

Universidade Federal de Ouro Preto - UFOP - Escola de Minas - Colegiado do curso de Engenharia de Controle e Automação - CECAU



Túlio Xavier Rossi

Mapeamento Tridimensional de Ambientes Internos Utilizando um Sensor LIDAR

Monografia de Graduação em Engenharia de Controle e Automação

Ouro Preto, 2019

Túlio Xavier Rossi

Mapeamento Tridimensional de Ambientes Internos Utilizando um Sensor LIDAR

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: prof. Elias José de Rezende Freitas

Coorientador: prof. Agnaldo José da Rocha Reis

Ouro Preto, 2019

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

R831m Rossi, Tulio Xavier .
Mapeamento Tridimensional de Ambientes Internos Utilizando um Sensor LIDAR. [manuscrito] / Tulio Xavier Rossi. - 2020.
73 f.: il.: color., gráf., tab..

Orientador: Prof. Me. Elias José de Rezende Freitas .
Coorientador: Prof. Dr. Agnaldo José da Rocha Reis.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas.

1. LIDAR (Light Detection And Ranging) . 2. Mapeamento - Imagem tridimensional . 3. Ambiente interno - Mapeamento tridimensional. I. Freitas , Elias José de Rezende. II. Reis, Agnaldo José da Rocha. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 681.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB: 1716

A comissão avaliadora constituída pelos professores Elias José de Rezende Freitas, Agnaldo José da Rocha Reis, Adrielle de Carvalho Santana e Julio Cezar David de Melo atesta que a monografia intitulada "Mapeamento Tridimensional de Ambientes Internos Utilizando um Sensor LIDAR" foi defendida e aprovada em 13 de dezembro de 2019.

Prof. Me. Elias José de Rezende Freitas - Orientador

Prof. Dr. Agnaldo José da Rocha Reis – Coorientador

Profa. Me. Adrielle de Carvalho Santana – Professora Convidada

Prof. PhD. Julio Cezar David de Melo – Professor Convidado

Dedico este trabalho à minha família, que sempre lutou ao meu lado e me ajudou a superar as dificuldades encontradas nessa longa jornada. Esta monografia é a prova de que seus esforços pela minha educação valeram a pena.

Agradecimentos

Agradeço primeiramente aos meus pais Marcílio e Sidneia, que me deram apoio e incentivo nas horas difíceis e foram peça chave para tornar esse sonho possível. Obrigado, Tomaz, irmão querido, por ser tão companheiro. Á minha namorada Thais, que jamais me negou apoio e carinho, e, ao meu filho Miguel.

Um enorme agradecimento aos meus orientadores. Elias, por ter dedicado tanto tempo para me orientar neste trabalho, pela confiança que depositou em mim e pela imensurável paciência para as correções deste trabalho. Ao Agnaldo, por nunca ter medido esforços para me ajudar, não apenas neste trabalho, mas em toda a graduação.

Também agradeço aos docentes da Universidade Federal de Ouro Preto por me proporcionarem um conhecimento de excelência, na qual carrego com orgulho. Agradeço aos excepcionais professores Ríspoli, Paulo e Alan por todo conhecimento partilhado e por serem uma fonte de inspiração para mim.

Ao Júlio pelas oportunidades, e por me ensinar na prática todo o conhecimento que adquiri durante a minha graduação.

A todos, um muito obrigado, vocês foram fundamentais para minha formação, e merecem meu eterno agradecimento.

"Lembre-se de olhar para o alto, para as estrelas, e não para baixo, para os seus pés."
(Stephen Hawking)

Resumo

Aplicações de mapeamento tridimensional estão sendo cada vez mais utilizadas na robótica, principalmente, por possibilitar uma representação mais completa de um ambiente, quando se comparado ao mapeamento bidimensional. Dessa maneira, pode-se permitir, por exemplo, que robôs consigam usufruir desse conhecimento completo do ambiente para realização de tarefas mais complexas. Nesse contexto, objetiva-se com esse trabalho desenvolver um protótipo robótico com o sensoriamento 3D feito com base em um sensor LIDAR, capaz de efetuar o mapeamento, tanto em um ambiente simulado para testes iniciais, quanto em um ambiente real, com dados de posicionamento obtidos a partir de um sensor IMU e um *encoder*. O desenvolvimento do projeto baseou-se em três etapas de implementação: (i) simulação no ambiente de software Gazebo; (ii) implementação da orientação com base nos dados de uma IMU, onde para obter uma estimativa dos ângulos de rotação e para diminuição de ruído aplicou-se o filtro de Kalman e (iii) implementação do posicionamento, onde os componentes foram dispostos em uma estrutura com rodas conectadas a um *encoder*, a fim de garantir a mobilidade do protótipo e obter a translação do sensor. Com essas três etapas concluídas, foi possível gerar uma matriz homogênea de rotação e translação coerente com a movimentação real e, então, gerar uma nuvem de pontos do ambiente. Os resultados do trabalho, mostram que no geral o filtro de Kalman foi efetivo para a diminuição de ruídos nos sensores, como era esperado, e tanto a nuvem de pontos como o mapa gerado apresentaram extrema semelhança com o ambiente testado.

Palavras-chaves: LIDAR, Mapeamento, Tridimensional.

Abstract

Three-dimensional mapping applications are being increasingly used in robotics, because it allows a more complete representation of an environment when compared to two-dimensional mapping. In this way, robots can be allowed, for example, to take advantage of this knowledge of the environment to perform more complex tasks. In this context, the objective of this work is to develop a robotic prototype with 3D sensing based on a LIDAR sensor, capable of mapping, both in a simulated environment for initial tests and in a real environment, with positioning data obtained from an IMU sensor and an encoder. The project development was based on three implementation steps: (i) simulation in the Gazebo software environment; (ii) implementation of orientation based on data from an IMU, where to obtain an estimate of the rotation angles and noise reduction the Kalman filter was applied and (iii) implementation of positioning, where the components were arranged in a structure with wheels connected to an encoder to ensure prototype mobility and sensor translation. With these three steps completed, it was possible to generate a homogeneous matrix of rotation and translation consistent with the actual movement and then generate a point cloud of the environment. The results of the work show that in general the Kalman filter was effective for reducing noise in the sensors, as expected, and both the point cloud and the map generated were extremely similar to the environment tested.

Key-words: LIDAR, Mapping, Indoor, 3D.

Lista de ilustrações

Figura 1 Exemplo de mapa tridimensional 17

Figura 2 Tópicos de comunicação no ROS.Fonte:`https://bit.ly/392IhAV` . . . 20

Figura 3 Representação dos eixos de rotação. 24

Figura 4 Sequência de rotação demonstrando graficamente a não comutatividade dos ângulos de Euler. 25

Figura 5 Placa comemorativa da descoberta dos *quaternions* na ponte de Brougham, na Irlanda. 26

Figura 6 Diagrama de blocos representando a ordem de implementação base do projeto para se obter o mapa 3D. 33

Figura 7 Diagrama de blocos referente às conexões entre os componentes utilizados e suas formas de comunicação. 33

Figura 8 Sendor LIDAR modelo YDLIDARX4. 34

Figura 9 Representação do funcionamento de um sensor LIDAR com base no princípio de triangulação. 35

Figura 10 Comportamento do funcionamento de um sensor LIDAR com base no principio de triangulação com aproximação do objeto. 35

Figura 11 Comportamento do funcionamento de um sensor LIDAR com base no principio de triangulação com o afastamento do objeto. 36

Figura 12 Representação do funcionamento de um sensor LIDAR com base no princípio do tempo de vôo. 37

Figura 13 Placa de desenvolvimento ESP32. 38

Figura 14 IMU modelo GY-87. 39

Figura 15 Representação dos graus de liberdade oferecidos ao sistema pelo MPU6050. 39

Figura 16 Principio de funcionamento acelerômetro MEMS. 40

Figura 17 Estrutura móvel com sensores acoplados para efetuar o mapeamento tridimensional. 41

Figura 18 Modelo robótico utilizado no ambiente simulado pelo Gazebo. 42

Figura 19 Ambiente simulado com o modelo robótico já implementando, mas sem a presença de obstáculos. 42

Figura 20 Robô em um ambiente simulado na presença de obstáculos. 43

Figura 21 Sensores acoplados em um mesmo referencial. 48

Figura 22 Estrutura móvel com sensores acoplados para efetuar o mapeamento tridimencional. 49

Figura 23 *Encoder* fixado à roda da estrutura móvel. 49
Figura 24 Nuvem de pontos e mapa tridimensional obtidos através da simulação. 50
Figura 25 Dados da rotação em torno do eixo X (*Roll*) obtidos pelo sensor IMU enquanto mantido em posição inercial. 51
Figura 26 Dados da rotação em torno do eixo Y (*Pitch*) obtidos pelo sensor IMU enquanto mantido em posição inercial. 52
Figura 27 Dados da rotação em torno do eixo X (*Roll*) obtidos pelo sensor IMU quando gerado uma movimentação aleatória no sensor. 52
Figura 28 Dados da rotação em torno do eixo Y (*Pitch*) obtidos pelo sensor IMU quando gerado uma movimentação aleatória no sensor. 53
Figura 29 Dados da rotação em torno do eixo Z (*Yaw*) obtidos pelo sensor IMU quando gerado uma movimentação aleatória no sensor. 53
Figura 30 Fotografia do local mapeado e mapa tridimensional obtido. 54
Figura 31 Ambiente a ser mapeado, utilizando metodologia SLAM. 55
Figura 32 Resultado mapeamento tridimensional pelos métodos SLAM. 56

Lista de tabelas

Tabela 1 Relação de tipos de mensagens do ROS e suas variáveis associadas(HENDRIX, 2019). 20

Tabela 2 Informações técnicas do sensor LIDAR modelo YDLIDAR X4. 34

Tabela 3 Comparação técnica entre ESP32 e Arduino UNO R3. 37

Tabela 4 Informações técnicas da placa de desenvolvimento ESP32. 38

Tabela 5 Preço dos componentes utilizados. 57

Lista de abreviaturas e siglas

A	Ampere
ADC	Analog-to-Digital Converter
B	Byte
b	Bit
DAC	Digital-to-Analog Converter
Hz	Hertz
IMU	Inertial Measurement Unit
IP	Internet Protocol
LASER	Light Amplification by Stimulated Emission of Radiation
LIDAR	Light Detection And Ranging
PCD	Point Cloud Data
PWM	Pulse Width Modulation
RGB-D	Red Green Blue - Depth
ROS	Robot Operating System
RPC	Remote Procedure Call
SLAM	Simultaneous Localization and Mapping
SONAR	Sound Navigation and Ranging
TCP	Transmission Control Protocol
UART	Universal Asynchrounous Receiver/Transmiter ou Receptor/Transmissor
V	Volts
XML	Extensible Markup Language

Lista de símbolos

α	Letra grega minúscula Alpha
β	Letra grega minúscula Beta
γ	Letra grega minúscula Gamma
Δ	Letra grega maiúscula Delta
ϕ	Letra grega minúscula Phi
θ	Letra grega minúscula Theta

Sumário

1 Introdução . . . 16
1.1 Formulação do Problema . . . 16
1.2 Justificativa . . . 16
1.3 Objetivo . . . 16
1.4 Organização do Trabalho . . . 17

2 Referência Bibliográfica . . . 18
2.1 Robot Operating System (ROS) . . . 18
2.1.1 Comunicação . . . 18
2.1.1.1 ROS Mestre . . . 18
2.1.1.2 Nós . . . 19
2.1.1.3 Publicação e Subscrição de mensagens . . . 19
2.1.1.4 Tópicos . . . 19
2.1.1.5 Mensagens . . . 20
2.1.1.6 Parâmetros . . . 20
2.1.2 Ferramentas do ROS . . . 21
2.1.2.1 ROS Visualization . . . 21
2.1.2.2 Gazebo . . . 21
2.1.2.3 OctoMap . . . 21
2.2 Rotações no plano tridimensional . . . 22
2.2.1 Matriz de rotação . . . 22
2.2.1.1 Matrizes com coordenadas homogêneas . . . 23
2.2.2 Ângulos de Euler . . . 23
2.2.2.1 Gimbal lock . . . 24
2.2.3 Quaternions . . . 25
2.2.3.1 Contexto histórico . . . 26
2.2.3.2 Propriedades básicas dos *quaternions* . . . 26
2.2.3.3 Equivalência entre *quaternion* e matrizes de rotação . . . 28
2.3 Filtro de Kalman . . . 28
2.3.1 Espaço de Estado . . . 29
2.3.1.1 Predição de Estado e a Matriz de covariância . . . 29
2.3.2 Ganho de Kalman . . . 30
2.3.3 Cálculo da estimativa e do erro de covariância . . . 30

3 Metodologia . . . 32
3.1 Componentes . . . 32

3.1.1 YDLIDAR-X4 . . . 32
3.1.1.1 Princípio de triangulação . . . 34
3.1.1.2 Tempo de vôo (ToF) . . . 36
3.1.2 Placa de desenvolvimento - ESP32 . . . 37
3.1.3 Unidade Inercial de Medida(IMU) . . . 38
3.1.3.1 Acelerômetro e Giroscópio MPU-6050 . . . 39
3.1.4 Encoder Rotativo KY-040 . . . 40
3.2 Etapa Simulada . . . 41
3.2.1 Ambiente virtual Gazebo . . . 41
3.3 Etapa Real . . . 42
3.3.1 Parâmetros de configuração LIDAR . . . 43
3.3.2 Obtenção *Roll*, *Pitch* e *Yaw* pelo Giroscópio . . . 43
3.3.3 Obtenção de *Roll* e *Pitch* pelo Acelerômetro . . . 44
3.3.4 Obtenção do deslocamento . . . 45
3.3.5 Transformação Euler - *Quaternion* . . . 46
3.4 Efetividade do filtro de Kalman . . . 47
3.4.1 Efetividade do filtro de Kalman em situação estacionária . . . 47
3.4.2 Efetividade do filtro de Kalman para movimentos gerados aleatoriamente . . . 48
3.4.3 Mapa gerado apenas pela rotação . . . 48
3.4.4 Localização e Mapeamento Simultâneos . . . 48

4 Resultados . . . 50
4.1 Etapa Simulada . . . 50
4.2 Aplicação do filtro de Kalman . . . 51
4.2.1 Efetividade do filtro de Kalman em situação estacionária . . . 51
4.2.1.1 *Roll* . . . 51
4.2.1.2 *Pitch* . . . 51
4.2.2 Efetividade do filtro de Kalman para movimentos gerados aleatoriamente . . . 51
4.2.2.1 *Roll* . . . 51
4.2.2.2 *Pitch* . . . 52
4.2.2.3 *Yaw* . . . 52
4.3 Mapa tridimensional . . . 53

5 Conclusão . . . 57

Conclusão . . . 57
5.1 Trabalhos Futuros . . . 57

Referências . . . 59

1 Introdução

1.1 Formulação do Problema

A necessidade de dados mais exatos para o mapeamento tridimensional (3D) motiva o desenvolvimento de novas tecnologias, tornando sua utilização usual para as mais amplas aplicações, que sucessivamente substituem o uso do mapeamento bidimensional, pois conseguem solucionar e prever problemas de maior complexidade.

1.2 Justificativa

Contudo, na literatura podemos observar que o emprego de métodos utilizando sensores do tipo *Light Detection And Ranging* (LIDAR) vem tomando frente para o registro de nuvem de pontos, e, consequentemente, sendo utilizado para mapeamento e reconstrução de ambientes (PAVAN; SANTOS, 2015). Isso ocorre, principalmente, devido ao fato dos sensores LIDAR gerarem um resultado rápido, preciso e seguro quando se comparado a mapeamento utilizando outras tecnologias como *Sound Navigation and Ranging* (SONAR) ou câmeras RGB-D.

A tecnologia LIDAR pertencendo à área de sensoriamento, que tem por finalidade a medição primária de distância entre o sensor e uma superfície, por meio da emissão de pulsos *Light Amplification by Stimulated Emission of Radiation* (LASER), que, quando associados às informações de posicionamento, permitem criar uma nuvem de pontos tanto em âmbito bidimensionais quanto tridimensionais. (MACIEL, 2011).

A utilização do mapeamento tridimensional pode ser empregado em áreas de mineração (Wu et al., 2018), arquitetura, construções civis e, inclusive, para exploração de ambientes, como é fantasiado no filme de ficção científica futurístico Prometheus, onde um conjunto de drones dotados com tecnologia LIDAR fazem o mapeamento de uma caverna e a nuvem de pontos resultante é enviado em tempo real ao usuário (PROMETHEUS, 2012).

1.3 Objetivo

Com base na necessidade e vantagens de se obter um mapa tridimensional, o objetivo deste trabalho é desenvolver um protótipo robótico móvel com o sensoriamento feito com base em um sensor LIDAR, capaz de efetuar o mapeamento em um ambiente

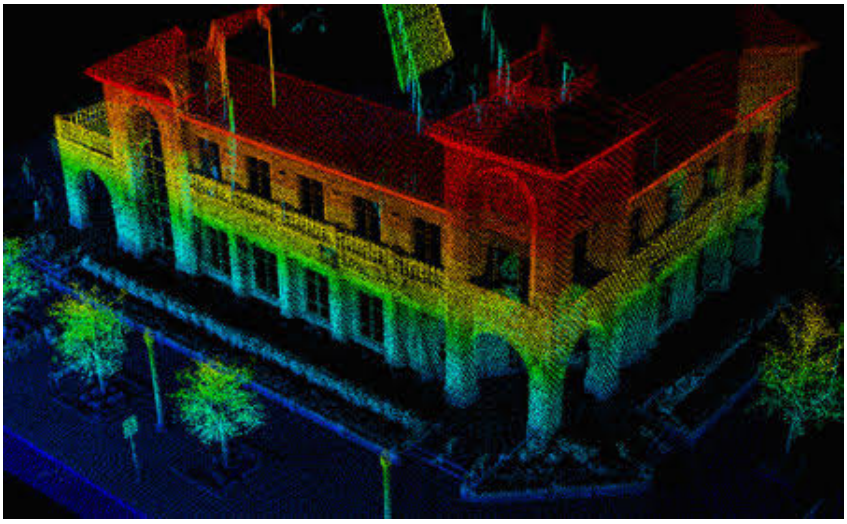


Figura 1: Exemplo de mapa tridimensional construído a partir de uma nuvem de pontos geradas por um sensor LIDAR. Fonte:https://www.sanborn.com.

simulado para testes e, posteriormente, fazer sua implementação em um ambiente real com dados de posicionamento gerados a partir de um sensor IMU e um *encoder*, o resultado é um mapa tridimensional desse ambiente fornecido ao usuário.

1.4 Organização do Trabalho

Este trabalho está estruturado da maneira como se segue. O Capítulo 1 apresenta uma introdução ao tema desenvolvido, especificando os objetivos a serem alcançados. O Capítulo 2 faz uma breve revisão bibliográfica, exemplificando os principais bases teóricas adotados neste trabalho para a realização dos métodos propostos no Capítulo 3. No Capítulo 4 são apresentados os resultados de mapeamento obtidos e o tratamento de dados utilizados para se obter o posicionamento do robô. Por fim, o Capítulo 5 é dedicado às considerações finais e a propostas para futuras melhorias deste trabalho.

2 Referência Bibliográfica

Este capítulo apresenta conceitos encontrados na literatura na qual foram utilizados neste presente trabalho para o desenvolvimento de um robô capaz de efetuar o mapeamento tridimensional do ambiente, bem como fornecer a sua localização ao usuário.

2.1 Robot Operating System (ROS)

Para o desenvolvimento de software e comunicação entre hardwares, foi usada a plataforma ROS (*Robot Operating System*), que permite executar projetos robóticos de forma individual ou grupal, permite criação de pacotes em múltiplas linguagens, provê serviços de um sistema operacional, controle de dispositivos, comunicação entre processos e é capaz de proporcionar a intercomunicação de hardwares (BOHREN, 2010; ROMERO, 2014; GVDHOORN, 2017).

Apesar das semelhanças com um sistema operacional (agendamento de processos, carregamento, monitoramento e ratamento de erros), de acordo com (BOHREN, 2010) não se pode classificar o ROS como um sistema operacional, pois o mesmo não pode ser instalado diretamente em um dispositivo robótico, ou seja, o mesmo trabalha sobre um sistema operacional. Sem uma classificação clara, em (DOE, 2018) o ROS é denominado como *"meta-operating system"* devido ao fato de necessitar estar instalado em cima de sistema operacional para funcionar.

2.1.1 Comunicação

Quando o ROS está em execução uma rede *peer-to-peer* de processos é criada, sendo que a comunicação pode ser feita utilizando, por exemplo, protocolos baseados em TCP/IP (TCPROS) ou em XMLRPC. Nas próximas subseções são apresentados esses protocolos e são apresentadas algumas definições e componentes da comunicação no ROS (SU; GUO; SHIAU, 2008).

2.1.1.1 ROS Mestre

O ROS Mestre funciona de forma análoga a um servidor que armazena informação para os serviços e os nós (GVDHOORN, 2017). Os nós se comunicam com o mestre para receberem informação a respeito de outros nós cadastrados e fazer a comunicação entre

eles. O mestre também envia informações de *callbacks* para quando essas informações forem alteradas, permitindo a criação de conexões dinâmicas, simultaneamente à execução.

A comunicação entre os nós (Seção 2.1.1.2) com o mestre é feito de forma direta, onde o mestre apenas providência a informação necessária, como, por exemplo um servidor DNS (*Domain Name System*). Os nós que se inscrevem em um tópico (Seção 2.1.1.4) requerem dados dos nós que publicam aquele tópico, sendo feito a conexão por meio de protocolo de conexão combinada, sendo TCPROS o mais utilizado pelo ROS, que possui funcionalidade semelhante ao TCP/IP padrão mas requer intermediação do Mestre (PECKA, 2018).

2.1.1.2 Nós

Os "Nós"são definidos como os menores processos computacionais do ROS, que quando em execução formam uma rede *peer-to-peer* de processos, que se acoplam ao ROS mestre usando o sistema de comunicação do TCPROS ou XMLRPC (PYO et al., 2017).

Pode-se dizer então, que de forma geral os nós funcionam de forma unidirecional, e mantém conexão continuas (enviando ou recebendo mensagens), o que é crucial para atuações em tempo real (ROMERO, 2014).

2.1.1.3 Publicação e Subscrição de mensagens

A ação de transmitir mensagens para o tópico desejado é denominado de "publicar"(do inglês *Publish*), e o nó caracterizado como *"Publisher"* armazena suas próprias informações e tópicos no mestre e envia uma mensagem aos *Subscribers* conectados ao mesmo tópico de interesse. O *Publisher* é declarado no nó, e pode ser declarado diversas vezes no mesmo nó (PYO et al., 2017).

Já a ação de receber mensagens relacionadas ao tópico desejado é denominada de subscrição (do inglês *Subscribe*), já um *subscriber* é um nó que registra suas próprias informações e tópicos no ROS mestre, que recebe informações do *Publisher* que armazena informações no tópico. Ele é declarado no nó, e pode ser declarado diversas vezes no mesmo nó.

2.1.1.4 Tópicos

Como exemplificado em (ROMERO, 2014), os tópicos podem ser mais facilmente entendidos como "tópicos de comunicação" entre os *Publishers* e *Subscribers*, onde cada tópico pode ser utilizado por diversos *Publishers* e *Subscribers*. Assim um único *Publisher* e *Subscriber* pode ser atrelado a diversos tópicos como mostra a Figura 2. Entretanto, durante a comunicação os nós fazem publicações no tópico desejado, sem saber com quem estão se comunicando (ROMERO, 2014).

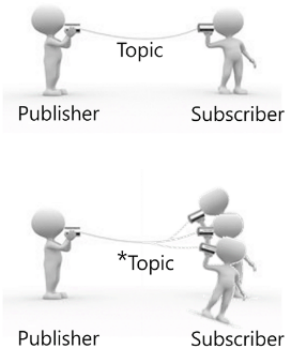


Figura 2: Tópicos de comunicação no ROS.Fonte:https://bit.ly/392IhAV

Tabela 1: Relação de tipos de mensagens do ROS e suas variáveis associadas(HENDRIX, 2019).

Tipo de dado ROS	**Tipo de dado C++**	**Tipo de dado Python**
Bool	uint8_t	Bool
Int8	int8_t	Int
Uint8	uint8_t	Int
Int16	int16_t	Int
Uint16	uint_t	Int
Int32	int32_t	Int
Uint32	uint32_t	Int
Int64	int64_t	Long
Uint64	Uint64_t	Long
Float32	Float	Float
Float64	Double	Float
String	Std::string	Str
Time	Ros::Time	Rospy.time
Duration	Ros::duration	Rospy.Duration
Fixed-length	Boost::array, std::vector	tuple
Variable-length	Std::vector	Tuple
Uint8[]	Std::vector	Bytes
Bool[]	Std:::vect<unit8_t>	List of bool

2.1.1.5 Mensagens

Um nó recebe e envia dados entre outros nós através das mensagens, e essas mensagens podem ser dos tipos contidos na Tabela 1 (HENDRIX, 2019).

2.1.1.6 Parâmetros

No ROS, os parâmetros referem-se às configurações dos nós. Os valores dos parâmetros podem ser lidos e alterados quando necessário, podendo ser muito útil quando são alterados em tempo real. Por exemplo, como usado neste trabalho, a alteração dos parâmetros pode ser realizada para definir a velocidade de motores ou, por exemplo, a

distância máxima e mínima da medição do sensor LIDAR.

2.1.2 Ferramentas do ROS

Nesta subseção serão apresentados as ferramentas do ROS, utilizadas para a obtenção, visualização e tratamento dos dados registrados pelos sensores do projeto na obtenção do resultado desejado.

2.1.2.1 ROS Visualization

O RViZ (*ROS visualization*) é uma ferramenta de visualização do ROS que permite ao usuário visualização dos dados e estados de sensores, seja ele real ou simulado. Com isso, neste trabalho foi possível obter em tempo real e fornecer ao usuário a visualização dos dados de distância do laser para um obstáculo, e até mesmo, do ambiente tridimensional PCD (*Point Cloud*), gerado pelo sensor LIDAR.

O RViZ também permite ao usuário enviar comandos para interagir nos movimentos do robô, receber dados através dos nós, além de permitir ao usuário operar robôs, respeitando seus graus de liberdade, podendo pode ser útil para simulação ou até mesmo o próprio controle do robô (PYO et al., 2017).

2.1.2.2 Gazebo

Gazebo é uma ferramenta do ROS que permite a criação, visualização e manipulação de um ambiente virtual usado para teste validação de metodologias. Além disso o Gazebo permite comunicação com as demais ferramentas do ROS, possibilitando a simulação de protótipos robóticos simulados, e interação dos mesmos em um ambiente virtual.

2.1.2.3 OctoMap

A biblioteca OctoMap é um *framework* que permite a implementação de um mapa de ocupação em grade com base em dados estatísticos, fornecendo uma estrutura de dados para mapeamento adequado à robótica, a partir de um robusto algorítimo em C++ que permite a criação de um mapa tridimensional completo, conseguindo distinguir entre áreas ocupadas, vazias e desconhecidas (HORNUNG et al., 2013).

Além da flexibilidade de permitir diferentes tipos de resolução e um único mapa com várias resoluções, o OctoMap permite a atualização do mapa em tempo real, isso tudo devido a um sistema de cálculos probabilístico que eliminam os ruídos do sensor e às movimentações dinâmicas do ambiente. Permitindo que um mesmo mapa seja atualizado por vários robôs de forma simultânea (HORNUNG et al., 2013).

Para fins de armazenamento, o OctoMap consegue gerar um mapa extremamente compacto quando comparado a outros modelos de mapeamento tridimensional, sendo de grande relevância para a economia em memória computacional (HORNUNG et al., 2013).

2.2 Rotações no plano tridimensional

Neta seção serão analisadas as rotações em volta dos três eixos principais X, Y, Z, além de apresentar algumas das formas de fazer a representação das rotações em $\Re^3$ como:

- Matriz de rotação
- Ângulos de Euler
- Quatérnion

2.2.1 Matriz de rotação

A matriz de rotação é muito utilizada na robótica por permitir representar todas as operações básicas de rotação de um corpo na forma matricial.

Como deduzido em (LOPES, 2002), pode-se representar uma rotação de um ângulo α em torno do eixo X da seguinte forma:

$$R_{(x,\alpha)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{2.1}$$

analogamente, tem-se que uma rotação de um ângulo genérico ϕ em torno do eixo Y pode ser dada por:

$$R_{(y,\phi)} = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \tag{2.2}$$

e uma rotação de um ângulo ψ em torno do eixo Z pode ser dado por:

$$R_{(z,\psi)} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

2.2.1.1 Matrizes com coordenadas homogêneas

Em um espaço tridimensional, os movimentos de rotação atrelados a objetos, normalmente estão também associados a movimentos de translação, mudança de escala e efeitos de perspectiva. Sendo assim as matrizes 3x3 não permitem representar todas essas transformações. Para solucionar o caso de translação pode-se utilizar matrizes 4x4 com coordenadas homogêneas, da seguinte maneira (LOPES, 2002):

$$H = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ f_{1\times3} & e \end{bmatrix} \tag{2.4}$$

em que H representa a matriz homogênea, $R_{3\times3}$ a matriz de rotação, $t_{3\times1}$ o movimento de translação, $f_{1\times3}$ o efeito de perspectiva e e o fator de escala. Como o efeito de perspectiva, e fator de escala não são de interesse deste presente trabalho, a matriz 4x4 pode ser representada da seguinte forma:

$$H = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ f_{1\times3} & 1 \end{bmatrix} \tag{2.5}$$

sendo $f_{1\times3}$ uma matriz nula.

2.2.2 Ângulos de Euler

Uma rotação definida pelos ângulos de Euler(α, β, γ) é o resultado da rotação sequencial em torno dos eixos de coordenadas. Já que as rotações em torno dos eixos não comutam, a ordem pela qual as rotações são aplicadas é importante. Por meio dos ângulos de Euler é possível representar as rotações no plano tridimensional de 12 formas, sendo elas: $xyz, xyx, yzx, yzy, zxy, zxz, xzy, xzx, yxz, yxy, zyz$.

De acordo com (PINHEIRO, 2006) a forma de representação dos ângulos de Euler mais popular é o modelo zyx, comumente utilizado na aeronáutica para descrever os movimentos de três graus de liberdade dos aviões, como é possível observar na Figura 3:

Em geral, os ângulos de Euler é o modelo mais comuns de se obter a representação da rotação de um objeto no espaço, por meio de três ângulos independentes que representam a rotação em torno de cada um dos três eixos. Porem, apesar de amplamente utilizado por conta da simplicidade e confiança, esse método enfrenta certas dificuldades, principalmente, referentes ao problema de *Gimbal Lock* (Seção 2.2.2.1) e o fato das operações não serem comutativas, implicando que a ordem de rotação é importante para a obtenção do resultado final, como exemplificado na Figura 4 (PINHEIRO, 2006; BIASI, 2016).

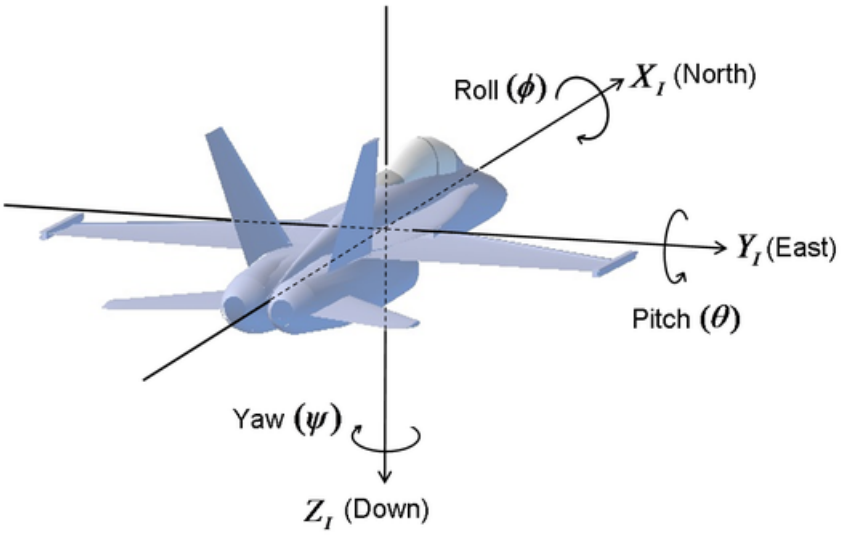


Figura 3: Imagem de um avião representando os eixos de rotação, na qual o *Roll* é a rotação em volta da direção do avião (X_1), *Pitch* a rotação em torno do eixo horizontal do avião (Y_1) e *Yaw* a rotação em torno do eixo vertical (Z_1).

2.2.2.1 Gimbal lock

De acordo com (BIASI, 2016; MENEGUZZI, 2008), dependendo da ordem de multiplicações para se obter a matriz de rotação final, é possível que em alguns casos um eixo se confunda com outro, sendo essa singularidade chamada de *Gimbal Lock*, ela ocorre quando dois eixos ficam em uma configuração paralela, assim, um dos graus de liberdade é perdido, fazendo com que o sistema que antes era tridimensional se torne um sistema bidimensional, tornando necessária a utilização de outra representação, via *quaternions* (MENEGUZZI, 2008).

Para exemplificar esse problema, (PINHEIRO, 2006) propõe um exemplo que seria dotado por uma série de rotações: A primeira rotação é um ângulo α em volta do eixo X, a segunda é de 90º em torno do eixo Y e a terceira de γ em torno do eixo Z. Com isso, pode-se notar que a rotação em torno do eixo Y gerou um aliamento dos eixos X e Z, provocando a perda de um grau de liberdade.

Matematicamente, pode-se representar esse movimento da seguinte forma:

$$R_{x,y,z}(\alpha, \frac{\pi}{2}, \gamma) = R_z(\gamma)R_y(\frac{\pi}{2})R_x(\alpha) \tag{2.6}$$

$$R_{x,y,z}(\alpha, \beta, \gamma) = \begin{bmatrix} 0 & \cos\gamma\sin\alpha - \sin\gamma\cos\alpha & \cos\gamma\cos\alpha + \sin\gamma\sin\alpha \\ 0 & \sin\gamma\sin\alpha + \cos\gamma\cos\alpha & \sin\gamma\cos\alpha - \cos\gamma\sin\alpha \\ -1 & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \tag{2.7}$$

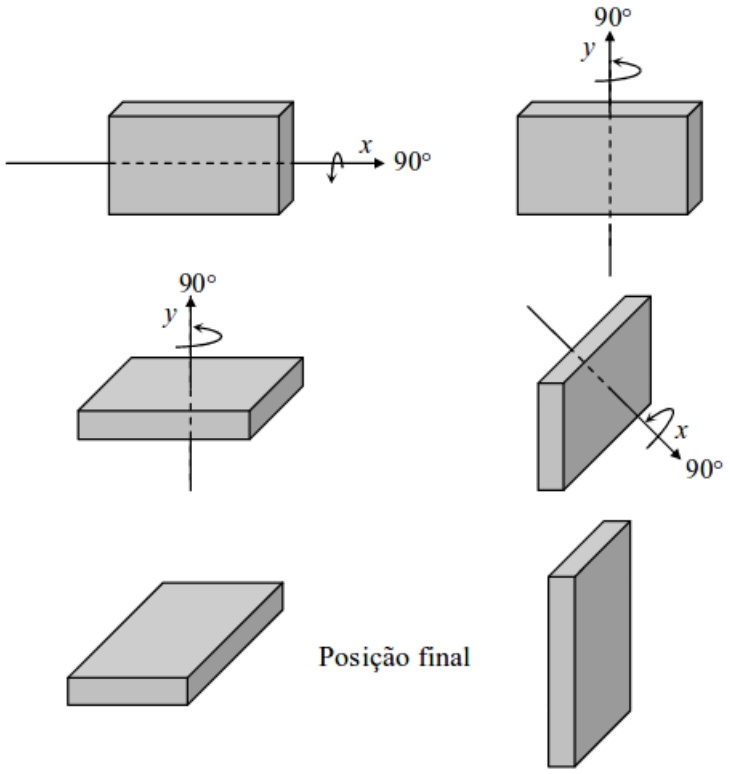


Figura 4: Sequência de rotação demonstrando graficamente a não comutatividade dos ângulos de Euler. Fonte: (BIASI, 2016).

Substituindo os valores na Equação 2.7, tem se que:

$$R_{x,y,z}(\alpha, \frac{\pi}{2}, \gamma) = \begin{bmatrix} 0 & \sin(\alpha-\gamma) & \cos(\alpha-\gamma) \\ 0 & \cos(\alpha-\gamma) & -\sin(\alpha-\gamma) \\ -1 & 0 & 0 \end{bmatrix} \tag{2.8}$$

A Equação 2.8 prova que a rotação depende apenas da diferença entre α e γ, portando, pode-se concluir que esse caso se enquadrou na singularidade do *Gimbal Lock*, havendo a perda de um grau de liberdade, pois as variações de α e γ resultam em rotações em volta do mesmo eixo.

2.2.3 Quaternions

De acordo com (COSTA, 2014) os *quaternions* não obedecem à álgebra tradicional e o fato dele ser representado por quatro dimensões elimina alguns problemas enfrentados pelos ângulos de Euler, citados na seção anterior.

Neste presente trabalho a utilização dos *quaternions* se faz necessária, pois o ROS só

permite a transformação das rotações para um ponto referencial apenas se, e somente se, as mensagens enviadas pelo tópico satisfazerem as características de um elemento *quaternion*. Essas características serão apresentadas na seção de propriedades dos *quaternions*.

2.2.3.1 Contexto histórico

A descoberta do *quaternion* foi atribuída ao matemático Willian Rowan Hamilton, onde seu propósito era representar o plano tridimensional utilizando as formas aritméticas dos números complexos (HAMILTON, 1866; MUKUNDAN, 2002).

Uma das motivações de Hamilton foi descrita por (BIRKHOFF; LANE, 1997) em que ele desejava utilizar números complexos tridimensionais para descrever as rotações no espaço, dessa forma, Hamilton percebeu que seriam necessários quatro números para descrever uma rotação, seguida de uma mudança de escala: um número correspondente à mudança de escala, outro número para indicar o ângulo de rotação e os dois restantes para indicar o plano de rotação, encontrando, assim, a solução para o problema.

Introduzindo em uma estrutura não comutativa, (HAMILTON, 1866) encontrou a multiplicação na forma: $w + ai + bj + ck$. "A partir disso chamou seus números complexos em $\Re^4$ de *quaternions*"(BIRKHOFF; LANE, 1997).



Figura 5: Placa comemorativa da descoberta dos *quaternions* na ponte de Brougham, na Irlanda. Fonte: https//www.fep.up.pt/docentes/jamatos/images/sam_0811.jpg.

2.2.3.2 Propriedades básicas dos *quaternions*

A fim de apresentar algumas das relações matemáticas básicas dos *quaternions* essa seção traz algumas das propriedades da descoberta de Hamilton.

Semelhante aos números complexos que possuem parte real e imaginária, em geral, um *quaternion* é também formado por uma parte real na qual representaremos por w, e outra imaginária composta por três eixos, na qual chamaremos de i, j e k. (BIASI, 2016).

Dessa forma podemos dizer que:

$$q = w + ai + bj + ck \tag{2.9}$$

Para efetuarmos multiplicações de números imaginários de naturezas diferentes, devemos seguir as seguintes propriedades (HAMILTON, 1866):

$$i^2 = j^2 = k^2 = -1 \tag{2.10}$$

$$ij = -ji = k \tag{2.11}$$

$$jk = -kj = i \tag{2.12}$$

$$ki = -ik = j \tag{2.13}$$

Com base nas equações apresentadas acima, pode-se observar que o produto de dois números imaginários de natureza diferente não apresenta resultado comutativo, então de acordo com (BIASI, 2016) podemos concluir que rotações tridimensionais também não comutam.

Em (PINHEIRO, 2006) são citados algumas das principais propriedades dos *quaternions*, sendo algumas delas:

Produto entre *quaternions* q_1 e q_2:

$$\begin{aligned} q_1 q_2 = a_1 a_2 - (b_1 b_2 + c_1 c_2 + d_1 d_2) + a_1(b_2 i + c_2 j + d_2 k) + a_2(b_1 i + c_1 j + d_1 k) + \\ (c_1 d_2 - d_1 c_2)i + (d_1 b_2 - b_1 d_2)j + (b_1 c_2 - c_1 b_2)k \end{aligned} \tag{2.14}$$

O conjugado de um *quaternion* é representado por $\bar{q}$, é definido como:

$$\bar{q} = (s, -\vec{v}) \tag{2.15}$$

O inverso de um *quaternion* q é representado por q^{-1} e definido de tal forma que:

$$qq^{-1} = q^{-1}q = 1 \tag{2.16}$$

A magnitude do *quaternion* é obtida através do produto do *quaternion* pelo seu conjugado. Sendo assim:

$$\left|q^2\right| = q\bar{q} = s^2 + \left|v^2\right| \tag{2.17}$$

Além dessa, uma vasta quantidade de propriedades matemáticas dos *quaternions* podem ser encontrados na base obra dos *quaternions* (HAMILTON, 1866).

2.2.3.3 Equivalência entre *quaternion* e matrizes de rotação

Nós podemos representar um *quaternion* em uma de matriz de rotação R em função de um *quaternion* unitário q, que podendo ser definido pela seguinte matriz deduzida em (PINHEIRO, 2006):

$$R = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{2.18}$$

Ainda de acordo com (PINHEIRO, 2006), se R é ortogonal e seu determinante é 1 pode-se concluir que R é uma matriz de rotação. Assim, estabelecendo equivalência com uma matriz de rotação genérica, pode-se determinar a matriz homogênea com números *quaternions* da seguinte forma:

$$R(q)^T L(q) = \begin{bmatrix} 1 & 0^T \\ 0 & R \end{bmatrix} \tag{2.19}$$

2.3 Filtro de Kalman

O filtro de Kalman, corresponde a um método matemático que utiliza estatística preditiva em que o maior peso do filtro é atribuído ao valor de menor incerteza, na qual tem por finalidade remover os ruídos dos sensores com base em estatística preditiva.

De acordo com (COSTA, 2014) o filtro de Kalman tende a ser mais preciso que usar apenas os dados dos sensores, pois ele efetua uma média entre o valor predito e o valor medido pelor sensores, melhorando a estimativa de incerteza. Porem, para que o filtro de Kalman seja aplicado é necessário um modelo matemático do sistema e o conhecimento da variação dos valores medidos pelos sensores. Portanto, quanto melhor o modelo matemático melhor será a próxima previsão de estado.

2.3.1 Espaço de Estado

Sabe-se que um espaço de estados se caracteriza como um conjunto de equações diferenciais de primeira ordem, então de acordo com (COSTA, 2014), podemos fazer essa representação na forma matricial:

$$\begin{bmatrix} \dot{x_1} \\ \vdots \\ \dot{x_n} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} \tag{2.20}$$

ou seja,

$$\dot{x} = Ax + Bu \tag{2.21}$$

em que x representa o vetor de estados que tem as características do sistema, $\dot{x}$ é a derivada do vetor de estados, A corresponde à matriz de transição de estados, B à matriz de entradas e u é o vetor de entrada

De forma análoga, um sistema de equações discretas também pode ser escrito na forma matricial:

$$\begin{bmatrix} \dot{x_1} \\ \vdots \\ \dot{x_n} \end{bmatrix}_{k+1} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}_k + \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}_k \tag{2.22}$$

De tal forma que x_{k+1} é dado por:

$$x_{k+1} = Ax_k + Bu_k \tag{2.23}$$

em que x_k é vetor de estado no instante de tempo discreto atual e x_{k+1} é o vetor de estado no próximo instante de tempo discreto.

Pode-se entender o filtro de Kalman em duas etapas principais: Predição e Atualização/Correcção, detalhadas nas próximas subseções.

2.3.1.1 Predição de Estado e a Matriz de covariância

Para prevermos o estado, deve-se conhecer o modelo do sistema:

$$\bar{X}_k = A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1} \tag{2.24}$$

onde $\bar{X}$ é o vetor de estados predito e $\hat{X}_{k-1}$ o vetor de estados estimado no tempo discreto anterior.

De maneira detalhada, (COSTA, 2014) mostra que a matriz de covariância predita pode ser obtida através:

$$\bar{P}_k = A_{k-1}\hat{P}_{k-1}A_{k-1}^T + Q \tag{2.25}$$

onde $\bar{P}_k$ é a matriz de covariância predita. Os termos na diagonal dessa matriz predizem a variância de cada nível de estado. Os demais termos apresentam a variância predita entre as variáveis de estado, ou seja, o quanto a variação em uma variável influencia sobre as demais. $\hat{P}_{k-1}$ é a matriz de covariância estimada no tempo anterior e Q corresponde a matriz de covariância do ruido de transição de estado, que é constante e modela a variação de ruido existente na transição de estado, devendo ser fornecido.

A partir de então a atualização do vetor de estado e matriz de covariância serão atualizados com base nos valores preditos e nos valores medidos pelos sensores.

2.3.2 Ganho de Kalman

Após a obtenção da matriz $\bar{P}_k$, deve-se calcular a matriz de ganho de Kalman (k_k). Essa matriz define o peso que será dado para o erro de covariância. K_k pode ser obtida da seguinte forma:

$$K_k = \bar{P}_k H^T (H\bar{P}_k H^T + R)^{-1} \tag{2.26}$$

onde K_k é a matriz do ganho de Kalman no tempo atual. H é a matriz de observação que também é constante e entra como parâmetros do modelo e tem o propósito de definir quais variáveis de estado podem ser medidas pelos sensores. R é a matriz de covariância do ruido de observação. Essa matriz é constante e modela a variância de ruido existente nas medidas dos sensores, sua função é similar à de Q.

2.3.3 Cálculo da estimativa e do erro de covariância

Após a obtenção do ganho de Kalman, como demostrado na seção 2.3.2 pode-se estimar o vetor de estado com base no modelo matemático e medidas realizadas.

$$\hat{x}_k = \bar{x}_k + K_k(z_k - H\bar{x}_k) \tag{2.27}$$

onde $\hat{x}_k$ é o vetor de estado estimado e z_k é o vetor de estado medido.

Podemos observar que a estimativa final é uma medida entre o valor observado e o valor predito, ponderada pela matriz do ganho de Kalman. Portando pode-se observar que quanto menor o ganho de Kalman, menor será a influência do valor observado na estimativa e maior será a influência do valor predito na estimativa.

Já a covariância dos estados pode ser obtida por:

$$\hat{P}_k = \bar{P}_k - K_k - H\bar{P}_k \tag{2.28}$$

Mais detalhes das deduções podem ser encontradas em (COSTA, 2014).

3 Metodologia

A ideia principal do trabalho é desenvolver um robô com o propósito de fornecer ao usuário um mapeamento tridimensional de ambientes de interesse. Para tal, um sensor IMU(*Inertial Measurement Units*) provê ao sistema dados de rotação da estrutura em relação a um referencial inercial e um sensor *encoder* rotativo fornece o deslocamento sofrido pela estrutura a fim de garantir os dados de translação. Dessa forma, utilizando uma matriz de transformação homogênea, é possível transformar todos os pontos obtidos pelo sensor LIDAR para esse referencial inercial, no qual será construído o mapa. A fim de fornecer um mapa de baixo custo computacional e em tempo real, utilizou-se o octomap, que permite implementar uma grade de ocupação tridimensional a partir dos dados providos dos sensores.

O desenvolvimento do projeto se baseou em 3 etapas de implementação. Na primeira etapa, os métodos foram testados em um ambiente simulado na plataforma de testes, utilizando o Gazebo (Seção 3.2.1). Na segunda etapa: foi iniciada a implementação do protótipo físico sem a atribuição do movimento de translação (Seção 3.4.3). Por último, os componentes foram dispostos em uma estrutura com rodas garantindo a mobilidade do protótipo e possibilidade de se obter dados do movimento de translação por meio do sensor *encoder* (Seção 3.4.4). A estrutura do projeto, bem como a ordem de tratamento computacional pode ser observada na Figura 6, por meio de um diagrama de blocos que apresenta as principais dados e módulos utilizados demonstrando o fluxo em que o projeto se segue. Um diagrama de blocos explicitando as formas de conexão utilizada entre os dispositivos pode ser observada na Figura 7;

3.1 Componentes

Nesta seção serão apresentados mais detalhadamente os componentes físicos utilizados para o desenvolvimento do protótipo, bem como uma introdução ao princípio básico de funcionamento de cada um deles e suas principais características.

3.1.1 YDLIDAR-X4

O YDLIDAR X4 é um sensor de medição de distância a laser que se tem como base o princípio de triangulação (Seção 3.1.1.1). O sensor foi desenvolvido pela companhia Shenzhen Yuedeng Technology Co., Ltd. (Figura 8).

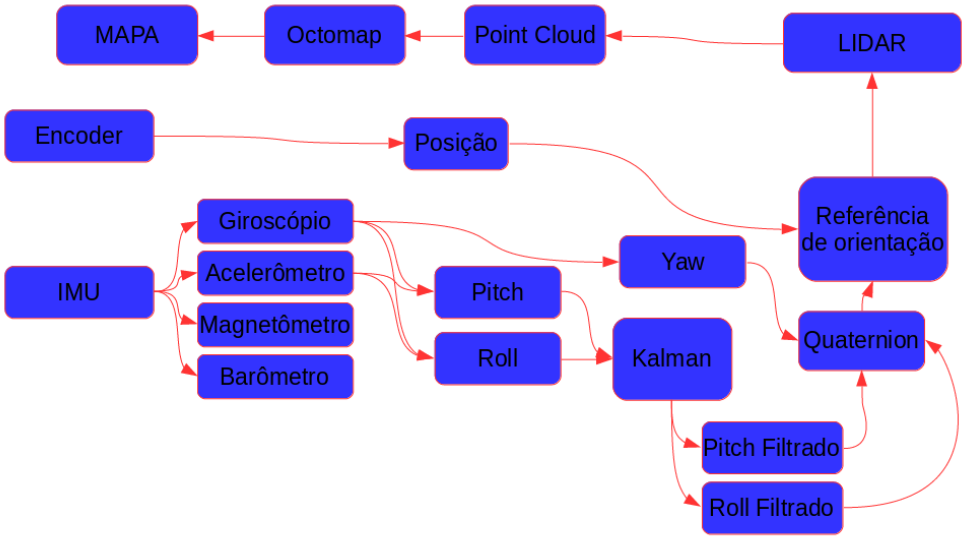


Figura 6: Diagrama de blocos representando a ordem de implementação base do projeto para se obter o mapa 3D.

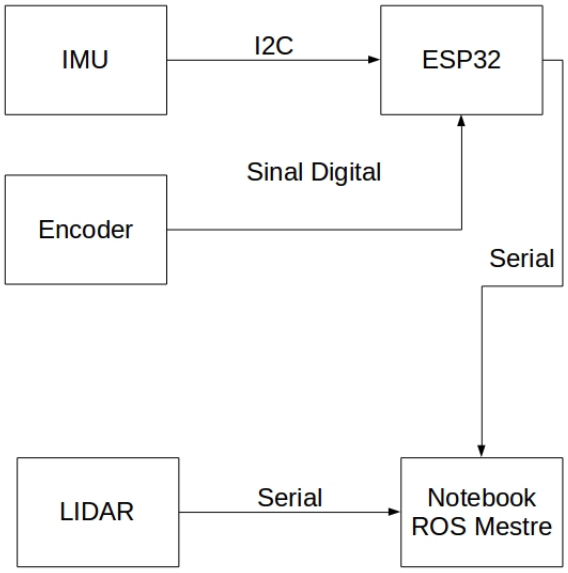


Figura 7: Diagrama de blocos referente às conexões entre os componentes utilizados e suas formas de comunicação.

O YDLIDAR X4 utiliza uma porta serial *Universal Asynchronous Receiver/Transmitter*(UART) para comunicação, faz uma varredura de leitura em 360º apresentando erro de ±1% além permite realizar medidas de 0,12 metro até 10 metros de distância, conforme apresentado na Tabela 2. O motor presente no modelo utilizado é controlado por um sinal *Pulse Width Modulation*(PWM), sendo configurado neste trabalho por meio dos parâmetros do ROS, garantindo frequência de escaneamento 7Hz e taxa de amostragem estabelecida em 5KHz (EAI TECHNOLOGY, 2015).



Figura 8: Sendor LIDAR modelo YDLIDARX4.

Tabela 2: Informações técnicas do sensor LIDAR modelo YDLIDAR X4.

Parâmetros	**Valor**
Scanning range	360°
Range	0.12 - 10m
Range Resolution	<1% da distância
Average resolution	∼0.5°
Scan Frequency	6-12Hz
Range Frequency	5000Hz
Max current	480mA
Laser Wavelenght	775 - 795nm
Laser Power	3 - 5mW

3.1.1.1 Princípio de triangulação

O sensor LIDAR empregado neste trabalho funciona com princípio de triangulação, que é uma das metodologias empregadas para cálculos de medida de distância até o objeto de interesse (EAI TECHNOLOGY, 2015). Para isso, o sensor emite um feixe LASER que é refletido e captado por um sensor fotossensível presente no equipamento. Dessa forma, quando o objeto se aproxima ou distancia do emissor, o feixe de luz refletido toca o sensor fotossensível em outra posição, onde, pelo princípio de triangulação geométrica é estimada a posição do objeto (Dehong et al., 2017).

Com base na Figura 9, pode-se deduzir que o ângulo de incidência na lente pode ser descrito por:

$$\theta = \arctan\left(\frac{h}{d}\right) \tag{3.1}$$

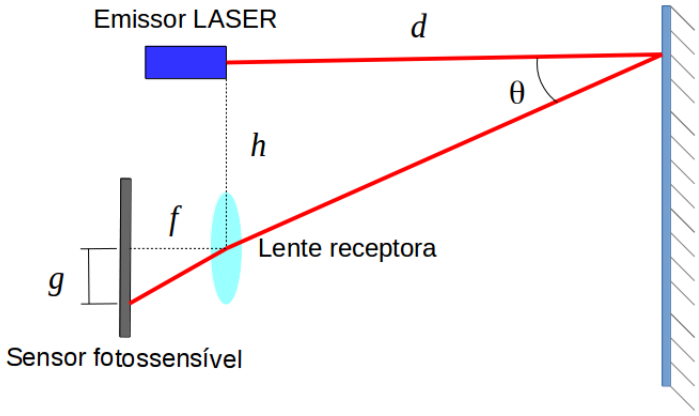


Figura 9: Representação do funcionamento de um sensor LIDAR com base no princípio de triangulação.

em que θ é o ângulo formado entre o feixe LASER emitido e o recebido, h é distância entre o LASER emissor e o receptor fotossensível e d é a distância de feixe emitido até o objeto de interesse.

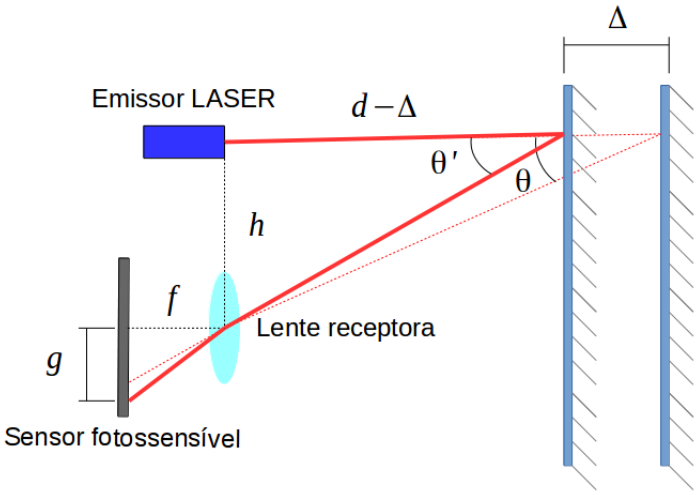


Figura 10: Comportamento do funcionamento de um sensor LIDAR com base no principio de triangulação com aproximação do objeto.

Seja Δ a variação da distância entre o sensor LIDAR e o objeto de interesse, então, de forma análoga à Equação 9 temos que:

$$\theta' = \arctan\left(\frac{h}{d-\Delta}\right) \tag{3.2}$$

$$\theta'' = \arctan\left(\frac{h}{d+\Delta}\right) \tag{3.3}$$

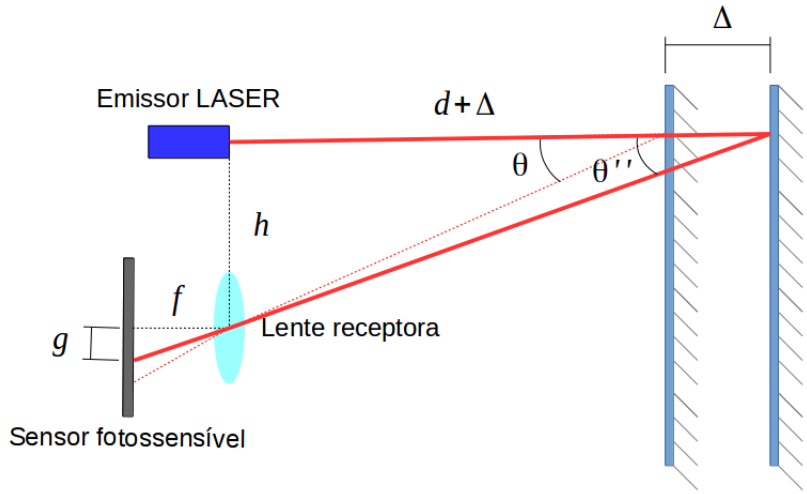


Figura 11: Comportamento do funcionamento de um sensor LIDAR com base no principio de triangulação com o afastamento do objeto.

Com base nas expressões acima, pode-se provar que o ângulo de incidência no receptor varia com relação à distância entre o emissor e o objeto. Ou seja, quanto maior a distância do sensor ao objeto menor o ângulo de incidência:

$$\theta'' < \theta < \theta' \tag{3.4}$$

A posição em que o feixe de luz tocará no sensor fotossensível, considerando a lente é dada por:

$$\left(\frac{h}{d}\right) = \left(\frac{f}{g}\right) \tag{3.5}$$

em que f é o foco da lente e g é a posição em que o feixe toca o sensor, obtendo-se:

$$g = \left(\frac{fd}{h}\right) \tag{3.6}$$

3.1.1.2 Tempo de vôo (ToF)

Outro método empregado para cálculos de distância em sensores LIDAR é o TOF, que estima a distância por meio do tempo que o feixe LASER emitido demora para retornar ao sensor receptor.

Considerando que o ângulo entre o Pulso LASER e o detector é um valor muito próximo de 0º, e sabendo que:

$$d = v\Delta t \tag{3.7}$$

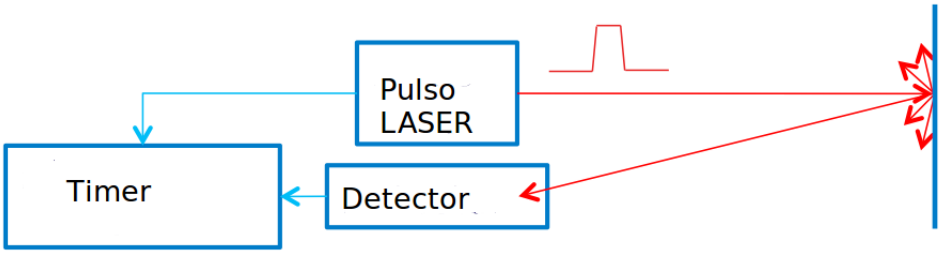


Figura 12: Representação do funcionamento de um sensor LIDAR com base no princípio do tempo de vôo.

Tabela 3: Comparação técnica entre ESP32 e Arduino UNO R3.

	ESP32	Arduino UNO R3
Núcleos	2	1
Arquitetura	32 bits	8 bits
Clock	160MHz	16MHz
WiFi	Sim	Não
Bluetooth	Sim	Não
RAM	512KB	2KB
Flash	4Mb	32KB
GPIO	36	14
Interfaces	SPI/I2C/UART/I2S/CAN	SPI/I2C/UART
ADC	18	6
DAC	2	0

em que d é a distância percorrida, v é a velocidade do objeto e Δt é a variação do tempo no instante desejado. Em que v representa a velocidade da luz, pode-se estimar a distância por:

$$d = \frac{c\Delta t}{2} \tag{3.8}$$

3.1.2 Placa de desenvolvimento - ESP32

Inicialmente, durante a etapa de implementação, foi utilizado o Arduino UNO R3, uma placa baseada no microcontrolador ATmega328, porém dado o elevado consumo computacional, para o microcontrolador, o uso dessa plataforma ficou inviável, gerando erros de sincronização com o ROS. Dessa forma escolheu-se o ESP32 para leitura e tratamento dos dados dos sensores IMU e *encoder*, pois o mesmo apresenta uma maior robustez computacional como apresentado na Tabela 3.

A placa de desenvolvimento ESP32 emprega um microprocessador Tensilica Xtensa LX6 dual-core, inclui uma antena integrada, RF tipo balun, amplificador de potência, receptor de baixo ruído amplificado, filtros e gerenciamento de energia dos módulos. O



Figura 13: Placa de desenvolvimento ESP32.

Tabela 4: Informações técnicas da placa de desenvolvimento ESP32.

Parâmetros	**Valor**
Processador	LX6 32-bit Dual-core, operando em 2-240 MHz
FLASH	4MB
RAM	520 kB
GPIO	34, com 3.3V e 12mA
ADC	18, com resolução de 12 bits
Wi-Fi	2,4 GHz, 802.11 b/g/n
Bluetooth	Bluetooth Low Energy v4.2 (BLE)

ESP32 foi criado e desenvolvido por Espressif Sistemas, uma empresa Chinesa e algumas de suas especificações técnicas mais relevantes são apresentados na Tabela 4. Mais detalhes podem ser encontrados em seu manual (ESPRESSIF SYSTEMS, 2018).

3.1.3 Unidade Inercial de Medida(IMU)

Neste trabalho, foi utilizado o GY-87, um IMU com dez graus de liberdade cujo principal propósito da sua utilização no projeto é fornecer os dados de rotação em relação ao referencial inercial, bem como permitir ao usuário a utilização simultânea de seus diversos sensores para realização da fusão sensorial. Esse sensor IMU é composto pelos sensores: MPU-6050 (acelerômetro e giroscópio), HMC5883L (magnetômetro) e o BMP180 (barômetro). Sua comunicação é feita diretamente com o ESP-32 por meio do barramento de comunicação serial pelo protocolo I2C.

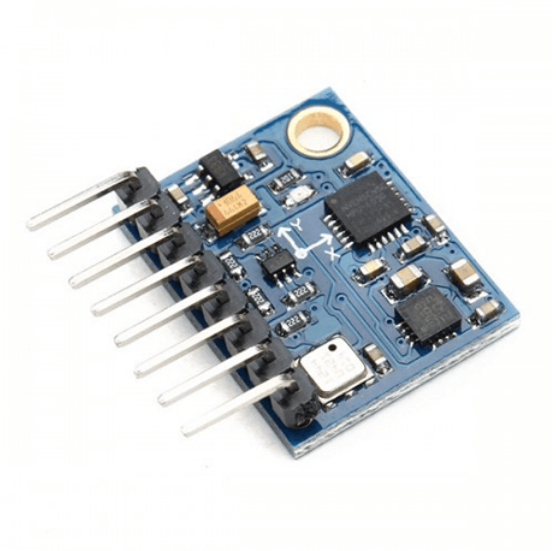


Figura 14: IMU modelo GY-87.

3.1.3.1 Acelerômetro e Giroscópio MPU-6050

O MPU6050 é um chip MEMS (superfície de elementos micro-usinados) capacitivo, que contém um sensor acelerômetro com três graus de liberdade e giroscópio também com três graus de liberdade, garantindo ao todo 6 graus de liberdade ao sistema como visualizado na Figura 15.

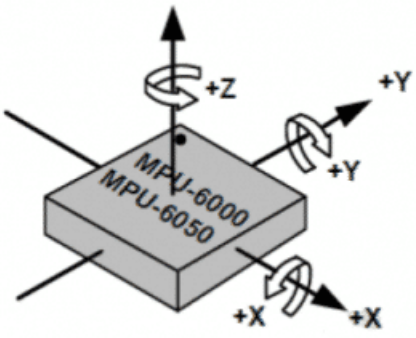


Figura 15: Representação dos graus de liberdade oferecidos ao sistema pelo MPU6050.

Esse tipo de acelerômetro é formado por vigas moveis, também associada ao nome "massas moveis"e placas capacitivas fixas também chamadas como eletrodos fixos. Quando essas vigas moveis são submetidas a uma aceleração ocorre a variação da distância entre

elas, ocasionando a variação da capacitância medida pelas placas capacitivas do sistema, sendo calculada por (VIEIRA; JUNIOR, 2013):

$$C = \frac{\epsilon_0 \epsilon_r A}{D} \tag{3.9}$$

onde ϵ_0 é a constante de permissividade no vácuo, ϵ_r é a permissividade relativa entre as placas, A é a areá de sobreposição entre as placas capacitivas fixas e a viga móvel e D a distância entre a placa capacitiva e a viga móvel.

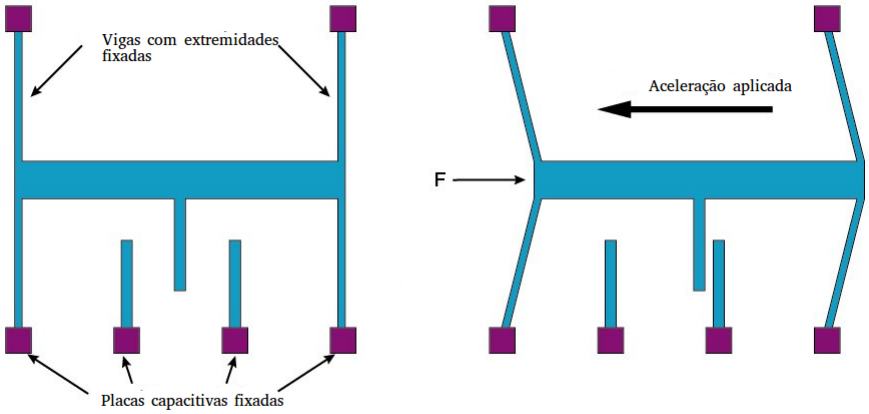


Figura 16: Princípio de funcionamento acelerômetro MEMS. Fonte: Silicon Far East

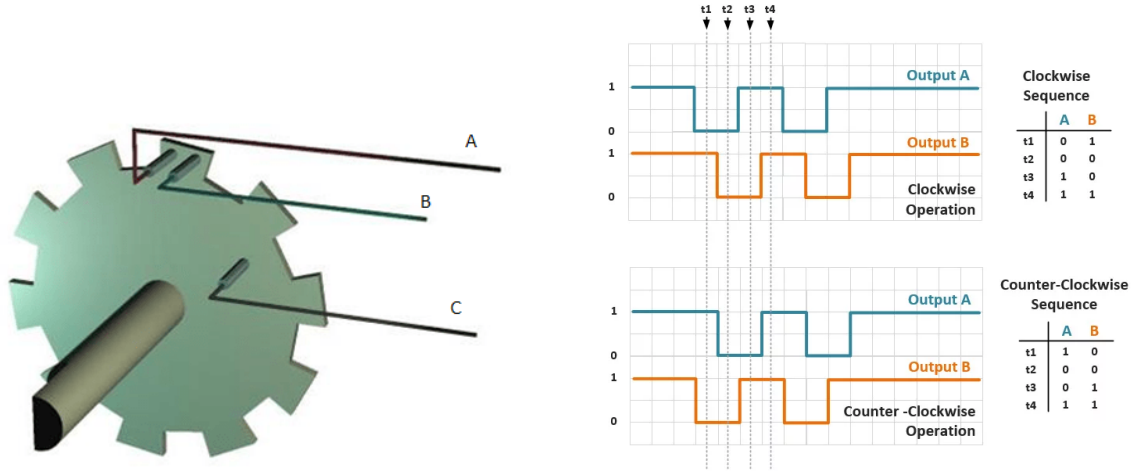
De acordo com (FUX, 2008), normalmente, ao se usar um acelerômetro, o principal objetivo do desenvolvedor é extrair a aceleração linear da gravidade. Em (NETO, 2004) é descrito como uma força de atração de corpos celestes, e que no caso, a gravidade da Terra é de aproximadamente $9,81 m/s^2$, denotada também pela letra g.

Um dos principais problemas encontrados para se trabalhar com essa aceleração linear é o fato desses sensores possuírem uma precisão baixa, gerada pelo excesso de ruídos. Além disso, por possuir seus eixos diferentes dos eixos relativos da terra, sendo extremamente difícil estimar com precisão a intensidade da força gravitacional em cada um dos eixos (MENEGUZZI, 2008).

Esse chip é dotado de um giroscópio MEMS, como se pode ver em (FORHAN, 2010), o princípio de funcionamento do giroscópio MEMS é baseado na transferência de energia entre uma massa e um movimento de rotação, na qual, de forma análoga ao acelerômetro MEMS, a massa se move em relação a um referencial fixo.

3.1.4 Encoder Rotativo KY-040

O *encoder* usado no projeto possui um total de 20 passos por revolução, que equivale a uma variação de 18º em torno do seu eixo, sendo utilizado para fornecer ao usuário o deslocamento total de translação.



(a) Esquematização do funcionamento interno de um *encoder* rotativo. Fonte: https://bit.ly/2J8YclL.

(b) Sequência de sinais gerados através da rotação do *encoder*. Fonte: https://bit.ly/2J8YclL.

Figura 17: Estrutura móvel com sensores acoplados para efetuar o mapeamento tridimensional.

O seu funcionamento, como explicitado em (HANDSON TECHNOLOGY, 2018) e demostrado na Figura 17, consiste em uma placa metálica condutora com conectores fixos, denominados A, B e C. Quando o eixo do *encoder* sofre uma rotação é gerada uma variação de sinais nos seus terminais, como apresentado na Figura 17b. E essa sequência de variação de sinal é utilizada para determinar em qual sentido esse movimento foi efetuado.

3.2 Etapa Simulada

3.2.1 Ambiente virtual Gazebo

Para se efetuar a simulação do projeto, foi o utilizado o ROS Kinetc, em um sistema operacional Ubuntu 16.04. Um robô simples foi modelado composto por duas rodas, um chassi, um sensor LIDAR disposto em posição vertical no intuito de efetuar a leitura dos obstáculos gerados no ambiente simulado como é possível observar na Figura 18. A ideia principal é que quando aplicado o movimento de translação a ele, fosse possível mapear o ambiente através da variação da matriz homogênea e dos valores registrados pelo sensor LIDAR.

Para que fosse possível testar a funcionalidade da simulação foram adicionados alguns sólidos geométricos ao ambiente, como na Figura 20, a fim de simular obstáculos que poderiam ser encontrados em ambiente real. O intuito é que assim que esses objetos entrarem no campo de leitura do sensor LIDAR seja gerado uma nuvem de pontos. Assim seria possível visualizar esses objetos em um mapa tridimensional, comprovando a efetividade do sensor LIDAR e os métodos apresentados.

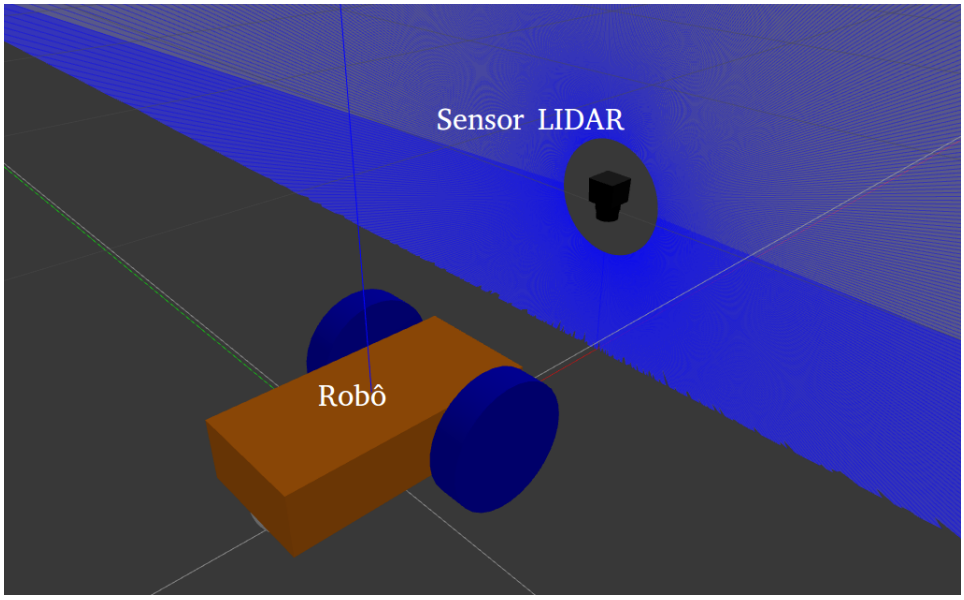


Figura 18: Modelo robótico utilizado no ambiente simulado pelo Gazebo.

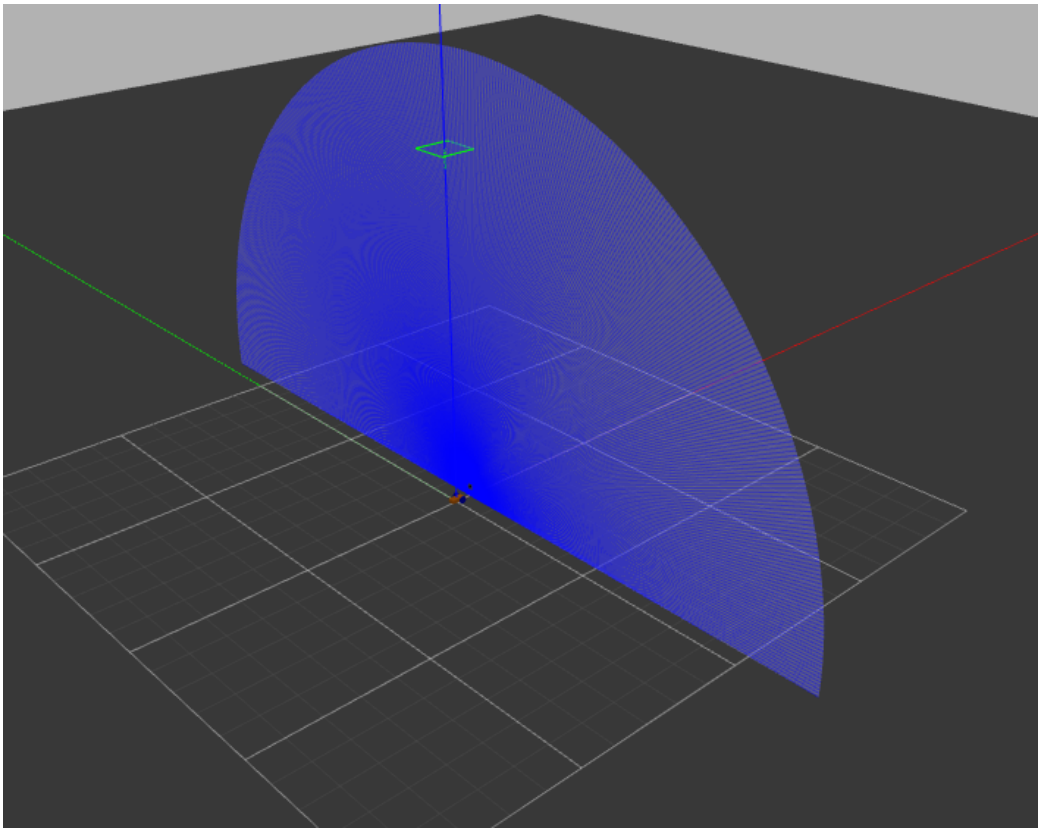


Figura 19: Ambiente simulado com o modelo robótico já implementando, mas sem a presença de obstáculos.

3.3 Etapa Real

Para implementar um sistema real, que obtivesse um resultado semelhante à simulação, foi utilizado o sensor IMU GY-87 conectado ao ESP-32, permitindo estimar a posição do objeto e, assim, gerar uma matriz homogênea referente à posição real.

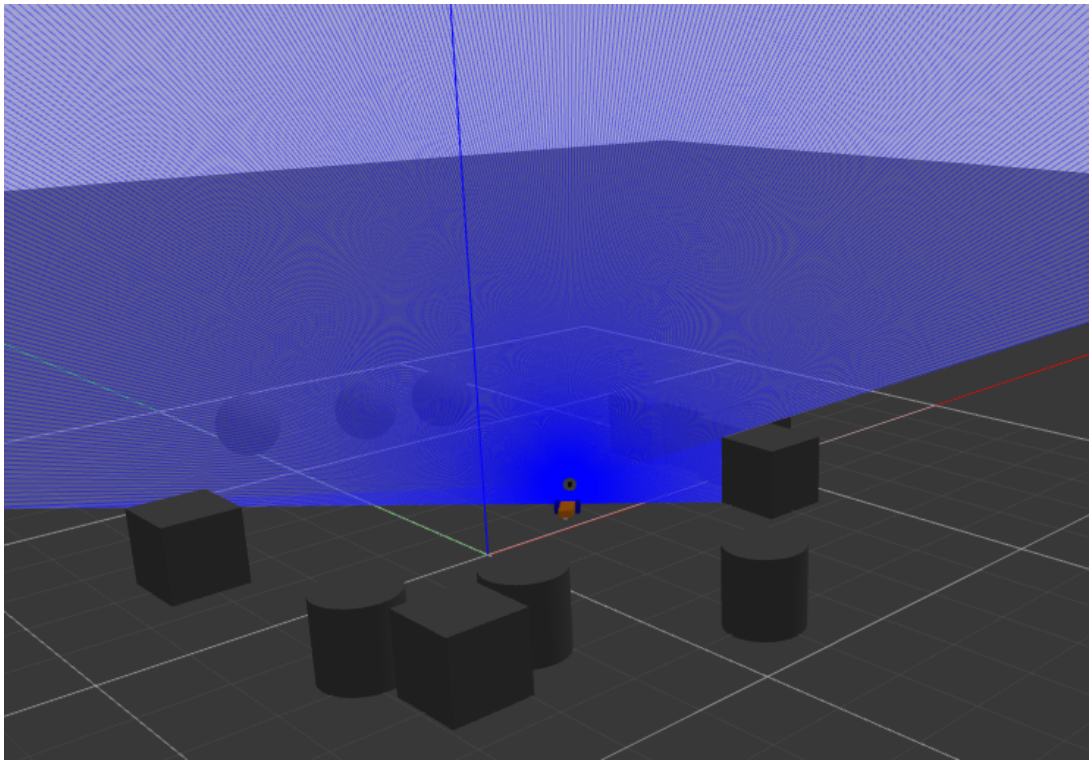


Figura 20: Robô em um ambiente simulado na presença de obstáculos.

Para a comunicação entre o ESP-32 e o computador foi utilizado as funções do pacote Rosserial, que permite a interação do ROS por meio da porta USB por comunicação serial.

3.3.1 Parâmetros de configuração LIDAR

Para integração do YDLIDAR-X4 ao ROS foi utilizado o *driver* disponibilizado pela EAI no seguinte link: https://github.com/EAIBOT/ydlidar. Os parâmetros foram configurados de forma a atender as especificações do projeto, de modo que o sensor efetuasse uma varredura em 360º, em uma frequência de rotação de 7Hz e amostragem de 5kHz.

3.3.2 Obtenção *Roll*, *Pitch* e *Yaw* pelo Giroscópio

A obtenção da rotação por meio dos dados gerados pelo giroscópio, foi efetuado por meio de uma simples integração das medidas geradas em cada eixo de rotação, implementada no ESP32 na forma discreta.

$$r = \int G_x dt \tag{3.10}$$

$$p = \int G_y dt \tag{3.11}$$

$$y = \int G_z dt \tag{3.12}$$

sendo G_x, G_y e G_z as velocidades angulares obtidas pelo giroscópio para os eixos X, Y e Z, respectivamente.

3.3.3 Obtenção de *Roll* e *Pitch* pelo Acelerômetro

A variação da rotação dos eixos de *Pitch* e *Roll*, proveniente da leitura do sensor acelerômetro, pode ser descrita da seguinte forma (PEDLEY, 2013):

$$Roll = arctan\left(\frac{A_y}{A_z}\right) \tag{3.13}$$

$$Pitch = arctan\left(\frac{-A_x}{\sqrt{A_y^2 + A_z^2}}\right) \tag{3.14}$$

em que A_x é a aceleração no eixo X, A_y é a aceleração no eixo Y e A_z é a aceleração no eixo Z.

Porém devido à ausência da força gravitacional no eixo horizontal, não é possível estimar a rotação em torno do eixo Z se fazendo dos dados obtidos pelo acelerômetro (Yaw). Assim, a implementação da aquisição de dados feita no microcontrolador foi construída da forma como se segue:

Algorithm 1 Obter rotação em Ângulos de Euler

$accX, accY, accZ \leftarrow$ Dados do Acelerômetro
$gyroX, gyroY, gyroZ \leftarrow$ Dados do Giroscópio

$Roll \leftarrow atan(accY, accZ)$
$Pitch \leftarrow atan(-accX/sqrt(accY * accY + accZ * accZ))$
$Yaw \leftarrow \sum gyroZ$

$Roll \leftarrow Kalman(Roll)$
$Pitch \leftarrow Kalman(Pitch)$
return $Roll, Pitch, Yaw$

3.3.4 Obtenção do deslocamento

De maneira semelhante, para se obter a velocidade pelos dados do sensor IMU basta fazer a integração dos dados. Tem-se assim:

$$V_x = \int A_x dt \tag{3.15}$$

$$V_y = \int A_y dt \tag{3.16}$$

$$V_z = \int A_z dt \tag{3.17}$$

Já para se obter o deslocamento em cada um dos eixos do plano cartesiano pode-se novamente integrar:

$$D_x = \int V_x dt \tag{3.18}$$

$$D_y = \int V_y dt \tag{3.19}$$

$$D_z = \int V_z dt \tag{3.20}$$

em que A_x é a aceleração no eixo X, A_y é a aceleração no eixo Y, A_z é a aceleração no eixo Z, V_x é a velocidade no eixo X, V_y é a velocidade no eixo Y, V_z é a velocidade no eixo Z, D_x é o deslocamento no eixo X, D_y é o deslocamento no eixo Y e D_z é o deslocamento no eixo Z.

Pelo fato dos acelerômetros serem imprecisos e possuir ruídos, ao utilizar com a integração dos valores para se obter o deslocamento, ocorre uma integração dos ruídos, acumulando o erro nos dados. Assim, em questão de pouco tempo, obtém-se um deslocamento com valor completamente irreal. (CRESTANI, 2015; FUX, 2008; MENEGUZZI, 2008)

Por conta desse acúmulo de erro, não foi possível estimar o deslocamento utilizando diretamente o acelerômetro. Então como método alternativo foi adicionado um *encoder* rotativo fixado no eixo de uma das rodas traseiras da estrutura, obtendo o deslocamento por meio da quantidade de rotações sofridas pela roda da estrutura.

O *encoder* usado apresenta 20 passos, podendo-se estimar a distância percorrida em cada passo por:

$$d_p = \frac{D\pi}{N} \tag{3.21}$$

sendo d_p a distância deslocada em cada passo do *encoder* rotativo, D o diâmetro da roda em metros, e N a quantidade de passos do *encoder* rotativo. Para esse projeto tem-se que:

$$d_p = \frac{0,118\pi}{20} \cong 0,0185m \tag{3.22}$$

Ou seja, cada passo do *enconder* representa um deslocamento de 0.0185m. A partir disso é feito uma somatória da quantidade de rotações que o *encoder* registrou, sendo possível estimar o deslocamento total da estrutura.

3.3.5 Transformação Euler - *Quaternion*

A conversão de Euler para *quaternion* se faz por meio da multiplicação de *quaternions*: cada ângulo Euler é convertido em um par eixo-ângulo. Esses três pares são convertidos para *quaternions* e, finalmente multiplicados, para que o resultado final represente o *quaternion* desejado (PINHEIRO, 2006).

Por exemplo, pela conversão zyx (*Yaw*,*Pitch*,*Roll*), obtém-se:

$$q_x = \cos\frac{\alpha}{2} + \sin\frac{\alpha}{2}i \tag{3.23}$$

$$q_y = \cos\frac{\beta}{2} + \sin\frac{\beta}{2}j \tag{3.24}$$

$$q_z = \cos\frac{\gamma}{2} + \sin\frac{\gamma}{2}k \tag{3.25}$$

em que a rotação em torno do eixo Z e representado pelo ângulo γ, a rotação em torno do eixo Y é representado pelo ângulo β e a rotação em torno do eixo X é representado pelo ângulo α.

Ao se multiplicar esses *quaternion*, obtém-se o *quaternion* desejado, $q = q_z q_y q_x$, da seguinte forma:

$$q_0 = \cos\frac{\alpha}{2}\cos\frac{\beta}{2}\cos\frac{\gamma}{2} + \sin\frac{\alpha}{2}\sin\frac{\beta}{2}\sin\frac{\gamma}{2} \tag{3.26}$$

$$q_1 = \sin\frac{\alpha}{2}\cos\frac{\beta}{2}\cos\frac{\gamma}{2} - \cos\frac{\alpha}{2}\sin\frac{\beta}{2}\sin\frac{\gamma}{2} \tag{3.27}$$

$$q_2 = \cos\frac{\alpha}{2}\sin\frac{\beta}{2}\cos\frac{\gamma}{2} + \sin\frac{\alpha}{2}\cos\frac{\beta}{2}\sin\frac{\gamma}{2} \tag{3.28}$$

$$q_3 = \cos\frac{\alpha}{2}\cos\frac{\beta}{2}\sin\frac{\gamma}{2} - \sin\frac{\alpha}{2}\sin\frac{\beta}{2}\cos\frac{\gamma}{2} \tag{3.29}$$

Também podendo ser representado na forma de matriz como:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\alpha}{2}\cos\frac{\beta}{2}\cos\frac{\gamma}{2} + \sin\frac{\alpha}{2}\sin\frac{\beta}{2}\sin\frac{\gamma}{2} \\ \sin\frac{\alpha}{2}\cos\frac{\beta}{2}\cos\frac{\gamma}{2} - \cos\frac{\alpha}{2}\sin\frac{\beta}{2}\sin\frac{\gamma}{2} \\ \cos\frac{\alpha}{2}\sin\frac{\beta}{2}\cos\frac{\gamma}{2} + \sin\frac{\alpha}{2}\cos\frac{\beta}{2}\sin\frac{\gamma}{2} \\ \cos\frac{\alpha}{2}\cos\frac{\beta}{2}\sin\frac{\gamma}{2} - \sin\frac{\alpha}{2}\sin\frac{\beta}{2}\cos\frac{\gamma}{2} \end{bmatrix} \tag{3.30}$$

Desta forma, foi adotado o seguinte algorítimo para a transformação do ângulos de Euler em *Quaternion*.

Algorithm 2 Transformar Euler em *Quaternion*

cy $= cos(Yaw \cdot 0.5)$
sy $= sin(Yaw \cdot 0.5)$
cr $= cos(Roll \cdot 0.5)$
sr $= sin(Roll \cdot 0.5)$
cp $= -cos(Pitch \cdot 0.5)$
sp $= -sin(Pitch \cdot 0.5)$

$q[0] = sy \cdot cr \cdot cp - cy \cdot sr \cdot sp$
$q[1] = cy \cdot cr \cdot sp + sy \cdot sr \cdot cp$
$q[2] = cy \cdot sr \cdot cp - sy \cdot cr \cdot sp$
$q[3] = cy \cdot cr \cdot cp + sy \cdot sr \cdot sp$
return float $q[4]$

3.4 Efetividade do filtro de Kalman

Para a simplificação do trabalho foi utilizado a biblioteca "Kalman.h", onde o filtro de Kalman foi modelado para atender as especificações de um sensor IMU genérico. Maiores informações, incluindo todas as etapas da modelagem do filtro de Kalman, para esse caso em especifico, pode ser encontrada em (TKJELECTRONICS, 1992).

3.4.1 Efetividade do filtro de Kalman em situação estacionária

Primeiramente, para a verificação da efetividade de filtro de Kalman no tratamento dos dados do IMU em posição estática, o sensor IMU foi mantido em posição inercial, enquanto os dados obtidos foram sendo registrados. A partir disso, foram gerados gráficos representando as rotações *Roll*, *Pitch* e *Yaw*, que permitem comparar os resultados com a utilização do filtro de Kalman e sem sua utilização.

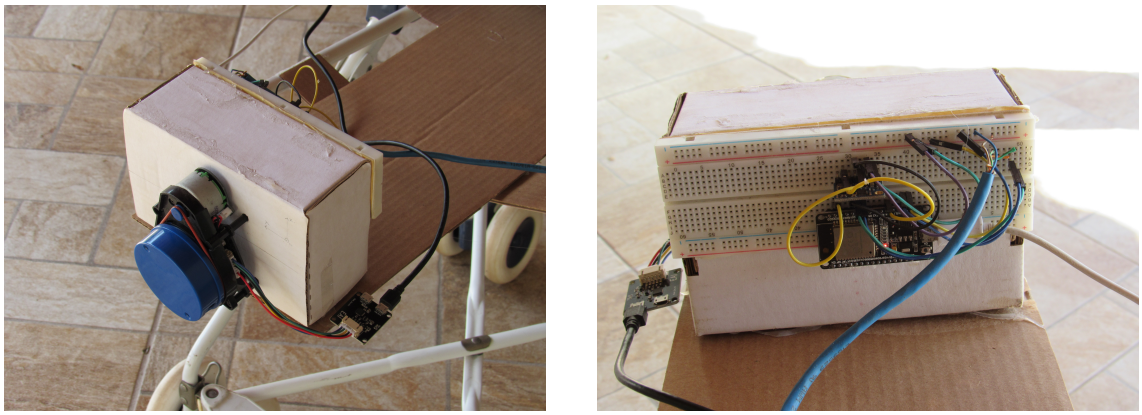
3.4.2 Efetividade do filtro de Kalman para movimentos gerados aleatoriamente

Para analisar a efetividade do filtro de Kalman durante a movimentação aleatória do IMU, o sensor foi mantido por um curto período de tempo em posição estacionária e, então, foi aplicada uma força aleatória a fim gerar um movimento de rotação em todos os eixos, por fim, o sensor foi novamente disposto em sua posição inicial. Para garantir que o sensor permanecesse exatamente no ponto inicial foi utilizado a quina de uma estrutura de construção civil, pois a mesma não sofre nenhuma variação física considerável em curto período de tempo a ponto de gerar variações nos valores do sensor.

A partir dos dados registrados foram gerados gráficos que representam as rotações em *Roll*, *Pitch* e *Yaw*, permitindo comparar os resultados com a utilização do filtro de Kalman e sem sua utilização.

3.4.3 Mapa gerado apenas pela rotação

Os métodos propostos foram testados, inicialmente, apenas considerando o movimento de rotação. Para isso foi montado um paralelepípedo, unindo o sensor IMU ao sensor LIDAR em um mesmo referencial, como mostrado na Figura 21. Dessa forma ambos sensores sofrem forças e movimentos equivalentes obtendo, assim, dados de rotação coerentes aos movimentos realizados no sensor LIDAR.



(a) Vista frontal. (b) Vista posterior.

Figura 21: Sensores acoplados em um mesmo referencial.

A partir disso, o ESP32 e o sensor LIDAR foram conectados a um computador com o ROS, a fim de gerar uma nuvem de pontos e, então, gerar o mapa tridimensional.

3.4.4 Localização e Mapeamento Simultâneos

Para que fosse possível obter um dado de deslocamento coerente como a movimentação real, os componentes do projeto foram dispostos em uma estrutura com quatro pares de rodas apresentado na Figura 22, sendo que em uma das rodas traseiras um *encoder*

foi fixado ao eixo da mesma (23). O sensor LIDAR foi disposto para efetuar a leitura em posição vertical, assim como foi feito na etapa de simulação, gerando uma nuvem de pontos com base no movimento de translação do protótipo.



(a) (b)

Figura 22: Estrutura móvel com sensores acoplados para efetuar o mapeamento tridimencional.

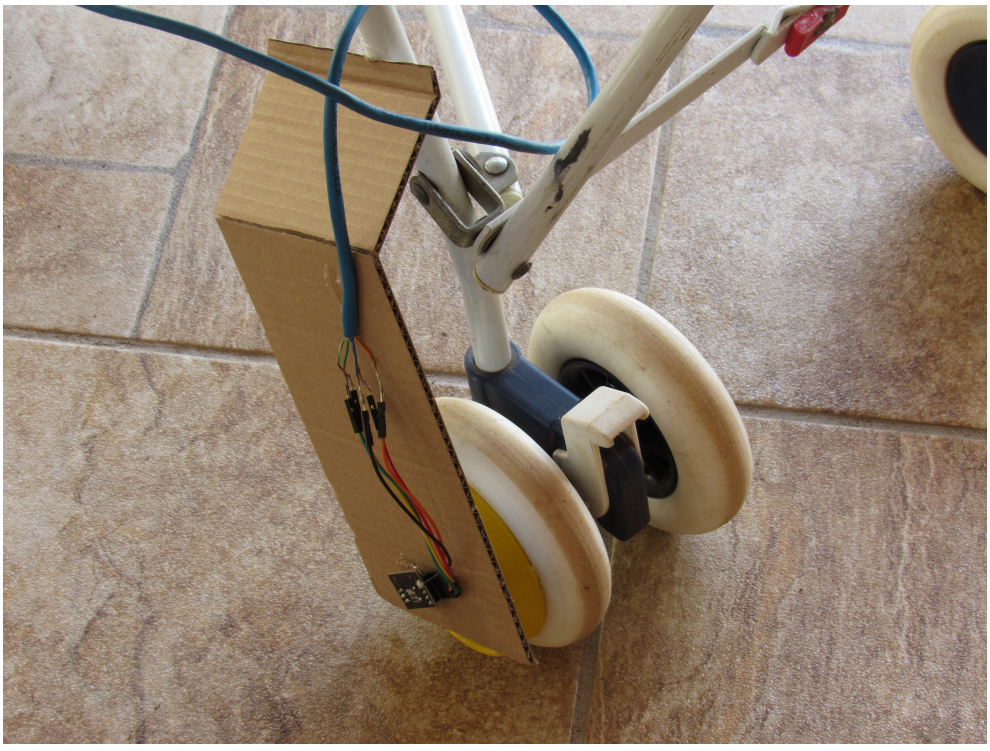


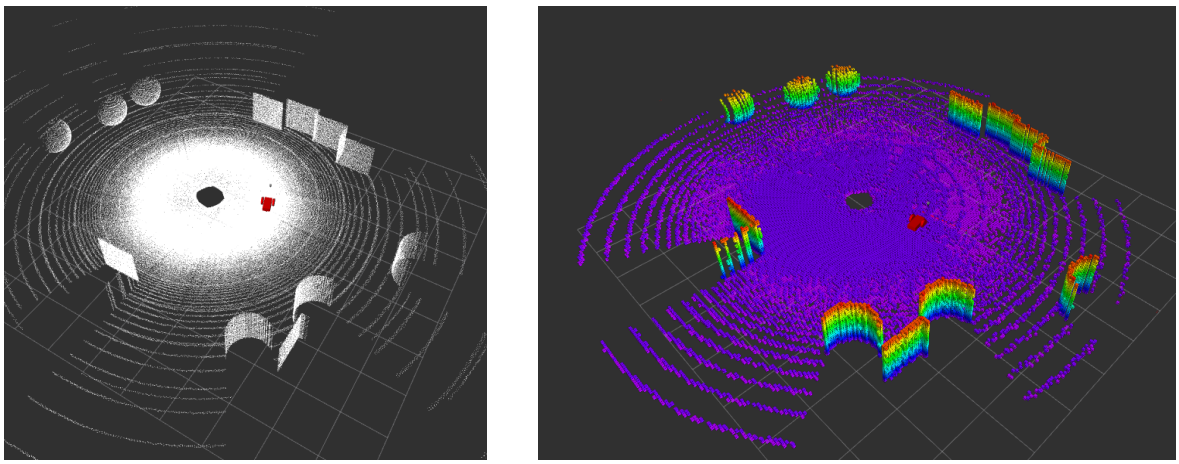
Figura 23: *Encoder* fixado à roda da estrutura móvel.

4 Resultados

Este capítulo apresenta os resultados obtidos do mapeamento tanto em ambiente simulado proposto na seção 3.2.1, quanto em ambiente real utilizando as metodologias proposta na seção 3.4.3 e seção 3.4.4, bem como o resultado do tratamento de dados do IMU utilizando o filtro de Kalman.

4.1 Etapa Simulada

Na etapa simulada, os métodos apresentados se mostrar eficientes, principalmente, pelo fato dos dados serem gerados por softwares, ou seja, sem a presença de ruídos, o que garantiu uma extrema precisão do posicionamento do robô e do mapeamento.



(a) (b)

Figura 24: (a) Nuvem de pontos registrada pelo RViz, após o robô efetuar uma varredura do ambiente simulado. (b) Mapa desse mesmo ambiente gerado pela grade de ocupação tridimensional Octomap, em que as cores representam a altura dos dados com base no referencial.

Nessa etapa, foi feito o mapeamento tridimensional do ambiente da Figura 20, a partir de uma nuvem de pontos representada na Figura 24a gerando um o mapa da Figura 24b. Pode-se observar que o resultado foi coerente com o ambiente simulado, comprovando o funcionamento dos métodos aplicados até o momento e permitindo o início da criação do protótipo para aplicação dos métodos em ambiente real.

4.2 Aplicação do filtro de Kalman

4.2.1 Efetividade do filtro de Kalman em situação estacionária

4.2.1.1 *Roll*

Após obter dados, conforme o capitulo anterior, por meio de uma simples análise gráfica da Figura 25, para o estado estacionário, observa-se que o filtro de Kalman reduz o ruído do sensor, tornando os dados de rotação em torno do eixo X (*Roll*) muito mais coerentes com o realidade.

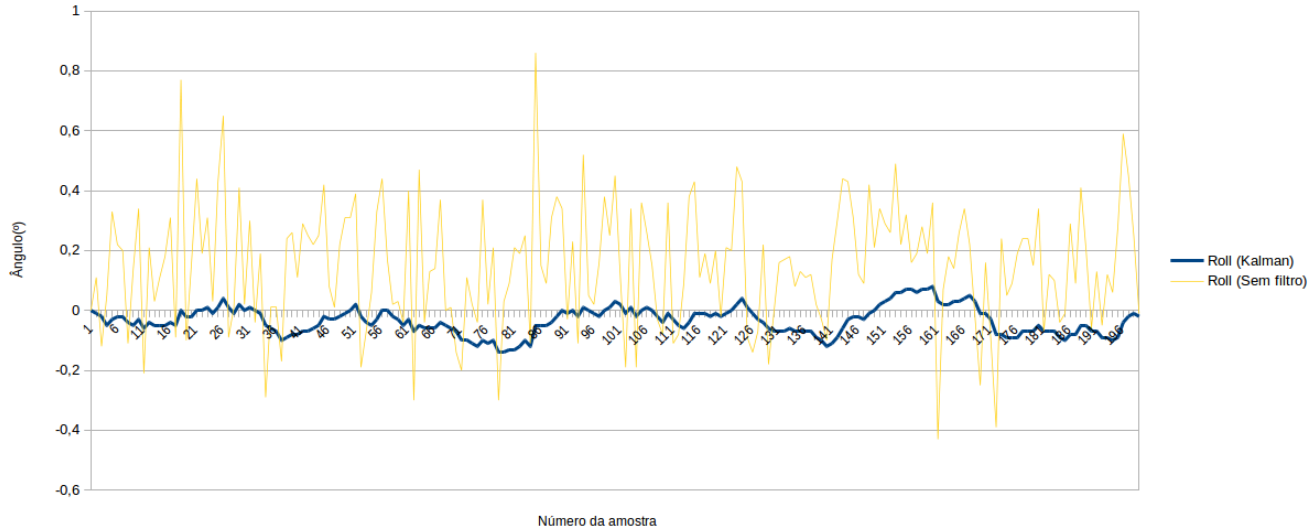


Figura 25: Dados da rotação em torno do eixo X (*Roll*) obtidos pelo sensor IMU enquanto mantido em posição inercial.

4.2.1.2 *Pitch*

Semelhante ao resultado obtido para a rotação em torno do eixo X (*Roll*), analisando a Figura 26, para estado estacionário, pode-se concluir que o filtro de Kalman também reduz significativamente de rotação em torno do eixo Y (*Pitch*).

4.2.2 Efetividade do filtro de Kalman para movimentos gerados aleatoriamente

4.2.2.1 *Roll*

Pela Figura 27, pode-se concluir que o filtro de Kalman faz com que os dados de rotação em torno do eixo X (*Roll*) apresente menos variações aleatórias, de forma com que ele consiga acompanhar as rotações geradas, filtrando os ruídos de alta frequência. É interessante notar que no final do teste, obteve-se um ângulo equivalente ao inicial.

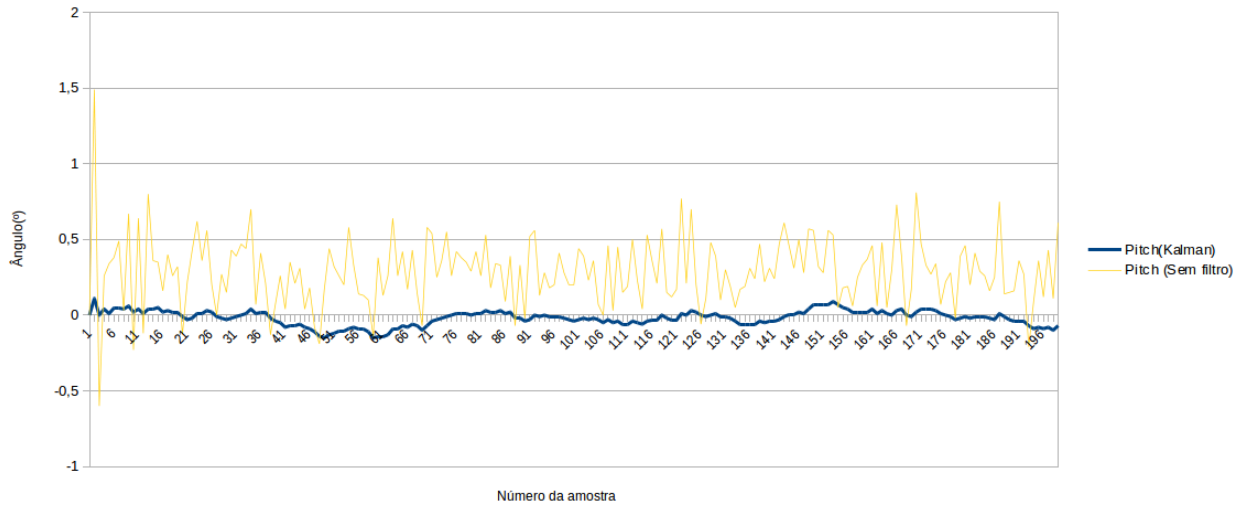


Figura 26: Dados da rotação em torno do eixo Y (*Pitch*) obtidos pelo sensor IMU enquanto mantido em posição inercial.

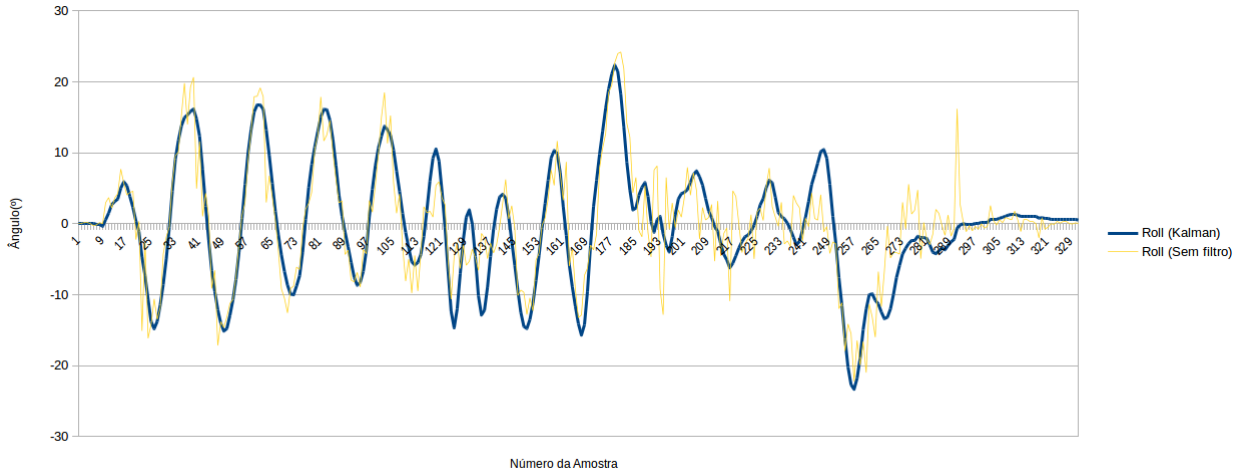


Figura 27: Dados da rotação em torno do eixo X (*Roll*) obtidos pelo sensor IMU quando gerado uma movimentação aleatória no sensor.

4.2.2.2 *Pitch*

De forma análoga ao movimento de rotação em torno do eixo X (*Roll*) e analisando a Figura 28, conclui-se que o filtro de Kalman fez com que os dados de rotação em torno do eixo Y (*Pitch*) apresentasse menos variações aleatórias, de forma com que ele consiga acompanhar as rotações geradas, eliminando o ruído presente no sinal. Nota-se que no final do teste, obteve-se, também, um ângulo equivalente ao inicial.

4.2.2.3 *Yaw*

Pelo fato de não ter sido possível a utilização do filtro de Kalman para aperfeiçoamento do valor de rotação em torno do eixo Z (*Yaw*), na Figura 29 pode-se notar que esse apresentou ruídos que não foram possíveis de serem tratados pelos métodos apresentados,

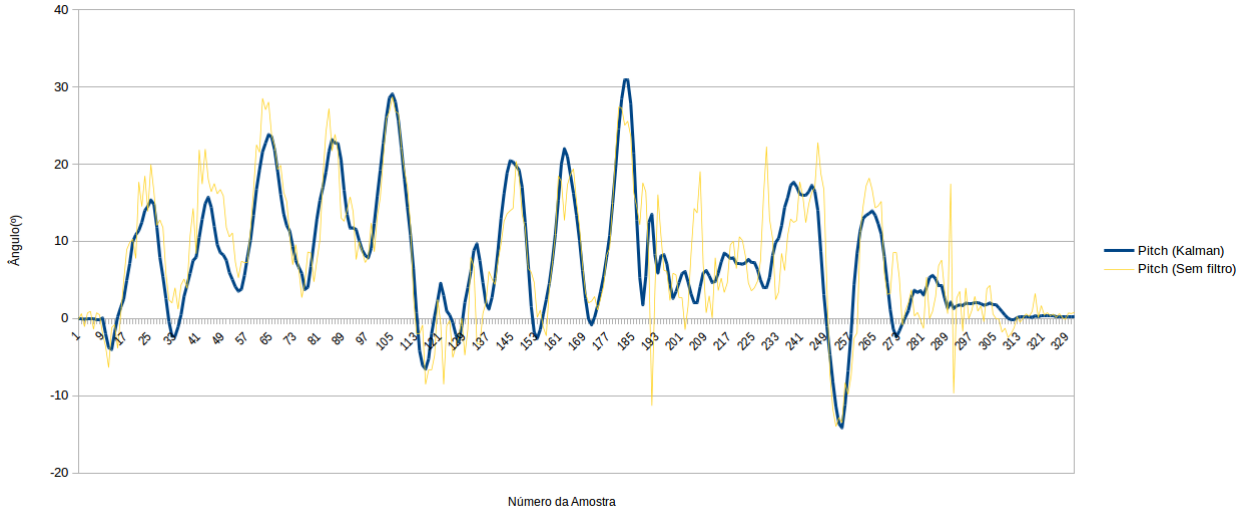


Figura 28: Dados da rotação em torno do eixo Y (*Pitch*) obtidos pelo sensor IMU quando gerado uma movimentação aleatória no sensor.

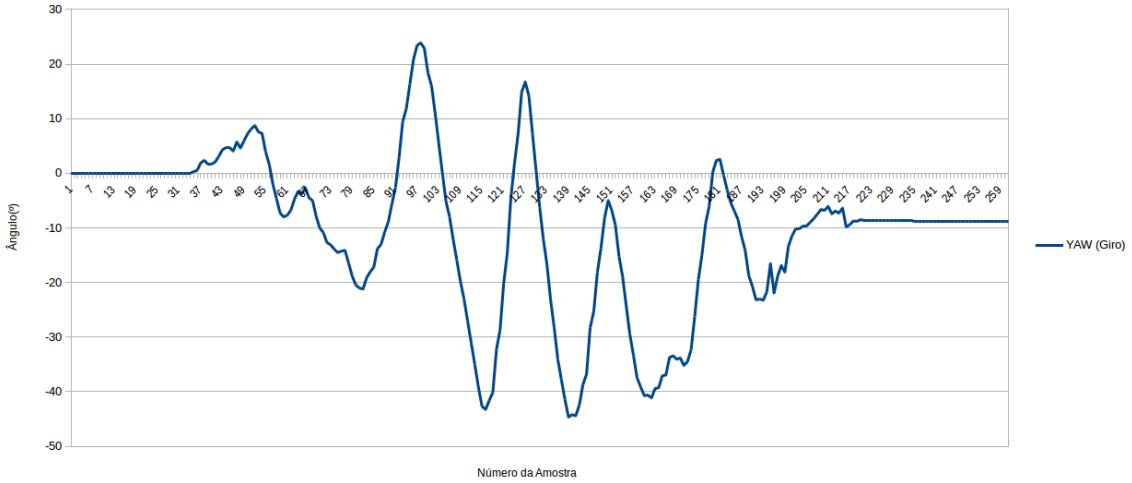


Figura 29: Dados da rotação em torno do eixo Z (*Yaw*) obtidos pelo sensor IMU quando gerado uma movimentação aleatória no sensor.

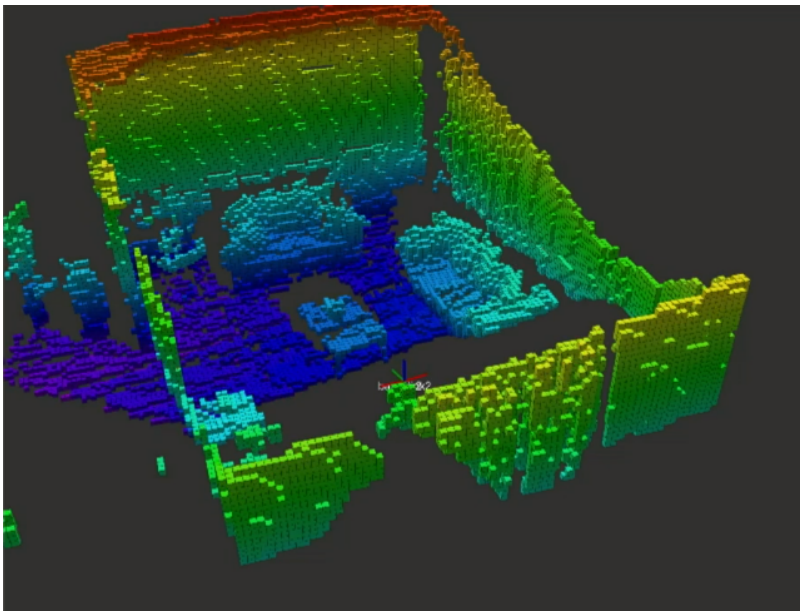
acumulando erros. Note que no final do experimento a estimativa do ângulo é incoerente com o valor inicial.

4.3 Mapa tridimensional

Inicialmente, considerando apenas o movimento de rotação foi possível obter o mapa apresentado na Figura 31b. Comparando a Figura 31a com a Figura 31b pode-se verificar que tanto o ambiente bem como os obstáculos presentes foram bem mapeados, reconstruindo, satisfatoriamente o ambiente. Porém, por não ter sido implementado uma correção de tratamento torno do eixo Z é possível notar pequenas distorções no mapa gerado, que gradativamente foram acumulando erros de angulação cada vez maiores.



(a)



(b)

Figura 30: (a) uma fotografia do local mapeado utilizando os métodos descritos na seção 3.4.3. (b) mapa do mesmo ambiente gerado pela grade de ocupação tridimensional Octomap, onde as cores representam a altura dos dados com base no referencial.

O experimento com mais detalhes pode ser visto no seguinte vídeo: https://www.youtube.com/watch?v=0lkZMW5HmtM.

O mapeamento tridimensional utilizando a metodologia de Localização e Mapeamento Simultâneos(SLAM) foi feito em uma varanda em "L" de aproximadamente 25 metros de extensão apresentado na Figura 31, pois essa permite gerar um mapa tridimensional fazendo-se tanto de movimentos de translação quanto de rotação.

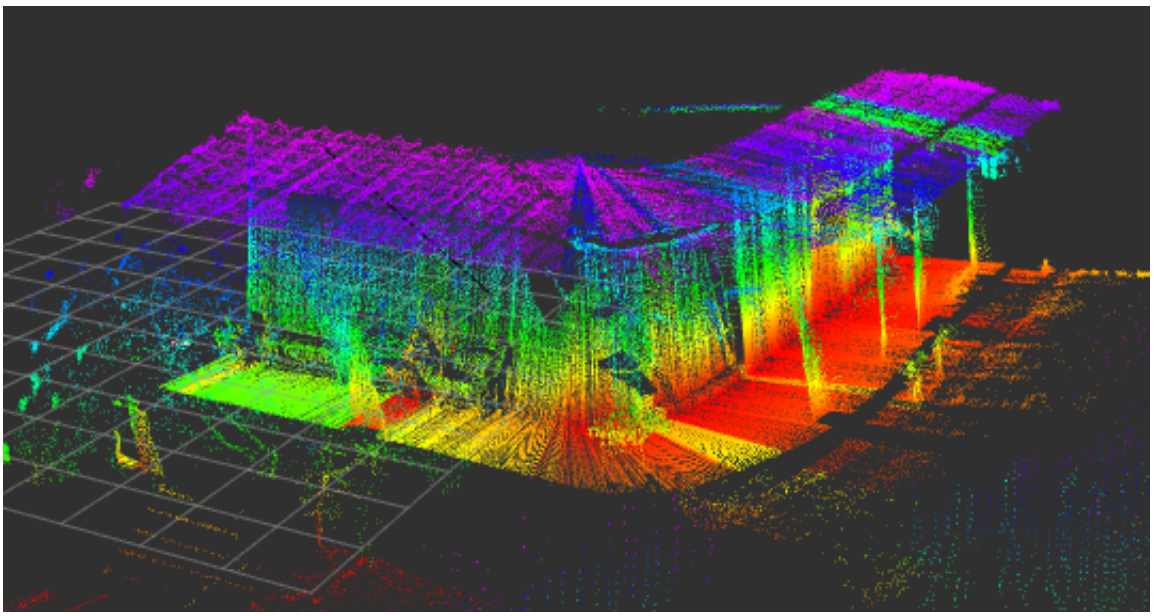
A Figura 32 mostra o resultado gerado a partir da aquisição de dados nesse ambiente,



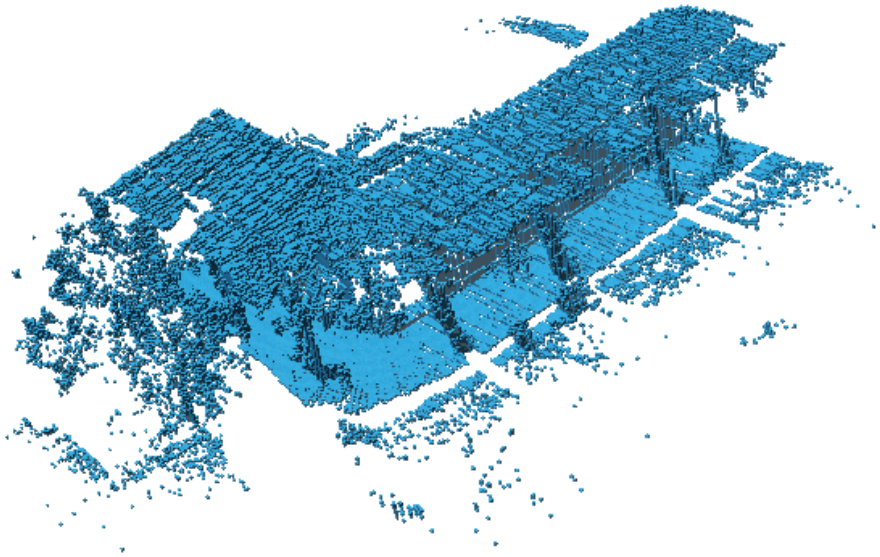
(a) (b)

Figura 31: (a) Fotografia do lado maior da varanda .(b) Fotografia demonstrando a estrutura em "L"da varanda.

na qual apresentou um mapa de maior extensão e de maior complexidade em comparação ao mapa gerado exclusivamente pelo movimento de rotação. Note que tanto os objetos presentes no ambiente quanto as vigas da construção foram bem mapeadas pelo sistema de mapeamento. Pode-se observar, também, que mesmo sem a utilização do filtro de Kalman para tratamento da rotação em torno do eixo Z (*Yaw*) o mapa não apresentou distorções significativas.



(a)



(b)

Figura 32: (a) Nuvem de pontos gerada através dos dados do sensor LIDAR com base nos movimentos de rotação e translação.(b) mapa desse mesmo ambiente gerado pela grade de ocupação tridimensional Octomap com base na nuvem de pontos.

5 Conclusão

De acordo com os resultados apresentados pode-se concluir que os métodos foram suficientes para a gerar um mapa tridimensional em ambiente simulado por meio das ferramentas do ROS, validando o objetivo inicialmente teorizada.

Em ambiente real, mesmo utilizando sensores de baixo custo conforme a Tabela 5, os métodos propostos para tratamento de dados foram suficiente para fornecer ao usuário os movimentos de rotação e translação de forma precisa, que junto às analises de dados realizadas, foi possível comprovar a eficiência do filtro de Kalman para esta aplicação.

Devido aos dados extremamente ruidosos do acelerômetro, preferiu-se não utiliza-lo para estimar velocidade e localização, adotando a utilização do sensor *encoder* rotativo como solução para adquirir os movimentos de translação, na qual apresentou um resultado satisfatório, que permitiu a realização do SLAM.

Tabela 5: Preço dos componentes utilizados.

Componente	Custo
YDLIDARX4	R$350,00
GY-87	R$37,00
Encoder	R$10,00
TOTAL	R$397,00

5.1 Trabalhos Futuros

- Implementar uma comunicação *Wireless* entre os *hardwares*, para permitir melhor mobilidade do robô e remover a necessidade de manutenção dos fios.
- Incluir metodologias para melhorar a estimativa do angulo de rotação em torno do eixo Z (*Yaw*).
- Implementar o projeto em vários robôs, criando uma rede de cooperação, de maneira que seja possível efetuar o mapeamento de vários pontos de forma simultânea, garantindo uma maior velocidade e confiabilidade para completar o mapeamento desejado.
- Possibilitar que o robô se movimente autonomamente, para que seja possível efetuar o mapeamento de ambientes, por exemplo, hostis ao ser humano ou inclusive diminuir o trabalho humano necessário para concluir o mapeamento.

- Modificar o *encoder* e o sensor IMU para outros de melhor resolução, para garantir um menor acúmulo de erro, e consequentemente menos distorções no mapa.
- Validar o experimento em ambiente com desnível.

Referências

BIASI, M. G. Sergio Coutinho de. *Utilização de quatérnios para representação de rotações em 3D*. 2016. Disponível em: <https://docplayer.com.br/22587181-Utilizacao-de-quaternios-para-representacao-de-rotacoes-em-3d.html>. Citado 5 vezes nas páginas 23, 24, 25, 26 e 27.

BIRKHOFF, G.; LANE, S. M. Book. *A survey of modern algebra / Garrett Birkhoff, Saunders Mac Lane*: A.K. Peters Wellesley, Mass, 1997. xi, 500 p. : p. ISBN 1568810687. Citado na página 26.

BOHREN, J. *ROS Crash-Course, Part I*: Introduction to ros distribution, build system and infrastructure. 2010. https://courses.cs.washington.edu/courses/cse466/11au/calendar/ros_cc_1_intro-jrsedit.pdf. Citado na página 18.

COSTA, M. A. M. M. S. M. *Controle de Veículos Aéreos Quadrirrotores Uso de Filtros de Kalman para Minimização de Erros na Unidade de Medida Inercial*. Dissertação (Mestrado) — Pontíficia Universidade Católica do Rio de Janeiro, 2014. Citado 5 vezes nas páginas 25, 28, 29, 30 e 31.

CRESTANI, T. *Desenvolvimento de API Para Rastreamento Híbrido de Dispositivos Móveis em Ambientes Internos*. 2015. Curso de Ciência da Computação, Instituto Federal Catarinense - Ifc Câmpus Videira,. Citado na página 45.

Dehong, C. et al. Design and implementation of lidar navigation system based on triangulation measurement. In: *2017 29th Chinese Control And Decision Conference (CCDC)*, 2017. p. 6060–6063. ISSN 1948-9447. Citado na página 34.

DOE, J. *ROS Introduction*. 2018. http://wiki.ros.org/ROS/Introduction. Acessado em 22 de Agosto de 2019. Citado na página 18.

EAI TECHNOLOGY. *YDLIDAR X4 DATASHEET*, 2015. Disponível em: <https:/-/www.elecrow.com/download/X4_Lidar_Datasheet.pdf>. Citado 2 vezes nas páginas 33 e 34.

ESPRESSIF SYSTEMS. *ESP32 Technical Reference Manual*, 2018. Disponível em: <https://www.espressif.com/sites/default/files/documentation-/esp32_technical_reference_manual_en.pdf>. Citado na página 38.

FORHAN, N. A. E. Giroscopios mems. In: , 2010. Disponível em: <http://mtc-m16d.sid-.inpe.br/col/sid.inpe.br/mtc-m19@80/2010/01.25.18.42/doc/publicacao.pdf>. Citado na página 40.

FUX, S. *Development of a planar low cost Inertial Measurement Unit for UAVs and MAVs*. Tese (Doutorado), 04 2008. Citado 2 vezes nas páginas 40 e 45.

GVDHOORN. *ROS Catkin*. 2017. http://wiki.ros.org/catkin/. Citado na página 18.

HAMILTON, W. R. *Elements of Quaternions*: Cambridge University Press, 1866. (Cambridge Library Collection - Mathematics). Citado 3 vezes nas páginas 26, 27 e 28.

HANDSON TECHNOLOGY. *Rotary Encoderfor Arduino/Raspberry*. 2018. Disponível em: <https://www.handsontec.com/dataspecs/module/Rotary\%20Encoder.pdf>. Acesso em: 10 agosto 2019. Citado na página 41.

HENDRIX, A. *ROS msg*. 2019. http://wiki.ros.org/msg. Acessado em 22 de Agosto de 2019. Citado 2 vezes nas páginas 11 e 20.

HORNUNG, A. et al. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. 2013. Software available at http://octomap.github.com. Disponível em: <http://octomap.github.com>. Citado 2 vezes nas páginas 21 e 22.

LOPES, A. M. *Modelação Cinemática e Dinâmica de Manipuladores de Estrutura em Sériel*. Dissertação (Mestrado) — FEUP - Faculdade de Engenharia Universidade do Porto, 2002. Citado 2 vezes nas páginas 22 e 23.

MACIEL, A. O. Aplicações: Mapeamento móvel utilizando tecnologia lidar. *Anais XV Simpósio Brasileiro de Sensoriamento Remoto*, 2011. p. 5455–5462, 2011. Citado na página 16.

MENEGUZZI, R. F. T. M. M. C. L. Utilização de giroscópio e acelerômetro para identificação de movimentação em ambientes tridimensionais. *Feira de Iniciação cientifica e extensão - Instituto Federal Catarinense - Câmpus Videira.*, 2008. Setembro 2008. Citado 3 vezes nas páginas 24, 40 e 45.

MUKUNDAN, R. Quaternions: From classical mechanics to computer graphics, and beyond. 2002. 01 2002. Citado na página 26.

NETO, J. *Mecânica Newtoniana, Lgrangiana e Hamiltoniana*. Editora Livraria da Física, 2004. ISBN 9788588325265. Disponível em: <https://books.google.com.br-/books?id=FyHOW_tvT8YC>. Citado na página 40.

PAVAN, N. L.; SANTOS, D. R. d. Um Método Automatico Para Registro De Dados Laser Scanning Terrestre Usando Superficies Planas. *BCG - Boletim de Ciencias Geodesicas*, 2015. scielo, v. 21, p. 572 – 589, 09 2015. ISSN 1982-2170. Citado na página 16.

PECKA, M. *ROS CMakeLists*. 2018. http://wiki.ros.org/catkin/CMakeLists.txt. Acessado em 10 de Outubro de 2018. Citado na página 19.

PEDLEY, M. Tilt sensing using a three-axis accelerometer. 2013. 06 2013. Citado na página 44.

PINHEIRO, M. L. G. Quaterniões : cálculo numérico e simbólico. 2006. p. 110, 10 2006. Citado 5 vezes nas páginas 23, 24, 27, 28 e 46.

PROMETHEUS: 20th Century Fox, 2012. Citado na página 16.

PYO, Y. et al. *ROS Robot Programming*: ROBOTIS, 2017. ISBN 9791196230715. Citado 2 vezes nas páginas 19 e 21.

ROMERO, A. M. *ROS Concepts*. 2014. http://wiki.ros.org/ROS/Concepts. Citado 2 vezes nas páginas 18 e 19.

SU, K. L.; GUO, H.; SHIAU, S. V. Development of a module based platform for mobile robots. In: , 2008. p. 1–6. ISBN 978-1-4244-2674-4. Citado na página 18.

TKJELECTRONICS. *KalmanFilter by TKJElectronics*. 1992. Disponível em: <https:/-/github.com/TKJElectronics/KalmanFilter>. Acesso em: 05 agosto 2019. Citado na página 47.

VIEIRA, M. E. A.; JUNIOR, S. L. S. Análise geral sobre o acelerômetro. In: *Semana de Eletrônica e Automação - UTFPR*, 2013. v. 4, p. 1–6. Citado na página 40.

Wu, D. et al. A lidar slam based on point-line features for underground mining vehicle. In: *2018 Chinese Automation Congress (CAC)*, 2018. p. 2879–2883. Citado na página 16.