore representa os espaços livres e ocupadas ver seção 3.3. A entrada são arquivos .binvox e a saída arquivos .bt

O arquivo .bt pode ser visualizado através do programa octovis (<http://wiki.ros.org/octovis>). Na Figura 20 o resultado da conversão do modelo original para o Octomap.

Figura - Octomap construído

Uma imagem contendo captura de tela

Descrição gerada automaticamente

Fonte: Matheus Kraemer

A informação de *leaf size* na Figura 20 é o tamanho das arestas do menor voxel, portanto pode ser entendido como a resolução do mapa obtido.

* Carregar modelo 3D e mapa de colisão

Para utilizar o modelo 3D na aplicação é necessário colocar os arquivos em uma pasta específica para que o Gazebo identifique o modelo. Para tornar o mapa de colisão identificável pelo *MoveIt* o arquivo .bt pode ser colocado em qualquer diretório. Este diretório deve ser lembrado pois será necessário informar o caminho do mapa de colisão para que o módulo atualizador de mapa alimente o *MoveIt.*

Para tornar o modelo 3D executável pelo Gazebo é necessário colocar o modelo 3D no diretório ~/.gazebo/models. A pasta pode ter qualquer nome e deve conter os seguintes arquivos:

* model.sdf

O arquivo SDF (*Simulation Description Format*) contém a descrição de visual físico e de colisão do modelo, dos arquivos de malha utilizados e a posição inicial do modelo. A Figura 21 mostra que o modelo é posicionado em z = -15 metros. Isso se deve a um erro de modelagem no arquivo original, que possui offsets em X, Y, e Z.

Figura - Código modelo 3D arquivo SDF

Fonte: arquivo pessoal

* model.config

O arquivo model.config contém a definição de nome, versão do modelo, nome do arquivo SDF.

Figura - Código modelo 3D arquivo config

Fonte: Arquivo pessoal

* model.dae: arquivo COLLADA do modelo 3D
  + 1. Módulo de atualização de mapas

O módulo de atualizacao de mapas realiza a leitura do arquivo *Octomap* do modelo 3D convertido na seção 5.4.3, realiza a modificação de alguns parâmetros do mapa, e atualiza, com uma determinada frequência, um tópico chamado de *planning\_scene* incluindo o mapa, parâmetros e a transformada entre a base do VANT simulado e o sistema de referencial global do ambiente simulado.

O tópico *planning\_scene* é utilizado pelo *MoveIt* como fonte de informações sobre o mapa de colisão e pelo RViz para visualização dos planejamentos e execuções de rotas. Principais publicações e subscrições:

* Publicações
  + *planning\_scene*
* Subscrições:
  + *tf*
  + *octomap\_full*

A seguir as partes do código referentes às suas principais funções serão detalhadas

#### *Imports* e inicialização

Na Figura 23 podemos observar o código referente aos *imports* do módulo.

Figura - Módulo atualizador de mapa: Imports



Fonte: Arquivo pessoal

As explicações sobre o código da Figura 23 seguem abaixo:

* Linhas 2 e 3: referências a biblioteca que se comunica com o ROS. Através desta biblioteca é que se torna possível ler, escrever, e utilizar tópicos e serviços disponíveis no ROS.
* Entre linhas 4 e 6: estes comandos referenciam para os tipos de mensagens utilizados pelo módulo atualizador de mapas. Para publicar ou subscrever em algum tópico é necessário estabelecer um tipo de mensagem no mecanismo de comunicação, esse tipo deve ser o mesmo tipo publicado no caso de uma subscrição
* *Linha 7*: este comando referência uma biblioteca utilizada para ler transformadas entre *frames* no ROS.
* Linha 8: inicialização da varável que receberá os dados de *planning\_scene*.
* Linha 9: *flag* utilizada no módulo para saber quando os dados do *Octomap server* foram recebidos
* Linha 10: variável utilizada no módulo para armazenar os dados do arquivo *Octomap* recebidos do *octomap\_server*
* Linha 11: inicialização do objeto que recolhe valores de transformadas entre *frames* que são fornecidas na camada ROS. Este objeto irá servir para ler a transformada entre o frame do VANT simulado e o sistema de referência do mundo simulado
* Linha 12: Inicialização de um objeto temporizador do ROS. Neste caso quando acionado trava a execução do programa por 4 segundos, pois está configurado para 0.25 Hz
* Linha 14: este comando utiliza a biblioteca do ROS para Python para inicializar um nodo chamado

Na Figura 24 podemos observar a inicialização do nodo.

Figura - Módulo atualizador de mapa: inicialização



Fonte: Arquivo Pessoal

As explicações sobre o código da Figura 24 seguem abaixo:

* Linha 42: este comando criar um nodo chamado atualizador\_mapa
* Linha 43: este comando inicializa a subscrição no nodo *octomap\_full*. Este tópico é publicado pelo nodo *octomap\_server*. O nodo *octomap\_server* lê o arquivo de *octomap* e publica com o tipo de dados *Octomap*, portando esse subscritor é do tipo *Octomap.* O comando especifica a função *cb* como função de *callback* ao receber a mensagem. Detalhes na seção de função de suporte
* Linha 44: este comando cria um objeto pelo qual se realiza publicações com o tipo de mensagem *PlanningScene*. As atualizações que o nodo atualizador de mapa realizada, posição do VANT e mapa, são concatenas e publicadas através deste objeto.

#### Funções de suporte

A função detalhada nessa seção é a função *cb* que recebe os valores do *Octomap* via tópico. A implementação da função de suporte *cb* é mostrada abaixo na Figura 25:

Figura - Módulo atualizador de mapa: função cb



Fonte: Arquivo pessoal

Os detalhes são detalhados abaixo:

* Entre as linhas 16 e19: este comando referência para variáveis de globais do código.
* Linha 20: este comando salva os dados lidos na variável *octomap*
* Entre 22 e 32: nestas linhas a variável do tipo *PlanningSceneWorld* psw é inicializada preenchida com informações de tempo, frame de referência, octomap e a posição do octomap em relação ao frame de referência. Vale notar aqui que há uma correção na posição do modelo, pois o modelo escolhido continha um erro de dimensionamento com offset em Z e Y, portanto aqui este erro é corrigido.
* Entre linhas 34 e 40: compreendido nestas linhas está a inicialização da vaiável mapMsg do tipo *PlanningScene*. Esta variável recebe em seu atributo *world* a variável psw. Ao final do procedimento a variável *flag* é configurada para 1 indicando ao software o recebimento do mapa de custo.

#### Main

A primeira parte da implementação da função de suporte *main* é mostrada abaixo na Figura 26:

Figura - Módulo atualizador de mapa: main parte 1



Fonte: arquivo pessoal

Os detalhes são detalhados abaixo:

* Linha 46: este comando trava o programa até o recebimento da primeira transformada entre a base do VANT e o frame do sistema simulado.
* Linha 48: este comando mantem o *while loop* executando até que o ROS *master* seja interrompido.
* Linha 50: esta condição espera que que a *flag* de recebimento do mapa de custo esteja ativa, para então executar os procedimentos necessários para publicar a *PlanningScene*
* Linha 55: Este comando realiza um tentativa de ler a transformada entre a base do VANT e o frame de coordenadas do sistema simulado para salvar nas variáveis trans e rot os valores de translação e rotação respectivamente.
* Entre 60 e 70: Nestas linhas é criado uma área de segurança no elo *base\_link\_intertia*. Este elo ele representa a base do VANT, e para assegurar que o mesmo não vá se aproximar de obstáculos é criado um enchimento de 0.2 m em torno do elo. A informação do enchimento de elo é inserida na variável *mapMsg.*
* Entre 72 e 79: Nestas linhas as características das juntas são definidas e inseridas na variável *mapMsg*. São configurados os nomes e a posição de cada junta. O sistema de controle de juntas não utiliza as informações de juntas por este meio, portanto os valores podem ser zerados. Estes valores são por motivos de visualização
* Linha 81: Aqui o nome do modelo do robot que está na cena é configurado e armazenado na variável *mapMsg*.

A segunda parte do código de implementação da função *main* é exibido na Figura 27

Figura - Módulo atualizador de mapa: main parte 2

Fonte: Arquivo pessoal

Os detalhes são detalhados abaixo:

* Entre 98 e 108: Nestas linhas é criada a variável *transform* e nesta armazenadas as informações de translação e rotação do VANT em relação ao eixo coordenado do ambiente simulado. Por fim a variável *transform* é inserida na propriedade *robot\_state* da variável mapMsg.
* Linha 110: A variável *mapMsg* é publicada no tópico *PlanningScene* pelo publicador *pub*.
* Entre 112 e 144: Estas linhas são percorridas caso o mapa de colisão ainda não tenha sido recebido pelo nodo atualizador de mapas. Caso percorre o tempo de 4 segundos é esperado.
  + 1. Módulo de coleta pontos
    2. Módulo de executor de rotas
    3. Gerenciador de voo + simulação de controlador de voo

## Teste

O teste foi realizado de tal e tal forma,

Inicialização

* Estação de comando

Conexão com uma rede LAN com internet

A configuração de ROS master foi realizada

O launch file foi executado

Explicar launch file

* Celular Moto C Plus

Celular na mesma rede

O aplicativo foi inicializado

IP do master inserido

Coleta de pontos

Execução de pontos

Onde se aborda a implementação/implantação da solução escolhida.

Deve-se fazer uma descrição do "projeto" e/ou software que se fez, diagramas, Casos de Uso, interfaces gráficas, ambientes de desenvolvimento (e porque os escolheram), TIs usadas (e porque as selecionaram), etc.

Aqui também deve ser colocada a Metodologia usada (não confundir isso com a lista das fases do cronograma), no sentido de “como” (técnicas e procedimentos) cada uma das fases foi planejada e executada. Por exemplo, com que técnica ou procedimento se testou, avaliou ou validou a solução, se verificou se ela realmente atendeu aos anseios dos usuários, se ela atendeu a todos os requisitos especificados, etc.

# Título (resultados)

Analise e especificao

Onde se apresentam e **são** **discutidos** os resultados (simulados, piloto ou reais) obtidos no decorrer da pesquisa/desenvolvimento.

Nesta discussão é importante dar meios para que o leitor entenda os resultados obtidos e como foram obtidos.

Deve-se fazer uma análise dos resultados: análise crítica profunda sobre o que foi feito, os resultados atingidos, os prós e contras, impactos, como isso resolveu o problema, os ganhos obtidos, etc.

^

Automaatizar a conversão de modulos

Colcocar camera para atualizar modelo

Atualizar modelo durante voo

Usar um VANT novo

# 7 ConSIDERAÇÕES FINAIS e Perspectivas

Síntese pessoal, objetiva, sucinta e interpretada dos resultados do trabalho.

Grosso modo, deve-se apresentar um resumo do que foi feito, dos resultados globais (frente aos objetivos inicialmente traçados). Exemplos:

- o método deu certo? funcionou? deu o resultado esperado? Foi melhor que o método anterior?

- impactos organizacionais, tecnológicos, financeiros, éticos, ecológicos, etc., tidos (ou potencialmente a ter) com a introdução do que foi proposto.

De forma complementar, se pertinente, sugestões para trabalhos futuros de continuação.

# REFERÊNCIAS

[1] <http://www.brasil.gov.br/noticias/infraestrutura/2011/11/energia-eletrica-chega-a-97-8-dos-domicilios-brasileiros-mostra-censo-demografico>

[2] <http://www.aneel.gov.br/home?p_p_id=101&p_p_lifecycle=0&p_p_state=maximized&p_p_mode=view&_101_struts_action=%2Fasset_publisher%2Fview_content&_101_returnToFullPageURL=%2F&_101_assetEntryId=14476909&_101_type=content&_101_groupId=654800&_101_urlTitle=faq&inheritRedirect=true>

[3] <http://ons.org.br/paginas/sobre-o-sin/o-que-e-o-sin>

[4] <http://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-160/topico-168/Anuario2018vf.pdf>

[5] [http://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-245/topico-261/DEA%20001\_2017%20-%20Proje%C3%A7%C3%B5es%20da%20Demanda%20de%20Energia%20El%C3%A9trica%202017-2026\_VF[1].pdf](http://www.epe.gov.br/sites-pt/publicacoes-dados-abertos/publicacoes/PublicacoesArquivos/publicacao-245/topico-261/DEA%20001_2017%20-%20Proje%C3%A7%C3%B5es%20da%20Demanda%20de%20Energia%20El%C3%A9trica%202017-2026_VF%5b1%5d.pdf)

[6] <https://www.ccee.org.br/portal/faces/pages_publico/onde-atuamos/setor_eletrico?_adf.ctrl-state=vrcxfrzk1_50&_afrLoop=48308490450560#!>

[7] <http://www2.aneel.gov.br/arquivos/PDF/atlas3ed.pdf>

<https://www.ros.org/about-ros/>

<https://robohub.org/ros-101-intro-to-the-robot-operating-system/>

<http://wiki.ros.org/Nodes>

<http://wiki.ros.org/Packages>

<http://wiki.ros.org/Topics>

<http://wiki.ros.org/Services>

<http://wiki.ros.org/actionlib>

<http://gazebosim.org/>

SCHÜTZ, E. **Reengenharia mental**: reeducação de hábitos e programação de metas. Florianópolis: Insular, 1997. 104 p. 1 espaço simples entre cada referência

SÓDERSTEN, B.; GEOFREY, R. **International economics**. 3. ed. London: MacMillan, 1994. 714 p.

Importante: mencionar apenas as obras **efetivamente utilizadas**! Toda obra citada no texto deve estar nesta lista de referências.

Deve-se seguir as orientações da NBR 6023/2002.

***Observação final***: As recomendações acima são de dois tipos:

* de organização da monografia* (devem ser cumpridas rigorosamente) e

* de conteúdo* (sugere-se que sejam cumpridas).

Recomenda-se bom senso e, acima de tudo, o cumprimento à risca das sugestões dos seus orientadores.

# APÊNDICE A – Título (opcional)

Textos elaborados pelo próprio autor, a fim de complementar a sua argumentação

# Anexo a – título (opcional)

Documentos não elaborados pelo autor, utilizados de fundamentação (mapas, leis, estatutos).