Insper

Veículo Terrestre Autônomo para Monitoramento de Talhões de Silvicultura e Plantações de Frutas

Felipe Catapano Emrich Melo
Gabriel Brunoro Motta Tumang
Luana de Matos Sorpreso
Rafael Eli Katri

Trabalho de Conclusão de Curso

Relatório do Projeto Intermediário - Capstone

São Paulo - SP – Brasil
Agosto 2024

Felipe Catapano Emrich Melo
Gabriel Brunoro Motta Tumang
Luana de Matos Sorpreso
Rafael Eli Katri

Veículo Terrestre Autônomo para Monitoramento de Talhões de Silvicultura e Plantações de Frutas

Trabalho de Conclusão de Curso

Relatório do Projeto Intermediário - Capstone

Relatório apresentado aos respectivos cursos de Graduação em Engenharia (Computação/Mecânica/Mecatrônica), como requisito parcial para a obtenção do título de Bacharel no respectivo curso.

Professor Orientador: Prof. Dr. Vinicius Licks

Mentor: Thiago Teixeira Santos

Coordenador TCC/Capstone: Prof. Dr. Luciano Pereira Soares

São Paulo - SP – Brasil
Agosto 2024

Felipe Catapano Emrich Melo
Gabriel Brunoro Motta Tumang
Luana de Matos Sorpreso
Rafael Eli Katri

Veículo Terrestre Autônomo para Monitoramento de Talhões de Silvicultura e Plantações de Frutas

Relatório apresentado aos respectivos cursos de Graduação em Engenharia (Computação/Mecânica/Mecatrônica), como requisito parcial para a obtenção do título de Bacharel no respectivo curso.

Professor Orientador: Prof. Dr. Vinicius Licks

Banca Examinadora

Prof. Dr. Vinicius Licks

Insper

Prof. Dr. Fabio Ferraz Junior

Insper

Prof. Dr. Fabricio Jailson Barth

Insper

Prof. Me. Fabio Roberto de Miranda

Insper

RESUMO

Este projeto visa desenvolver um sistema de navegação autônoma para um robô de monitoramento de silvicultura e pomares montado pela iteração anterior deste Capstone. O projeto tem como propósito automatizar o processo de monitoramento de tais plantações, possibilitando a otimização da mão de obra humana. O programa para esta funcionalidade nova será desenvolvido utilizando o *framework* ROS 2, e mais especificamente o pacote de navegação NAV 2. Para conseguir atingir o objetivo final, foi estudado pelo grupo múltiplas soluções e algoritmos de SLAM compatíveis com fusão de sensores de múltiplos tipos, para mapeamento multissensorial robusto em ambientes não estruturados com terreno irregular. Para achar a solução ideal para os requisitos de projeto, foi adotado a ferramenta da matriz de soluções.

Palavras-chave: Navegação Autônoma; Monitoramento de plantações; SLAM; ROS 2; NAV 2; UGV;

ABSTRACT

This project aims to develop an autonomous navigation system for a forestry and orchard monitoring robot assembled by the previous iteration of this Capstone. The project aims to automate the monitoring process of such plantations, enabling the optimization of human labor. The program for this new functionality will be developed using the ROS 2 framework, and more specifically the NAV 2 navigation package. To achieve the final objective, the group studied multiple solutions and SLAM algorithms compatible with sensor fusion of multiple types, for robust multisensory mapping in unstructured environments with uneven terrain. To find the ideal solution for the project requirements, the solution matrix tool was adopted.

Keywords: Autonomous Navigation; Plantation Monitoring; SLAM; ROS 2; NAV 2; UGV.

1 Introdução

1.1 Proposta do projeto

O objetivo do projeto é a implementação de um sistema de navegação autônoma e de monitoramento de trajetória para um UGV, ou seja, um veículo terrestre não tripulado. O veículo deve ser capaz de planejar seu movimento e enviar informações de posicionamento enquanto navega por corredores de talhões, a unidade básica de plantio agrícola. A iteração atual faz parte de um contexto maior de projetos de UGV's para uso agrícola, representando continuidade ao Capstone anterior responsável pela montagem do robô a ser utilizado, capaz de atuar em pomares e talhões de silvicultura (ARAÚJO *et al*, 2024).

Diferentemente da iteração passada, o UGV não é operado manualmente com um controle remoto. O robô deverá, com o auxílio de uma ferramenta de software interativa, receber um conjunto de pontos chamados de *waypoints* e, sem comandos adicionais do operador, ser capaz de navegar de ponto a ponto autonomamente. Para isso, foi desenvolvido um módulo separado de navegação autônoma, composto por um conjunto de sensores e uma unidade computacional dedicada, que deverá processar tarefas de localização, mapeamento e navegação e se comunicar com o serviço de atuação dos motores do módulo base do UGV. Isso se deve principalmente pelo caráter crítico da atuação dos motores do veículo, que não deve ser comprometido pelos processos alto nível da navegação autônoma.

A navegação autônoma e o acompanhamento do percurso via interface interativa servirão de base para aplicar futuramente técnicas de agricultura de precisão, principalmente no que se trata de monitoramento de plantio. Esse caso de uso traz valor para o agronegócio dado que pode ser usado para controle de pragas e planejamento de plantações. Esse tipo de monitoramento poderá ser explorado em futuras iterações, que beneficiam da navegação do robô pelos talhões e a publicação de informações de telemetria que informam o seu posicionamento.

O UGV, desde a iteração anterior, utiliza a framework ROS. Robot Operatintg System (ROS) é um *framework open-source* de robótica que padroniza o desenvolvimento de robôs e facilita a colaboração entre usuários. Nesse sentido, o ROS oferece padrões de comunicação entre sensores, atuadores e computadores, permitindo que o desenvolvedor não se preocupe com a implementação baixo nível da transmissão de dados. Além disso, o ROS disponibiliza ferramentas de visualização e definição de *waypoints*.

O framework permite que projetos sejam convertidos em pacotes que podem ser facilmente instalados e acessados por outros desenvolvedores. Desse modo, a comunidade ROS de robótica mantém pacotes open-source sobre as mais diversas áreas de atuação no campo de robótica. ROS também é relevante para aplicações no mercado de trabalho, sendo que mais de 900 empresas o utilizaram em 2023, com um aumento de 22.9% em relação ao ano anterior (SCOTT, 2024).

No que diz respeito a navegação, os pacotes ROS utilizados fazem parte da plataforma NAV 2. Esses pacotes são desenvolvidos pela Open Navigation, uma empresa que cria soluções open-source para tarefas de navegação. O NAV 2 é uma plataforma utilizada no mercado, sendo usada por empresas como Toyota, LG e Bosch (NAV 2, 2024). Um grande diferencial da plataforma é sua modularidade, sendo que é fácil customizar o comportamento de seus planejadores, ou até mesmo criar comportamentos usando o sistema de plugins (DOCS NAV 2, 2024). Outro ponto forte é a disponibilidade de plugins prontos com algoritmos de planejamento e suavização de rotas, árvores de comportamento, entre outros (DOCS NAV 2, 2024).

Um sistema de navegação autônoma depende de técnicas de localização e mapeamento para obter informações do posicionamento do robô em relação a um ponto de referência (Corke, 2011, p. 125). Uma solução para isso é uso de GPS, um sistema que estima a posição diretamente com uso de satélites. Sensores GPS-RTK são comumente usados, oferecendo precisão na escala de centímetros. Entretanto, para o caso de uso do projeto, no contexto da silvicultura com áreas densas de árvores, é conhecido que essas áreas sofrem de bloqueios e reflexões de sinal (AGUIAR *et al*, 2020), o que gera instabilidades.

Nesse sentido, uma técnica relevante é o SLAM, que consiste na localização e mapeamento simultâneos de um ambiente não explorado (CORKE, 2011, p. 167). Existem diversas implementações de SLAM, incluindo SLAM baseado em LiDAR's 3D e SLAM visual que são relevantes para ambientes externos (FASIOLO *et al*, 2023). Outra questão relevante é o SLAM multi-sensor, que realiza fusão de sensores para evitar o acúmulo de erros na coleta dos sensores, também chamado de *drift* (FASIOLO *et al*, 2023). Isso é relevante em ambientes externos, que são sujeitos a efeitos climáticos desfavoráveis, uma vez que as leituras de um sensor podem compensar falhas do outro (ZHU, 2024).

Soluções de SLAM usando LiDAR 2D são conhecidas, computacionalmente eficientes e robustas para ambientes fechados, entretanto elas não têm a mesma eficácia no contexto do

projeto como fonte única de sensoriamento (REGER, 2022) e podem assumir um papel auxiliar na fusão de sensores. Esse auxílio pode se dar tanto no refinamento da odometria visual, quanto na otimização do mapa com a identificação de *loop closures* (LABBÉ, 2019).

Com essas questões em mente, propõe-se utilizar uma câmera de profundidade Intel Realsense D435i e um LiDAR 2D Robotis LDS-02 para as tarefas de localização e mapeamento, em conjunto com o módulo de GPS. Será utilizado o pacote ROS RTAB-MAP, um projeto open-source da Université de Sherbrooke, que se diferencia por oferecer suporte a uma variedade de sensores e pela possibilidade de realizar SLAM multi-sensor. A câmera Realsense será utilizada para extrair odometria visual e coletar *features* usando o modo RGB-D e o IMU interno da câmera será usado como referência para aperfeiçoar o SLAM. O LiDAR 2D também será utilizado no refinamento da odometria visual e na otimização do mapa com a identificação de *loop closures*.

1.2 Origem do Desafio

Com base no relatório final do semestre passado foi pensado na origem do desafio, considerando que o resultado de ambos os projetos é o mesmo. A Embrapa (Empresa Brasileira de Pesquisa Agropecuária) é uma empresa que foca em pesquisa para a agropecuária no Brasil. Se trata de uma empresa pública vinculada ao Ministério da Agricultura e Pecuária (MAPA), criada em 1973.

O objetivo do desenvolvimento do UGV no contexto agrícola é facilitar o monitoramento de talhões de fruticultura e silvicultura, permitindo que esse trabalho se torne mais preciso e rápido. O talhão é uma parte do solo que representa a unidade mínima de cultivo.

A separação por talhões é de extrema relevância para a fruticultura pois permite um controle por área da fruta, dessa forma, é possível isolar um talhão que esteja contaminado por pragas, por exemplo. Além disso, cada talhão pode conter um tipo de fruta específico, o que faz com que seja possível uma maior organização no plantio e na colheita. Já no caso da silvicultura, o talhão também se mostra muito importante, uma vez que, a área florestal é dividida em unidades territoriais de acordo com características específicas da floresta, assim cada talhão tem o seu método silvicultural adequado.

Dessa forma, é evidente que o monitoramento dos talhões é essencial para o controle de pragas e o planejamento de toda a produção e colheita. Por conta da grande relevância do setor agrícola no Brasil, o monitoramento de talhões realizado por seres humanos pode ser dificultado por conta do grande número de talhões ou até mesmo sua extensão, dessa forma, a inclusão de

UGVs nessa etapa do processo produtivo pode ser de grande auxílio para a realização dessa tarefa, além de permitir que haja um planejamento maior sobre aquela área.

1.3 Escopo do projeto

Este projeto tem como principal objetivo transformar o robô montado pela equipe anterior em um UGV capaz de seguir *waypoints* evitando colisões com obstáculos. Além disso, será disponibilizado alguma forma de monitoramento em tempo real da posição e orientação do robô seguindo algum critério de localização, assim como possíveis mensagens de status.

1.4 Mapeamento dos stakeholders

Os *stakeholders* do projeto são constituídos por alunos do Insper selecionados pelo programa *Capstone*, o professor orientador responsável pela equipe, e um representante da Embrapa que serve como mentor e o ponto de contato entre a equipe do Insper e a empresa. A seguir, na **Error! Reference source not found.**, está descrito visualmente as funções e posições de cada *stakeholder*.

Stakeholder	Posição	Propósito	Funções
Felipe Catapano Emrich Melo	Membro da Equipe do Projeto	Aprendizado	Executor e Gestor do Projeto
Gabriel Brunoro Motta Tumang	Membro da Equipe do Projeto	Aprendizado	Executor e Gestor do Projeto
Luana de Matos Sorpreso	Membro da Equipe do Projeto	Aprendizado	Executor e Gestor do Projeto
Rafael Eli Katri	Membro da Equipe do Projeto	Aprendizado	Executor e Gestor do Projeto
Vinicius Licks	Orientador	Contribuição para o Insper	Validação do Projeto
Thiago Teixeira Santos	Mentor (Embrapa)	Contribuição para o desenvolvimento de projetos da Embrapa	Validação do Projeto

Tabela 1 - mapeamento dos stakeholders (autoria própria)

1.5 Questões Éticas e Profissionais

Para esse projeto, foram levadas em consideração questões éticas e profissionais e seus impactos ambientais econômicos e sociais. Essas questões éticas e profissionais foram pensadas com base no relatório final do semestre passado.

Um dos principais impactos sociais da utilização de UGVs na agricultura é a substituição dos trabalhadores por conta da automação de um trabalho antes realizado pelo homem, assim, muitos empregos podem ser perdidos por conta dessa nova tecnologia. Levar

isso em consideração é relevante como forma de prever os impactos dessa implementação e com isso planejar ações para reduzir o impacto negativo na vida das pessoas.

Além disso, deve ser levado em conta aspectos ambientais na produção, uso e descarte dos UGVs. Dessa forma, é necessário estudar aspectos de contaminação de solos, rios, entre outros, para então poder evitar impactos ambientais tanto na produção desses robôs, quanto no seu descarte e manutenção.

Outro aspecto importante é na utilização dos UGVs. O principal objetivo deles é coletar informações e dados dos respectivos talhões, assim, é necessário garantir a segurança e privacidade desses dados, ou seja, proteger-se contra o mau uso e o vazamento de informações coletadas pelo robô.

1.6 Normas Técnicas

Neste projeto foram utilizadas as seguintes normas técnicas, desde o planejamento até a execução.

Uma das normas utilizadas é a IEEE 1872.2-2021. Ela faz parte do instituto IEEE, uma organização responsável por normas nas diversas áreas da engenharia elétrica e eletrônica (IEEE, 2024), e mais especificamente, pertence ao RAS, o setor de robótica e automação (IEEE-RAS, 2024). Essa norma padroniza conceitos e definições relevantes para robótica autônoma, nisso se inclui também padrões de arquitetura para aplicações de navegação autônoma (IEEE-RAS, 2022). Esse tipo de norma que apresenta definições de conceitos de uma área é conhecido como norma ontológica.

Outra norma relevante é a ISO 18646-2:2024. Essa norma pertence a ISO, a organização internacional de normalização, que tem membros representantes em mais de 170 países (ISO, 2024). Essa norma em específico trata sobre a padronização na navegação de robôs, o que inclui acurácia em dados de posicionamento, capacidade de desviar de obstáculos, qualidade do mapeamento, entre outros tópicos (ISO, 2024).

1.7 Recursos

O projeto tem disponibilidade de recursos de hardware e de infraestrutura para ser executado. Foram priorizados o uso de sensores e computadores disponíveis prontamente em laboratórios. Isso se deve ao custo considerável que componentes de robótica podem alcançar e pelo tempo vago no cronograma que o envio pode causar. A seguir segue as informações de quais recursos foram utilizados.

O UGV utilizado é baseado no Open Source Rover, um projeto de código aberto da Nasa, que consiste em um robô com custo acessível e de rápida manufatura similar aos modelos que exploram a superfície de Marte. Esse UGV já foi montado previamente por um grupo do Capstone Insper do primeiro semestre de 2024, que também modificou o modelo da Nasa para ter suportes para câmeras e sensor lidar, os quais podem ser utilizados no processo de navegação autônoma. Portanto, o UGV que será utilizado já está montado e suficientemente pronto para uso. (ARAÚJO et al., 2024, p. 88).

Figura 1 - Foto do OSR, (JPL OPEN SOURCE ROVER, 2023)

Em termos de infraestrutura, o Insper oferece recursos como os laboratórios, que permitem acesso a materiais, máquinas e ambientes especializados para o trabalho. Além disso, especialistas da área, como técnicos e professores, podem ser consultados no andamento do projeto.

Para a montagem do módulo de navegação autônoma, os recursos necessários foram:

- Computador Dell OptiPlex 7000 MFF
- LiDAR Robotis LDS-02
- Câmera Intel Realsense D435i
- GPS Quectel L80

Em testes prévios, também foram utilizados:

- Kit de desenvolvedor Nvidia Jetson TX2
- Computador Intel NUC D34010WYK

1.8 Revisão do Estado da Arte

O projeto atual está dentro do contexto de navegação autônoma em ambientes agrícolas e no uso de interfaces humano-robô (HRI) para envio e recebimento de dados. Esses tópicos se relacionam com outras áreas de estudo, como localização e mapeamento, algoritmos de planejamento de rotas, entre outros.

1.8.1 Aplicações de robótica e monitoramento na agricultura de precisão

Nas últimas décadas, técnicas de SLAM foram extensivamente aprimoradas e estendidas para várias tarefas dentro da agricultura. Isso é especialmente motivado pelo rápido desenvolvimento do maquinário agrícola inteligente e proporciona a liberação de parte significativa da mão de obra em regiões onde há a prática de agricultura. Aplicações nessa área onde SLAM já foi e ainda é usado de forma significativa incluem, mas não limitadas a (HAIZHOU *et al*, 2022):

- Reconstrução tridimensional de alta fidelidade
- Detecção e mapeamento de troncos de árvores
- Previsão de rendimento
- Monitoramento da altura da cultura e colheita
- Polinização autônoma
- Capina de precisão autônoma
- Controle de pragas

1.8.2 Sensores para localização no contexto agrícola

Existe uma variedade de sensores que podem ser usados para a obtenção de dados de localização (HAIZHOU, 2022). A tarefa de localização consiste em relacionar a posição atual do veículo a um sistema de coordenadas, sendo possível utilizar uma abordagem global como o GNSS, que usa satélites para obter referências de georreferenciamento, ou locais, que usam sensores para estimar o deslocamento em relação a períodos anteriores (FASIOLO *et al*, 2023). Segue a descrição de alguns sensores que são utilizados no estado da arte do cenário agrícola.

Em relação a localização global, um tipo de sensor que tem destaque pela sua precisão é o *Real-Time-Kinematic-GNSS* ou RTK-GNSS (FASIOLO *et al*, 2023). Sensores GNSS convencionais recebem informações de distanciamento de satélites e as usam diretamente para estimar a localização, usualmente atingindo precisões na escala de metros. Já sensores RTK-GNSS utilizam dois receptores para realizar a estimativa, um deles em uma estação fixa e outro no veículo. Com isso, o receptor da estação consegue ser usado para aumentar a precisão da estimativa, que costuma estar na escala de centímetros (FASIOLO *et al*, 2023).

Entretanto, em áreas de agricultura densas, como as do caso de uso do projeto, sensores GNSS convencionais e RTK sofrem de multi-reflexão e bloqueios de sinal, que afetam negativamente o uso do sensor (AGUIAR *et al*, 2020). Outra questão relevante é a fusão de sensores GNSS com unidades de medição inercial (IMU), que usam a técnica de *dead reckoning* para estimar dados de orientação, que usualmente não estão disponíveis para um receptor GNSS (FASIOLO *et al*, 2023). Além disso, receptores atuam com uma frequência média consideravelmente baixa de 1 Hz, enquanto IMU's tem frequências médias superiores a 1 kHz, o que torna a fusão entre os dois sensores interessante para preencher lacunas entre os dados de GNSS (FALSIOLO *et al*, 2023).

Já em relação a localização local, sensores LiDAR 3D e câmeras de profundidade são comumente utilizadas. LiDAR's são sensores que usam como base o tempo de reflexão de um laser para medir distâncias de seus arredores, sendo que técnicas como ICP podem ser utilizadas para comparar duas leituras diferentes e estimar o deslocamento entre elas (FASIOLO *et al*, 2023). Já câmeras de profundidade oferecem dados de imagem e profundidade, podendo ser baseadas em técnicas *stereo* ou *time of flight.*

Para localização visual, é comum o uso de técnicas de extração de *features*, que são o conjunto pontos de interesse coletados da imagem (*keypoints*) e suas informações relevantes *(descriptors)*, que em seguida são comparados entre os frames para estimar o deslocamento (LABBÉ, 2019).

Por um lado, LiDAR's são mais robustos em relação a mudanças de iluminação (FASIOLO *et al*, 2023), mas são consideravelmente mais caros e podem ser prejudicados por condições climáticas como chuvas e neblinas (REGER, 2022).

1.8.3 Comparação de implementações de SLAM Open Source

Uma técnica relevante para a navegação autônoma é o uso de SLAM, um método de mapear os arredores do robô enquanto ele anda, sem informações prévias. O SLAM entra no contexto de estratégias de localização, e pode ser dividido em tarefas de extração de *feature*s dos dados de sensores para a construção do mapa, o que é conhecido como Front-End, e na otimização do mapa, conhecido como Back-End (KAZEROUNI *et al*, 2022). Outra questão relevante para o SLAM é a identificação de pontos que já foram visitados previamente de modo a corrigir imperfeições do mapa, o que é conhecido como *loop closure* (KAZEROUNI *et al*, 2022).

Para contextualizar, determinados algoritmos de planejamento de rotas como A* ou Djikstra dependem de mapeamento para serem utilizados. Inclusive, o uso da plataforma padrão de navegação do ROS, NAV 2, depende de técnicas de mapeamento como o SLAM ou de AMCL, uma técnica que utiliza um mapa previamente feito.

Tanto o SLAM, quanto algoritmos de planejamento de rotas são técnicas conhecidas e já possuem diversas implementações desenvolvidas por grupos acadêmicos e distribuídas em pacotes ROS. Nesse sentido, o propósito do projeto “Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment” (FILIPENKO, 2018) é realizar o benchmark de diversos pacotes conhecidos de SLAM no framework ROS, utilizando uma variedade de configurações de sensores diferentes.

Em questão de hardware, o robô utilizado pelo projeto usa uma placa Nvidia Jetson TX1, um LiDAR 2D Hokuyo, uma câmera estérea e uma câmera convencional. Foram testados pacotes de SLAM baseados em LiDAR 2D e SLAM visual.

TABLE V: ABSOLUTE TRAJECTORY ERROR FOR DIFFERENT SYSTEMS BASED ON HECTOR SLAM TRAJECTORY

System	RMSE (m)	Mean (m)	Median (m)	Std. (m)	Min (m)	Max (m)
Cartographer	0.024	0.017	0.013	0.021	0.001	0.07
LSD SLAM	0.301	0.277	0.262	0.117	0.08	0.553
ORB SLAM (mono)	0.166	0.159	0.164	0.047	0.047	0.257
DSO	0.459	0.403	0.419	0.219	0.007	0.764
ZEDfu	0.726	0.631	0.692	0.358	0.002	1.323
RTAB map	0.163	0.138	0.110	0.085	0.004	0.349
ORB SLAM (stereo)	0.190	0.151	0.102	0.115	0.004	0.414
S-PTAM (no loop cl.)	0.338	0.268	0.244	0.206	0.001	0.768
S-PTAM (loop cl.)	0.295	0.257	0.242	0.145	0.006	1.119

Tabela 2 - comparação entre técnicas de SLAM (Filipenko, 2018)

Os pacotes que tiveram os melhores resultados foram RTAB-MAP, ORB-SLAM e Cartographer. As soluções de LiDAR 2D, como Cartographer, obtiveram os melhores resultados. Já em relação ao SLAM visual, os pacotes RTAB-MAP e ORB SLAM se sobressaíram, sendo que o primeiro teve dificuldades em ambientes com paredes monocromáticas e o último teve dificuldades em calcular a escala das distâncias na imagem para produzir o mapa.

Vale ressaltar que as condições de teste não são iguais ao da proposta de projeto, dado que eles foram feitos em ambientes fechados. Isso pode causar um viés na análise de desempenho, principalmente a favor do uso de LiDAR 2D, uma vez que ambientes abertos não são favoráveis para uso de LiDAR 2D como fonte única de sensoriamento (REGER, 2022).

O diferencial das soluções de SLAM visual que usam câmeras de profundidade é o mapeamento em 3D. Além disso, pacotes como o RTAB-MAP oferecem opções de SLAM multi-sensor e projeção do mapa para o plano 2D, o que é útil para a integração com o pacote de navegação.

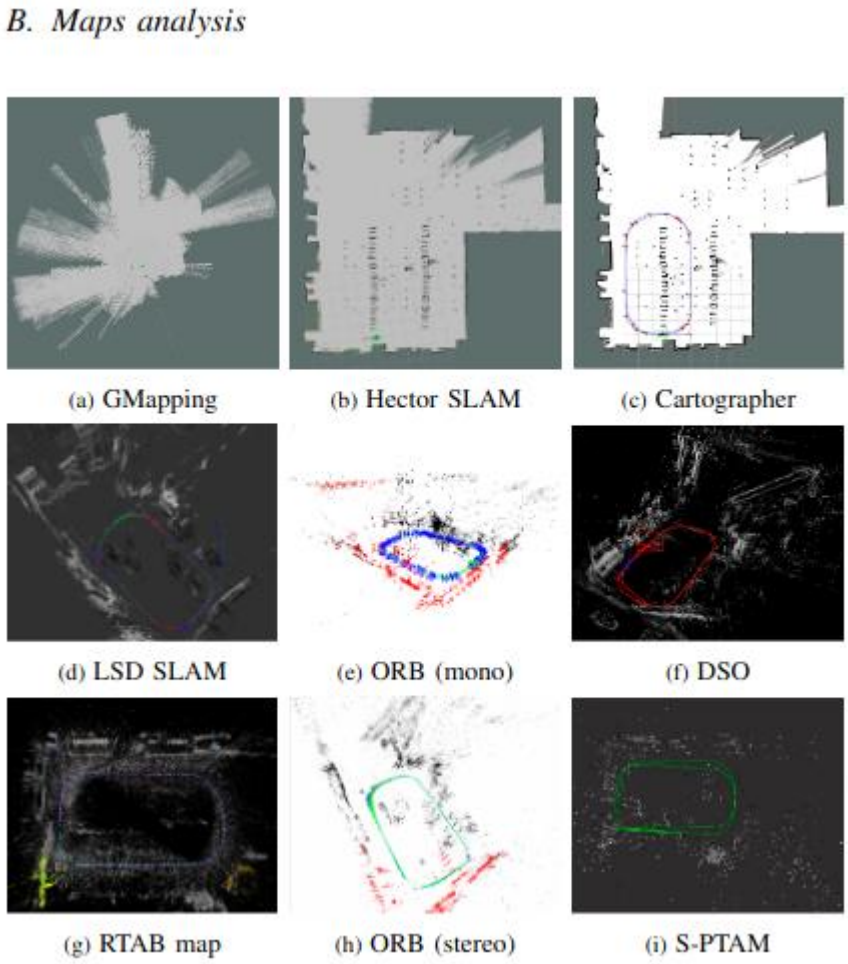

Figura 2 - visualização dos mapas criados por diferentes métodos de SLAM (Filipenko, 2018)

1.8.4 ROS

Considerando o estado da arte, é importante também ressaltar a adoção do framework ROS 2 ao invés de sua implementação antecessora. O primeiro ROS, que foi desenvolvido com foco originalmente em uso acadêmico, causou impacto na indústria devido ao crescimento de bibliotecas de código aberto, entretanto não possui recursos considerados importantes para projetos comerciais. Devido a isso, foram iniciados extensivos esforços por parte da comunidade para a migração de soluções. Desde 2023, o número de downloads de ROS 2 superou o de ROS 1, chegando a 57.88% do total desse ano, o que indica que ele está cada vez sendo mais adotado pela comunidade de robótica (SCOTT, 2024). Além disso, em relação ao ano anterior, o número de downloads e pacotes ROS 2 instalados aumentou em 18.05% e 41.44%, respectivamente (SCOTT, 2024).

Melhorias em relação ao projeto original incluem melhoria de qualidade de serviço na transmissão de mensagens, integração plena entre as diferentes implementações das API’s de

linguagens, que agora são convertidas para o padrão RCL feito em C, e a possibilidade de implementar tarefas em tempo real (ROS 2 DESIGN, 2024).

Uma vez que cada distribuição do ROS está diretamente atrelada a uma distribuição do sistema operacional Ubuntu Linux, o fim da vida da versão compatível com a última versão disponibilizada do ROS original (denominada Noetic Ninjemys) implica no encerramento de suporte oficial a partir de maio de 2025. Parte das migrações observadas, portanto, se deve ao fato de que a falta de atualizações e recursos de segurança representa um risco severo para aplicações comerciais, uma vez que se estaria usando pacotes com vulnerabilidades conhecidas.

1.8.5 Projetos semelhantes

De modo a entender como estes e outros desafios semelhantes de navegação autônoma são atendidos, foi realizada uma pesquisa de projetos semelhantes.

A Clearpath Robotics é uma empresa de robótica que atende o segmento de UGV's de agricultura. A empresa canadense apresenta robôs especializados para a movimentação em áreas externas (CLEARPATH ROBOTICS, 2024) e softwares proprietários para a navegação autônoma em plantios (CLEARPATH ROBOTICS, 2024). A Clearpath também oferece kits de sensores para adaptar robôs proprietários ou de fora da empresa para o uso de navegação autônoma. Um diferencial deles é o suporte de sistemas open-source como ROS, que permite fácil colaboração e integração com outros projetos.

Como o seu principal software de navegação autônoma para ambientes agrícolas, OutdoorNav, não é open-source, não é possível saber as técnicas de navegação utilizadas. Sabe-se que a estrutura do problema é semelhante, sendo que o caminho a ser feito é descrito também com waypoints.

Entretanto, pelos kits de sensores é possível ter noção do que é utilizado em situações práticas no mercado. A versão padrão do kit, de maior custo e desempenho, acompanha um sensor LiDAR 3D, GPS RTK de duas antenas e um IMU (CLEARPATH ROBOTICS, 2024). Já a versão para iniciantes acompanha duas câmeras de profundidade Realsense e um sensor GPS RTK (CLEARPATH ROBOTICS, 2024). Ambos os kits apresentam uma unidade de processamento separada do computador principal, sendo que esse possui versões com placas da série Nvidia Jetson ou computadores Mini ITX (CLEARPATH ROBOTICS, 2024).

Outro projeto relevante é o Rovy (SCHNABEL, 2020), feito por um entusiasta de robótica que buscava rodar SLAM e navegação autônoma em uma Raspberry PI 4. Para isso, foi necessário fazer mudanças de otimização de código em diversos pacotes de mapeamento, navegação e até em firmware de sensores. Dentre esses se encontram o pacote RTAB-MAP, librealsense e NAV 1. Percebe-se que a tarefa de otimização é complexa, a qual foge do escopo do projeto atual.

No fim, o projeto teve sucesso em rodar SLAM visual baseado em câmeras de profundidade simultaneamente com o pacote NAV 1 para o planejamento de rotas. Ele otimiza o cálculo da odometria visual usando uma câmera Realsense T265, a qual faz os cálculos internamente sem depender da Raspberry Pi. O projeto também conta com uma Realsense D435 para alimentar os dados de SLAM. O Rovy usa o RTAB-MAP, sendo capaz de gerar mapas em 3D, e usa um planejador de rotas customizado, otimizado para o seu uso (SCHNABEL, 2020).

A análise desse projeto é interessante porque demonstra as limitações do uso de uma Raspberry Pi 4 para o processamento simultâneo de SLAM, navegação e acionamento de motores, e o potencial de como contorná-las. Apesar de ser possível, a solução precisa de um trabalho extenso de otimização.

2 Metodologia

2.1 Gestão do Projeto

Para o desenvolvimento e gestão do projeto será utilizada a metodologia de PDP de Pahl & Beitz (Pahl et al, 2005). Tal método define uma sequência lógica de design e produção de produtos de engenharia. Para auxiliar a equipe durante o desenvolvimento será utilizado a plataforma Notion que integra ferramentas como Kanban, calendário online e uma linha de tempo das tarefas.

2.1.1 Metodologia de Pahl & Beitz

A metodologia de Pahl & Beitz define um fluxo de trabalho metódico e iterativo para o processo de planejamento e execução de um projeto de engenharia. A ferramenta contém 4 grandes fases: investigação, design conceitual, design de materialização e detalhamento. Cada

uma dessas fases elencadas anteriormente agrupa diversas tarefas e resultados esperados, podendo uma macro fase sobrepor a outra em certos períodos.

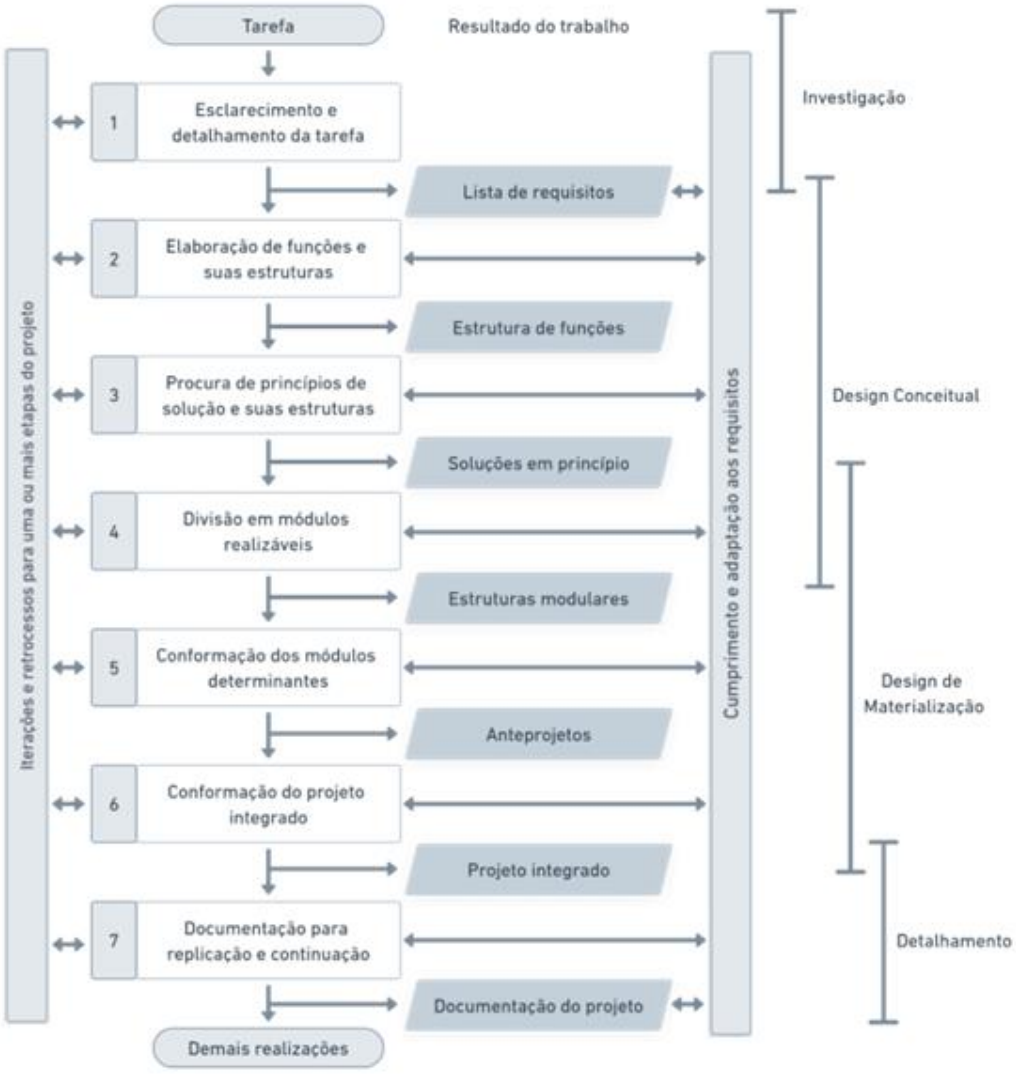

Figura 3 - Estrutura do método Pahl & Beitz adaptada (Apud Captsone grupo do peixe-robo 2024.1)

2.1.2 Ferramentas de Gestão

Para auxiliar o grupo a manter a organização e acompanhamento das tarefas, assim como marcar reuniões e guardar documentos, foi utilizada a plataforma Notion. Neste serviço é possível criar páginas personalizadas que contém múltiplas funcionalidades diferentes, desde documentação e orientações para usuários, mas também uma ferramenta completa de acompanhamento de tarefas.

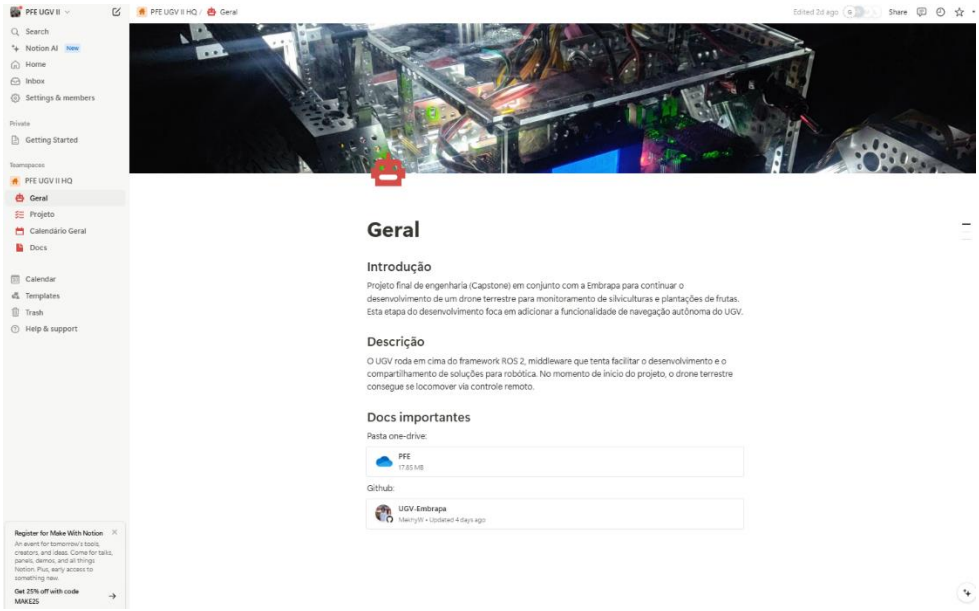

Figura 4 - *landing page* do Notion criado pela equipe (autoria própria)

Para este projeto foi criado 4 seções: Geral, Projeto, Calendário Geral e Docs. O Geral contempla uma breve explicação do projeto, assim como um meio de acesso rápido a outras ferramentas mais específicas utilizado pela equipe. Dentro de Projeto está configurada uma página completa para acompanhamento do projeto, contendo uma lista de todas as tarefas, um Kanban e uma linha de tempo (cronograma) que dá para ser visualizada de múltiplas formas utilizando filtros diferentes. Por fim, o Calendário Geral é utilizado para manter um controle sobre reuniões marcadas e eventos relacionados ao projeto, e o Docs é uma página dedicada para associar possíveis tarefas a documentos específicos delas.

2.2 Análise de Riscos

A análise de riscos é feita com base no capítulo 11 do guia PMBOK. Os riscos do projeto são analisados em duas dimensões, a probabilidade de ocorrência e a sua gravidade, ou seja, as consequências caso ele realmente ocorra. Os riscos podem ser classificados como técnicos, organizacionais, externos ou de gestão de projetos. Uma vez listados os riscos, ele recebe um índice de probabilidade de ocorrência e um indice de gravidade para o escopo do projeto, qualidade, cronograma e custo. Assim, o maior dos indices de gravidade é multiplicado pelo indice de probabilidade para determinar a prioridade do risco com relação aos demais, ou seja, aquele que tiver um maior valor resultado dessa multiplicação, é o risco com maior prioridade. Além disso, são pensadas em ações preventivas para cada um deles. Na tabela a seguir o que está marcado em vermelho representa um risco maior e em amarelo um risco moderado.

ID do risco	Tipo de Risco	Descrição do risco	Impacto	Índice de gravidade do impacto (G)				Probabilidade de ocorrencia do risco (O)	Avaliação da prioridade do risco (P=MédiaGxO)	Medida de Prevenção (resposta ao risco)	Responsável
				Escopo	Qualidade	Tempo	Custo				
1	Técnicos	Acumulo de erros nos sensores	Navegação errônea	Alto 0.4	Alto 0.4	Baixo 0.1	Muito Baixo 0.05	Média 0.5	0.20		Rafael
2		Caso o robô vá muito rápido ele pode perder a referência odometria visual	Navegação errônea	Alto 0.4	Alto 0.4	Baixo 0.1	Muito Baixo 0.05	Média 0.5	0.20	Limitar velocidade de movimento do robô	Felipe
3		Erros de fabricação das PCI's	Atraso no cronograma	Muito Baixo 0.05	Muito Baixo 0.05	Moderado 0.2	Moderado 0.2	Baixa 0.3	0.06	Utilizar métodos corretos de fabricação e consultar com técnicos	Gabriel
4		Mal contatos dos fios do robô	Probelams gerais de funcionamento	Moderado 0.2	Alto 0.4	Baixo 0.1	Muito Baixo 0.05	Baixa 0.3	0.12	Verificar conexões e prender fios de forma adequada	Gabriel
5		Perda de comunicação com centro de controle	Perda de dados e parada total da operação	Alto 0.4	Alto 0.4	Baixo 0.1	Muito Baixo 0.05	Média 0.5	0.20	Manter robô em range de operação determinado	Rafael
6		Erros de montagem mecânica	Integridade física do robô comprometida	Muito Baixo 0.05	Moderado 0.2	Baixo 0.1	Baixo 0.1	Alta 0.7	0.14	Substituir porcas por porcas autotravantes e revisar montagem	Luana

Figura 5 - Tabela de riscos técnicos (autoria própria)

ID do risco	Tipo de Risco	Descrição do risco	Impacto	Índice de gravidade do impacto (G)				Probabilidade de ocorrencia do risco (O)	Avaliação da prioridade do risco (P=MédiaGxO)	Medida de Prevenção (resposta ao risco)	Responsável
				Escopo	Qualidade	Tempo	Custo				
7	Organizacionais	Problemas de comunicação com cliente	Atraso no cronograma	Moderado 0.2	Muito Baixo 0.05	Moderado 0.2	Muito Baixo 0.05	Média 0.5	0.10	Marcar previamente reuniões recorrentes	Luana
8		Desacordo entre a equipe executora e o cliente	Dificuldade em definir escopo	Moderado 0.2	Muito Baixo 0.05	Baixo 0.1	Muito Baixo 0.05	Baixa 0.3	0.06	Comunicação frequente e clara	Luana
9		Desencontro entre membros da equipe	Falta de alinhamento entre partes do	Baixo 0.1	Moderado 0.2	Baixo 0.1	Muito Baixo 0.05	Baixa 0.3	0.06	Marcar previamente reuniões recorrentes	Gabriel
10	Gestão de projetos	Falta de atualização da ferramenta de gestão	acompanhamento do projeto pelos	Moderado 0.2	Baixo 0.1	Moderado 0.2	Muito Baixo 0.05	Média 0.5	0.10	Ter um responsável especifco para esta função	Gabriel
11		inadequada do	Atraso no projeto	Moderado 0.2	Moderado 0.2	Moderado 0.2	Muito Baixo 0.05	Média 0.5	0.10	Revisar com professores e dedicar tempo necessário	Gabriel
12	Externos	seguindo tendências	Produto obsoleto em pouco tempo	Muito Alto 0.8	Muito Baixo 0.05	Muito Baixo 0.05	Muito Baixo 0.05	Muito Baixa 0.1	0.08	Pesquisas do Estado da arte adequadas	Felipe

Figura 6 - Tabela de riscos organizacionais, de gestão de projetos e externos (autoria própria)

2.3 WBS e Cronograma

Para organização de tarefas e prazos o grupo elaborou um WBS e com isso um cronograma. O WBS tem como sua entrega final a navegação autônoma do UGV, tendo sido dividido em 6 entregáveis. O Desenvolvimento foi dividido com 2 níveis de detalhe, sendo o primeiro composto por uma divisão mais geral entre software, eletrônica e mecânica, tendo cada uma dessas divisões seus pacotes de trabalho.

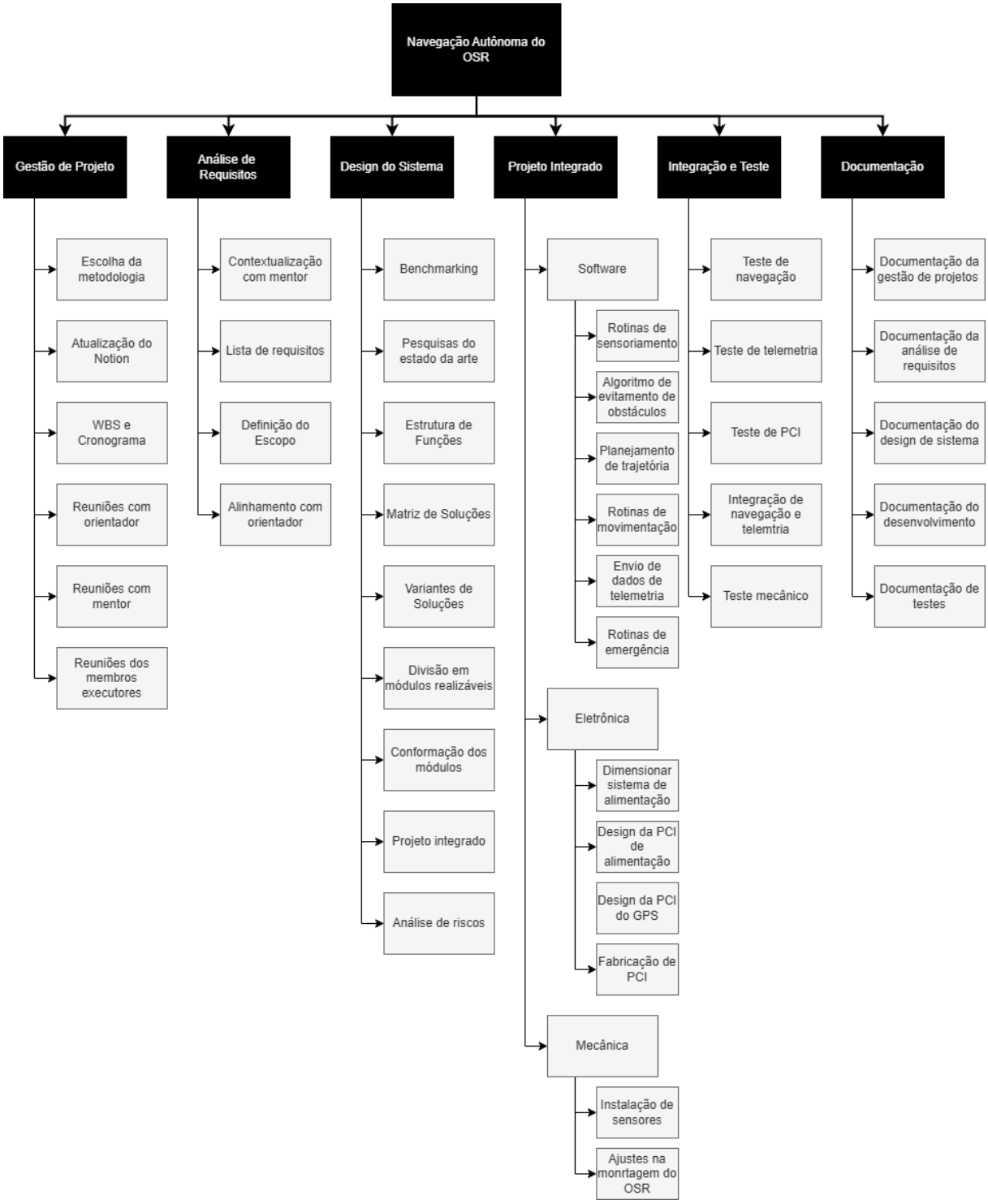

Figura 7 - Work Breakdown Structure (WBS) criado pela equipe (autoria própria)

Com todos as tarefas organizadas é possível estabelecer um tempo de realização para cada uma delas baseado no que os integrantes do grupo acreditaram ser o necessário para cada tarefa, assim podendo organizar quando cada parte do projeto deve ser iniciada e finalizada para que tudo seja entregue. Foi estabelecido que dia 02/10, o dia da banca intermediária, será o primeiro Gate do projeto, e dia 21/11 será o Gate final.

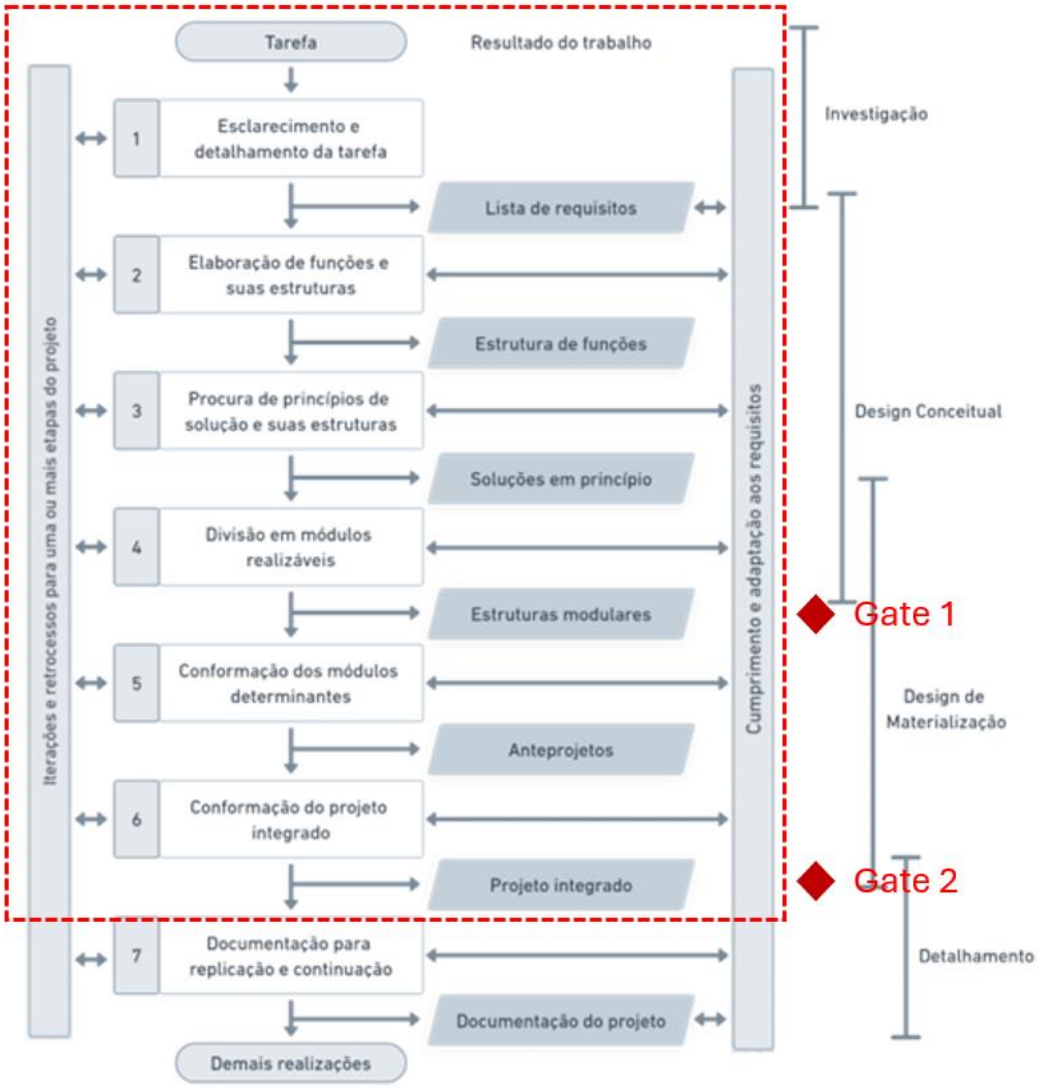

Figura 8 - definição dos gates de acordo com a metofologia Pahl & Beitz (autoria própria)

Com o WBS feito e os Milestones decididos, as tarefas foram divididas entre os membros e foi definido um cronograma a ser seguido pela equipe.

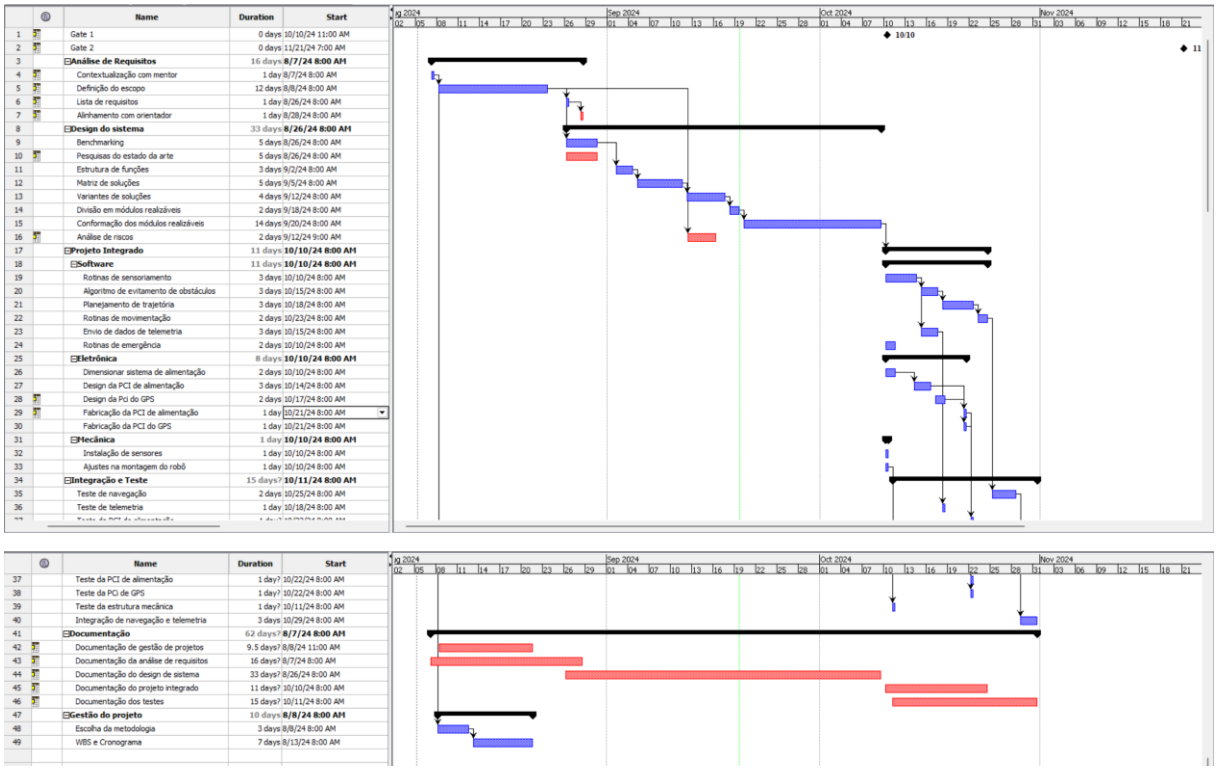

Figura 9 - cronograma criado pela equipe (autoria própria)

3 Desenvolvimento do Projeto

3.1 Lista de Requisitos e Elaboração das Funções

Seguindo a metodologia de Pahl & Beitz escolhida, inicialmente foi desenvolvida uma lista de requisitos para o projeto, com o objetivo de elencar as necessidades técnicas e exigências do cliente.

Lista de requisitos
Utilizar a plataforma do rover OSR da NASA
Utilizar ROS 2
Conseguir seguir waypoints disponibilizados pelo operador
Navegar autonomamente evitando obstáculos
Ser capaz de para quando se deparar com um obstáculo intransponível
Enviar informações de localização e pose ao operador em tempo real
Informar operador sobre o status atual do UGV

Figura 10 - Lista de requisitos (autoria própria)

Seguido da lista de requisitos é necessário determinar a função global do projeto para então dividi-lo em subfunções que vão compor a totalidade do funcionamento do robô. A função global é o objetivo principal que se tem com o projeto considerando as entradas e saídas do sistema, mais especificamente de energia, sinais e materiais, no caso, a função global é a navegação autônoma do robô, considerando que há entrada de energia elétrica de waypoints e grandezas do ambiente e saída de movimento, ruído, luzes, dados de posição e pose do robô e informações de status. Não há entrada e saída de materiais.

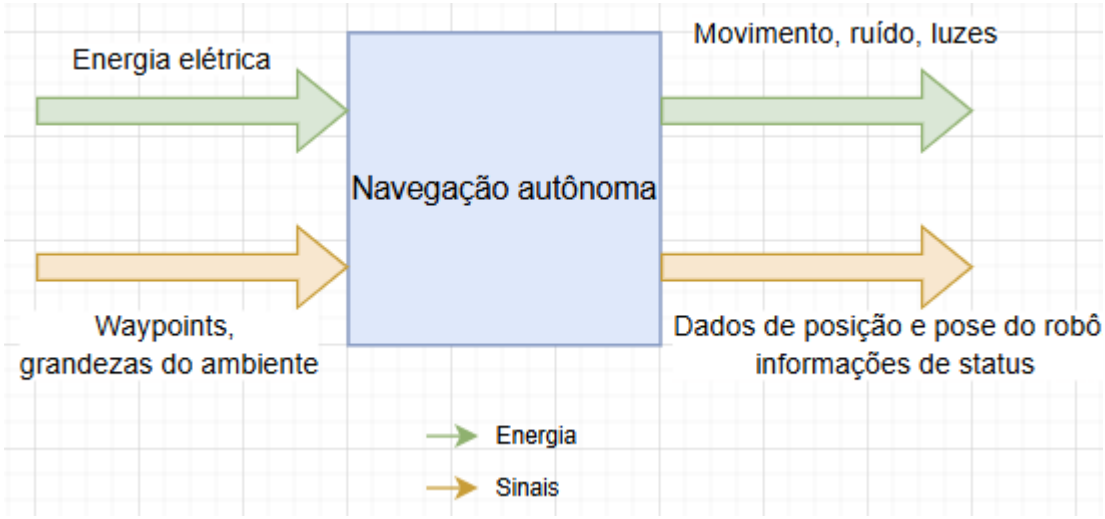

Figura 11 - Função global (autoria própria)

Após isso, a função macro do projeto que é a navegação autônoma foi dividida em subfunções para esquematizar todos os processos que são necessários para que a navegação

autônoma ocorra. Assim, o esquema foi dividido em quatro partes, a que ocorre no robô e a que ocorre em um ambiente de controle, a intermediação entre essas partes e a interface com o usuário.

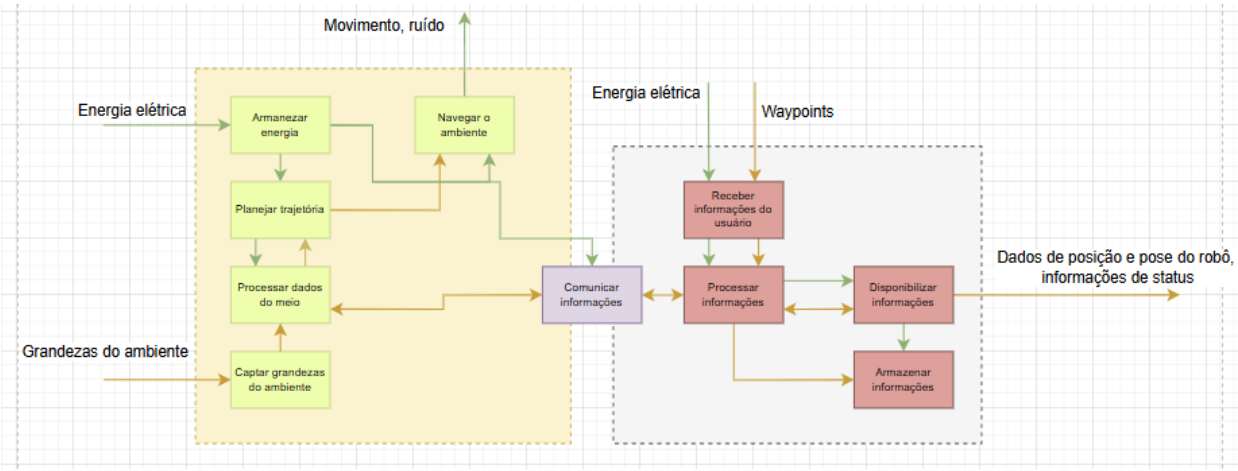

Figura 12 - Estrutura de funções (autoria própria)

Além disso, existe uma legenda para facilitar a compreensão do esquema da figura 8 acima.

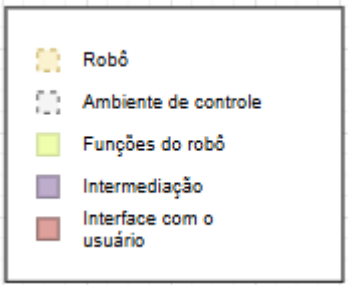

Figura 13 - Legenda da estrutura de funções (autoria própria)

3.2 Matriz de soluções

Com isso, é possível que se inicie a elaboração da matriz de soluções que inclui ideias para resolver cada uma das subfunções apresentadas, obtidas através de pesquisas e conhecimentos prévios. Assim, obtém-se variantes de solução para que então a melhor opção seja selecionada de acordo com critérios previamente elaborados.

A matriz de soluções é dividida em três, o subsistema de processamento de dados e planejamento de trajetórias, o subsistema de energia e movimentação e o subsistema de telemetria e IHM para maior organização de informações. Cada um desses subsistemas possui

subfunções relacionadas a eles e cada uma das subfunções seus blocos, ou seja, partes que compõe aquela subfunção. Assim, é possível indicar soluções para cada um dos blocos, ao invés de uma solução mais geral para cada subfunção. Essa divisão foi pensada com base no projeto do capstone de 2024.1 do peixe robô (Versolato et al., 2024, p. 48).

Subsistema de Processamento de Dados e Planejamento de Trajetória								
Subfunção	Blocos	Soluções						
		1	2	3	4	5	6	7
Captar Grandezas do ambiente	Sensores	Lidar 2D	Lidar 3D	Câmera RGB	Câmera RGB-D	Sensor Ultrassom	Radar	
Processar dados do meio	Frameworks	Baremetal	RTOS	ROS 1	ROS 2			
	Hardware de procesamento	FPGA	Jetson Orion NX	Jetson TX2	Intel NUC	Raspberry PI	Intel Optiplex	Microrcontroladores
	Arquitetura	1 hardware de processamento	1 para controle e 1 para evitamento de obstáculos					
	Localização	GPS	SLAM	Dead Reckoning	AMCL			
Planejar trajetória	Algoritmos de planejamento de trajetória	Smac Hybrid-A*	Smac Lattice Planner	NavFn	Theta Star	Smac		
	Estratégia de evitamento de obstáculos	Regulated Pure Pursuit	MPC	DWB	Vector Pursuit	Detectar limites da plantação e centralizar (Row detection)	Estratégia reativa	

Figura 14 - Matriz de soluções do subsistema de processamento de dados e planejamento de trajetória (autoria própria)

Subsistema de Energia e Movimentação							
Subfunção	Blocos	Soluções					
		1	2	3	4	5	6
Armazenar Energia	Forma de energização	Cabo	Bateria Ion-Litio	Bateria LiPO	Bateria Chumbo-Ácido	Pilhas	Paineis solares
	Monitoramento de carga	Multímetro	Contador de coulomb				
Navegar o ambiente	Métodos de movimentação terrestre	4 Rodas	6 Rodas	Esteira			
	Arquitetura possíveis	Differential drive	Holonomic drive	Ackermann Steering	Skid Steering		
	Atuadores	Motor DC	Motor Brushless	Motor de Passo			

Figura 15 - Matriz de soluções do subsistema de energia e movimentação (autoria própria)

Subsistema de Telemetria e IHM							
Subfunção	Blocos	Soluções					
		1	2	3	4	5	6
Comunicar Informações	Meio de comunicação	Cabo elétrico	Ondas eletromagnéticas	Cabo óptico	Sinais acústicos		
	Tecnologias de transmissão	Ethernet	Bluetooth	LoRa	Wi-Fi	Internet	
	Hardware de comunicação	Dongle 4G	Antena rádio				
Processar Informações	Frameworks	Baremetal	ROS 1	ROS 2			
	Hardware de processamento	Computador pessoal	Nuvem	Servidores locais			
Receber informações do usuário	Interface	Computador	Smartphone				
Disponibilizar informações	Interface	Computador	Smartphone	Display customizado			
	Framework	RVIZ	Graphana	Flask	Streamlit		
Armazenar Informações	Tecnologia de armazenamento	Local (Disco rígido/SSD)	NAS	Nuvem			
	Framework	MySQL	MongoDB	AWS S3			

Figura 16 - Matriz de soluções do subsistema de telemetria e IHM (autoria própria)

3.3 Redução da Matriz de Soluções

Para selecionar a solução ideal segue-se a metodologia explicada a seguir. Com as matrizes prontas, é realizada uma redução preliminar, onde são eliminadas soluções inconsistentes e que não seguem os requisitos selecionados. Para essa redução é levado em conta aspectos de segurança, ergonomia, produção, controle, montagem, transporte, manutenção, reciclagem, gastos e entre outros aspectos importantes, gerando um critério de avaliação das soluções. Após isso, são definidas métricas de avaliação mais específicas para cada uma das soluções fazendo com que reste apenas as melhores. No final, isso são montadas variantes de solução, que são as combinações possíveis entre as soluções restantes e cada uma das variantes passa por uma avaliação de acordo com critérios definidos pelo grupo para que então a melhor combinação de soluções seja escolhida.

A primeira redução feita foi excluindo as soluções que não vão de acordo com os requisitos do projeto, dessa forma, o que está em amarelo nas imagens seguintes representa aquilo que é necessário ser utilizado, assim, todas as outras possíveis soluções das respectivas linhas podem ser eliminadas. Após isso, foi eliminado aquilo que se mostrava inviável para o projeto, por conta de preço, tempo, manutenção, segurança, entre outros aspectos, o que está representado em vermelho nas imagens.

Subsistema de Processamento de Dados e Planejamento de Trajetória								
Subfunção	Blocos	Soluções						
		1	2	3	4	5	6	7
Captar Grandezas do ambiente	Sensores	Lidar 2D	Lidar 3D	Câmera RGB Stereo	Câmera RGB-D	Sensor Ultrassom	Radar	
Processar dados do meio	Frameworks	Baremetal	RTOS	ROS 1	ROS 2			
	Hardware de procesamento	FPGA	Jetson Orion NX	Jetson TX2	Intel NUC D34010WYK	Raspberry PI	Dell Optiplex	Microrcontroladores
	Arquitetura	1 hardware de processamento	1 para controle e 1 para evitamento de obstáculos					
	Localização	GPS	GPS RTK	SLAM	Dead Reckoning	AMCL		
Planejar trajetória	Algoritmos de planejamento de trajetória	Smac Hybrid-A*	Smac Lattice Planner	NavFn	Theta Star	Smac		
	Estratégia de evitamento de obstáculos	Regulated Pure Pursuit	MPC	DWB	Vector Pursuit	Detectar limites da plantação e centralizar (Row detection)	Estratégia reativa	

Figura 17 - Redução da matriz de soluções do subsistema de processamento de dados e planejamento de trajetória (autoria própria)

Subsistema de Energia e Movimentação							
Subfunção	Blocos	Soluções					
		1	2	3	4	5	6
Armazenar Energia	Forma de energização	Cabo	Bateria Ion-Litio	Bateria LiPO	Bateria Chumbo-Ácido	Pilhas	Paineis solares
	Monitoramento de carga	Multímetro	Contador de coulomb				
Navegar o ambiente	Métodos de movimentação terrestre	4 Rodas	6 Rodas	Esteira			
	Arquitetura possíveis	Differential drive	Holonomic drive	Ackermann Steering	Skid Steering		
	Atuadores	Motor DC	Motor Brushless	Motor de Passo			

Figura 18 - Redução da matriz de soluções do subsistema de energia e movimentação (autoria própria)

Subsistema de Telemetria e IHM							
Subfunção	Blocos	Soluções					
		1	2	3	4	5	6
Comunicar Informações	Meio de comunicação	Cabo elétrico	Ondas eletromagnéticas	Cabo óptico	Sinais acústicos		
	Tecnologias de transmissão	Ethernet	Bluetooth	LoRa	Wi-Fi	Internet	
	Hardware de comunicação	Dongle 4G	Antena rádio				
Processar Informações	Frameworks	Baremetal	ROS 1	ROS 2			
	Hardware de processamento	Computador pessoal	Nuvem	Servidores locais			
Receber informações	Interface	Computador	Smartphone				
Disponibilizar informações	Interface	Computador	Smartphone	Display customizado			
	Framework	RVIZ	Graphana	Flask	Streamlit		
Armazenar Informações	Tecnologia de armazenamento	Local (Disco rígido/SSD)	NAS	Nuvem			
	Framework	MySQL	MongoDB	AWS S3			

Figura 19 - Redução da matriz de soluções do subsistema de telemetria e IHM (autoria própria)

Após essa redução preliminar, continuou-se o processo. Para isso, foi utilizado métricas para definir quais opções não eram bons o suficiente em comparação com as demais. Os critérios definidos pelo grupo estão representados na figura a seguir:

A	Foram dadas as condições de compatibilidade
B	As exigências de lista de requisitos foram satisfeitas
C	Realizáveis em princípio
D	Custo aceitavel
E	Observada tecnologia de segurança direta
F	Solução favorecida no campo específico

Figura 20 - Métricas para redução

Os resultados desta análise foram os seguintes, considerando que está marcado em verde as soluções que foram consideradas as melhores:

			A	B	C	D	E	F	Observações
Captar Grandezas do ambiente	Sensores	1	+	+	+	+	+	+	Utilizável, mas tem restrições
		2	+	+	+	+	+	-	Desfavorecida no campo
		3	+	+	+	+	+	+	
		4	-	+	+	+	+	-	incompatível e desafovercido no campo
Processar dados do meio	Hardware de processamento	1	+	+	-	+	+	-	Solução de alta complexidade e pouco customizável
		2	-	+	-	+	+	+	Versão do Ubuntu não é compatível com ROS 2
		3	-	+	-	+	+	-	Versão do Ubuntu não é compatível com ROS 2
		4	+	+	+	+	+	+	Porte pequeno e tem compatibilidade com ROS 2
		5	+	+	+	+	+	+	Hardware potente e de porte pequeno
		6	-	+	+	+	+	-	Microcontroladores não são favorecidos no âmbito de navegação autônoma utilizando ROS 2
	Arquitetura	1	+	+	+	+	+	+	O hardware pode não ser capaz de entregar o desempenho computacional necessário
		2	+	+	+	+	+	+	Separa a parte de controle de movimentação e de navegação autônoma, podendo garantir que o hardware consiga desempenhar de forma adequada
	Localização	1	+	+	-	+	+	-	Não tem a precisão necessária para localização
		2	+	+	+	+	+	+	Mais utilizado no campo
		3	+	+	+	+	+	-	Técnica antiquada e pouco precisa
		4	-	+	+	+	+	+	Não tem mapa pronto
Planejar Trajetória	Algoritmos de planejamento de trajetória	1	+	+	+	+	+	+	Oferece suporte para Ackermann steering
		2	+	+	+	+	+	+	Oferece suporte para Ackermann steering e differential drive
		3	+	+	+	+	+	-	UGV consegue simular differential drive mas não é circular. Poderia ser aproximado para um circulo.
		4	+	+	+	+	+	-	UGV consegue simular differential drive mas não é circular. Poderia ser aproximado para um circulo.
		5	+	+	+	+	+	-	UGV consegue simular differential drive mas não é circular. Poderia ser aproximado para um circulo.
	Estratégia de evitamento de obstáculos	1	+	+	+	+	+	+	Cálculos simples e foco em Path following
		2	+	+	+	+	+	-	Usa uma função de score para avaliar trajetórias e controlar a velocidade segundo um objetivo customizado. O modelo é alimentado com ruído e usa o resultado da função para aprimorar os pesoas e convergir em um resultado. Requer um considerável poder computacional.
		3	+	+	+	+	+	+	Foco em ambientes dinâmicos, requer mais poder computacional que o RPP
		4	+	+	+	+	+	+	Similar ao RPP, mas com melhorias para casos de contorno. Complexidade menor que DWA e maior que RPP
		5	+	+	+	+	+		
		6	+	+	+	+	-		

Figura 21 - Redução da matriz de soluções (autoria própria)

			A	B	C	D	E	F	Observações
Comunicar informações	Tecnologia de transmissão	1	+	+	-	+	+	-	Baixo alcance
		2	+	+	+	+	+	+	Solução de baixo custo e fácil implementação
		3	+	+	+	+	+	-	Baixo alcance
		4	+	+	+	+	+	+	Solução de ponta (IoT)
	Hardware de comunicação	1	+	+	+	+	+	+	Solução de ponta (IoT)
		2	+	+	+	+	+	+	Solução de baixo custo e fácil implementação
Processar informações	Frameworks	1	+	+	-	+	+	-	Alta complexidade e dedicação de tempo
		2	+	+	+	+	+	+	
		3	+	+	+	+	+	+	Versão mais recente do ROS. É o mesmo framework que estará sendo utilizado embarcado no robô.
	Hardware de processamento	1							
		2							
Receber informações do usuário	Interface	1	+	+	+	+	+	+	
		2	+	+	-	+	+	+	Demandaria desenvolvimento de plataforma mobile customizada, o que seria altamente complexo
Disponibilizar informações	Interface	1	+	+	+	+	+	+	
		2	+	+	-	+	+	+	
		3	+	+	-	+	+	-	
	Frameworks	1	+	+	+	+	+	+	Integrado com ROS2
		2	+	+	-	+	+	+	Adiciona complexidade fora fora do escopo do projeto
		3	+	+	-	+	+	+	Adiciona complexidade fora fora do escopo do projeto
		4	+	+	-	+	+	+	Adiciona complexidade fora fora do escopo do projeto
Armazenar informações	Tecnologia de armazenamento	1	+	+	+	+	+	+	Simples e prático
		2	+	+	+	+	+	+	Solução de ponta (IoT)
	Frameworks	1	+	+	+	+	+	+	Não há necessidade de uma database relacional, porém é funcional
		2	+	+	+	+	+	+	Programa de database não relacional simples e eficaz
		3	+	+	+	+	+	+	Solução de ponta (IoT)

Figura 22 - Redução da matriz de soluções (autoria própria)

Com isso, é possível formar as variantes de solução:

	Navegação autônoma									Telemetria			
	Captar Grandezas do ambiente	Processar dados do meio			Planejamento de trajetória		Comunicar informações		Processar informações		Receber informações do usuário	Disponibilizar informações		Armazenar Informações	
	Sensores	Hardware de processamento	Arquitetura	Localização	Algoritmos de planejamento de	Estratégia de evitamento de	Tecnologia de transmissão	Hardware de comunicação	Frameworks	Hardware de processamento	Interface	Interface	Frameworks	Tecnologia de armazenamento	Frameworks
	Lidar 2D	Raspberry Pi	1 HW de processamento	SLAM	Smac Hybrid-A*	Regulated Pure Pursuit	Lora	Dongle 4G	ROS 1	Computador pessoal	Computador	Computador	RVIZ	Local	MongoDB
	Câmera RGB-D	Dell Optiplex	2 HW de processamento		Smac Lattice Planner	DWB	Internet	Antena Rádio	ROS 2	Nuvem				Nuvem	AWS S3
						Vector Pursuit									

Variante	Captar Grandezas do ambiente	Processar dados do meio			Planejamento de trajetória		Comunicar informações		Processar informações		informações do	Disponibilizar informações		Armazenar Informações	
	Sensores	Hardware de processamento	Arquitetura	Localização	Algoritmos de planejamento de trajetória	Estratégia de evitamento de obstáculos	Tecnologia de transmissão	Hardware de comunicação	Frameworks	Hardware de processamento	Interface	Interface	Frameworks	Tecnologia de armazenamento	Frameworks
1	Lidar 2D	Raspberry Pi	1 HW de processamento	SLAM	Smac Hybrid-A*	Regulated Pure Pursuit	Lora	Antena Rádio	ROS 1	Computador pessoal	Computador	Computador	RVIZ	Local	MongoDB
2	Câmera RGB-D	Dell Optimax	2 HW de processamento	SLAM	Smac Hybrid-A*	Vector Pursuit	Lora	Antena Rádio	ROS 2	Computador pessoal	Computador	Computador	RVIZ	Local	MongoDB
3	Câmera RGB-D	Dell Optimax	2 HW de processamento	SLAM	Smac Hybrid-A*	DWB	Internet	Dongle 4G	ROS 2	Nuvem	Computador	Computador	RVIZ	Nuvem	AWS S3

Figura 23 - Variantes de solução obtidas (autoria própria)

A primeira variante representa algo que o grupo considera mais simples, a terceira representa a variante ideal e a segunda um intermediário entre os dois. Assim, através de uma análise, a variante selecionada foi a segunda.

3.4 Instalação do ambiente de desenvolvimento e dependências de Software

Antes de instalar as dependências de Software no computador embarcado do robô, foi realizado uma pesquisa sobre quais pacotes ROS são necessários e como configurá-los.

Figura 24 - Versão do ROS 2 utilizada

A versão do ROS utilizada é a ROS 2 Humble Hawksbill, que oferece suporte para os seguintes sistemas operacionais: Windows 10, Ubuntu 22.04 Jammy e Red Hat Enterprise 8. A versão do módulo de navegação foi mantida em relação a iteração anterior para permitir a comunicação com a Raspberry Pi do módulo, dado que não existe suporte oficial para a transmissão de dados entre distribuições diferentes do ROS (ROS CROSS-DISTRIBUTION COMMUNICATION, 2022).

A escolha se deu principalmente pela disponibilidade de binários prontos para os pacotes ROS, uma vez que existe garantia que o ambiente de desenvolvimento será replicável

entre diferentes computadores usando o mesmo sistema operacional. Segundo o padrão REP 143 (REP 143, 2014), todos os pacotes ROS oficiais devem estar disponíveis dentro da plataforma Rosdistro. Até o momento, o Windows nunca recebeu suporte na plataforma Rosdistro. Até 2021, com o lançamento da distribuição ROS 2 Galactic Geochelone, a plataforma apenas oferecia suporte para sistemas operacionais Ubuntu, sendo que até o momento atual nem todos os pacotes oficiais foram portados para Red Hat. Portanto, entre os três, foi escolhido o Ubuntu 22.04.

Os binários da plataforma Rosdistro são instalados usando o gerenciador de pacotes APT. Em seguida, eles são configurados pelo gerenciador de pacotes Colcon para serem acessados pelo ROS. Para a instalação dos binários foi realizado o comando *sudo apt install ros-humble-{nome_do_pacote}*. No fim do processo, deve-se executar o comando *source /opt/ros/humble/setup.bash* para configurar o ambiente ROS.

Os seguintes comandos foram efetuados para instalar pacotes a partir de binários da plataforma Rosdistro:

- *sudo apt install ros-humble-desktop*
- *Sudo apt install ros-humble-librealsense2**
- *sudo apt install ros-humble-realsense2-**
- *sudo apt install ros-humble-rtabmap-ros*
- *sudo apt install ros-humble-imu-tools*
- *sudo apt install ros-humble-control-msgs*

Em seguida foi criado um diretório de Workspace Colcon, de modo a gerenciar pacotes ROS que não possuem binários oficiais Rosdistro. Para isso, basta criar um diretório com o comando *mkdir-p ~/osr_ws/src,* usar a ferramenta Git para baixar os repositórios de interesse dentro do diretório *src*, e rodar o comando *colcon build –symlink-install* no diretório *osr_ws*. Por fim, deve-se rodar o comando *source ~/osr_ws/install/setup.bash* para atualizar as mudanças no ambiente ROS.

Os seguintes repositórios foram baixados na pasta *src* para usar esse método:

- O repositório *Robotis/ld08_driver (*WILL SON; HYE-JONG KIM, 2022*)* para o funcionamento do LiDAR

- O repositório *UGV-Embrapa* (Autoria própria) com os pacotes do projeto anterior e do atual

3.5 Configuração de sensores

Antes de serem usados, os sensores precisam estar configurados e calibrados corretamente. Segue as instruções para o preparo de cada sensor.

Figura 25 - Intel Realsense D435i (fonte: ntelrealsense, 2019)

Para a configuração da Realsense D435i é necessário calibrar o seu IMU, que não vem com calibração de fábrica (GETTING IMU DATA FROM D435I AND T265, 2021). Um *script* Python de calibração está disponível no repositório do kit de desenvolvimento de software da Realsense, chamado *Librealsense*. O arquivo rs-imu-calibration.py está contido no diretório tools e depende da biblioteca *pyrealsense2*, que está disponível índice de pacotes Pypi e pode ser instalado pelo comando: *pip install pyrealsense2*.

O script espera que o usuário posicione a câmera em 6 orientações diferentes, mantendo o dispositivo parado até atingir a estabilização necessária. As orientações necessárias estão descritas em um guia da Intel (INTEL REALSENSE DEPTH D435I IMU CALIBRATION, 2019). Por fim é atribuída uma nota a qualidade da calibração, sendo considerada uma nota boa valores próximos de 98. A nota atingida foi de 98.05.

Figura 26 - LiDAR LDS-02 (fonte: ROBOTIS e-Manual, 2024)

Em relação ao LiDAR LDS-02, é necessário configurar as regras de *udev* para permitir seu funcionamento. As regras de *udev* servem para executar scripts em resposta a eventos de dispositivos, como conectar ou desconectar um cabo de uma entrada USB (AN INTRODUCTION TO UDEV: THE LINUX SUBSYSTEM FOR MANAGING DEVICE EVENTS, 2018). O arquivo de configuração *99-turtlebot3-cdc.rules* se encontra no repositório *Robotis/turtlebot3*, dentro do pacote *turtlebot3_bringup*.

Para atualizar as regras de *udev* é preciso copiar o arquivo para o diretório */etc/udev/rules.d/* e executar os seguintes comandos: *sudo udevadm control --reload-rules* e *sudo udevadm trigger.*

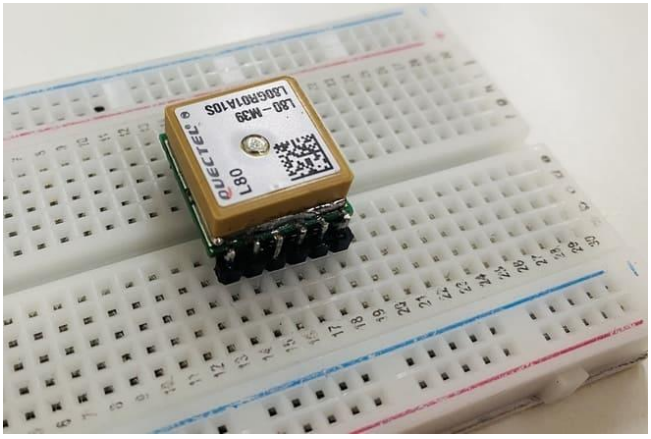

Figura 27 - GPS Quectel L80 M39 (fonte: how2electronics, 2023)

Para que seja possível ter um monitoramento da posição do UGV durante sua operação, a equipe adquiriu um dispositivo GPS chamado *GPS3 Click* que conta com um módulo Quectel L-80 M39. Este módulo tem precisão de posicionamento menor que 2,5 metros e tem baixo consumo de energia. Após testes com o dispositivo, se viu que sem uma antena externa, o módulo tem dificuldade de se conectar a satélites, sendo quase inutilizável em ambientes fechados.

3.6 Criação do pacote *osr_autonomous*

Com o ambiente ROS e os sensores preparados, o próximo passo necessário é rodar os *nodes* ou nós, os arquivos executáveis de um pacote ROS, com os parâmetros desejados para inicializar a solução do projeto. Um nó pode ser executado a partir do terminal com o comando *ros2 run nome_do_pacote nome_do_executável*. Entretanto, para um caso de uso complexo como o do projeto atual, que exige um uso de diversos nós e uma variedade de parâmetros

customizáveis, é uma boa prática criar um pacote próprio e estruturar a aplicação com arquivos *launch*.

Arquivos de *launch* realizam a orquestração de uma aplicação, executando todos os nós e seus respectivos parâmetros com um só comando. No ROS 2, o formato dos arquivos de launch mudou de XML para Python, possibilitando implementações de lógica que a linguagem de markup não permitia. Com isso em mente, foi criado um pacote chamado *osr_autonomous* para manter os arquivos de *launch* e eventuais arquivos YAML para a configuração de certos pacotes, como o NAV 2.

Até o momento, foram elaboradas duas versões de launch diferentes. Uma delas com fusão de sensores Visual-inercial e outra apenas com a câmera RGB-D. Até o fim do projeto, planeja-se integrar também o LiDAR 2D na solução. O foco para o relatório intermediário é a elaboração do SLAM, a publicação de dados de *pose* (posicionamento e orientação do veículo) e a capacidade de interagir com o NAV 2 para enviar Waypoints. O aperfeiçoamento dos algoritmos de motion planning do NAV 2 será feito posteriormente.

A seguir irá ser mencionado o papel de cada pacote ROS integrado nos arquivos *launch* e para isso irá ser descrito o *launch* com fusão de sensores, dado que todos os pacotes do outro *launch* estão contidos nele.

Os primeiros nós a serem incluídos no *launch* foram os relativos aos sensores. O pacote realsense2_camera foi utilizado com parâmetros para a publicação de imagens RGB, dados de profundidade, giroscópio e acelerômetro. Também foi configurada a opção que permite a sincronia dos dados de giroscópio e acelerômetro, que são publicados a uma taxa diferente, 200 Hz e 62.5 Hz respectivamente. Em seguida, o pacote *imu_filter_madgwick* recebe os dados da Realsense e une as informações de velocidade angular e aceleração para estimar a orientação da câmera com base em quaterniões.

Em seguida, é necessário configurar as transformações do sistema de coordenadas dos sensores para o *base_link*, o ponto de referência do UGV que fica em seu centro. Para isso é utilizado o pacote *tf 2*, que é nativo do ROS e é capaz de publicar transformações estáticas entre dois sistemas de coordenadas. Desse modo, informações como a odometria visual da câmera podem ser projetadas para o ponto de referência do veículo.

Também foi utilizado o pacote *robot_state_publisher* para publicar as transformações entre as juntas do robô e o *base_link*. Esse tipo de transformação não é estático, uma vez que a

orientação das juntas pode mudar dependendo do estado do robô. Os dados de posicionamento das juntas já são publicados pela Raspberry Pi desde a iteração anterior do projeto.

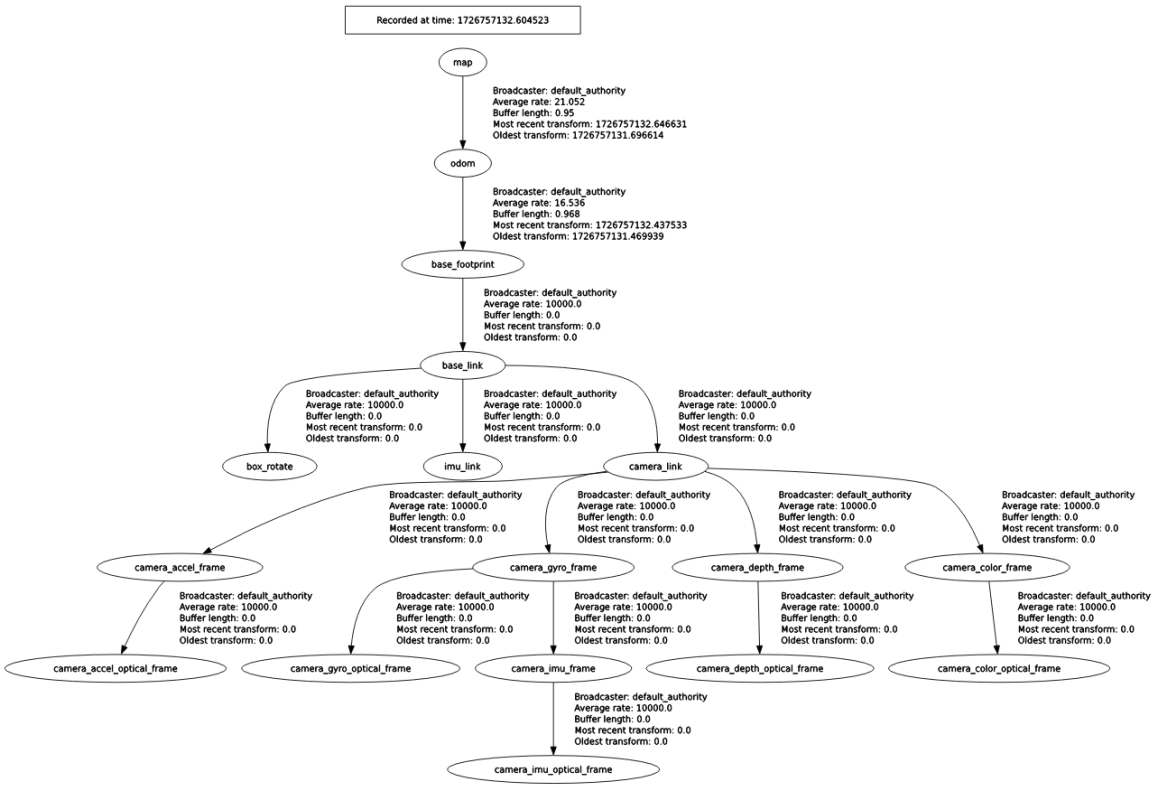

Figura 28 – árvore de transformações

Além disso, foi utilizado a *framework* NAV 2 para as tarefas de planejamento de rotas. Foram utilizados os plugins de planejamento global, planejamento local, árvore de comportamentos, suavização de rotas e monitoramento de colisões.

Vale ressaltar que o NAV 2 utiliza um sistema de configuração de parâmetros baseado em arquivos YAML. O *framework* espera dois arquivos, um com parâmetros envolvendo planejadores de rota e informações relacionadas aos plugins de navegação e o outro é utilizado para definir dados do mapa utilizado, como ponto inicial e resolução.

Por fim, o RTAB-MAP é utilizado para realizar os processos de SLAM e odometria visual. Os testes de parâmetros serão detalhados a seguir.

3.7 Testes de parâmetros com RTAB-MAP

Para a realização de testes iniciais com os parâmetros foi utilizado um computador com uma *AMD Ryzen 7 4800H*, que tem oito núcleos e clock base de 2.9 GHz, e 16 GB de RAM DDR4 de frequência 3200 MT/s. O intuito desses testes é encontrar os melhores parâmetros

para uma condição com um poder computacional considerável, para ter uma base de comparação na escolha do computador a ser instalado no veículo.

O primeiro teste foi feito com as condições padrão do RTAB-MAP para SLAM e odometria visual. Essas condições são: mapeamento em 6 graus de liberdade (3D), coleta de *features* visuais com *keypoints* no formato GFTT e *descriptors* no formato ORB para uso em etapas de odometria e geração de mapas locais, taxa de atualização do mapa global de 1Hz e odometria visual no modo *Frame to Map* (F2M), que compara os dados do frame atual com *features* de múltiplos frames anteriores para estimar a odometria (LABBÉ, 2019).

Para estimar a qualidade dos testes foi utilizado o comando *ros2 topic hz /odom* para monitorar a taxa de publicação da odometria visual e foi utilizado o RViz para verificar se mapeamento está condizente com a realidade.

Sob essas condições, o computador conseguiu processar o nó de odometria visual com uma média de 10 Hz. Em relação ao desempenho, observou-se que o sistema perdia a referência de odometria visual ocasionalmente, principalmente quando se aproxima de ambientes com pouca variedade de cor ou em casos de movimento rápido da câmera. O primeiro caso se deve a extração de features, o qual é prejudicado em ambientes com pouca diferenciação. O segundo caso pode ter sido causado por baixas taxas de publicação de odometria ou de atualização do mapa. Entretanto, o funcionamento pode ser considerado relativamente satisfatório com o uso do modo F2M, que é capaz de realizar a recuperação da odometria.

Figura 29 - Resultados de SLAM com RTAB. A esquerda é está o visualizador rtabmap_viz, com uma representação das features e da odometria. A direita o RViz está configurado para mostrar o mapa em Point Cloud e a projeção para Occupancy Grid

Foram também testadas opções menos complexas para o cálculo da odometria visual, com o fim de aumentar a taxa de publicação. A escolha dos parâmetros a serem testados foram

baseadas em guias oficiais da plataforma no site ROS Org (Mathieu Labbe, 2023) e na página do GitHub do RTAB-MAP.

Os testes utilizaram o modo *Frame To Frame* (F2F), que compara o frame atual apenas com o anterior. Outros parâmetros configurados foram o número máximo de *features* e o uso de técnicas de *optical flow* para analisar o movimento do *frame* como todo sem a necessidade de usar *descriptors* e comparar as *features* em si. Essas mudanças em específico conseguiram simplificar o processo de calcular a odometria visual, gerando uma taxa média de 28 Hz. Entretanto, a odometria visual é perdida com maior facilidade devido ao sistema F2F e a coleta de *features* simplificada.

Uma solução robusta deveria ser um caso intermediário, capaz de oferecer uma taxa adequada de odometria, ser capaz de retomar sua referência quando perdida e conseguir coletar um conjunto de *features* suficiente para calcular a transformação entre dois *frames* e estimar a *pose* atual.

Para isso foram escolhidos os seguintes parâmetros: mapeamento em 3 graus de liberdade (2D), projeção de mapas locais para 2D, taxa de atualização do mapa em 10 Hz, *features* que usam o método ORB de *keypoints* e *descriptors*, coleta de no máximo 1000 *features* por quadro e uso da técnica F2M para comparação do quadro atual com múltiplos quadros anteriores. Esses parâmetros geraram um resultado satisfatório e com uma taxa de publicação de odometria média de 16 Hz e um valores mínimos e máximos observados de 14 e 25 Hz, respectivamente.

É reconhecido pelos autores deste texto que não foram utilizadas métricas formais para os resultados apresentados dos testes realizados, uma vez que o foco para a versão intermediária deste relatório foi a validação das técnicas e pacotes utilizados e não a otimização precisa de parâmetros. Para a edição final, serão adotadas métricas mais formais no que diz respeito a qualidade da solução de SLAM e dos mapas gerados.

Figura 30 – Resultados do RTAB-MAP

Após esse processo foi realizado uma comparação entre os arquivos *launch* padrão e o com uso de IMU. A principal diferença observada é que a versão com IMU aparenta perder a referência da odometria visual com menos frequência ao realizar movimentos bruscos. Seria necessário formalizar uma métrica para afirmar isso categoricamente.

3.8 Seleção de computadores para o módulo de navegação autônoma

Um dos objetivos de usar um módulo separado para navegação é garantir que os recursos computacionais sejam suficientes para executar as tarefas. A placa responsável pelos processos de alto nível do UGV no módulo base é a Raspberry Pi 4B. Esse modelo possui dois núcleos, roda a 1.5 GHz em frequência de *clock* base e possui 8 GB de memória RAM (Rasberry Pi 4 Model B, 2024), o que pode ser considerado limitado para tarefas de navegação.

Pode se observar uma tendência de uso de computadores com maior poder computacional nesse meio. Considerando aplicações de mercado, como os UGV's Husky e Warthog da Clearpath Robotics (CLEARPATH ROBOTICS, 2024) ou o RB-VOGUI da Robotnik (ROBOTNIK, 2024), que usam processadores dedicados da Intel ou placas Nvidia Jetson. Além disso, a atuação dos motores do veículo é uma tarefa crítica que não deve ser comprometida pelo processamento de tarefas de alto nível, então a separação em módulos traz uma segurança nesse sentido.

Foi realizado uma pesquisa de computadores disponíveis no Insper para compor o módulo de navegação autônoma. Inicialmente foram encontradas as seguintes opções: Nvidia

Jetson TX2 e Intel NUC D34010WYK. Para avaliar a usabilidade de cada uma foram realizados testes para verificar sua performance.

3.8.1 Nvidia Jetson TX2

A linha Nvidia Jetson é uma série de placas de computação embarcada equipadas com GPU’s dedicadas. As placas usam CPU’s de arquitetura ARM com foco em desempenho energético. Essa linha é voltada para a área de robótica e computação de borda, possibilitando aplicações de aprendizado de máquina e computação visual em dispositivos embarcados (NVIDIA, 2024).

O Kit de desenvolvimento Jetson TX2 é uma placa que já passou do período de fim de vida, que acabou em 2020 (NVIDIA DEVELOPER, 2020). A versão disponível mais recente do Linux for Tegra (L4T), o sistema operacional proprietário baseado em Ubuntu com drivers Nvidia embutidos, não é baseado em Ubuntu 22.04, sendo limitado ao funcionamento do Ubuntu 18.04. Apesar dessa versão de sistema operacional não ter suporte para o ROS 2, foi encontrado um repositório mantido por um funcionário da Nvidia com descrição de contêiners Docker especializados para o uso de ROS na linha Jetson (FRANKLIN, 2024).

Figura 31 - Jetson TX2 developer kit (fonte: siliconhighwaydirect)

Dentre eles, existe um contêiner de ROS 2 Humble para o L4T 32.7.1 que pode ser utilizado. A maior problemática desse uso seria que a plataforma Rosdistro não oferece binários de pacote ROS para sistemas L4T, portanto todas as dependências devem ser compiladas do

código fonte. Entretanto, o uso de conteinerização do Docker ofereceria a replicabilidade desejada e facilidade de *deploy*.

A placa foi recebida com L4T 28, uma versão baseada em Ubuntu 16.04. Para o uso do contêiner foi necessário a instalação da versão mais recente do L4T com o uso da aplicação *Nvidia SDK Manager*. A aplicação apresentou erros na instalação da imagem do sistema operacional, não configurando interfaces de rede e serviços necessários para a continuação do instalador, como o SSH. Esse tipo de erro é reconhecido pela Nvidia na instalação da versão desejada e a empresa não ofereceu uma solução definitiva para isso, apenas soluções de contorno que podem gerar instabilidades (NVIDIA DEVELOPER, 2019).

Manualmente criando o serviço SSH e usando a interface de rede local sem fio (WLAN) para estabelecer a comunicação com computador executando o *Nvidia SDK Manager* foi possível prosseguir com a instalação. Entretanto, foram encontradas ainda mais instabilidades de rede que impossibilitavam o estabelecimento de conexão Ethernet, algo crucial para a comunicação com o módulo base. Novamente, a Nvidia reconhece o erro e oferece apenas uma solução de contorno, que consiste no uso de IP’s estáticos em uma configuração de rede específica (NVIDIA DEVELOPER, 2019). Tendo em vista as instabilidades das versões mais recentes do ambiente Jetson para o modelo TX2, somado ao fato do ambiente já não ter suporte oficial ao ROS 2, foram procuradas outras opções de computadores.

3.8.2 Intel NUC D34010WYK

A linha Intel NUC é especializada em minicomputadores, pensados para uso em escritórios ou para computação de borda, dependendo do modelo. Os computadores possuem processadores com arquitetura x86 e possuem desde versões com menor desempenho, como as da linha Intel Celeron, até versões de alto desempenho da linha Intel Core I9. A Intel desenvolveu os produtos NUC até 2023, no ano em que a empresa ofereceu os direitos de licenciamento para a ASUS.

A Intel NUC disponível é de um modelo produzido em 2013, o ano em que a linha NUC foi lançada. O modelo não recebe mais suporte da Intel ou ASUS, atingindo o seu fim de vida em 2018. Essa NUC tem um processador Intel Core i3-4010U que possui dois núcleos e clock básico de 1.7 GHz.

Figura 32 – Intel NUC 2013

Foram realizados dois testes com parâmetros diferentes do RTAB-MAP, ambos com desempenho consideravelmente baixo. O primeiro, com as condições padrões, obteve uma taxa de publicação de odometria média de 0.8 Hz. O segundo, que usa as condições mais simples do modo F2F e *optical flow* para o cálculo da odometria, atingiu uma taxa de publicação média de 11 Hz.

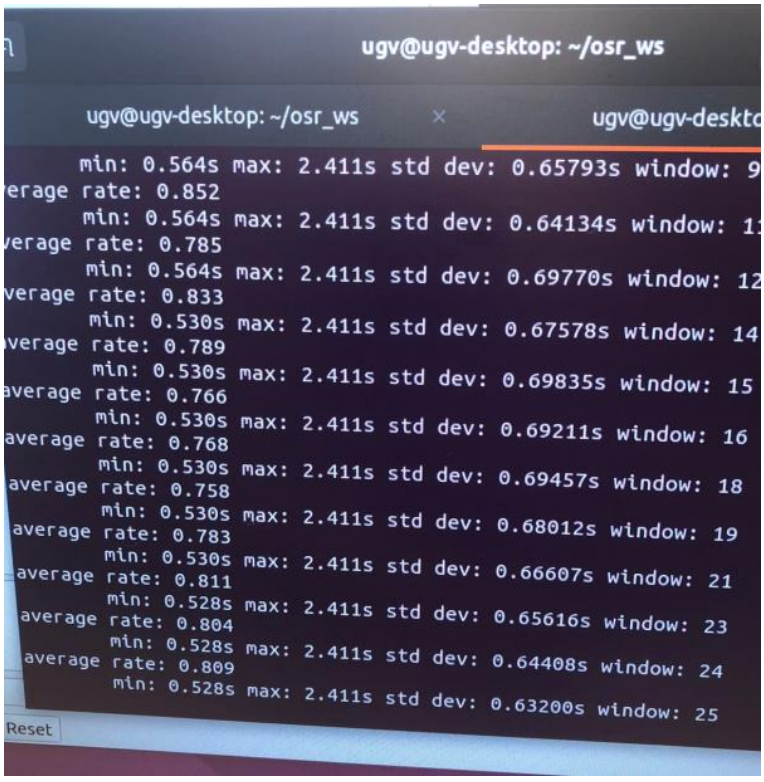

Figura 33 – Taxas de publicação de odometria visual, capturadas com o comando ros2 topic hz

O desempenho dos dois testes foi consideravelmente pior em relação a publicação da odometria, que estava a 10 Hz e 28 Hz no notebook, respectivamente. Em relação à qualidade do mapeamento, notou-se que a referência da odometria era perdida com uma frequência consideravelmente maior que no notebook, principalmente no segundo teste. Isso é grave dado que os processos de mapeamento e navegação autônoma dependem diretamente da disponibilidade da odometria.

Não foi realizado um teste com os parâmetros finais escolhidos, uma vez que eles foram decididos em um tempo posterior. Entretanto, a partir dos dados coletados nos dois testes é possível perceber que o poder computacional da Intel NUC é limitante para processar as tarefas de SLAM e odometria visual.

3.8.3 Dell OptiPlex 7000 MFF

A série OptiPlex de computadores Dell é voltada para uso empresarial, apresentando uma variedade de modelos de chassi e processadores. O tamanho dos computadores varia entre o tamanho de gabinete desktop com o modelo de chassi *All-in-One,* até o tamanho de um minicomputador com os modelos *Micro Form Factor* e *Ultra Form Factor*.

Após os testes com a Intel NUC, alternativas de computadores foram procuradas em laboratórios Insper. Foi encontrado o modelo OptiPlex 7000 MFF, que possui um processador Intel Core i5-12500 de 6 núcleos e frequência de clock base 3 GHz. Em relação a memória, o computador tem 8 GB de RAM DDR4 de 3200 MT/s.

Figura 34 - Dell OptiPlex 7000 MFF (fonte: acdtech.mu)

A configuração desse minicomputador é mais próxima ao dos testes no notebook. O desempenho de um núcleo individual desse modelo é superior ao do notebook testado. Entretanto, a OptiPlex possui dois núcleos a menos no processador e metade da memória

disponível. Ainda não foram realizados testes com o dispositivo, que apenas recentemente se encontrou disponível.

3.9 Interface Humano-Robô e NAV 2

Como interface humano-robô foi utilizada a aplicação RViz. A ferramenta é o visualizador padrão para o ROS, capaz de se inscrever em tópicos para o recebimento de dados e de representá-los visualmente em 3D. O RViz possui padrões de visualização para os formatos de mensagem ROS mais comuns como odometria, pose e *point cloud*, entre outros. Além disso, é possível configurar plugins customizados para representar um formato de dados visualmente, o que é utilizado pelo RTAB-MAP, por exemplo, para incorporar associação de *point cloud* com imagem RGB.

Outro papel interessante do RViz é sua associação com o pacote NAV 2. O RViz permite publicar dados de Pose 2D como objetivos de navegação e enviá-los ao canal esperado pelo NAV 2. Entretanto, isso não é suficiente para publicar uma trajetória de waypoints, dado que apenas um ponto serve como objetivo. Para resolver essa questão, um plugin de RViz oficial do pacote NAV 2, *waypoint_follower*, foi desenvolvido.

O NAV 2 possui um arquivo de *launch* pronto em seu pacote de *bringup* para inicializar seus plugins. Nele é possível visualizar informações como trajetórias locais e globais, *footprint* do robô e odometria. O plugin *waypoint_follower* também está disponível para publicar os waypoints.

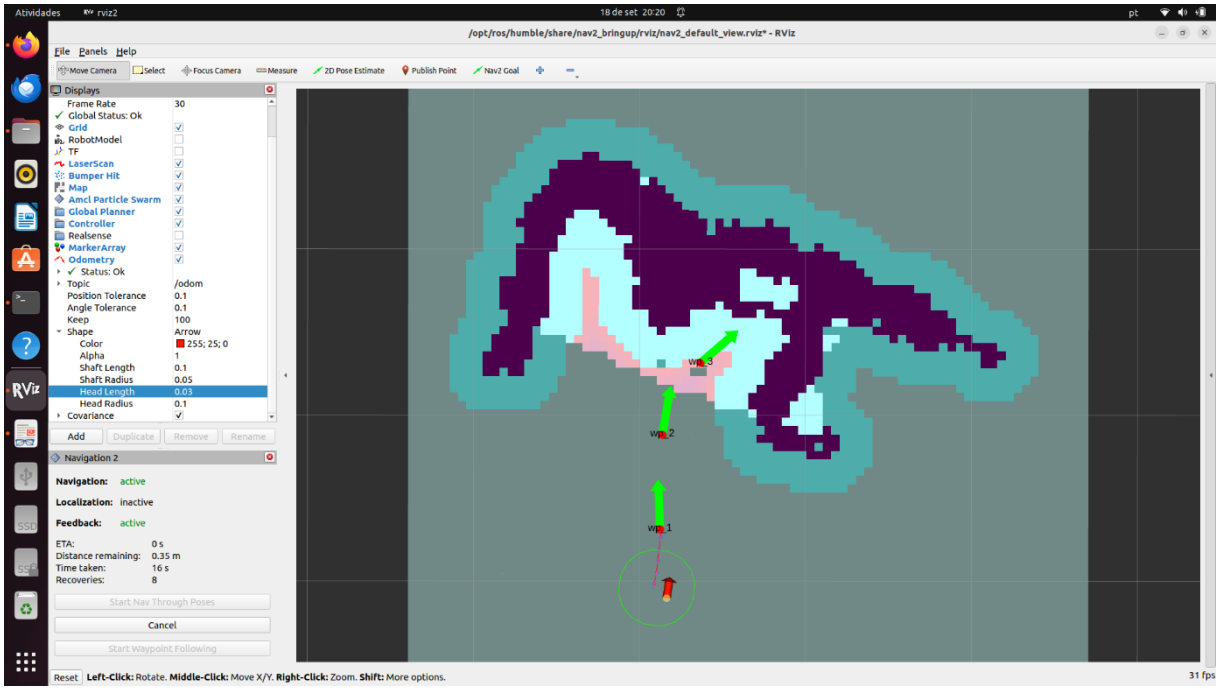

Figura 35 – Ambiente RViz configurado com plugins NAV 2. A circunferência representa a footprint do robô, a seta vermelha representa a odometria, a seta roxa representa a trajetória computada e os pontos vermelhos são os Waypoints

Também foi testado a comunicação entre computadores. Para isso, dois computadores se conectaram na mesma rede local: um rodando o *launch* do projeto e outro rodando apenas o RViz. O ROS 2 permite que qualquer dispositivo encontre outros dispositivos na mesma rede de forma descentralizada, sem a existência de um processo externo que organize isso. Para comparação, o ROS 1 usava o comando *roscore* para configurar o *ROS Master,* o processo principal de descoberta de tópicos, e qualquer máquina que desejasse se comunicar deveria informar o endereço do *master*. Portanto, apenas conectar os dispositivos na mesma rede é o suficiente para permitir a transferência de dados entre os computadores.

Em relação ao NAV 2, os parâmetros configurados ainda não foram testados no UGV, dado que o módulo de navegação autônoma ainda não foi instalado. Para selecionar os parâmetros, a documentação oficial do NAV 2 (NAV2, 2023) foi consultada e apenas os planejadores locais e globais foram configurados considerando características específicas do projeto, como o objetivo de seguir waypoints e o fato do UGV ter Ackermann *steering*.

3.10 Conclusão e melhorias futuras

Como resultados intermediários, portanto, foi configurado o ambiente de desenvolvimento ROS para o módulo de navegação autônoma e desenvolvido arquivos *launch* que orquestram a execução de nós para a tarefa de SLAM e navegação. Com o uso do RTAB-MAP, foi possível obter dados de odometria visual e realizar o SLAM com uma confiabilidade

considerável. Também foi configurado o visualizador RViz, responsável pela visualização do mapa e pela publicação de waypoints. Além disso, o pacote NAV 2 foi configurado com os parâmetros padrões para as condições do projeto. Vale ressaltar que os dados de odometria visual obtidos pelo RTAB-MAP contém as informações de *pose*, que já estão disponíveis para a telemetria.

Como melhorias futuras, é previsto, primeiramente, a instalação do módulo de navegação no UGV. O processo de instalação não foi efetuado até o momento devido às incertezas no processo de seleção de computadores anterior a disponibilidade da Dell OptiPlex 7000. Com isso, será possível realizar testes práticos com o NAV 2 para calibrar a navegação autônoma. Além disso, a instalação do LiDAR permitirá testes com técnicas que podem refinar o SLAM. Por fim, será necessário conseguir coletar dados do GPS com maior confiabilidade com o uso de uma antena externa.

4 Referências e Bibliografia

PAHL, G. et al. **Projeto na Engenharia:** Fundamentos do Desenvolvimento Eficaz de Produtos - Métodos e Aplicações. Tradução: Hans A. Werner. 6. ed. São Paulo: Blucher, 2005, 433 p.

CORKE, Peter. **Robotics, Vision and Control:** Fundamental Algorithms in MATLAB. 1. ed. Berlin: Springer, 2011. 570 p.

RASPBERRY PI 4 MODEL B. 2024. Disponível em: < https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>. Acesso em: 01 set. 2024.

ROS 2 DESIGN. 2024. Disponível em: <http://design.ros2.org/>. Acesso em: 01 set. 2024.

SCOTT, Katherine; FOOTE, Tully. 2023 ROS Metrics Report. São Francisco: ROS, 2023. Disponível em: <https://discourse.ros.org/uploads/short-url/gXltQJOg3jBEzsJWcGaQ2G8YpNw.pdf. Acesso em: 01 set. 2024.

NAV 2. 2024. Disponível em: <http://nav2.org/>. Acesso em: 01 set. 2024.

DOCS NAV 2. 2024. Disponível em: <https://docs.nav2.org/plugins/index.html/>. Acesso em: 01 set. 2024.

REDA, Mohamed. *et al. Path planning algorithms in the autonomous driving system: A comprehensive review*. Robotics and Autonomous Systems, v.174. 03 fev. 2024.

ZHU, Jun; LI, Hongyi; ZHANG, Tao. *Camera, LiDAR, and IMU Based Multi-Sensor Fusion SLAM: A Survey*. Tsinghua Science and Technology, v.29, n.2, p. 415-429. 2024.

ARAÚJO, Breno Alencar et al. UGV (Unmaned Ground Vehicle) Para Monitoramento De Talhões Em Fruticultura E Silvicultura. **Produção**, v. 3, n. 2, p. 88-90, 2024.

RTAB-MAP. 2023. Disponível em: <https://introlab.github.io/rtabmap/>. Acesso em: 01 set. 2024.

FILIPENKO, Maskim; Afanasyev, Ilya. *Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment*. 9th IEEE International Conference: Intelligent Systems 2018. 2018. Madeira, Portugal.

WHAT IS SLAM?. 2024. Disponível em <https://www.faro.com/pt-BR/Resource-Library/Article/What-is-SLAM>. Acesso em: 01 set. 2024.

REP 143. 2014. Disponível em <https://www.ros.org/reps/rep-0143.html>. Acesso em: 16 set. 2024.

SEARS-COLLINS, Addison. What is an occupancy grid?. **Automatic Addison**, 27 jun. 2021. Disponível em: https://automaticaddison.com/what-is-an-occupancy-grid-map/. Acesso em: 01 set. 2024.

DANIEL SCHNABEL. **rovy_base_controller**. 2020. Disponível em: < https://github.com/dschnabel/rovy_base_controller>. Acesso em: 01 set. 2024.

OCTO MAP. 2024. Disponível em: https://octomap.github.io/. Acesso em: 01 set. 2024.

HOW TO USE LIDAR FOR OUTDOOR USE. 2024. Disponível em: < https://www.hokuyo-aut.jp/products/data.php?id=89>. Acesso em: 01 set. 2024.

CLEARPATH ROBOTICS. 2024. Disponível em: < https://clearpathrobotics.com/>. Acesso em: 01 set. 2024.

ROBOTNIK. 2024. Disponível em: < https://robotnik.eu/products/mobile-robots/>. Acesso em: 15 set. 2024.

GETTING IMU DATA FROM D435I AND T265. 2021. Disponível em: <https://www.intelrealsense.com/how-to-getting-imu-data-from-d435i-and-t265/>. Acesso em: 16 set. 2024.

JPL OPEN-SOURCE ROVER. 2023. Disponível em: < https://open-source-rover.readthedocs.io>. Acesso em: 01 set. 2024.

IEEE. New Jersey, 2024. Disponível em: < https://www.ieee.org/>. Acesso em: 01 set. 2024.

IEEE-RAS. New Jersey, 2024. Disponível em: < https://www.ieee-ras.org/>. Acesso em: 01 set. 2024.

IEEE-RAS. **IEEE 1872.2-2021: Standard for Autonomous Robotics (AuR) Ontology**. New Jersey. 2022.

ISO. Geneva, 2024. Disponível em: < https://www.iso.org/home.html>. Acesso em: 01 set. 2024.

ISO. **ISO 18646-2:2024: Robotics — Performance criteria and related test methods for service robots**. Geneva. 2024.

PMI. **Um guia do conhecimento em gerenciamento de projetos**: Guia PMBOK 5. ed. EUA: Project Management Institute, 2013. 567.

INTEL REALSENSE DEPTH D435I IMU CALIBRATION. 2019. Disponível em: <https://www.intelrealsense.com/wp-content/uploads/2019/07/Intel_RealSense_Depth_D435i_IMU_Calibration.pdf>. Acesso em: 16 set. 2024.

AN INTRODUCTION TO UDEV: THE LINUX SUBSYSTEM FOR MANAGING DEVICE EVENTS. 2018. Disponível em: <https://opensource.com/article/18/11/udev>. Acesso em: 16 set. 2024.

VERSOLATO, André Rodrigues et al. **PEIXE ROBÔ: Veículo Submergível Semiautônomo**. 2024. Projeto final de engenharia - Insper, São Paulo, 2024.

FASIOLO, Diego Tiozzo. *et al. Towards autonomous mapping in agriculture: A review of supportive technologies for ground robotics*. Robotics and Autonomous Systems, v.169. 22 ago. 2023.

REGER, Matthias; STUMPENHAUSSEN, Jörn; BERNHARDT, Heinz. *Evaluation of LiDAR for the Free Navigation in Agriculture*. AgriEngineering, v.4. 31 mai. 2022.

LABBÉ, Mathieu; MICHAUD, François. *RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation*. Journal of Field Robotics, v.36. 08 fev. 2019.

HAIZHOU, Ding. e*t al*. *Recent developments and applications of simultaneous localization and mapping in agriculture*. Journal of Field Robotics, v.39. 29 jul. 2022.

ROS CROSS-DISTRIBUTION COMMUNICATION. 2022. Disponível em: <https://discourse.ros.org/t/ros-cross-distribution-communication/27335>. Acesso em: 19 set. 2024.

AGUIAR, André Silva. *et al. Localization and Mapping for Robots in Agriculture and Forestry: A Survey*. Robotics, v.9. 18 nov. 2020.

DUSTIN FRANKLIN. **jetson-containers**. 2024. Disponível em: < Commits · dusty-nv/jetson-containers · GitHub>. Acesso em: 19 set. 2024.

NVIDIA DEVELOPER. 2020. Disponível em: <EOL Notice for NVIDIA Jetson TX2 Developer Kit - Jetson & Embedded Systems / Jetson TX2 - NVIDIA Developer Forums>. Acesso em: 19 mês set. 2024.

ROBOTIS E-MANUAL. 2024. Disponível em: <TurtleBot3 (robotis.com)>. Acesso em: 19 set. 2024.

HOW2ELECTRONICS. 2023. Disponível em: <How to interface Quectel L80 GPS Module with Arduino (how2electronics.com)>. Acesso em: 19 set. 2024.

INTELREALSENSE. Disponível em: <imu_stereo_DT_d435_front-crop1a-1-1.png (1440×638) (intelrealsense.com)>. Acesso em: 19 set. 2024.

SILICONHIGHWAY. Disponível em: <NVIDIA Jetson TX2 Developer Kit - 945-82771-0005-000 (siliconhighwaydirect.com)>. Acesso em: 19 set. 2024.

ACDTECH.MU. Disponível em: <DELL OPTIPLEX 7000 MICRO(MFF) – 12TH GEN CORE I5 – 8GB – 256GB(NVMe) – WIN 10 PRO – ACD Tech>. Acesso em 19 set. 2024.

WILL SON; HYE-JONG KIM. **Robotis-GIT**. 2022. Disponível em: < GitHub - ROBOTIS-GIT/ld08_driver: ROS package for TurtleBot3 LD08 Lidar>. Acesso em: 19 set. 2024.

MATHIEU LABBE. **Advanced Parameter Tuning**. 2023. Disponível em: < rtabmap_ros/Tutorials/Advanced Parameter Tuning - ROS Wiki >. Acesso em: 19 set. 2024.

NVIDIA. 2024. Disponível em: < Embedded Systems Developer Kits & Modules from NVIDIA Jetson >. Acesso em: 19 set. 2024.

NAV2.SELECT AN ALGORITHM. 2023. Disponível em: <Setting Up Navigation Plugins — Nav2 1.0.0 documentation>. Acesso em: 19 set. 2024.

Ding, Haizhou et al. Recent developments and applications of simultaneous localization and mapping in agriculture. **Wiley**, 2022, p. 956-983, abr. 2022. Disponível em: < Recent developments and applications of simultaneous localization and mapping in agriculture - Ding - 2022 - Journal of Field Robotics - Wiley Online Library > acesso em: 19. Set. 2024.

KAZEROUNI, Iman Abaspur. e*t al*. *A survey of state-of-the-art on visual SLAM*. Expert Systems with Applications, v.205. 01 nov. 2022.