



UNIFOR

UNIVERSIDADE DE FORTALEZA

CENTRO DE CIÊNCIAS TECNOLÓGICAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

DAVI GUANABARA DE ARAGÃO

**NAVEGAÇÃO AUTÔNOMA DA PLATAFORMA JAGUAR EM AMBIENTE
ESTÁTICO UTILIZANDO O LIDAR UTM-30LX**

FORTALEZA – CEARÁ

2018

DAVI GUANABARA DE ARAGÃO

NAVEGAÇÃO AUTÔNOMA DA PLATAFORMA JAGUAR EM AMBIENTE ESTÁTICO
UTILIZANDO O LIDAR UTM-30LX

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Engenharia de
Controle e Automação do Centro de Ciências
Tecnológicas da Universidade de Fortaleza,
como requisito parcial à obtenção do grau
de bacharel em Engenharia de Controle e
Automação.

Orientador: José Everardo Bessa Maia

FORTALEZA – CEARÁ

2018

DAVI GUANABARA DE ARAGÃO

NAVEGAÇÃO AUTÔNOMA DA PLATAFORMA JAGUAR EM AMBIENTE ESTÁTICO UTILIZANDO O LIDAR UTM-30LX

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação do Centro de Ciências Tecnológicas da Universidade de Fortaleza, como requisito parcial à obtenção do grau de bacharel em Engenharia de Controle e Automação.

Aprovada em: 01 de Janeiro de 2019

BANCA EXAMINADORA

José Everardo Bessa Maia (Orientador)
Centro de Ciências Tecnológicas - CCT
Universidade de Fortaleza - UNIFOR

Membro da Banca Dois
Faculdade de Filosofia Dom Aureliano Matos – FAFIDAM
Universidade do Membro da Banca Dois - SIGLA

Membro da Banca Três
Centro de Ciências e Tecnologia - CCT
Universidade do Membro da Banca Três - SIGLA

Membro da Banca Quatro
Centro de Ciências e Tecnologia - CCT
Universidade do Membro da Banca Quatro - SIGLA

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse, ao longo de minha vida, e não somente nestes anos como universitário, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

Aos meus pais, pelo amor, incentivo e apoio incondicional.

Obrigado meus irmãos e sobrinhos, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente! A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. a palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

“É melhor lançar-se à luta em busca do triunfo mesmo expondo-se ao insucesso, que formar fila com os pobres de espírito, que nem gozam muito nem sofrem muito; E vivem nessa penumbra cinzenta sem conhecer nem vitória nem derrota.”

(Franklin Roosevelt)

LISTA DE ILUSTRAÇÕES

| | |
|--|----|
| Figura 1 – Aparelhos autônomos | 12 |
| Figura 2 – frames | 15 |
| Figura 3 – TF | 16 |
| Figura 4 – LIDAR Hokuyo | 16 |
| Figura 5 – Especificações | 17 |
| Figura 6 – Example Grid | 19 |
| Figura 7 – Hector process | 20 |
| Figura 8 – Valores Costmap | 21 |
| Figura 9 – Demonstração visual do costmap | 22 |
| Figura 10 – Jaguar | 23 |
| Figura 11 – Fluxograma de funcionamento | 24 |
| Figura 12 – Fluxograma de funcionamento | 26 |
| Figura 13 – Fonte de Tensão | 35 |
| Figura 14 – Fonte Ligado no Sensor | 35 |
| Figura 15 – Sensor e Pc | 36 |
| Figura 16 – Dados do Sensor | 38 |
| Figura 17 – LIDAR no rviz | 38 |
| Figura 18 – Configuração Rviz | 40 |
| Figura 19 – Gmapping Sendo Executado | 41 |
| Figura 20 – Labirinto | 41 |

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

| | | |
|--------------|---|----|
| 1 | INTRODUÇÃO | 11 |
| 2 | MOTIVAÇÃO | 13 |
| 2.1 | OBJETIVOS | 13 |
| 2.1.1 | Objetivo Geral | 13 |
| 2.1.2 | Objetivo Específico | 13 |
| 3 | FUNDAMENTAÇÃO TEÓRICA | 14 |
| 3.1 | ROS (ROBOTIC OPERATING SYSTEM) | 14 |
| 3.2 | SISTEMA DE COORDENADAS E O PACOTE TF | 14 |
| 3.3 | HOKUYO UTM 30LX | 16 |
| 3.4 | OCCUPANCY GRID | 17 |
| 3.5 | GMAPPING | 20 |
| 3.6 | HECTOR MAPPING | 20 |
| 3.7 | COSTMAP 2D | 21 |
| 3.8 | BASE PLANNER (PATH PLANNING AND CONTROL) | 22 |
| 3.9 | DRROBOTV2 PLAYER | 22 |
| 3.10 | JAGUAR | 22 |
| 4 | METODOLOGIA | 24 |
| 4.1 | RECURSOS | 24 |
| 4.2 | ALGORÍTMO | 24 |
| 4.2.1 | Aquisição de dados | 25 |
| 4.2.2 | Processamento de dados | 25 |
| 4.2.2.1 | Configuração do Hector-slam e transformações | 25 |
| 4.2.2.2 | Configuração do Move Base | 27 |
| 4.2.3 | Controle da plataforma móvel | 27 |
| 5 | RESULTADOS | 28 |
| 5.1 | RESULTADOS DA ETAPA AJUSTAR E INTERPRETAR O LIDAR | 28 |
| 6 | CONCLUSÕES E TRABALHOS FUTUROS | 29 |
| 6.1 | CONTRIBUIÇÕES DO TRABALHO | 29 |
| 6.2 | LIMITAÇÕES | 29 |
| 6.3 | TRABALHOS FUTUROS | 29 |
| | REFERÊNCIAS | 30 |

| | |
|--|----|
| APÊNDICES | 32 |
| APÊNDICE A – Ajustar e interpretar o LIDAR | 34 |
| A.1 PRE-CONFIGURAÇÕES | 34 |
| A.1.1 Links interessantes | 34 |
| A.2 LIGAR O UTM-30XL | 34 |
| A.2.1 Material | 34 |
| A.2.2 Configuração do sensor | 35 |
| A.2.2.1 Alimentação | 35 |
| A.2.2.2 Coletar dados | 36 |
| A.2.2.3 visualizar dados | 37 |
| A.3 PROCESSAMENTO DE DADOS | 38 |
| A.3.1 Gmapping | 39 |
| APÊNDICE B – Jaguar Autônomo | 42 |
| B.1 RECURSOS | 42 |
| B.2 PRÉ CONFIGURAÇÃO | 42 |
| B.3 ROS WORKSPACE | 42 |
| B.4 APLICAÇÃO NO JAGUAR | 43 |
| APÊNDICE C – Solução de problemas identificados na primeira etapa do projeto | 44 |
| C.1 DEBUG | 44 |
| C.2 PARTICIONAMENTO DO DISCO | 44 |
| C.3 HOKUYO | 44 |
| C.4 ERRO AO TESTAR O LASER NO RVIZ TRANSFORM [SENDER=UNKNOWN_PUBLISHER] | |
| C.5 PROBLEMAS QUANTO AOS FRAMES DO GMAPPING | 46 |
| C.6 COMANDOS INTERESSANTES AO USAR O TERMINAL | 48 |
| APÊNDICE D – hector-simulador-map.launch | 49 |
| APÊNDICE E – base-local-planner | 52 |
| APÊNDICE F – global-costmap-params | 53 |
| APÊNDICE G – local-costmap-params | 54 |
| APÊNDICE H – costmap-common-params | 55 |
| APÊNDICE I – mybot _w orld | 56 |
| ANEXOS | 57 |

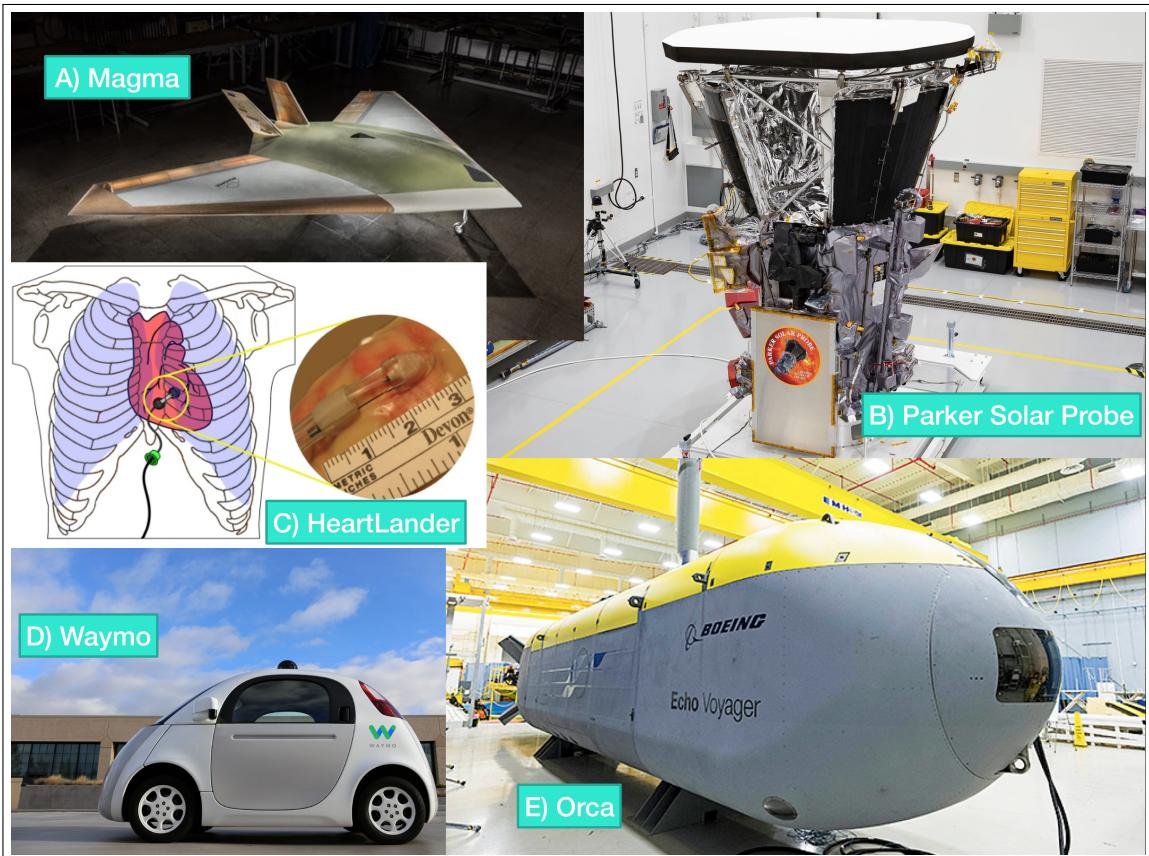
1 INTRODUÇÃO

Os sistemas autônomos, que se diferem de um automático por sua capacidade de tomar decisões sem intervenção humana, são extremamente relevantes atualmente, pois sua aplicação tem como impacto a mitigação da latência, a melhora da eficiência por reduzir os custos e a massa do aparelho, do uso dos instrumentos e no gerenciamento de sistemas complexos que se tornaram tão difíceis que é por vezes impossível para humanos à distância diagnosticar e resolver os problemas (FROST, 2011).

Dessa forma, os desafios que esses tipos de sistema enfrentam estão relacionados com a interação com o ambiente real, planejamento e execução da ação, interação com o humano e a percepção do ambiente,(HODICKY; PROCHAZKA, 2017).

Conforme esses desafios são superados, novos aparelhos que hospedam essa tecnologia são criados, possibilitando sua aplicação em diferentes áreas, mostradas na figura 1, como em: C) HeartLander (QARRA, 2015), Robô móvel em miniatura que fornece terapia minimamente invasiva para a superfície do coração; B) Parker Solar Probe (FOX *et al.*, 2016), sonda espacial da NASA que necessita ser autônoma para se manter segura sem a orientação vinda da terra; E) Orca (XLUUV),(LOCKHEED MARTIN CORPORATION,) um submarino não tripulado; A) MAGMA UAV, (COXWORTH,), um veículo aéreo não tripulado que não contém flaps; D) Google Waymo, (KRAFCIK,), um carro que não precisa de um motorista humano.

Figura 1 – Aparelhos autônomos



Fonte – A) (COXWORTH,), B) (REILY,), C) (THE ROBOTIC INSTITUTE,), D) (KRAFCIK,), E) (LOCKHEED MARTIN CORPORATION,)

A capacidade de navegar é uma característica comum entre todas as plataformas anteriormente citadas, sem a qual estas não seriam aptas a executar uma tarefa solicitada. Para tal, duas etapas são de extrema importância, a aquisição e o processamento de dados do sistema e ambiente.

A navegação é uma capacidade imprescindível para as plataformas citadas anteriormente consigam executar suas tarefas, e para tal, duas etapas são fundamentais, a aquisição e o processamento de dados do sistema e ambiente.

A aquisição de dados é feita através dos sensores e instrumentação, os quais mensuram as grandezas relacionadas à plataforma e ao ambiente, como o GPS (Global System Position, ou Sistema de Posicionamento Global), giroscópio, acelerômetro, barômetro, magnetômetro, câmera, *laser scanner*, *encoders*, termômetro, entre outros.

Processando esses dados com diferentes métodos de fusão sensorial, como o SLAM, é possível estimar a posição da plataforma em relação ao ambiente, e consequentemente, planejar e executar uma trajetória.

2 MOTIVAÇÃO

Como mostrado em (G1,), o Brasil é um dos últimos países no *ranking* de aptidão para receber os carros autônomos, apesar dos brasileiros estarem mais dispostos a obter essa tecnologia do que os japoneses. Um dos pontos da lista do que o Brasil precisa para receber essa tecnologia é a pesquisa e desenvolvimento de serviços de mobilidade, evidenciando assim a deficiência desse tipo de projeto nesse país. O atual trabalho se propõe a implementar um sistema de navegação autônoma com autonomia nível 5, no qual não há a intervenção humana, de forma alcançar um paralelo em escala reduzida dos carros autônomos, utilizando sensores e métodos comumente encontrado nesses (SILVA,).

Pesquisa e desenvolvimento de serviços de mobilidade é um dos pontos listados do que o Brasil precisa ter para receber essa tecnologia.

2.1 OBJETIVOS

2.1.1 Objetivo Geral

Programar o robô DRRobot Jaguar V4 with arm, para navegar de forma autônoma entre pontos de origem e destino fornecidos em uma sala retangular com obstáculos fixos, e avaliar o desempenho em laboratório utilizando o sensor LIDAR Hokuyo UTM-30LX Scanning Laser Rangefinder.

2.1.2 Objetivo Específico

1. Instalar o ROS Melodic na máquina virtual Ubuntu
2. Criar o workspace com os pacotes necessários
3. Coletar dados LIDAR através da conexão serial utilizando o ROS
4. Coletar dados LIDAR através da rede sem fio do Jaguar utilizando o ROS
5. Controlar o jaguar pelo computador através da rede sem fio utilizando o ROS
6. Configurar sistema SLAM
7. Configurar sistema de navegação
8. Integrar sistema de navegação com o Jaguar
9. Avaliar o desempenho em 30 testes de laboratório com os pontos de origem e destino escolhidos aleatoriamente na sala de teste. A medida de desempenho é o tempo necessário para o robô se deslocar entre dois pontos específicos.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 ROS (ROBOTIC OPERATING SYSTEM)

O ROS (Robotic Operating System) é um *framework* flexível que se tornou o programa chave do projeto por facilitar a criação de comportamentos complexos e robustos para os robôs de forma a integrar os diversos softwares desenvolvidos por diferentes laboratórios (OPEN SOURCE ROBOTICS FOUNDATION,).

Para tal, o ROS se baseou nos seguintes conceitos: ponto-a-ponto, onde os sistemas são conectados diretamente ao framework, de forma que esses podem tanto receber como enviar dados; Multi-linguagens, suportando diferentes linguagens como C++ e Python; Leve, no qual a complexidade foi deixada nas bibliotecas, permitindo a fácil extração do código e seu reuso; E livre, permitindo o desenvolvimento de programas comerciais e não comerciais (QUIGLEY *et al.*, 2009).

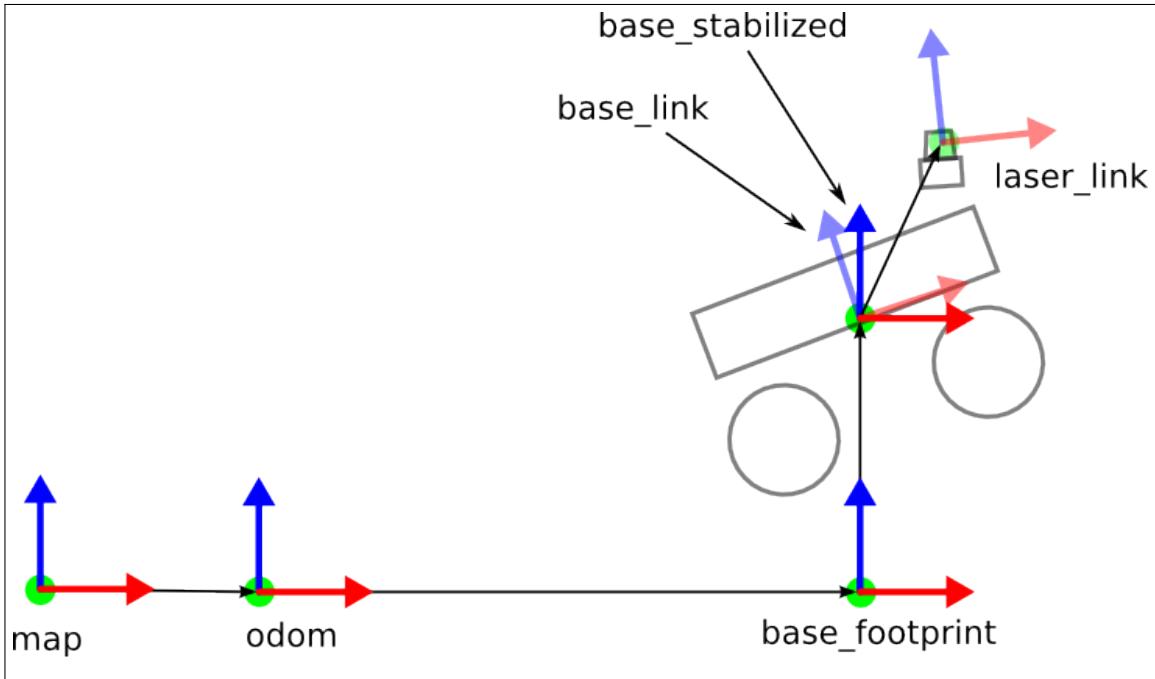
Objetivando prover funcionalidades úteis em uma maneira fácil de usar, de forma que possa ser prontamente reutilizado, os programas do ROS são organizados em pacotes (HANSEN, 2015), o qual é um conjunto de múltiplos módulos de códigos que trabalham em conjunto para atingir os seus objetivos (TECHOPEDIA,).

O presente trabalho utilizou diversos módulos, como o *hokuyo node*, para o acesso ao sensor *Hokuyo laser range finder 30 lx*, o *Hector* e o *Gmapping*, para o mapeamento do ambiente e a estimativa da localização do robô a partir dos dados fornecidos pelo sensor, o *costmap 2d*, para o planejamento e execução da navegação e, por fim, para o controle do jaguar, o *drrobotV2 player*.

3.2 SISTEMA DE COORDENADAS E O PACOTE TF

Aqui é chamado de *frame* o conjunto de vetores x, y e z, formando a tripla ordenada, de forma que seus valores se relacionam com a posição do ponto no espaço. O conjunto de *frames* utilizados foram uma adaptação do sistema do hector slam, mostrada na figura 2, no qual cada um, o *map*, *odom*, *base-footprint*, *base-stabilized*, *base-link* e *laserlink*, tem uma função desempenhada.

Figura 2 – frames



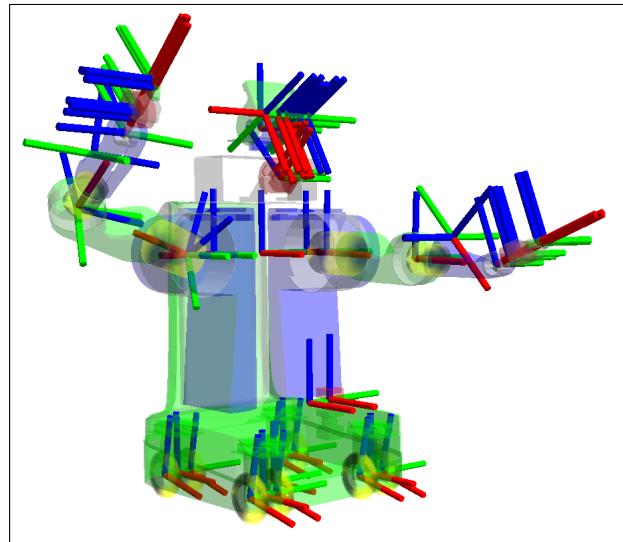
O frame "map" é usado como referência global da posição do robô, de forma que, o sistema constantemente recomputa a posição do robô em relação ao mapa. Por ser discreto, o frame "map" não deve ser usado como uma referência local. O frame "odom", por sua vez, também é usado como referência à posição do robô, porém, diferente do map, não é usado de forma global, e sim local, pois por ser contínuo prover maior precisão para esse tipo de aplicação. Para esse trabalho, não foi necessário os frames "base footprint" nem o "base stabilized". O frame "base link" foi renomeado para "base frame", e o "laser link" para "laser". O pacote hector-slam prover um frame adicional, o chamado "scanmatch frame", onde revela a nova posição do robô, após ter recebido novas informações do sensor laser.

Ele é gerado a partir da análise dos dados armazenados com os já coletados, de forma a estimar a localização relativa aos objetos.

Temos mais um frame, o chamado "base frame". Esse tem o propósito de estar atrelado ao robô, de forma a representá-lo nessa sistema.

Para criar, relacionar, manter, analisar e transformar os frames, é necessário usar o pacote TF, no qual apresenta como os pontos fortes, como escrito em (FOOTE, 2013), a eficiência e a flexibilidade. Na figura 3 podemos ver como uma série de frames podem ser relacionados de forma que seja possível estimar a posição de pontos interessantes do robô, como as juntas.

Figura 3 – TF



Fonte – (FOOTE, 2013)

3.3 HOKUYO UTM 30LX

Hokuyo UTM 30LX Scanning Laser Range Finder, sensor mostrado na Figura 12, tem um princípio de funcionamento simples, seguindo o padrão SCIP 2.0, o qual prevê uma interface flexível e eficiente para aplicações robóticas (MAEJIMA KAWATA,).

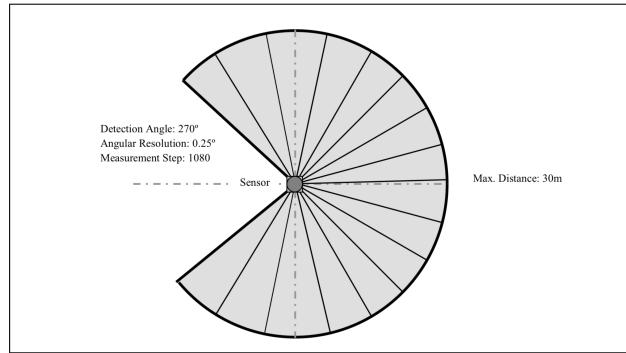
Figura 4 – LIDAR Hokuyo



Esse sensor rotaciona no sentido anti-horário quando olhado a partir da visão superior (MAEJIMA KAWATA,), disparando um pulso de laser de 905nm a cada passo de 0.25

graus, em uma área de 270 graus (KIMITANI HINO,), como mostrado na Figura 13. O laser refletido é captado pelo sensor, mensurando, em seguida, a diferença de tempo entre o disparo e a captação. Como a velocidade da luz na terra é conhecida e o tempo foi mensurado, é possível calcular a distância do objeto e sua posição relativa ao sensor (MOSKAL, 2008).

Figura 5 – Especificações



Esse instrumento garante a detecção de um objeto entre 0.1 e 30 metros, e até 60 metros de forma não garantida (KIMITANI HINO,). Sua estrutura confere uma proteção IP67, o qual provê uma proteção contra poeira e jatos de água (THE ENCLOSURE COMPANY (INTERNATIONAL) LTD,), e o laser é classificado como class 1 (KIMITANI HINO,), segundo o padrão ANSI (American National Standards Institute) e IEC (International Electrotechnical Commission), o qual esclarece que o laser não pode emitir radiação em intensidades que causem danos aos olhos em operação normal (ROCKWELL LASER INDUSTRIES,) o que significa que esse é o mais seguro, com o risco mínimo de causar problemas com o olho humano (SLINEY, 1980).

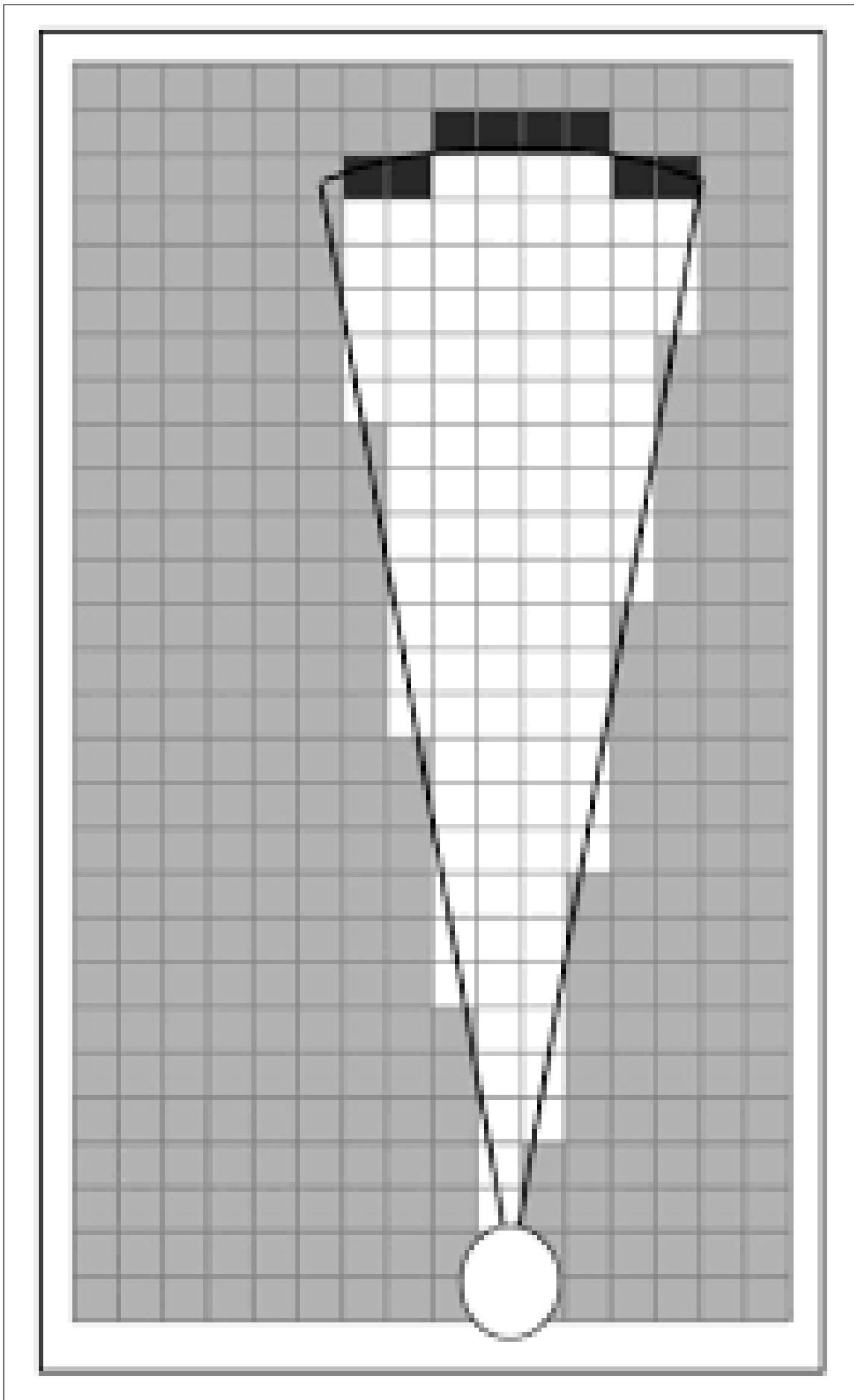
3.4 OCCUPANCY GRID

Ocupancy é definido como uma variável binária aleatória. Variável aleatória foi definida em (PETERNELLI, 2013, p. 47) "Uma função cujo valor é um número real determinado por cada elemento em um espaço amostral é chamado uma variável aleatória". Portanto, Occupancy está no espaço probabilístico, sendo possível adotar somente dois estados, livre e ocupado. Quando esse valor é desconhecido, considera-se como indefinido.

O occupancy grid map nada mais é do que uma matriz de variáveis occupancy, no qual foi usado para mapear o ambiente no qual o robô está inserido. Esse está graficamente representado na Figura 17, em que a cor preta é relacionado com uma célula ocupada, branca é

uma célula livre e marrom é uma célula indefinida.

Figura 6 – Example Grid



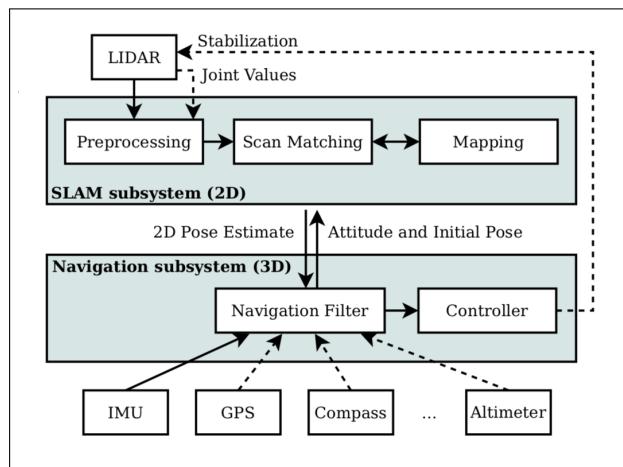
3.5 GMAPPING

O Gmapping SLAM é um pacote para ROS com métodos de SLAM, no qual seu correto funcionamento depende do fornecimento de dados de odometria para determinar se o mesmo tem que incorporar novas leituras e para o palpite inicial da localização do sensor. Como essas características foram descobertas após o início do seu uso, a sua aplicação nesse trabalho ficou restrito aos testes do sensor, não sendo utilizado posteriormente. Uma possível solução ao seu uso seria fornecer a odometria a partir do pacote laser-scan-matcher, no qual seria alimentado pelos dados do sensor IMU (Inertial Measurement unit).

3.6 HECTOR MAPPING

O Hector mapping é um pacote SLAM no qual seu processo, descrito graficamente na Figura 13, aproveita a alta taxa de atualização do LIDAR para o alinhamento das sucessivas leituras, processo chamado de Scan Matching, com o mapa já criado, sendo, dessa forma, capaz de estimar a posição do robô sem a necessidade de coletar dados de odometria (KOHLBRECHER et al., 2011).

Figura 7 – Hector process



Fonte – (KOHLBRECHER et al., 2011)

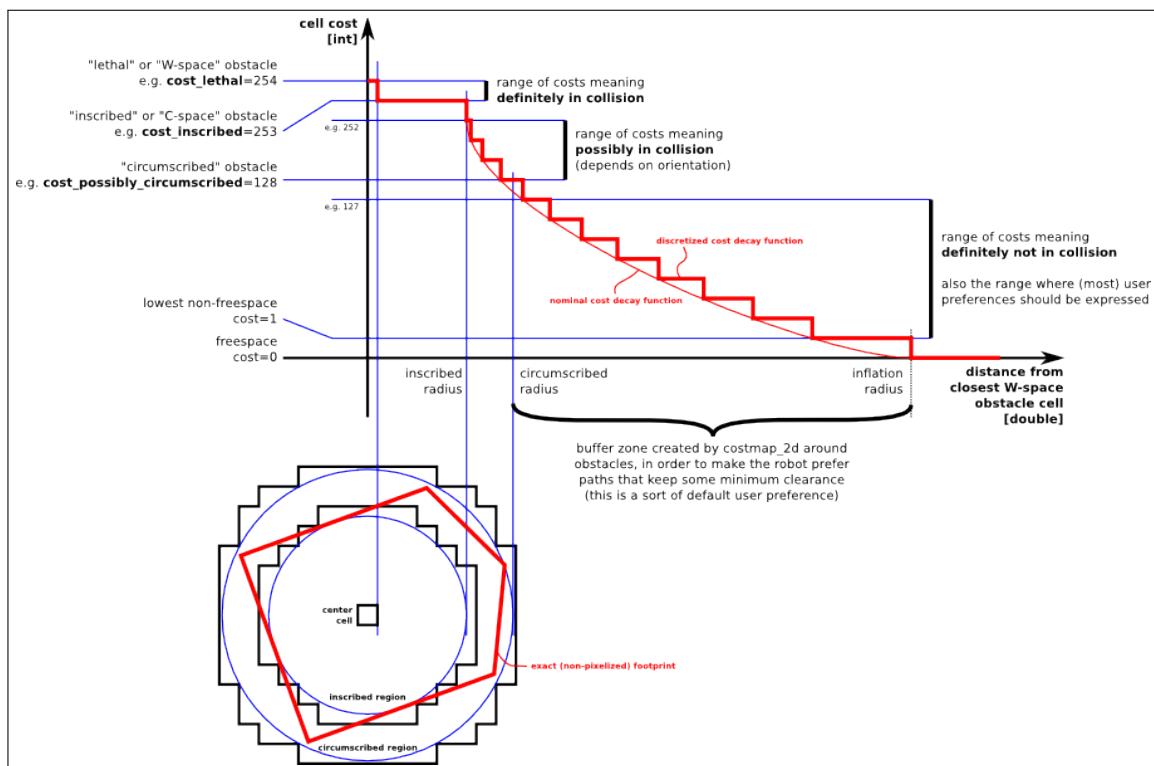
Para o intuito desse trabalho, não será usado um subsistema de navegação 3D, como mostrado na figura 13.

3.7 COSTMAP 2D

O pacote *costmap 2d* prover uma implementação que, com os dados captados pelo LIDAR e o mapa já armazenado, no formato *occupancy grid*, atualiza a posição dos obstáculos e prover o espaço navegável.

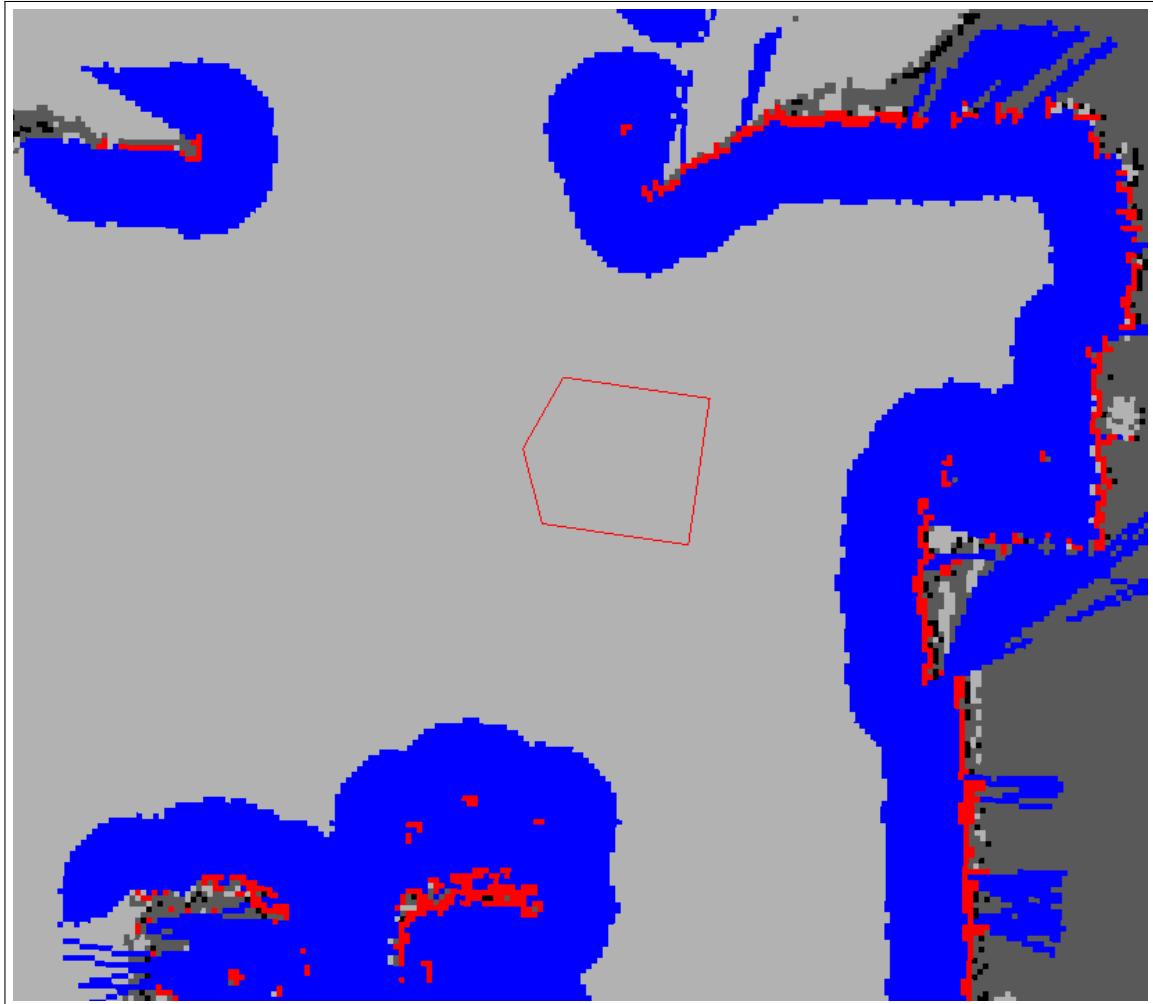
A Figura 9 mostra os possíveis valores que cada célula, após ter seu custo de navegação calculado, passa a conter, sendo 0 um espaço livre para a navegação, e 254 sendo colisão certa. A Figura 10 é uma representação gráfica dos valores calculados pelo *costmap*, de forma que cinza é a área navegável, azul é a área de risco de colisão e, por fim, a vermelha é a área de colisão certa.

Figura 8 – Valores Costmap



Fonte – (MARDER-EPPSTEIN DAVID V. LU!!,)

Figura 9 – Demonstração visual do costmap



Fonte – (MARDER-EPPSTEIN DAVID V. LU!!,)

Conforme novas leituras do LIDAR são recebidas pelo algoritmo, o estado de cada célula, livre ou ocupada, é atualizado a partir de um cálculo probabilístico fornecido por outro pacote, como o Hector Slam ou o Gmapping.

3.8 BASE PLANNER (PATH PLANNING AND CONTROL)

3.9 DRROBOTV2 PLAYER

3.10 JAGUAR

O Jaguar V4 with Manipulator Arm Mobile Robotic Platform, Figura 14, foi projetado para aplicações que requerem uma robusta manobrabilidade e manipulação de objetos. O mesmo é integrado com um sistema de medição inercial com nove graus de liberdade, três

oriundos de um acelerômetro, 3 de giroscópio, e os últimos 3 de um magnetrômetro. Além desse, há outros aparelhos integrados, como uma câmera com áudio e vídeo de alta resolução, GPS e LIDAR, todos conectados com uma rede sem fio 802.11N (ROBOT,).

Figura 10 – Jaguar



Fonte – Elaborado pelo autor

4 METODOLOGIA

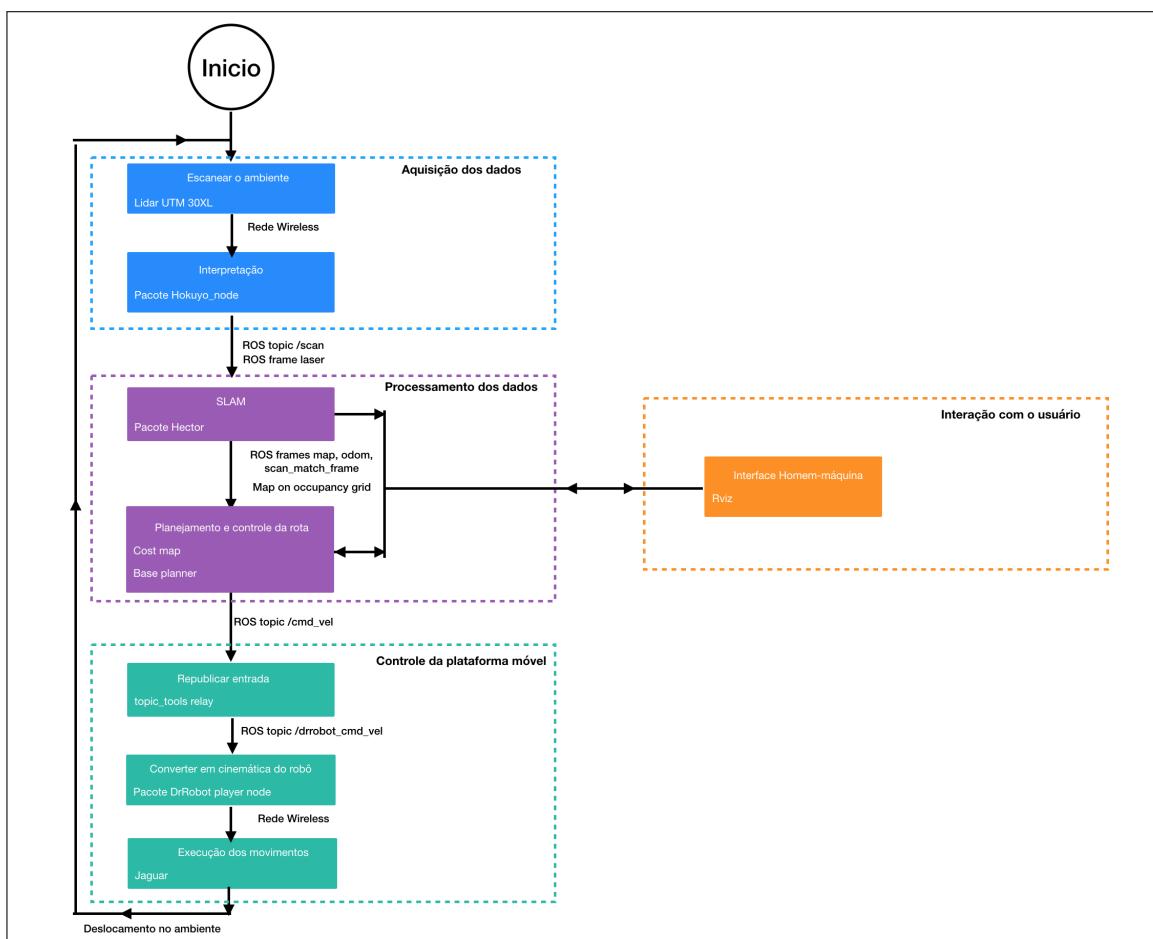
4.1 RECURSOS

Para essa etapa, foi utilizado: MacBook Pro (Retina, 15-inch, Mid 2015) 2,2 GHz Intel Core i7 16 GB 1600 MHz DDR3 Intel Iris Pro 1536 MB macOS Mojave Version 10.14.1 Parallels Desktop 14 home edition Ubuntu 18 ROS Melodic

4.2 ALGORÍTMO

O sistema de navegação autônoma desenvolvido nesse trabalho, mostrando na figura 11, é dividido em 4 etapas fundamentais: Aquisição de dados; Processamento de dados; Controle da plataforma móvel; E a interação com o usuário.

Figura 11 – Fluxograma de funcionamento



Fonte – Elaborado pelo autor

4.2.1 Aquisição de dados

A aquisição de dados é feita por meio do LIDAR UTM 30XL, o qual escaneia o ambiente e envia, através da rede sem fio do jaguar, as medidas coletadas para o pacote do ROS Hokuyo-node. Esse pacote, por sua vez, interpreta essas medidas, gerando o frame laser, o qual é o ponto inicial das medições e publicando dados LIDAR no tópico scan do ROS, deixando-os prontos para serem usados na etapa de processamento de dados.

O pacote Hokuyo-node necessita que o instrumento LIDAR esteja conectado em uma porta serial do computador. Como o LIDAR está montado no Jaguar, não é possível fazer essa conexão. A solução encontrada aqui foi criar uma conexão serial virtual, de forma que os dados do sensor devem ser recebidos via rede sem fio, o qual estão disponíveis no endereço de IP (Internet Protocol) 192.168.0.60:10002, redirecionados à uma porta serial virtual e por fim, acessadas pelo pacote Hokuyo-node.

4.2.2 Processamento de dados

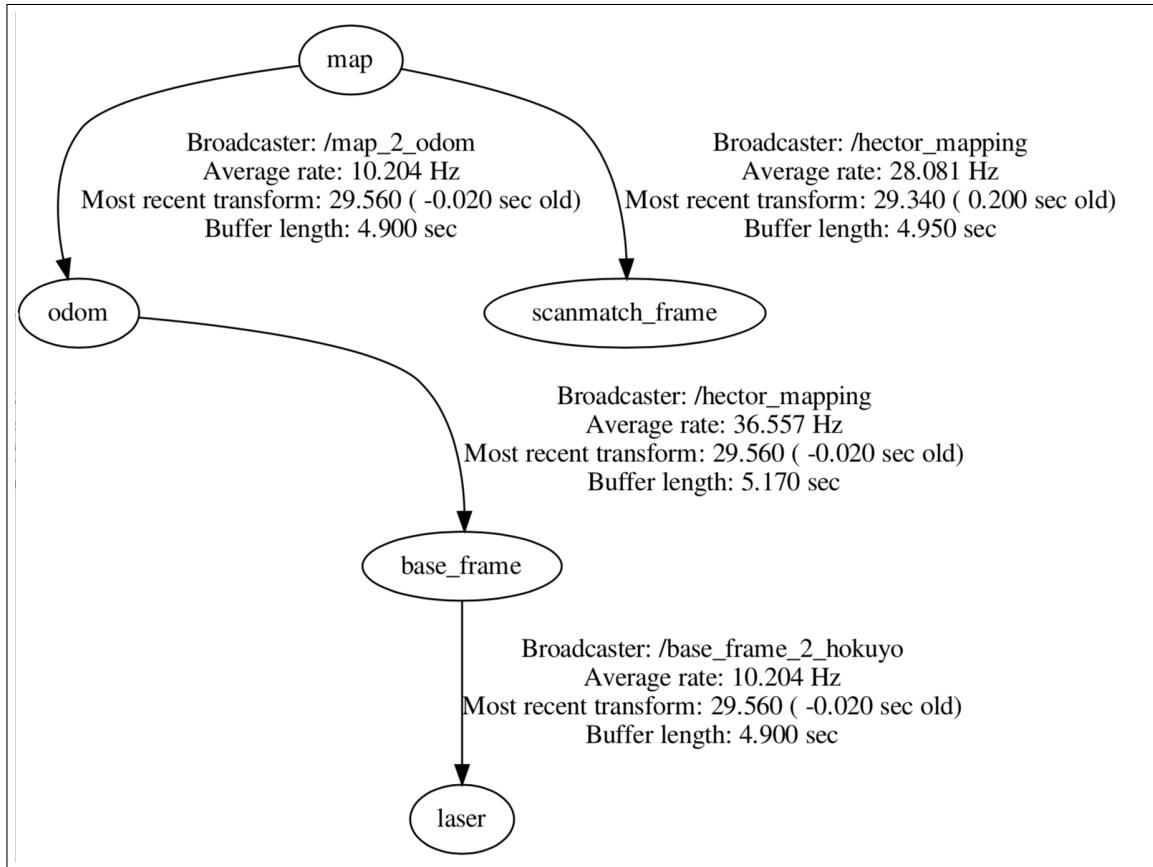
A configuração do processamento de dados foi dividido em três etapas, a configuração do Hector-slam, a interligação dos diferentes frames e a configuração do pacote move-base.

4.2.2.1 Configuração do Hector-slam e transformações

O hector-slam proporciona o SLAM, o qual é publicado no tópico map, e as transformações do frame map, que está relacionado com o mapa, para o odom, que se refere à odometria do robô, e ao frame scanmatch, que indica qual é a nova posição estimada do robô após receber a nova leitura do sensor. Porém, antes de configura-lo, devemos proporcionar a cadeia de relações de frames. Como o pacote oferece a relação até o frame odom, devemos relacioná-lo com o base-frame, que se refere ao robô, e desse para o frame laser, que se refere ao LIDAR. Essa configuração pode ser conferida no Apendice D, da linha 16 à 27.

A figura 12 mostra como os frames se relacionam quando o pacote hector-slam está em funcionamento.

Figura 12 – Fluxograma de funcionamento



Fonte – Elaborado pelo autor

Além das transformações, é necessário configurar o hector-slam da seguinte forma:

Inscrever no tópico scan para receber os dados LIDAR; Definir o nome do frame de base ao SLAM, chamado aqui de base-frame; Indicar que não há dados de odometria sendo coletados, tornando o frame de odometria igual ao frame base; Nomear o frame do scanmatch, aqui nomeado de scanmatch-frame; Nomear o frame do mapa como map; Ativar a publicação da transformação do frame map para o frame do scanmatch; publicar a transformação do frame map para o frame da odometria.

Essa configuração pode ser encontrada nas linhas 8 até 14 do Apêndice D.

Configurado, o Hector-slam entregará ao ROS os frames do mapa, scanmatch e odometria já interrelacionados e atualizados conforme os dados do LIDAR são recebidos. Porém, outras transformações relações de frames são necessárias e abordadas abaixo.

4.2.2.2 Configuração do Move Base

Por fim, devemos configurar e executar o pacote move-base, o qual é responsável por, entre outras funcionalidades, o planejamento e controle da rota, a partir dos dados coletados dos tópicos map e scan. Provendo a rota, no tópico path, e as variações linear e angular que o frame atrelado ao posicionamento do robô, o frame scanmatch, deve executar para navegar, no tópico cmd-vel. Essas funcionalidades são feitas através do costmap, em que suas configurações podem ser encontradas com mais detalhes nos Apêndices F, G e H. Para mais detalhes do base-local-planner, ver Apêndice E.

4.2.3 Controle da plataforma móvel

O controle do jaguar acontece por meio do pacote drrobot-player-node, o qual se inscreve no tópico drrobot-cmd-vel, para receber a variação linear e angular que o jaguar deve executar para seguir a rota elaborada anteriormente, e envia ao robô via sua rede sem fio, a intensidade e direção da rotação dos seus motores. Porém, como os dados das variações são publicados em um tópico diferente do que esse pacote se inscreve. Dessa forma, devemos redirecionar os dados oriundos do tópico cmd-vel para o tópico drrobot-cmd-vel. Esse redirecionamento é feito pelo pacote topic-tools, usando o comando relay.

5 RESULTADOS

5.1 RESULTADOS DA ETAPA AJUSTAR E INTERPRETAR O LIDAR

O processamento dos dados LIDAR, com o uso do gmapping e hector-slam, foi bem sucedido, mostrando que os pacotes usados são compatíveis com o sistema apresentado. Porém, a eficácia e a eficiência não puderam ser mensuradas a tempo da escrita desse trabalho.

6 CONCLUSÕES E TRABALHOS FUTUROS

6.1 CONTRIBUIÇÕES DO TRABALHO

6.2 LIMITAÇÕES

6.3 TRABALHOS FUTUROS

REFERÊNCIAS

- COXWORTH, B.** *BAE's latest UAV has no use for flaps.* Disponível em: <<https://newatlas.com/bae-magma-uav/52611/>>.
- FOOTE, T.** *tf: The transform library.* In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on.* [S.l.: s.n.], 2013. (Open-Source Software workshop), p. 1–6. ISSN 2325-0526.
- FOX, N. J.; VELLI, M. C.; BALE, S. D.; DECKER, R.; DRIESMAN, A.; HOWARD, R. A.; KASPER, J. C.; KINNISON, J.; KUSTERER, M.; LARIO, D.; LOCKWOOD, M. K.; MCCOMAS, D. J.; RAOUAFI, N. E.; SZABO, A. *The solar probe plus mission: Humanity's first visit to our star.* *Space Science Reviews*, v. 204, n. 1, p. 7–48, Dec 2016. ISSN 1572-9672. Disponível em: <<https://doi.org/10.1007/s11214-015-0211-6>>.**
- FROST, C. R.** *Challenges and opportunities for autonomous systems in space.* In: . [S.l.: s.n.], 2011.
- G1. Brasil é um dos últimos em ranking de aptidão para carros autônomos.** Disponível em: <<https://g1.globo.com/carros/noticia/brasil-e-um-dos-ultimos-em-ranking-de-aptidao-para-carros-autonomos.ghtml>>.
- HANSEN, K.** *Packages.* 2015. Disponível em: <<http://wiki.ros.org/Packages>>.
- HODICKY, J.; PROCHAZKA, D.** *Challenges in the implementation of autonomous systems into the battlefield.* In: *2017 International Conference on Military Technologies (ICMT).* [S.l.: s.n.], 2017. p. 743–747.
- KIMITANI HINO, M.** *Scanning Laser Range Sensor UTM-30LX Specification.* [S.l.]. Disponível em: <http://wiki.ros.org/hokuyo_node?action=AttachFile&do=get&target=UTM-30LX_Specification.pdf>.
- KOHLBRECHER, S.; MEYER, J.; STRYK, O. von; KLINGAUF, U.** *A flexible and scalable slam system with full 3d motion estimation.* In: *IEEE. Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR).* [S.l.], 2011.
- KRAFCIK, J.** *Say hello to Waymo: what's next for Google's self-driving car project.* Disponível em: <<https://medium.com/waymo/say-hello-to-waymo-whats-next-for-google-s-self-driving-car-project-b854578b24ee>>.
- LOCKHEED MARTIN CORPORATION.** *Orca - Extra Large Unmanned Undersea Vehicle.* Disponível em: <<https://www.lockheedmartin.com/en-us/products/orca-extra-large-unmanned-underwater-vehicle-xluuv.html>>.
- MAEJIMA KAWATA, M.** *Communication Protocol Specification For SCIP2.0 Standard.* [S.l.]. Disponível em: <http://wiki.ros.org/hokuyo_node?action=AttachFile&do=get&target=URG-Series_SCIP2_Compatible_Communication_Specification_ENG.pdf>.
- MARDER-EPPSTEIN DAVID V. LU!!;, D. H. E.** *costmap_{2d}.* Disponível em : <>.
- MOSKAL, P. L. M.** *LiDAR Fundamentals.* 2008. Disponível em: <https://www.dnr.wa.gov/publications/bc_fp_lidar_pres_moskal.ppt>.

OPEN SOURCE ROBOTICS FOUNDATION. **About ROS**. Disponível em: <<http://www.ros.org/about-ros/>>.

PETERNELLI, M. P. M. L. A. **Conhecendo o R - Uma Visão mais que Estatística**. [S.l.]: UFV, 2013.

QARRA, N. A. P. C. N. R. M. A. Z. S. E. The heartlander: A novel epicardial crawling robot for myocardial injections. Elsevier BV, 2015.

QUIGLEY, M.; CONLEY, K.; GERKEY, B. P.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. Ros: an open-source robot operating system. In: **ICRA Workshop on Open Source Software**. [S.l.: s.n.], 2009.

REILY, H. Disponível em: <<https://www.axios.com/parker-solar-probe-mission-to-touch-the-sun-cd27019d.html>>.

ROBOT, I. D. **Jaguar V4 with Arm**. Disponível em: <http://jaguar.drrobot.com/specification_V4Arm.asp>.

ROCKWELL LASER INDUSTRIES. **Laser Standards and Classifications**. Disponível em: <<http://www.rli.com/resources/articles/classification.aspx>>.

SILVA, T. O. C. **Brasil não está pronto para carro sem motorista**. Disponível em: <<https://economia.estadao.com.br/noticias/geral,brasil-nao-esta-pronto-para-carro-sem-motorista,70002202669>>.

SLINEY, J. M. D. H. **Safety with Lasers and Other Optical Sources**. [S.l.]: Springer US, 1980.

TECHOPEDIA. **Definition - What does Software Package mean?** Disponível em: <<https://www.techopedia.com/definition/4360/software-package>>.

THE ENCLOSURE COMPANY (INTERNATIONAL) LTD. **IP Rated Enclosures Explained**. Disponível em: <<http://www.enclosurecompany.com/ip-ratings-explained.php>>.

THE ROBOTIC INSTITUTE. **HeartLander**. Disponível em: <<https://www.cs.cmu.edu/~heartlander/index.html>>.

APÊNDICES

Aqui será apresentado o passo-a-passo de como replicar o que foi implementado neste trabalho e também os materiais usados.

APÊNDICE A – Ajustar e interpretar o LIDAR

A.1 PRE-CONFIGURAÇÕES

Antes de começar nos manuais de instalação, você deve fazer os tutoriais descrito nos links abaixo.

O programa usado para a criação das máquinas virtuais foi o Parallels versão 14.0.1. Você pode encontrar-lo em: <https://www.parallels.com/br/> obs: Se você estiver usando o parallels, não se esqueça de instalar também na máquina virtual o Parallels ToolBox, o qual irá ajudar bastante no desempenho e estabilidade da mesma

O sistema operacional utilizado foi o Ubuntu 18, o qual pode ser encontrado em: <<http://ubuntu.mirror.pop-sc.rnp.br/mirror/ubuntu-releases/18.04.1/ubuntu-18.04.1-desktop-amd64.iso>>

Caso você use um Mac, máquina usada nesse trabalho, use o link abaixo para particionar o disco: <<https://www.maketecheasier.com/install-dual-boot-ubuntu-mac/>>

Caso queira saber mais sobre o sensor: <<https://www.hokuyo-aut.jp/search/single.php?serial=169>>

Para Instalar o ROS Melodic: <<http://wiki.ros.org/melodic/Installation>>

Configurar workspace: <http://wiki.ros.org/catkin/Tutorials/create_a_workspace>

Para instalar o gazebo: <http://gazebosim.org/tutorials?tut=ros_installing>

A.1.1 Links interessantes

Para saber mais sobre o Rviz, acesse o link: <http://wiki.ros.org/rviz/UserGuide>

Tutorial sobre navegação autônoma: <http://moorerobots.com/blog/post/3>

A.2 LIGAR O UTM-30XL

A.2.1 Material

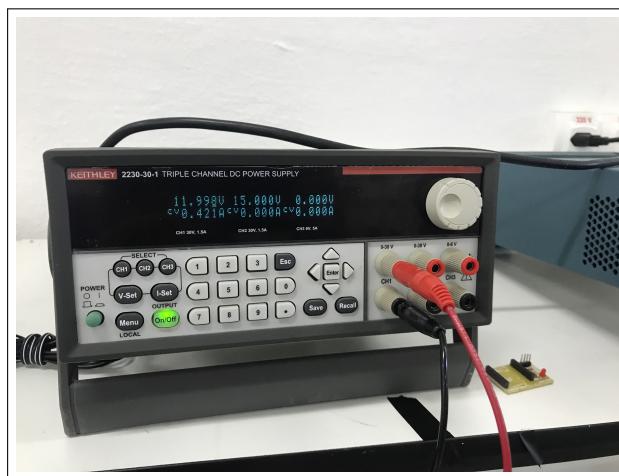
Aqui foi usado um Mac com o disco particionado em 50 gb, sistema operacional Linux Ubuntu 18.04, processador 2,2 GHz intel core i7, 16 GB 1600 MHz DDR3 de ram e Intel Iris Pro 1536 MB em gráficos. Foi também usado o sensor Hokuyo UTM-30LX Scanning Laser Rangefinder para a aquisição de dados, o Robotic Operational System (ROS) Melodic Desktop-Full, o Gazebo, e o Matlab.

A.2.2 Configuração do sensor

A.2.2.1 Alimentação

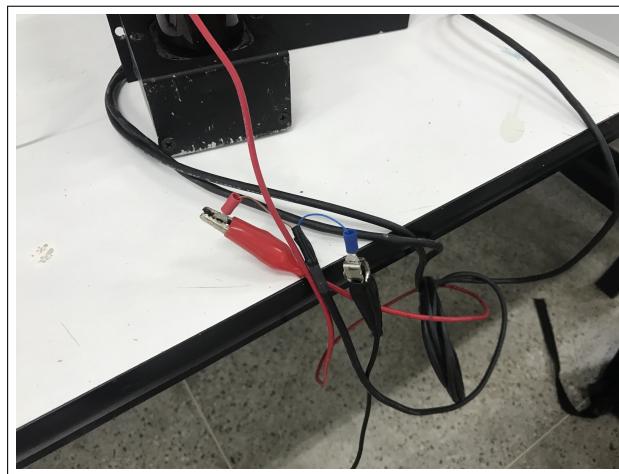
Para ligar o sensor você deve alimentar o cabo marrom com 12 volts e o cabo azul com 0 volt. O sensor consome até 1 ampere de corrente. Após os dois LEDs indicadores acenderem, um vermelho outro verde, conecte o cabo USB do sensor ao computador. Para saber mais sobre esse aparelho, leia os links abaixo. <<https://www.robotshop.com/media/files/pdf/utm-30lx-overview.pdf>> <http://www.lara.prd.fr/_media/imara/platforms/hardware/mobilink/lasers/utm-30lx_um.pdf>

Figura 13 – Fonte de Tensão



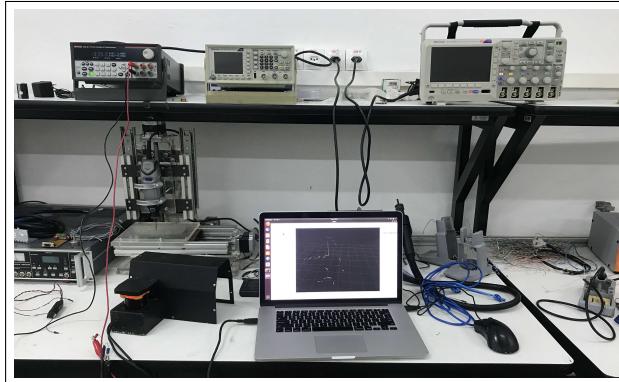
Fonte – Elaborado pelo autor

Figura 14 – Fonte Ligado no Sensor



Fonte – Elaborado pelo autor

Figura 15 – Sensor e Pc



Fonte – Elaborado pelo autor

A.2.2.2 Coletar dados

Baixe o package Laser Hokuyo - Driver Hokuyo Node em <https://github.com/ros-drivers/hokuyo_node> e cole na pasta src do seu workspace. Abra o terminal de comandos e mude o diretório para o seu workspace

```
1 cd ~/home/user/your_workspace
```

Execute o comando abaixo para configurar e instalar o package

```
1 catkin_make
```

Se houver algum erro na execução do comando anterior, refaça as etapas anteriores. Caso algum erro seja lançado na execução do comando anterior, é provável que seja necessário instalar antes outro package. Leia o apêndice Solução de Problemas.

Agora vamos saber se tudo está funcionando corretamente. Abra um terminal e initialize o roscore:

```
1 roscore
```

Execute o comando abaixo para configurar e instalar o package

```
1 catkin_make
```

Se houver algum erro na execução do comando anterior, refaça as etapas anteriores. Caso algum erro seja lançado na execução do comando anterior, é provável que seja necessário instalar antes outro package. Leia em Solução de Problemas.

Agora vamos saber se tudo está funcionando corretamente. Abra um terminal e initialize o roscore:

```
1 roscore
```

Abra um novo terminal e execute

```
1 source ~/your_workspace/devel/setup.bash
2 rosrun hokuyo_node hokuyo_node
```

Caso tenha algum problema aqui, verifique se há correção na sessão Solução de Erros

Obs: você deve executar o comando

```
1 source ~/your_workspace/devel/setup.bash
```

toda vez que abrir um novo terminal para associar o mesmo com o seu workspace, como discutido aqui: <<https://answers.ros.org/question/206876/how-often-do-i-need-to-source-setupbash/>>

Se tudo tiver ocorrido bem, o topic /scan já deve ter sido criado para receber os dados do LIDAR. Você pode verificar se os dados realmente estão chegando. Abra um novo terminal e digite

```
1 rostopic list
```

Deve aparecer um topic de nome /scan

Digite

```
1 rostopic echo /scan
```

A leitura do sensor deve aparecer na tela.

A.2.2.3 visualizar dados

Para executar o Rviz execute o seguinte comando:

Rosrun rviz rviz

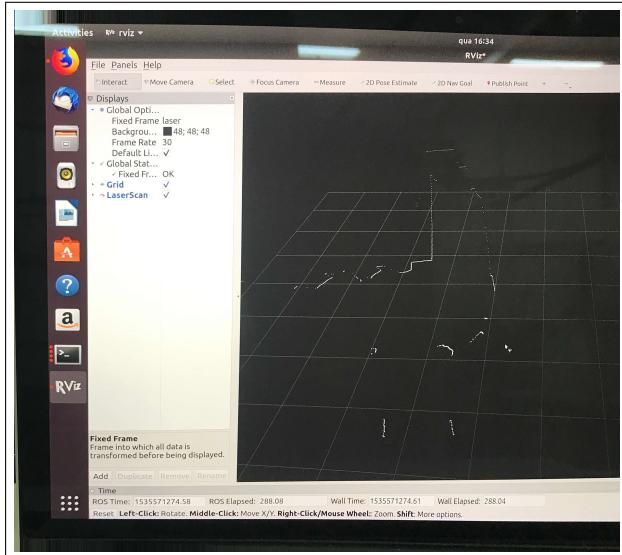
Na side bar do lado esquerdo, no canto inferior clique no botão “ADD” e procure o tópico /scan na lista, na janela que irá se abrir. Após adicionar o tópico /scan marque a check box na side bar. E você deverá visualizar os dados do laser.

Figura 16 – Dados do Sensor

Fonte – Elaborado pelo autor

No rviz deverá ser parecido como na imagem abaixo (no formato da sua sala).

Figura 17 – LIDAR no rviz



Fonte – Elaborado pelo autor

A.3 PROCESSAMENTO DE DADOS

Baixe os packages `slam_gmapping`, `open_slamgmapping`, `gmapping` nos links abaixo:

<https://github.com/ros-perception/openslam_gmapping> <https://github.com/ros-perception/slam_gmapping> Obs: você deve refazer os mesmos passos feitos para a instalação do driver do sensor. Para cada package adicionado na pasta src, você terá que executar um catkin_make.

Abra o terminal e execute

```
1 cd /home/user/your_workspace
```

Cole primeiro o package slam_gmapping em src no seu workspace, em seguida execute

```
1 catkin_make
```

Cole o package open_slamgmapping em src e execute

```
1 catkin_make
```

Por fim, cole gmapping em src e execute

```
1 catkin_make
```

Se tudo tive ocorrido bem, o package gmapping estará pronto para uso.

A.3.1 Gmapping

Encerre todas as execuções que envolvam o ROS. Para tal, recomendo que reinicie o computador. Em seguida: Abra um novo terminal e execute:

```
1 roscore
```

Abra um segundo terminal:

```
1 source ~/your_workspace/devel/setup.bash
2 rosrun hokuyo_node hokuyo_node
```

Abra um terceiro terminal:

```
1 rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 laser odom
100
```

Abra um quarto terminal:

```
1 rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 odom map
100
```

Execute em um sexto terminal:

```
1 rosrun gmapping slam_gmapping scan:=scan _linearUpdate:=0.0
   _angularUpdate:=0.0 _map_update_interval:=0.0 _particles:=3 _delta
   :=0.1 _base_frame:=laser
```

Para mais informações sobre o comando acima, leia Para mais informações:

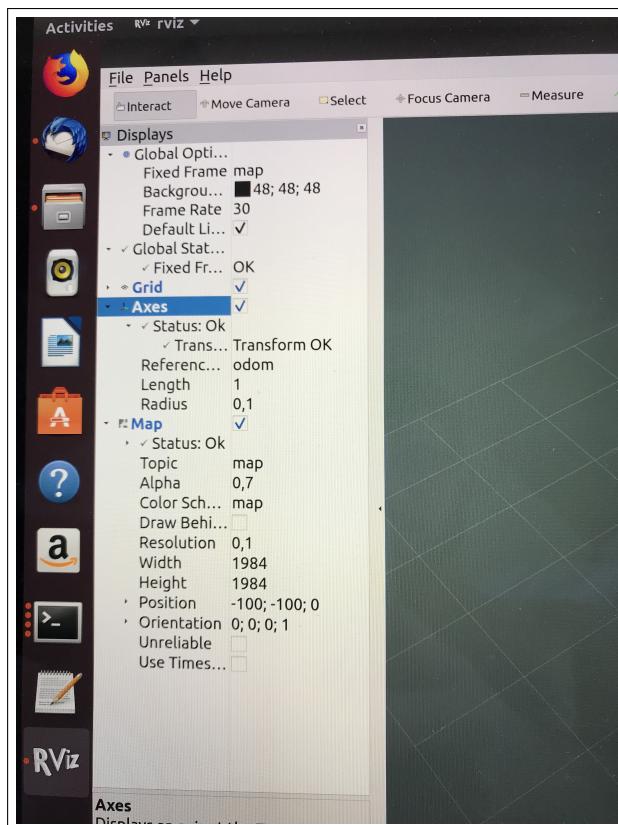
<http://wiki.ros.org/tf>

Execute em um sétimo terminal:

```
1 rviz rviz
```

No rviz, clique no botão ADD, no canto inferior esquerdo, e adicione Map e Axes. Em seguida, ajuste o topic do Map para /map, o topic do Axes para /odom e o Fixed Frame do Global Options para map. Após as configurações deverá parecer como o da foto.

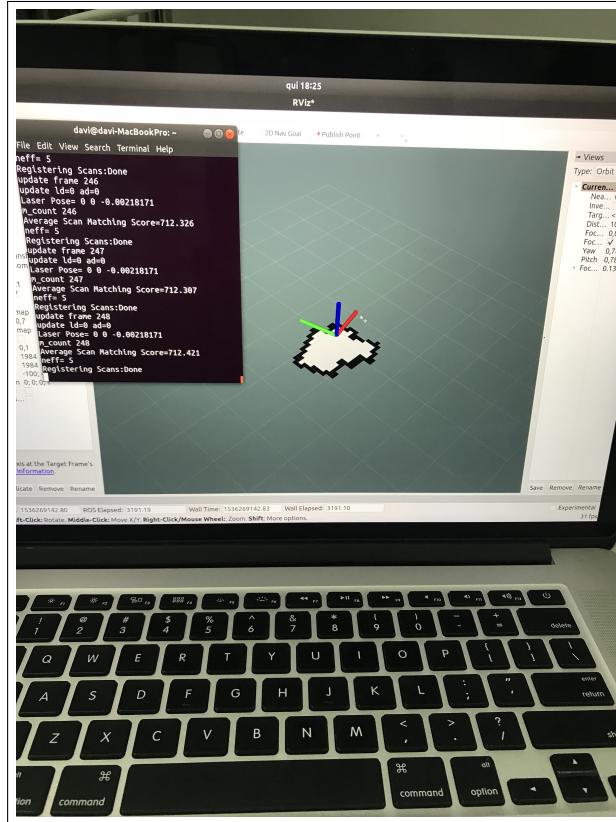
Figura 18 – Configuração Rviz



Fonte – Elaborado pelo autor

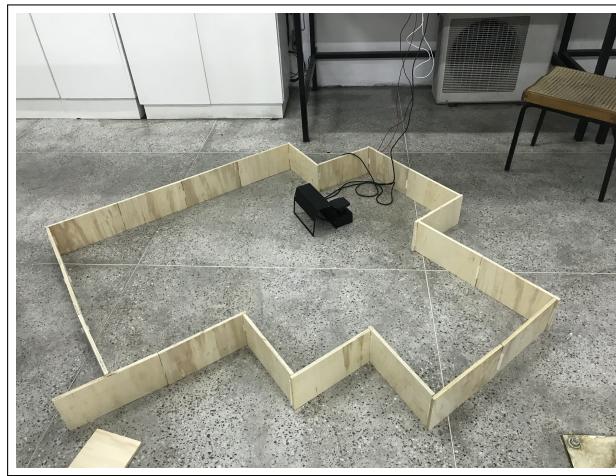
Se o sensor funcionar corretamente, seus dados podem ser interpretados e visualizados. Como citado na fundamentação teórica, o pacote até então utilizado, o gmapping, necessita

Figura 19 – Gmapping Sendo Executado



Fonte – Elaborado pelo autor

Figura 20 – Labirinto



Fonte – Elaborado pelo autor

de um sistema de odometria. Dessa forma, percebeu-se que o hector-slam é o mais adequado, já que esse trabalho se limita somente a captação dos dados do LIDAR, que portanto, não oferece um sistema de odometria.

APÊNDICE B – Jaguar Autônomo

B.1 RECURSOS

Para essa etapa, foi utilizado: MacBook Pro (Retina, 15-inch, Mid 2015) 2,2 GHz Intel Core i7 16 GB 1600 MHz DDR3 Intel Iris Pro 1536 MB macOS Mojave Version 10.14.1 Parallels Desktop 14 home edition Ubuntu 18 ROS Melodic

B.2 PRÉ CONFIGURAÇÃO

O projeto foi baseado no tutorial abaixo, no qual trata sobre navegação autônoma em um simulador: <http://moorerobots.com/blog/post/3> Utilizando o meu workspace, <<https://github.com/DaviGuanabara/jaguar-hokuyo-hector-slam>>, é possível fazer a etapa Mapping and Navigation: Our Robot do tutorial do link acima.

B.3 ROS WORKSPACE

Feche todos os terminais abertos até então.

abra o terminal e digite:

```

1 git clone https://github.com/DaviGuanabara/jaguar-hokuyo-hector-slam.
2 git
3 mkdir my_workspace
4 cd my_workspace
5 mkdir src
6 cd jaguar-hukuyo-hector-slam/jaguar\[_\]ws/src
7 ls

```

siga o exemplo abaixo para todas as pastas mostradas no terminal

```

1 sudo cp -r driver_common ~/jaguar-hukuyo-hector-slam/jaguar\[_\]ws/src
2 cd ..
3 cd ..
4 cd my_workspace
5 catkin_make
6 cd ..

```

```
7 | cd jaguar-hukuyo-hector-slam/jaguar\[_\]ws/src
```

Se tudo ocorrer bem, você terá instalado no seu computador todos os arquivos pacotes necessários.

B.4 APLICAÇÃO NO JAGUAR

```
1 roscore
```

Para acessar remotamente os dados do sensor do jaguar, conecte com a rede do mesmo e execute no terminal:

```
1 sudo mknod -m 666 /dev/ttyS51 c 4 115
2 sudo socat PTY,link=/dev/ttyS51, TCP4:192.168.0.60:10002
```

```
1 sudo chmod a=r+w /dev/pts/*
2 rosparam set hokuyo_node/port /dev/ttyS51
3 rosrun hokuyo_node hokuyo_node
```

em outro terminal, digite:

Para ativar o nó que controla o jaguar, digite:

```
1 roslaunch drrobotV2_player jaguar_v4.launch
```

abra mais um terminal e digite:

```
1 roslaunch hector_slam_example hector_simulador_map.launch
```

por fim, conecte o nó dos comandos gerados pelo rviz com o nó de envio dos comandos para o jaguar:

```
1 rosrun topic_tools relay cmd_vel drrobot_cmd_vel
```

Pronto, o robô já está pronto para ser controlado autonomamente. Você só precisa dizer aonde o robô deve ir no rviz, e os dados gerados irão diretamente para o jaguar.

APÊNDICE C – Solução de problemas identificados na primeira etapa do projeto

C.1 DEBUG

Para debugar alguma execução no ROS, abra um novo terminal e digite

```
1 rosrun rqt_logger_level rqt_logger_level
```

Obs: não debugar o laser! o programa vai travar e será necessário reiniciar o pc

C.2 PARTICIONAMENTO DO DISCO

Ao tentar fazer o dual boot, não conseguia adicionar uma partição ao disco. Mesmo tendo bastante espaço para o mesmo, o sistema acusava de não haver espaço suficiente. O problema foi gerado pelo Time Machine Snapshot, responsável pelo backup da máquina. Esse problema foi solucionado com a remoção dos Snapshots. A solução encontra-se em: <<https://www.macworld.com/article/3260635/mac/how-to-delete-time-machine-snapshots-on-your-mac.html>>

Não consegui instalar no próprio mac.

C.3 HOKUYO

Instalação do Hokuyo Node:

Caso seja lançado uma mensagem de erro, durante a execução do catkin_make, dizendo que faltou arquivos ou packages, você deve adicionar outros packages na pasta src do seu workspace (a mesma pasta do Hokuyo_node)

Para solucionar, vc deve fazer:

1. Retirar todos os packages da pasta src
 2. Adicionar o package driver_common na pasta src, este você pode encontrar em <https://github.com/ros-drivers/driver_common.git>
 3. Adicionar o package Hokuyo_node na pasta src
 4. Executar catkin_make no terminal no diretório do seu workspace
- <<https://answers.ros.org/question/240235/how-to-install-packages-from-github/>>

A idéia, como pode-se notar, é executar o catkin_make toda vez que um novo pacote for adicionado ao workspace. Além disso, devemos também adicionar somente um pacote por vez, para evitar problemas. Outros pacotes que podem ser requisitados:

<https://github.com/ros-drivers/urg_c> <https://github.com/ros-drivers/urg_node>

Hokuyo failed to open port – No authorized

Caso haja problema ao executar

```
1 rosrun hokuyo_node hokuyo_node
```

Por não ter autorização para acessar a porta usb, você deve ir para a raiz do sistema:

```
1 cd
```

e execute:

```
1 sudo chmod a+r /dev/ttyACM0
```

Como na imagem abaixo.

```
davi@davi-MacBookPro: ~
File Edit View Search Terminal Help
ubleshooting
ls /dev/ttyACM*
^C
davi@davi-MacBookPro:~$ ls /dev/ttyACM*
/dev/ttyACM0
davi@davi-MacBookPro:~$ SUBSYSTEMS=="usb", KERNEL=="ttyACM[0-9]*", ATTRS{manufacturer}=="Hokuyo Data Flex for USB", ATTRS{product}=="URG-Series USB Driver", PRO
GRAM=="/etc/ros/run.sh hokuyo_node getID %N q", SYMLINK+="sensors/hokuyo_%c"
ATTRS{manufacturer}==Hokuyo Data Flex for USB,: command not found
davi@davi-MacBookPro:~$ rosrun hokuyo_node hokuyo_node
[ERROR] [1535568794.738815950]: Exception thrown while opening Hokuyo.
Failed to open port: /dev/ttyACM0. Permission denied (errno = 13). You probably
don't have premission to open the port for reading and writing. (in hokuyo::lase
r::open) You may find further details at http://www.ros.org/wiki/hokuyo_node/Trou
bleshooting
^C
davi@davi-MacBookPro:~$ ls -l /dev/ttyACM0
crw-rw---- 1 root dialout 166, 0 ago 29 15:48 /dev/ttyACM0
davi@davi-MacBookPro:~$ sudo chmod a+r /dev/ttyACM0
davi@davi-MacBookPro:~$ rosrun hokuyo_node hokuyo_node
[ INFO] [1535568959.150482035]: Connected to device with ID: H1304418
[ INFO] [1535568959.175099938]: Starting calibration. This will take up a few se
conds.
[ INFO] [1535568959.947414846]: Calibration finished. Latency is: -0.0226
[ INFO] [1535568960.025030741]: Streaming data.
```

Instalar o hokuyo_node

add o driver_common catkin_make dps add hokuyo_node catkin_make

The requested port does not exist

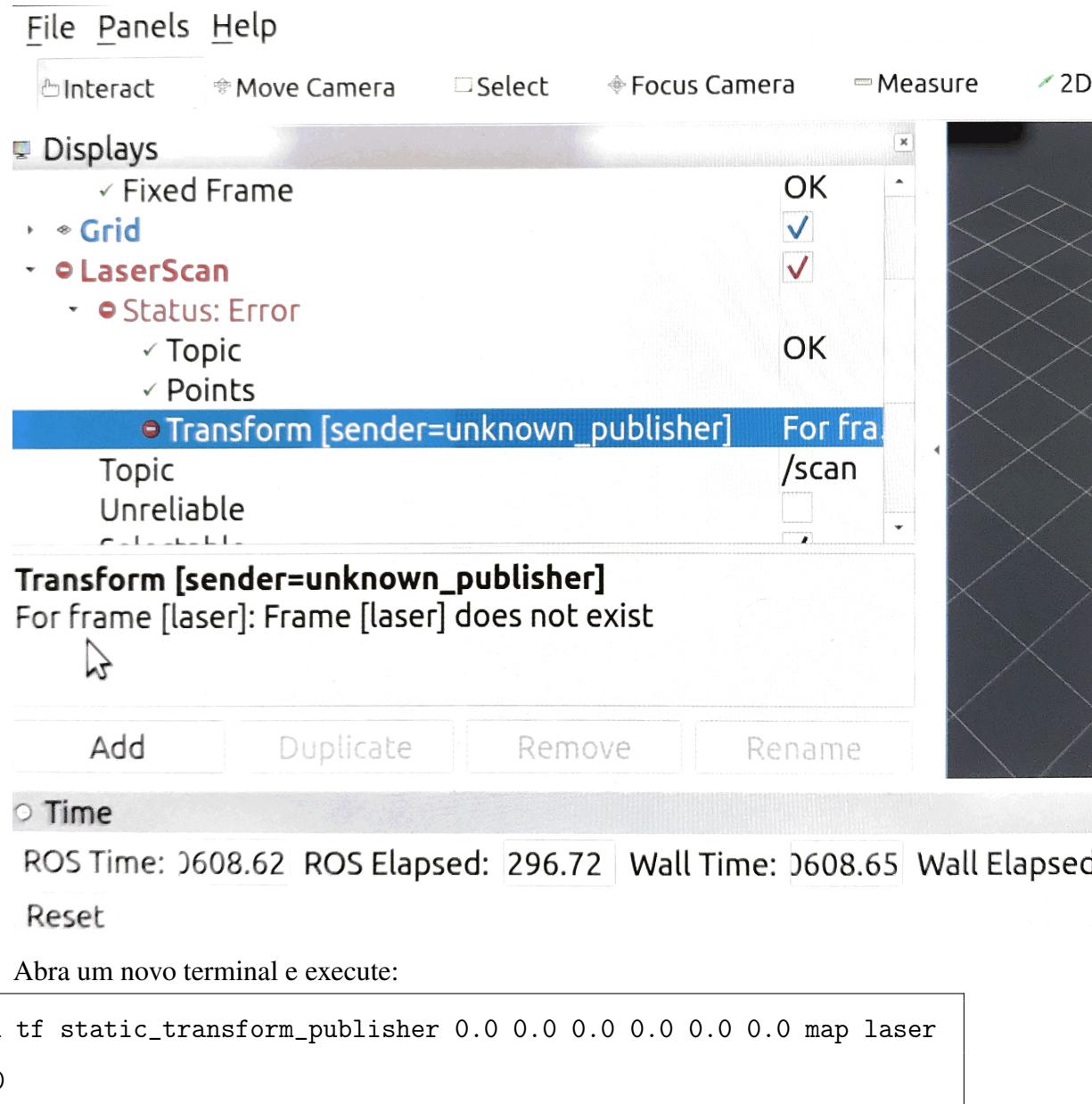
[ERROR] [1536087827.457595928]: Exception thrown while opening Hokuyo.

Failed to open port: /dev/ttyACM0. No such file or directory (errno = 2). The requested port does not exist. Is the hokuyo connected? Was the port name misspelled? (in hokuyo::laser::open)

You may find further details at <http://www.ros.org/wiki/hokuyo_node/Troubleshooting>

Verifique se o sensor está ligado, e o cabo conectado corretamente ao computador. Remova o cabo USB por alguns minutos e conecte-o novamente. Se o erro insistir, reinicie o computador.

C.4 ERRO AO TESTAR O LASER NO RVIZ TRANSFORM [SENDER=UNKNOWN_PUBLISHER]



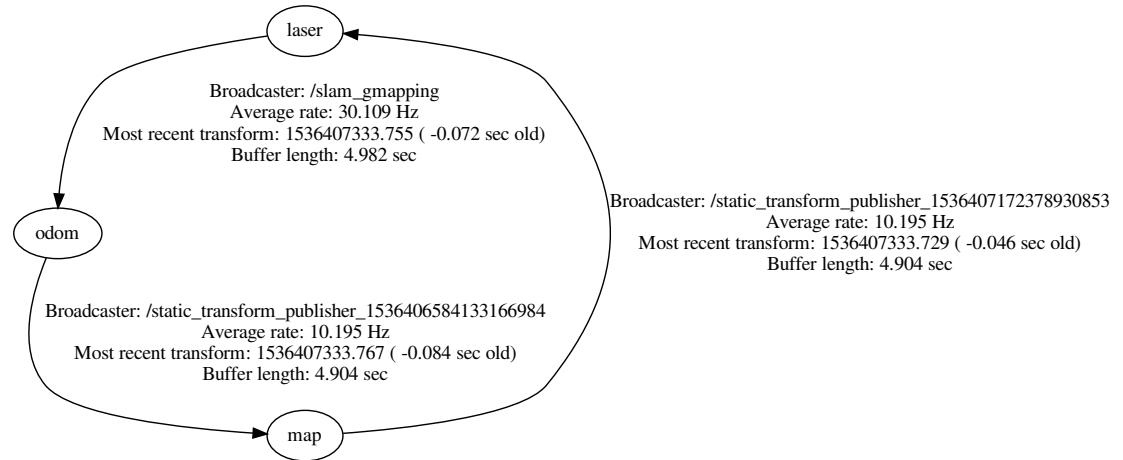
se algo n funcionar mais, refazer o workspace

C.5 PROBLEMAS QUANTO AOS FRAMES DO GMAPPING

Se voce executar em um terminal:

```
1 rosrun tf view_frames
```

um arquivo como na imagem abaixo deverá ser gerado



Se não estiver nesse formato, encerre todas as execuções que contenham “`static_transform`”
abra um novo terminal e execute:

```
1 rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 laser odom 100
```

Abra mais um terminal e execute:

```
1 rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 odom map 100
```

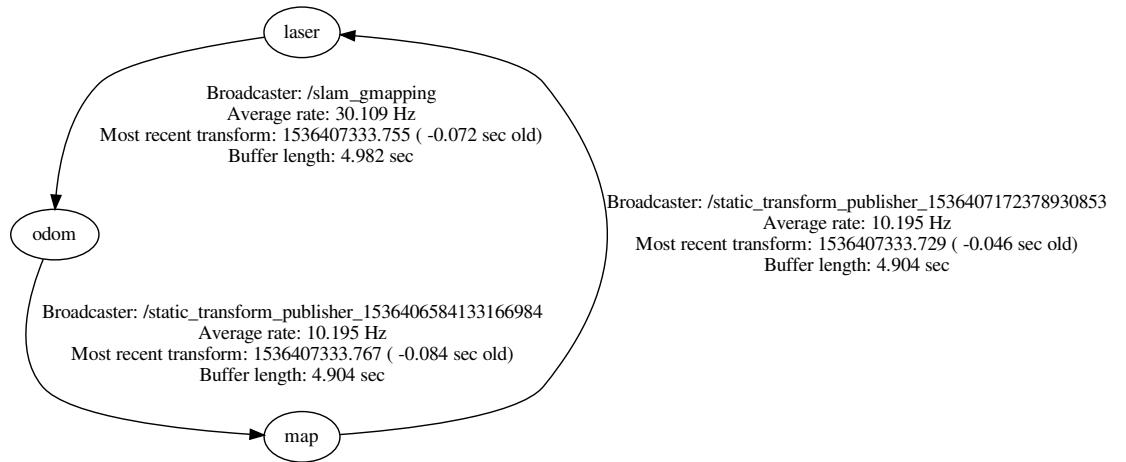
Por fim, execute

```
1 rosrun tf static_transform_publisher 0.0 0.0 0.0 0.0 0.0 0.0 map laser 100
```

Pronto! Ao executar

```
1 rosrun tf view_frames
```

A imagem abaixo deverá ser gerada.



C.6 COMADOS INTERESSANTES AO USAR O TERMINAL

para mostrar todas a funções associadas ao nome escrito no terminal, nome + tab (x3) + y

Cancelar uma execução: control + c (x2)

top

APÊNDICE D – hector-simulador-map.launch

```

1 <?xml version="1.0"?>
2
3 <launch>
4   <!--param name="use_sim_time" value="true"-->
5
6   <include file="$(find hector_slam_example)/launch/default_mapping.
7     launch"/>
8
9   <param name="use_sim_time" type="bool" value="True"/>
10  <param name="pub_map_scanmatch_transform" value="true"/>
11  <param name="tf_map_scanmatch_transform_frame_name" value="
12    scanmatch_frame"/>
13  <param name="pub_map_odom_transform" value="true"/>
14  <param name="map_frame" value="map"/>
15  <param name="base_frame" value="base_fame"/>
16  <param name="odom_frame" value="base_fame"/>
17
18
19
20  <node pkg="tf" type="static_transform_publisher" name="map_2_odom"
21    args="0.0 0.0 0.0 0.0 0.0 0.0 /map /odom 100" />
22
23
24  <node pkg="tf" type="static_transform_publisher" name="
25    odom_2_base_frame"
26    args="0.0 0.0 0.0 0.0 0.0 0.0 /odom /base_frame 100" />
27
28
29  <node pkg="tf" type="static_transform_publisher" name="
30    base_frame_2_hokuyo"
31    args="0.0 0.0 0.0 0.0 0.0 0.0 /base_frame /laser 100" />
32
33
34  <!--node pkg="tf" type="static_transform_publisher" name="
35    scanmatch_2_hokuyo"-->
```

```

27   args="0.0 0.0 0.0 0.0 0.0 0.0 /scanmatch_frame /laser 100" /-->
28
29
30
31 <node pkg="rviz" type="rviz" name="rviz" args="-d $(find
32   hector_slam_example)/launch/rviz_cfg.rviz"/>
33
34 <!-- Move base -->
35 <node pkg="move_base" type="move_base" respawn="false" name="
36   move_base" output="screen">
37   <rosparam file="$(find mybot_navigation)/config/
38     costmap_common_params.yaml" command="load" ns="global_costmap" />
39
40   <rosparam file="$(find mybot_navigation)/config/
41     costmap_common_params.yaml" command="load" ns="local_costmap" />
42   <rosparam file="$(find mybot_navigation)/config/
43     local_costmap_params.yaml" command="load" />
44   <rosparam file="$(find mybot_navigation)/config/
45     global_costmap_params.yaml" command="load" />
46   <rosparam file="$(find mybot_navigation)/config/
47     base_local_planner_params.yaml" command="load" />
48
49   <remap from="cmd_vel" to="cmd_vel"/>
50   <remap from="odom" to="/odom"/>
51   <remap from="scan" to="/scan"/>
52   <param name="move_base/DWAPlannerROS/yaw_goal_tolerance" value="1.0"
53     />
54   <param name="move_base/DWAPlannerROS/xy_goal_tolerance" value="1.0"
55     />
56
57 </node>
58
59 </launch>

```

APÊNDICE E – base-local-planner

```
1
2 TrajectoryPlannerROS:
3   max_vel_x: 0.5
4   min_vel_x: 0.01
5   max_vel_theta: 1.5
6   min_in_place_vel_theta: 0.01
7
8   acc_lim_theta: 1.5
9   acc_lim_x: 0.5
10  acc_lim_y: 0.5
11
12
13  holonomic_robot: false
```

APÊNDICE F – global-costmap-params

```
1 # static_map - True if using existing map
2
3 global_costmap:
4   global_frame: map
5   robot_base_frame: scanmatch_frame
6   update_frequency: 1.0
7   publish_frequency: 1.0
8   resolution: 0.05
9   static_map: true
10  width: 10.0
11  height: 10.0
```

APÊNDICE G – local-costmap-params

```
1 local_costmap:  
2   global_frame: map  
3   robot_base_frame: scanmatch_frame  
4   update_frequency: 2.0  
5   publish_frequency: 1.0  
6   static_map: false  
7   rolling_window: true  
8   width: 6.0  
9   height: 6.0  
10  resolution: 0.05
```

APÊNDICE H – costmap-common-params

```
1 obstacle_range: 2.5
2 raytrace_range: 3.0
3 #footprint: [[x0, y0], [x1, y1], ... [xn, yn]]
4 #robot_radius: ir_of_robot
5 robot_radius: 0.5 # distance a circular robot should be clear of the
6   obstacle
7 inflation_radius: 3.0
8 transform_tolerance: 500.0
9
10 # marking - add obstacle information to cost map
11 # clearing - clear obstacle information to cost map
12 laser_scan_sensor: {sensor_frame: hokuyo, data_type: LaserScan, topic:
13   /scan, marking: true, clearing: true}
14 #point_cloud_sensor: {sensor_frame: frame_name, data_type: PointCloud,
15   topic: topic_name, marking: true, clearing: true}
```

APÊNDICE I – mybot_{worl}d

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <launch>
3
4   <arg name="world" default="empty"/>
5   <arg name="paused" default="false"/>
6   <arg name="use_sim_time" default="false"/>
7   <arg name="gui" default="true"/>
8   <arg name="headless" default="false"/>
9   <arg name="debug" default="false"/>
10
11  <include file="$(find gazebo_ros)/launch/empty_world.launch">
12    <!--arg name="world_name" value="$(find mybot_gazebo)/worlds/mybot.
13      world"-->
14    <arg name="world_name" value="$(find mybot_gazebo)/worlds/
15      turtlebot_playground.world"/>
16    <arg name="paused" value="$(arg paused)"/>
17    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
18    <arg name="gui" value="$(arg gui)"/>
19    <arg name="headless" value="$(arg headless)"/>
20    <arg name="debug" value="$(arg debug)"/>
21  </include>
22
23
24  <param name="robot_description" command="$(find xacro)/xacro $(find
25    mybot_description)/urdf/mybot.xacro"/>
26
27  <node name="mybot_spawn" pkg="gazebo_ros" type="spawn_model" output=
28    "screen"
29    args="-urdf -param robot_description -model mybot" />
30
31</launch>
```

ANEXOS