



# INTEGRAÇÃO DE ROBÔS MÓVEIS COM LIDAR E ROS PARA IMPLEMENTAÇÃO DE LOCALIZAÇÃO E MAPEAMENTO



## **Orientador(s):**

Drº. Paulo César M. de Abreu Farias  
Drº. Tiago Trindade Ribeiro

**Salvador (BA)**

## **Discentes:**

Dhavidy de Almeida Silva  
Erick Bragança Martins Santos  
Lucas Pinheiro Soares

# Introdução

- Objetivos gerais e específicos



# Objetivos gerais e específicos

## Objetivo geral

*“Implementar um sistema de localização para um robô móvel car-like equipado com LiDAR 2D, utilizando ROS com algoritmos de SLAM para mapear e estimar a posição do robô.”*



# Objetivos gerais e específicos

## Objetivos específicos

- Adaptar a comunicação do robô para tópicos ROS.
- Integrar sensores de acelerômetro e giroscópio.
- Programar e implementar a odometria do robô.
- Aplicar algoritmos de SLAM utilizando o filtro de Kalman.
- Validar a solução em testes simulados e reais.



# Fundamentação Teórica

- Robô Car-Like
- Odometria
- ROS
- Controle

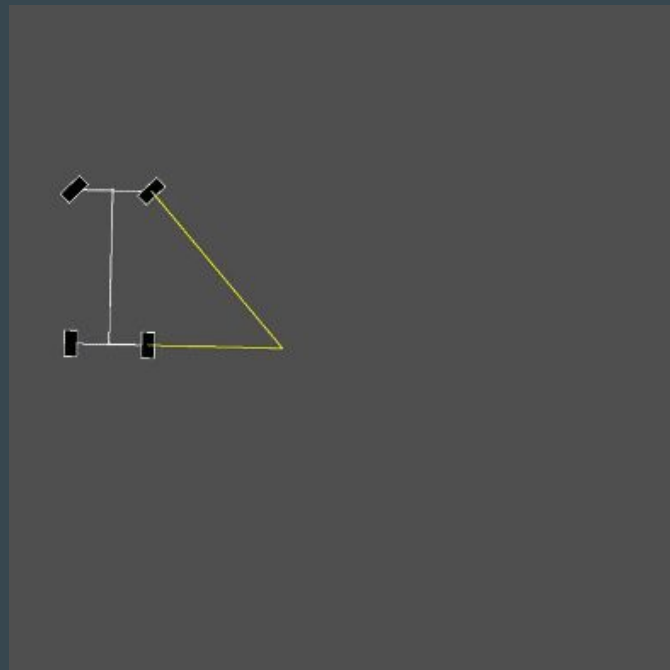
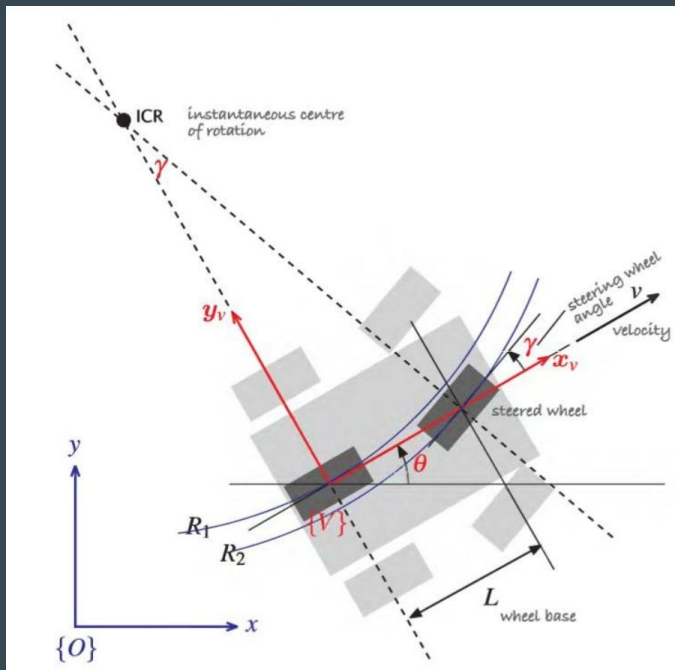


# Robô *Car-Like*

- Robôs *car-like* e direção de Ackermann



# Robôs *car-like* e direção de Ackermann



# Protótipo

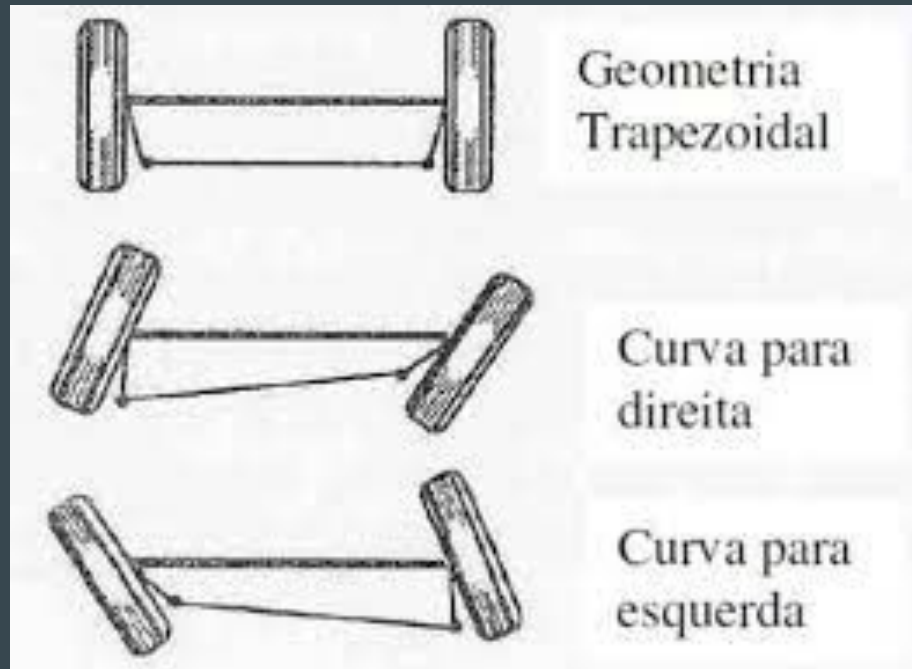
- Escolha dos componentes
- Diagrama elétrico
- Modelagem 3D
- Aquisição e controle





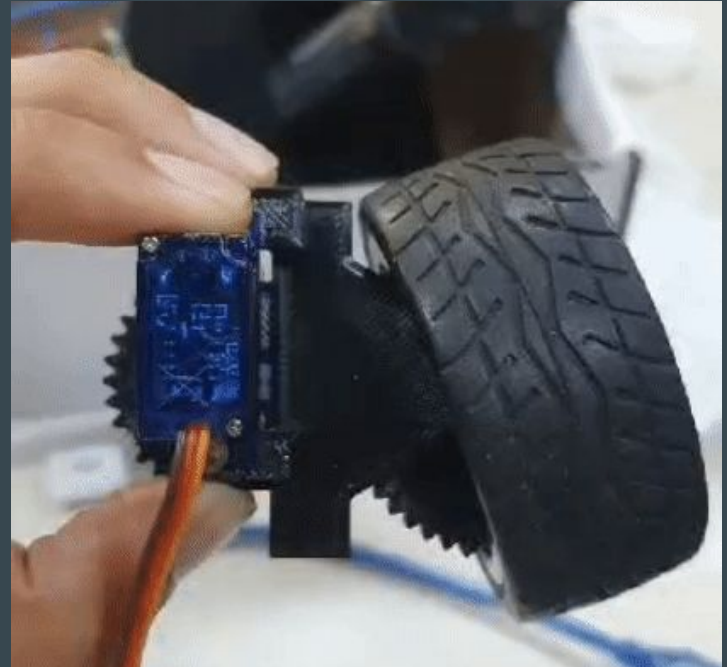
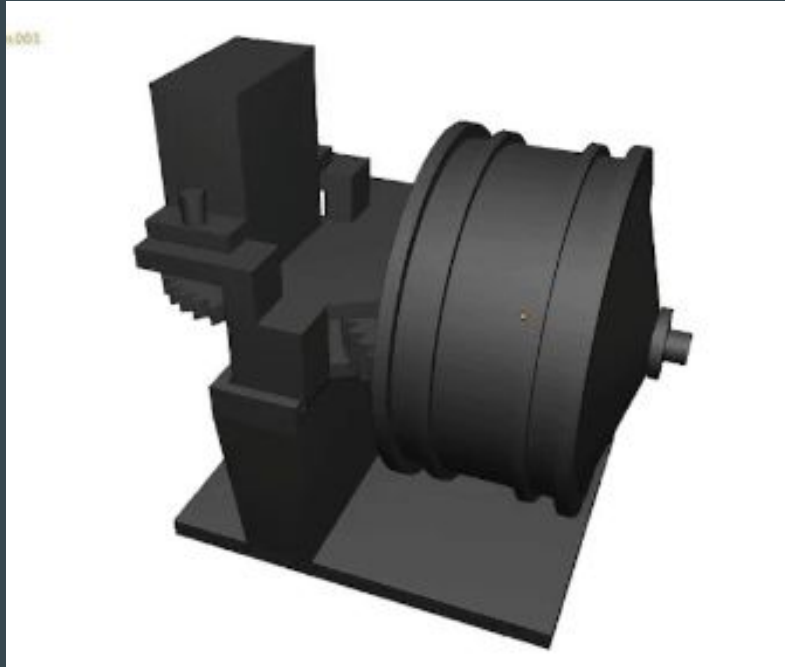
# Escolha dos componentes

## Direção



# Modelagem 3D

## Sistema de direção



# Escolha dos componentes

## Motor CC

- **Redução** 34,02:1
- 210 RPM
- 10 Kg•cm
- **Encoder** de quadratura de 11 pulsos por revolução



# Escolha dos componentes

## Bateria

- Samsung ICR18650-26F
- 18mm x 65mm
- 3,7V

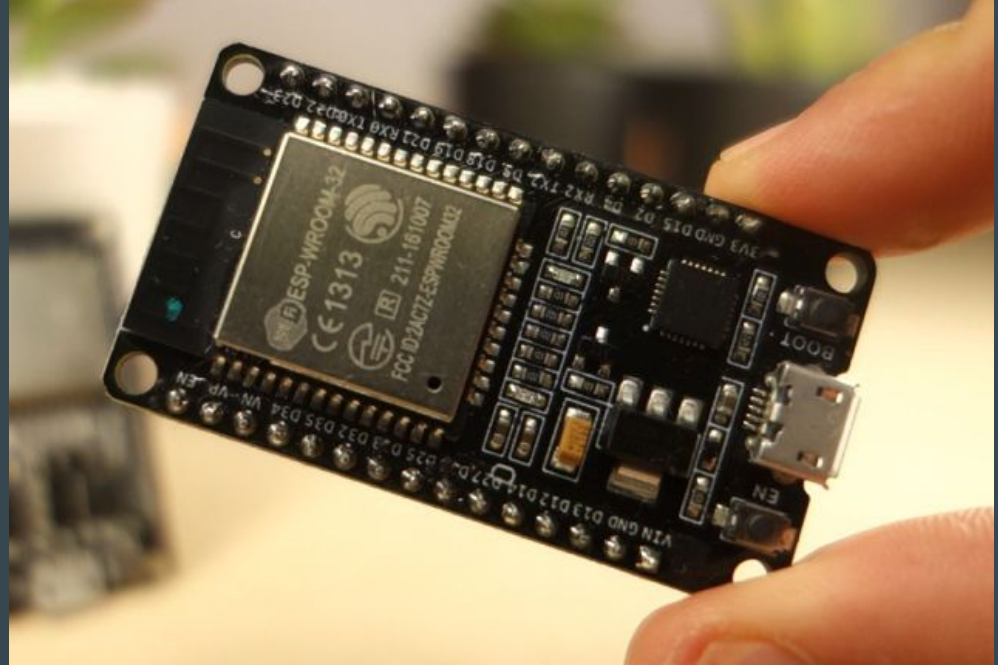
$$t_h = \frac{C_{min} \cdot \eta(I)}{I} \rightarrow \frac{2550mAh \cdot 0,950,5C}{2 \cdot 600mA + 100mA} = 1,86h.$$



# Escolha dos componentes

## Plataforma computacional

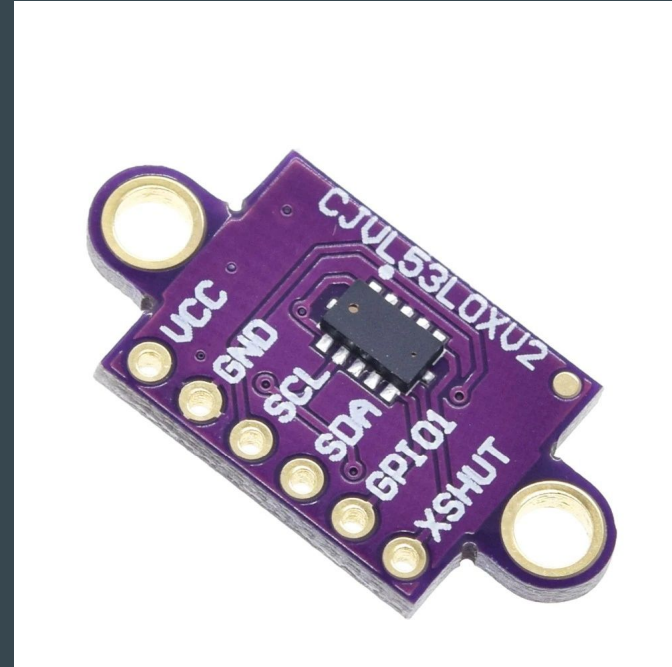
- ESP32 x Placas Jetson x Raspberry Pi



# Escolha dos componentes

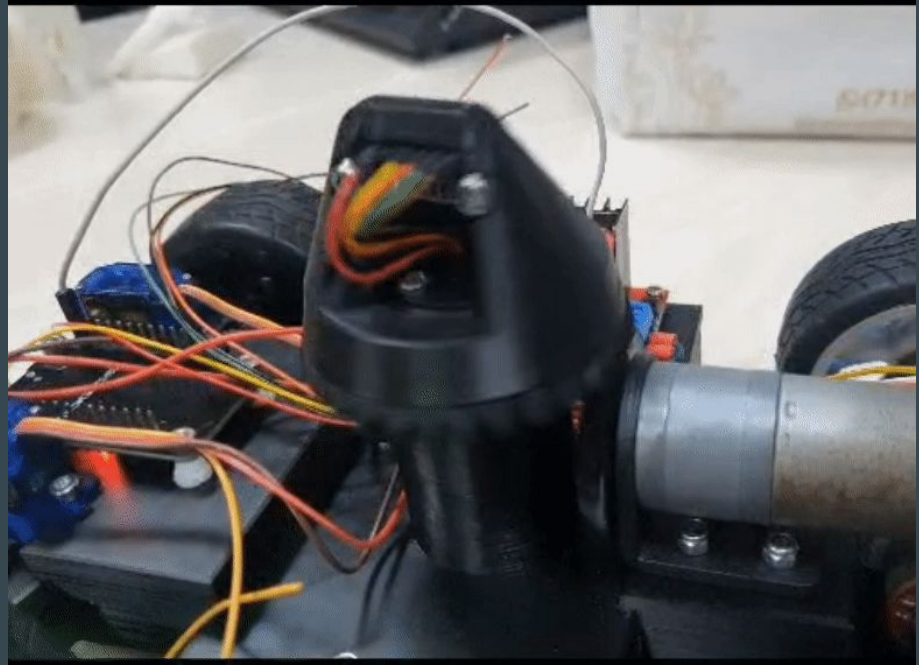
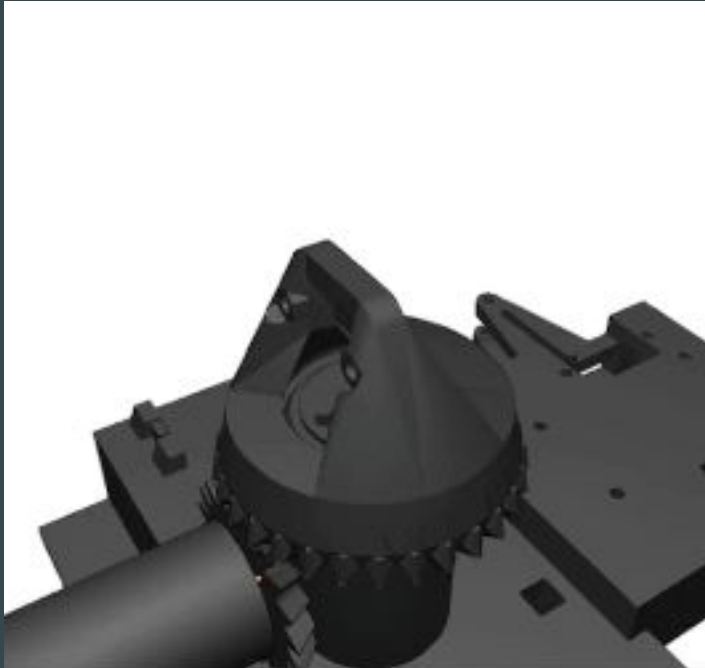
## Sensor de distância

- VL53L0X
  - Taxa de aquisição de até 50 Hz
  - Alcance de até 2 m
  - Precisão 1mm
  - Cone de visão de 25º
  - I2C



# Modelagem 3D

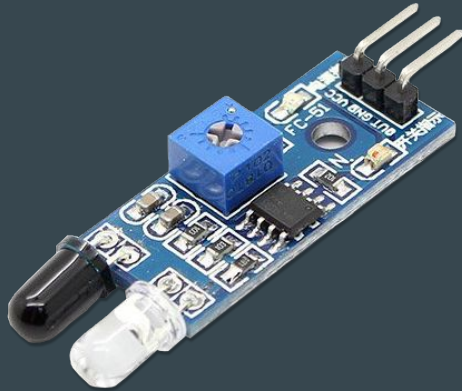
## LiDAR v2





# Escolha dos componentes

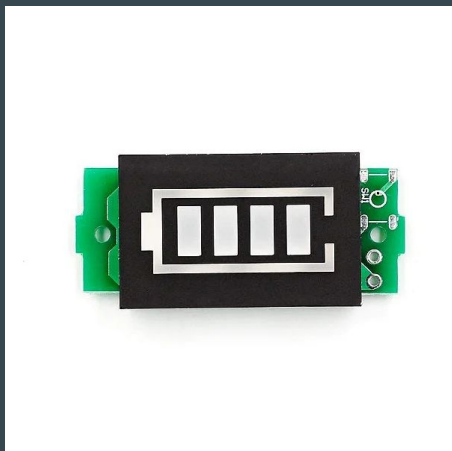
## Outros componentes



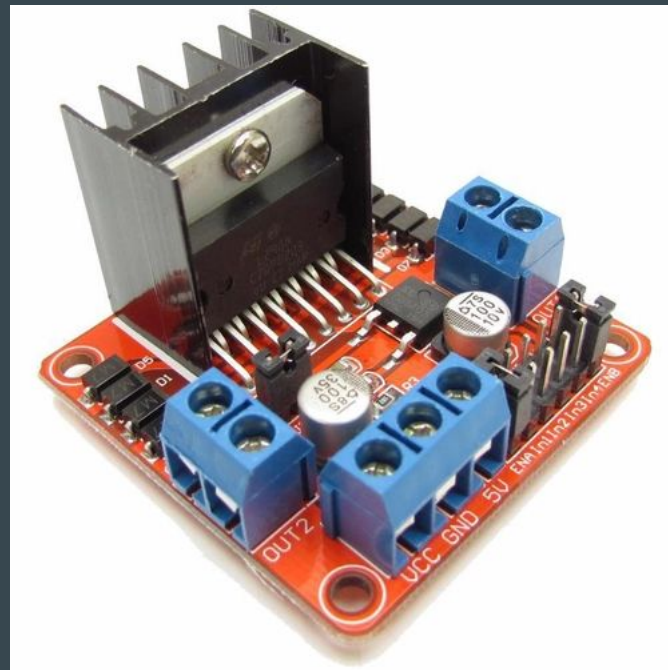
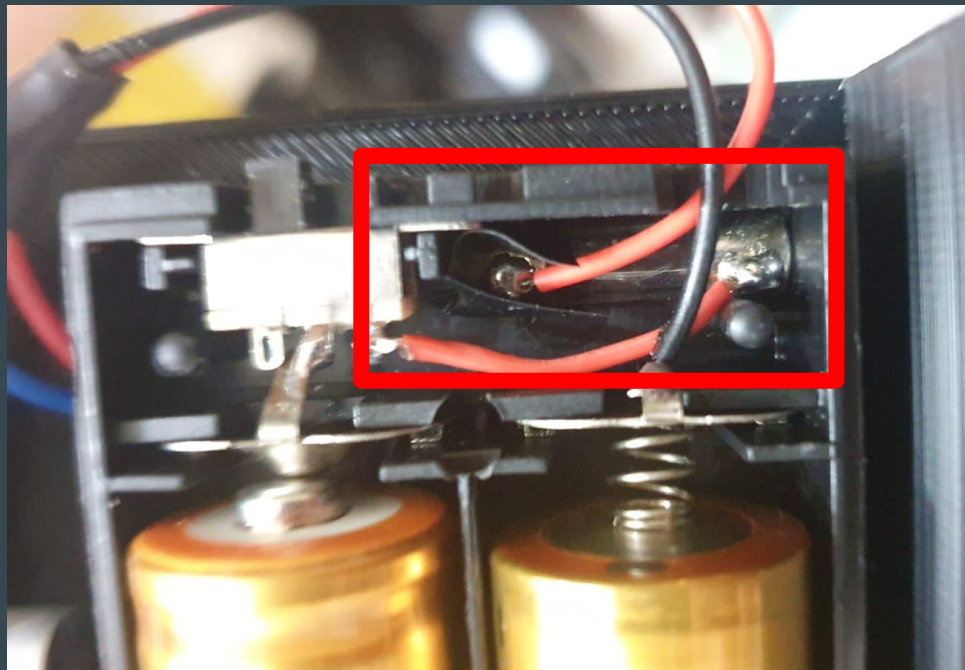


# Escolha dos componentes

## Outros componentes

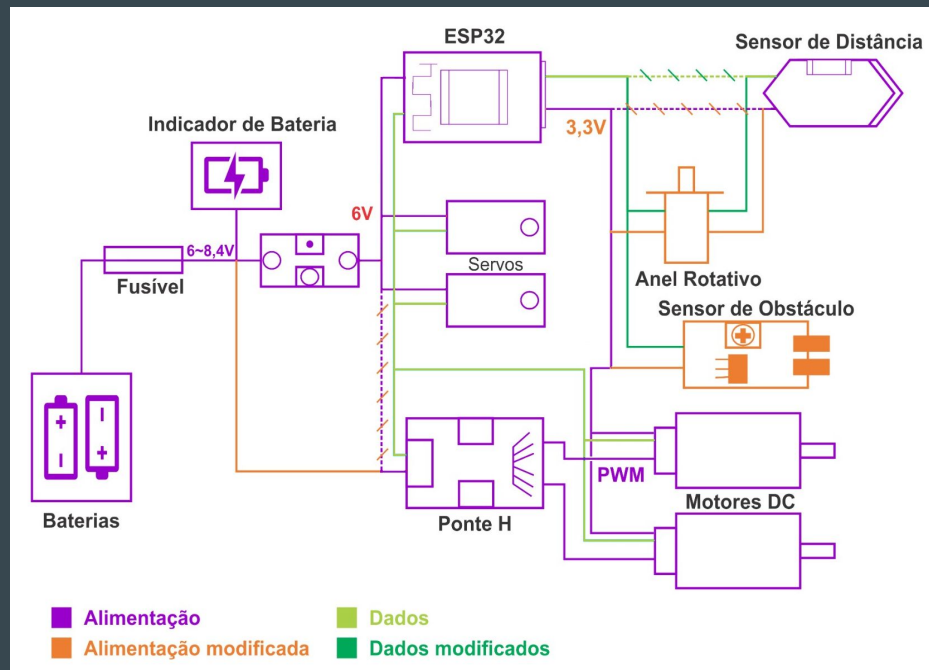


## Outros componentes



# Diagrama Elétrico

## Diagrama



# Modelagem 3D

## Chassi



# Aquisição e controle

## Comunicação

- Wi-Fi: “UDP\_Comm”
- Protocolo UDP
  - IP Host: 192.168.4.22
  - IP Cliente: 192.168.4.23
  - Port: 8080

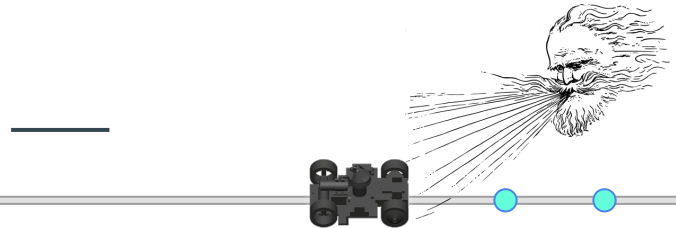
```
[Quantidade_de_Pontos]_[Dist0]_[Dist1]..._[Dist15]  
[Velocidade]_[Esterçamento]
```

Comando	Valor auxiliar	Descrição
go	0 255	Comando de controle do PWM de motor do LiDAR
Parar	X	Comando para parar a movimentação do protótipo
Points	4 16	Comando para configurar a quantidade de pontos por revolução do LiDAR
Amostragem	50 5000	Comando para configurar o tempo de amostragem da velocidade e ângulo das rodas
Deg	0 255	Comando de controle do PWM de motor de tração
Vel	0 500	Comando de controle de velocidade do protótipo em mm/s
Acelerar	X	Comando para acelerar o protótipo
Re	X	Comando para desacelerar o protótipo
Dir	-30 30	Comando para configurar o ângulo de esterçamento da roda virtual do modelo
Direita	X	Comando para incrementar a direção para a direita
Esquerda	X	Comando para incrementar a direção para a esquerda
off	X	Comando para desativar o LiDAR



# Acelerômetro

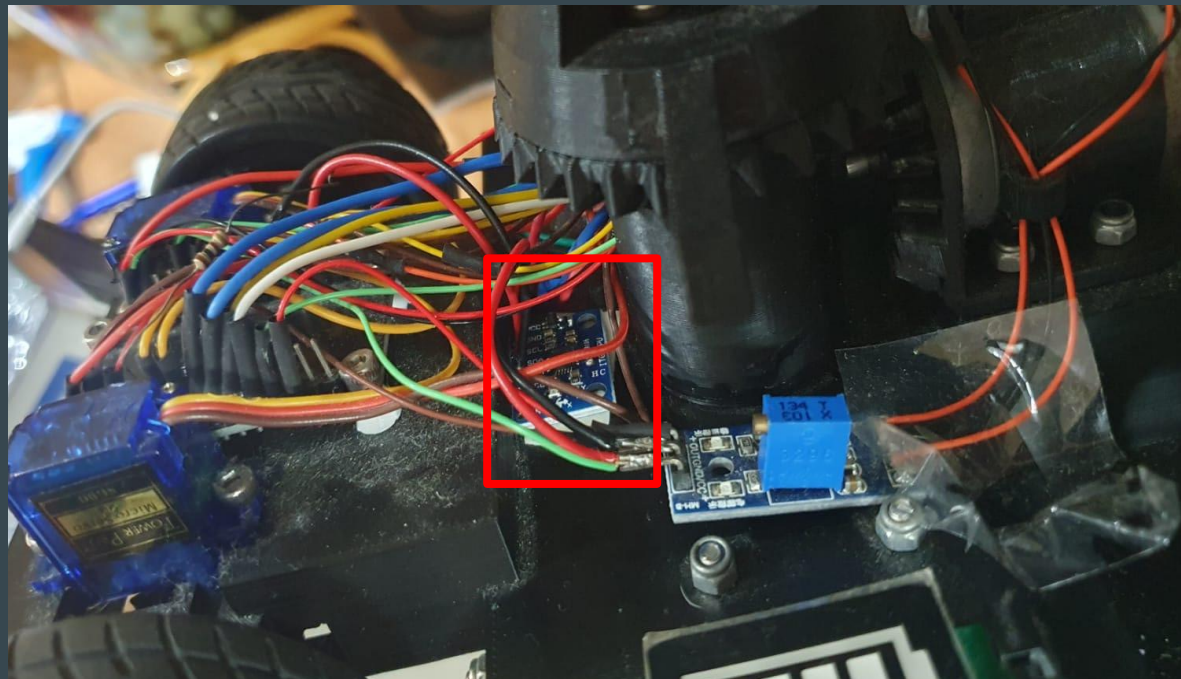
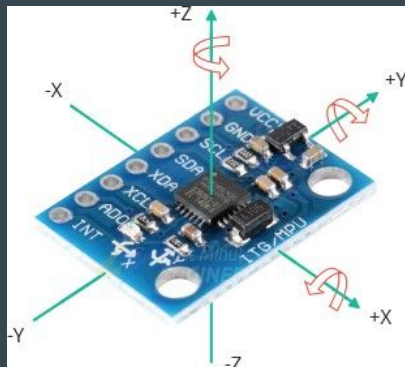
- Adição do acelerômetro





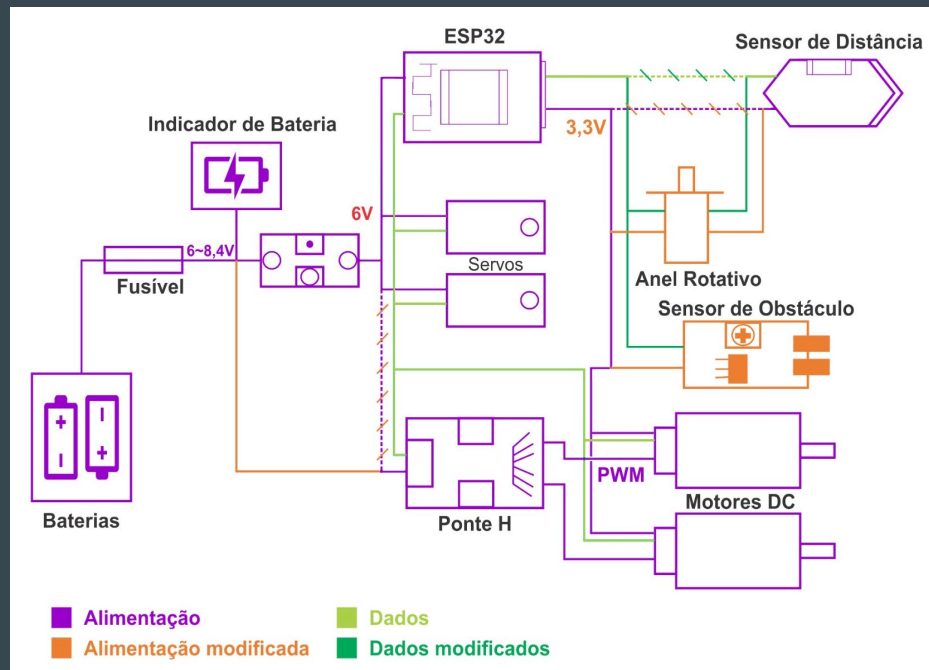
# Adição do acelerômetro MPU5060

- MPU5060
  - I2C
  - Aceleração
  - Velocidade angular



# Adição do acelerômetro

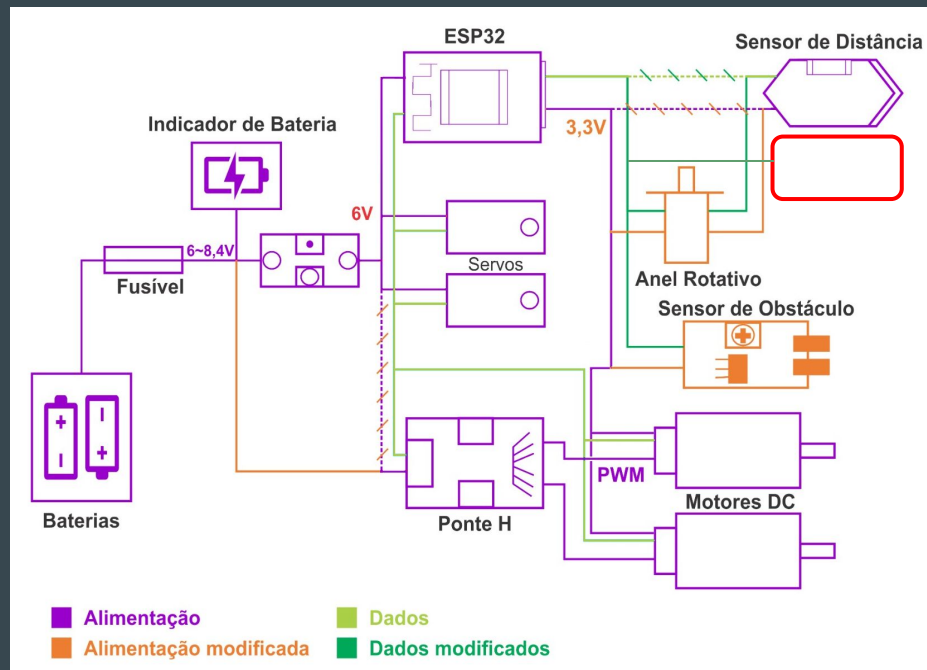
## Diagrama





# Adição do acelerômetro

## Diagrama



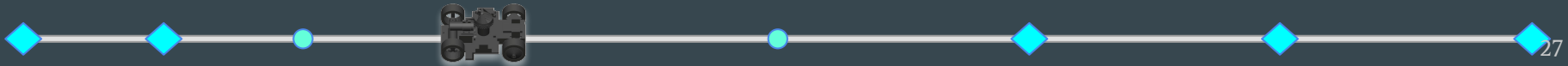
# Odometria

- Tipos de Odometria
- Cálculo de velocidade angular e deslocamento



# Tipos de Odometria

- Odometria baseada em encoders
- Odometria visual
- Odometria baseada em LiDAR



# Tipos de Odometria

- Odometria baseada em encoders

## Vantagens

- ✓ Simples e de baixo custo.
- ✓ Funciona bem em superfícies lisas e estáveis.
- ✓ Fácil de implementar com microcontroladores e ROS.

## Desvantagens

- ✗ Acúmulo de erro ao longo do tempo (drift).
- ✗ Deslizamento das rodas pode gerar medições imprecisas.
- ✗ Ineficiente em terrenos irregulares.



# Tipos de Odometria

- Odometria visual

## Vantagens

- ✓ Não sofre com deslizamento das rodas.
- ✓ Pode funcionar em superfícies irregulares ou em veículos sem rodas.
- ✓ Funciona bem em ambientes ricos em características visuais.

## Desvantagens

- ✗ Exige maior capacidade computacional.
- ✗ Depende de boa iluminação e texturas no ambiente.
- ✗ Pode falhar em ambientes com baixa visibilidade (pouca luz, superfícies lisas)



# Tipos de Odometria

- Odometria baseada em LiDAR

## Vantagens

- ✓ Alta precisão em ambientes estruturados.
- ✓ Funciona bem mesmo em condições de baixa iluminação.
- ✓ Não depende do contato com o solo, evitando erros de deslizamento.

## Desvantagens

- ✗ Mais caro que encoders e câmeras.
- ✗ Alto consumo de processamento.
- ✗ Sensível a superfícies reflexivas e pouca variação no ambiente.



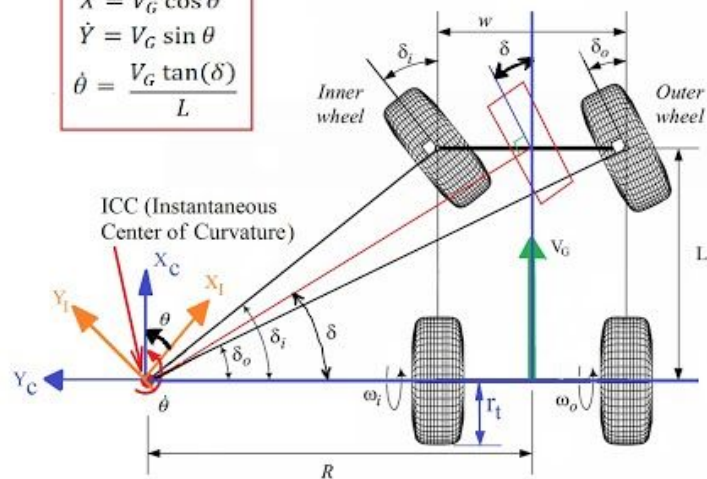
# Odometria

## Cálculo de velocidade angular e deslocamento

```
float pos_rad = pos * M_PI / 180.0;  
float rot = (vel / L) * tan(pos_rad);  
  
// Atualiza a posição  
xx += vel * dt * cos(theta);  
yy += vel * dt * sin(theta);  
theta += rot * dt;
```

$$\tan(\delta_i) = \frac{L}{R - \frac{w}{2}} \quad \tan(\delta_o) = \frac{L}{R + \frac{w}{2}}$$

$$\begin{aligned}\dot{X} &= V_G \cos \theta \\ \dot{Y} &= V_G \sin \theta \\ \dot{\theta} &= \frac{V_G \tan(\delta)}{L}\end{aligned}$$



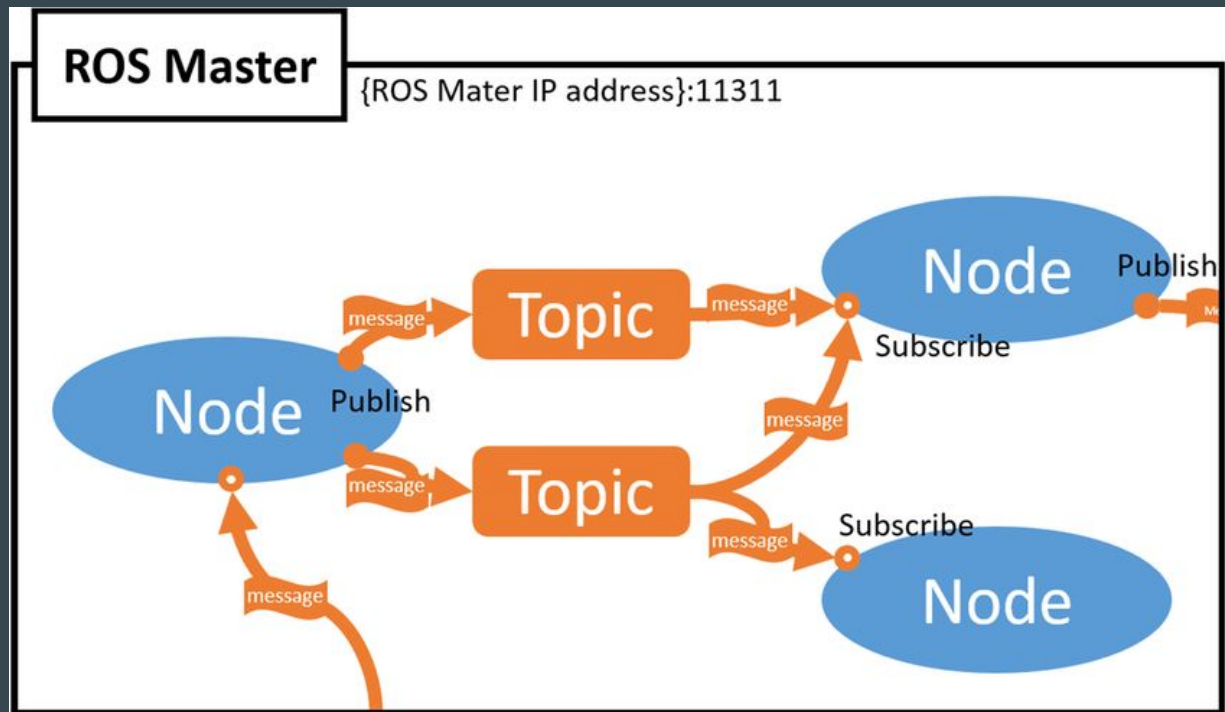
# ROS

- Estrutura de controle
- Estrutura de arquivos





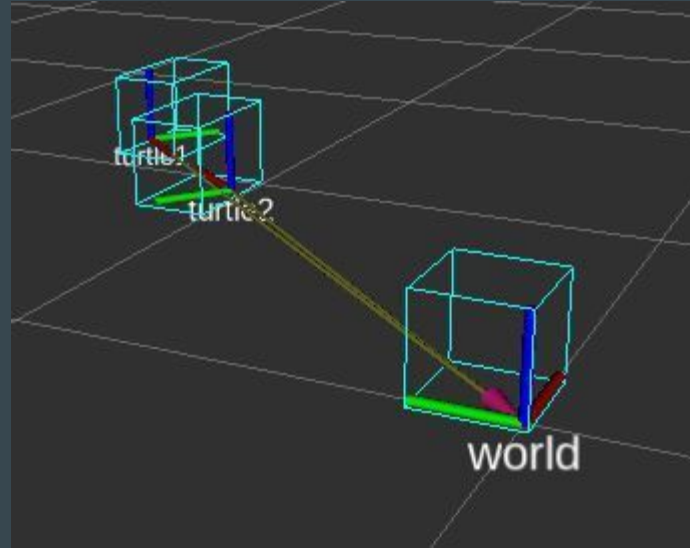
# Estrutura de controle



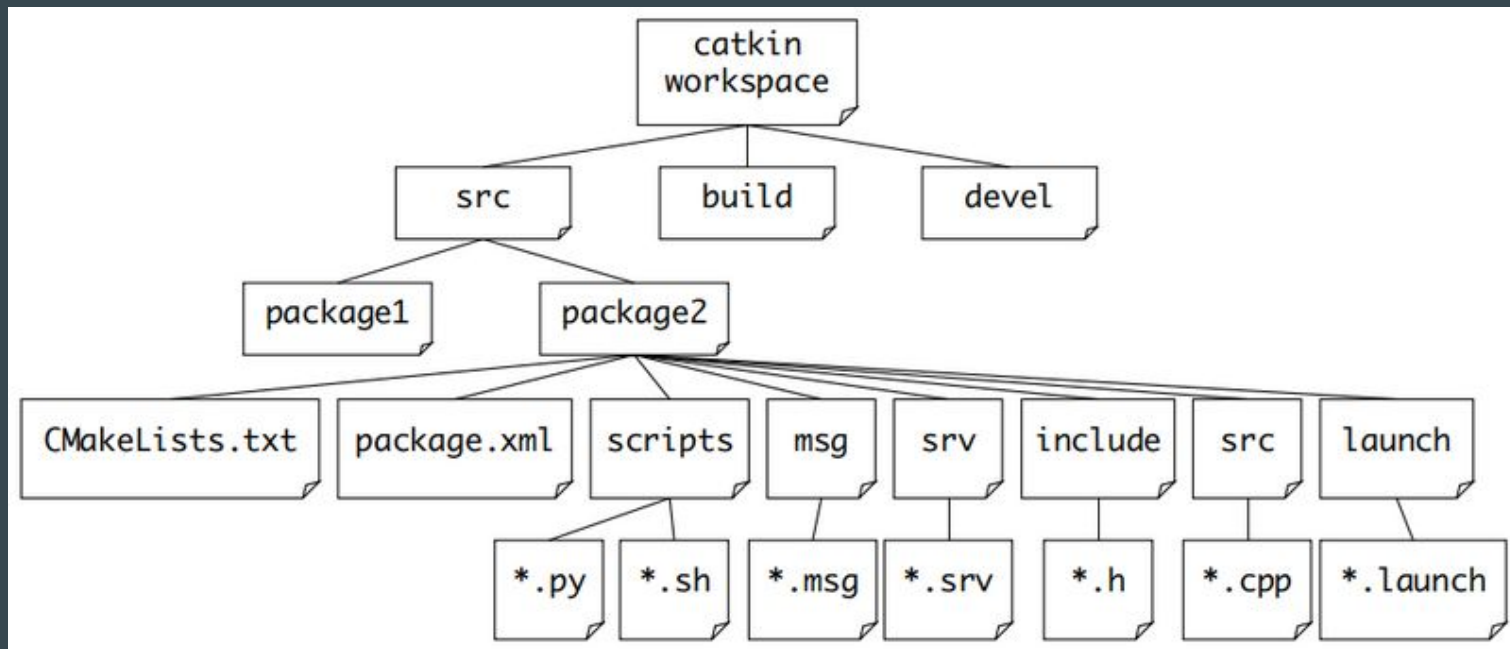
# Estrutura de controle

## Transformações

- Frame
  - tf2



# Estrutura de arquivos



# Estrutura de arquivos

```
> .vscode
> build
> devel
✓ src
  > beginner_tutorials
  > teleop_twist_keyboard
  ✓ udp_receiver
    > config
      ! ekf_template.yaml
    > include
  > launch
    📄 ekf.launch
    📄 gmapping.launch
  ✓ src
    📄 udp_dialog.cpp 4
    📄 udp_dialog.py
    📄 udp_receiver.cpp
    M CMakeLists.txt
    📄 CMakeLists.txt.save
    📄 package.xml
    M CMakeLists.txt 4
    📄 .catkin_workspace
    📄 frames.gv
    📄 frames.pdf
```

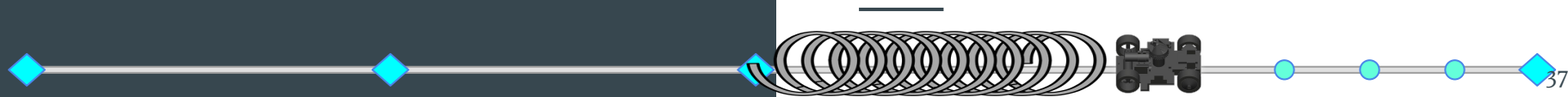
```
> .vscode
> build
> devel
✓ src
  > beginner_tutorials
  > teleop_twist_keyboard
  ✓ udp_receiver
    > config
      ! ekf_template.yaml
    > include
  > launch
    📄 ekf.launch
    📄 gmapping.launch
```

```
📄 gmapping.launch
✓ src
  📄 udp_dialog.cpp 4
  📄 udp_dialog.py
  📄 udp_receiver.cpp
  M CMakeLists.txt
  📄 CMakeLists.txt.save
  📄 package.xml
  M CMakeLists.txt 4
  📄 .catkin_workspace
  📄 frames.gv
  📄 frames.pdf
```



# Controle

- Comunicação e odometria
- Filtro de Kalman
- SLAM com gmapping



# Controle

## Comunicação e odometria

- Bibliotecas
- Variáveis globais
- Função de controle por comando de texto
- Função de recebimento de mensagem
  - Filtro de tipo de mensagem
    - Velocidade e imu
      - Cálculo de odometria e imu
      - Publicação de odometria
      - Publicação de imu
    - LiDAR
      - Publicação do /scan
  - Publicação das transformações
- Função de recebimento de /cmd\_vel (controle pelo teclado)
- Main
  - Configuração do nó
  - Configuração das bibliotecas e funções
  - Envio de mensagens



# Aquisição e controle

## Comunicação

```
[Microsegundos]_[Quantidade_de_Pontos]_[Dist0]_[Dist1]_..._[Dist15]
```

```
[Microsegundos]_[Velocidade]_[Esterçamento]_[AcelX]_[AcelY]_[AcelZ]_[RotX]_[RotY]_[RotZ]
```



# Controle Kalman

```
1  # Filter Configuration
2  frequency: 10
3  sensor_timeout: 0.1
4  silent_tf_failure: false
5  two_d_mode: true
6  transform_time_offset: 0.0
7  transform_timeout: 0.0
8  print_diagnostics: true
9
10 # Transform Settings
11 publish_tf: true
12 publish_acceleration: false
13 permit_corrected_publication: false
14
15 # Frame Configuration
16 map_frame: map
17 odom_frame: odom
18 base_link_frame: base_link
19 world_frame: odom
20
```






# Controle Kalman

```
21 # Odometry Input (odom0)
22 odom0: /odom
23 v odom0_config: [true, true, true,|
24                 true, true, true,
25                 true, true, true,
26                 true, true, true,
27                 false, false, false]
28 odom0_queue_size: 2
29 odom0_nodelay: false
30 odom0_differential: false
31 odom0_relative: false
32 odom0_pose_rejection_threshold: 5
33 odom0_twist_rejection_threshold: 1
```

```
35 # IMU Input (imu0)
36 imu0: /imu/data
37 v imu0_config: [false, false, false,
38                true, true, true,
39                false, false, false,
40                true, true, true,
41                true, true, true]
42 imu0_queue_size: 5
43 imu0_nodelay: false
44 imu0_differential: false
45 imu0_relative: true
46 imu0_pose_rejection_threshold: 0.8
47 imu0_twist_rejection_threshold: 0.8
48 imu0_linear_acceleration_rejection_threshold: 0.8
49 imu0_remove_gravitational_acceleration: true
50
```



# Controle Gmapping

```
src > udp_receiver > launch >  gmapping.launch
```

```
1 <launch>
2   <!-- Inicia o Gmapping -->
3   <node name="slam_gmapping" pkg="gmapping" type="slam_gmapping" output="screen">
4     <param name="base_frame" value="base_link" /> <!-- Frame base do robô -->
5     <param name="odom_frame" value="odom" /> <!-- Frame da odometria -->
6     <param name="map_frame" value="map" /> <!-- Frame do mapa -->
7     <param name="maxUrange" value="1.0" /> <!-- Alcance máximo do LIDAR (em metros) -->
8     <param name="minimumScore" value="6" /> <!-- Qualidade mínima para uma leitura ser usada -->
9     <param name="linearUpdate" value="0.1" /> <!-- Atualização do mapa após mover 20 cm -->
10    <param name="angularUpdate" value="0.1" /> <!-- Atualização do mapa após girar 0.1 rad -->
11    <param name="particles" value="30" /> <!-- Número de partículas usadas no filtro -->
12    <param name="xmin" value="-5.0" /> <!-- Limites do mapa (em metros) -->
13    <param name="ymin" value="-5.0" />
14    <param name="xmax" value="5.0" />
15    <param name="ymax" value="5.0" />
16    <param name="delta" value="0.05" /> <!-- Resolução do mapa (em metros/pixel) -->
17    <param name="iterations" value="10" />
18    <!-- Configuração do tópico do scan -->
19    <remap from="scan" to="/scan" />
20  </node>
21 </launch>
22
```

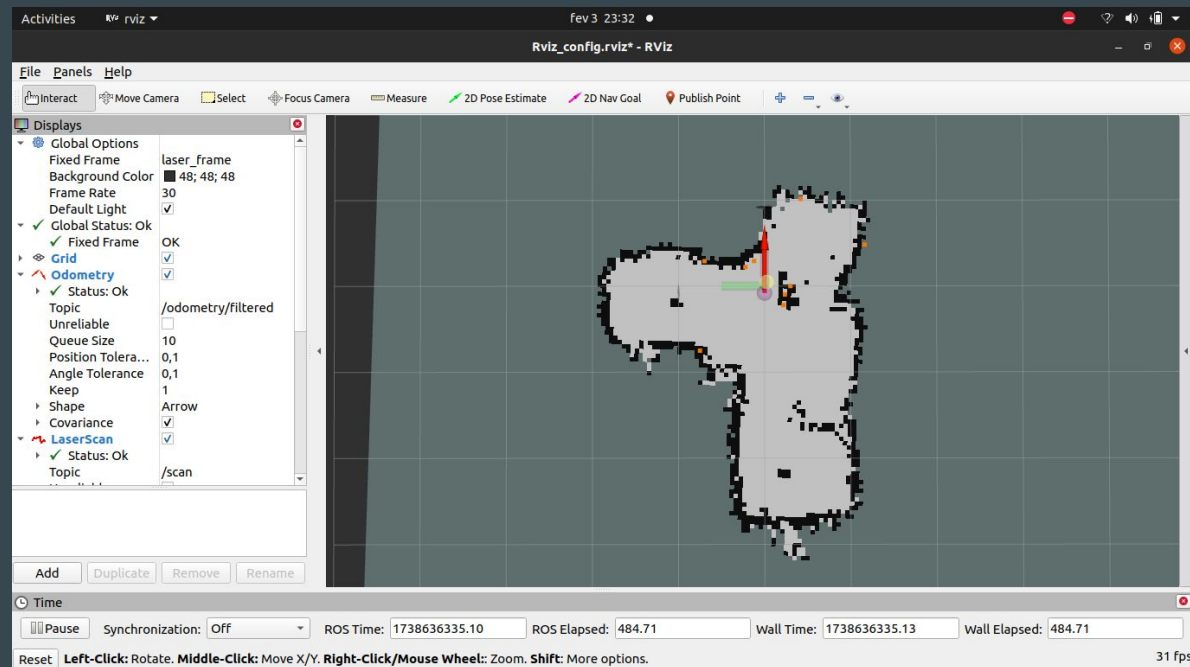


# Execução

- Rviz
- Tópicos
- Frames

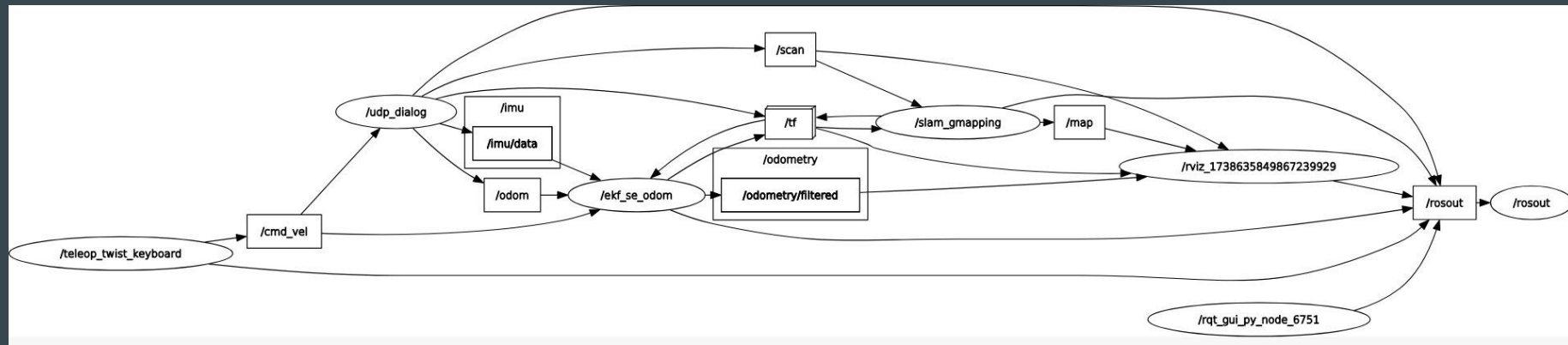


# Aquisição e controle Rviz



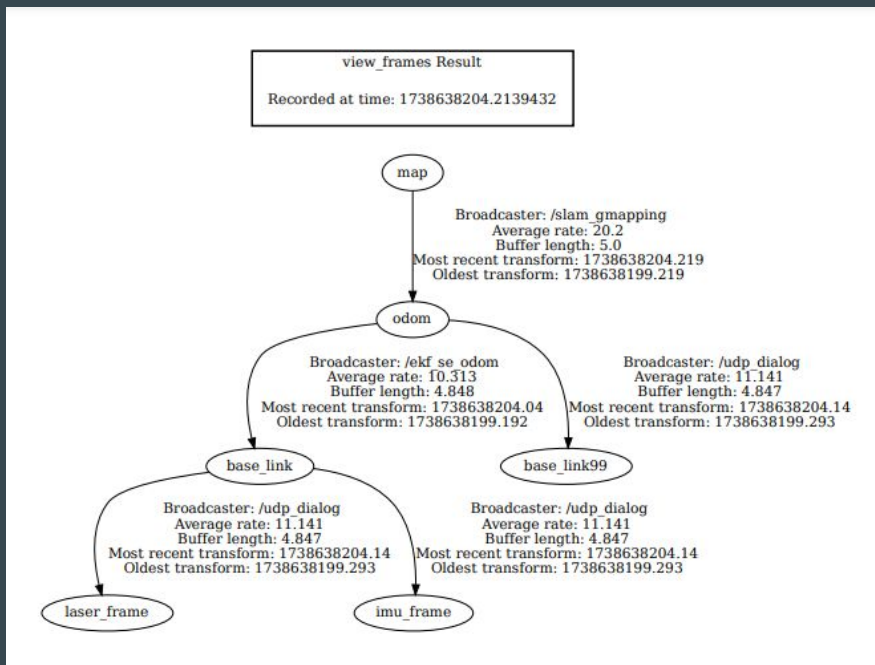
# Aquisição e controle

## Gmapping



# Aquisição e controle

## /tf



# Aquisição e controle

## /tf

```
tucss@tucss:~/Documents$ rostopic info /tf
Type: tf2_msgs/TFMessage

Publishers:
* /ekf_se_odom (http://tucss:45759/)
* /udp_dialog (http://tucss:39785/)
* /slam_gmapping (http://tucss:37349/)

Subscribers:
* /ekf_se_odom (http://tucss:45759/)
* /slam_gmapping (http://tucss:37349/)
* /rviz_1738635849867239929 (http://tucss:35711/)

tucss@tucss:~/Documents$
```



# Resultados

- Ambiente
- Operação





# Resultados

## Ambiente



# Resultados

## Ambiente



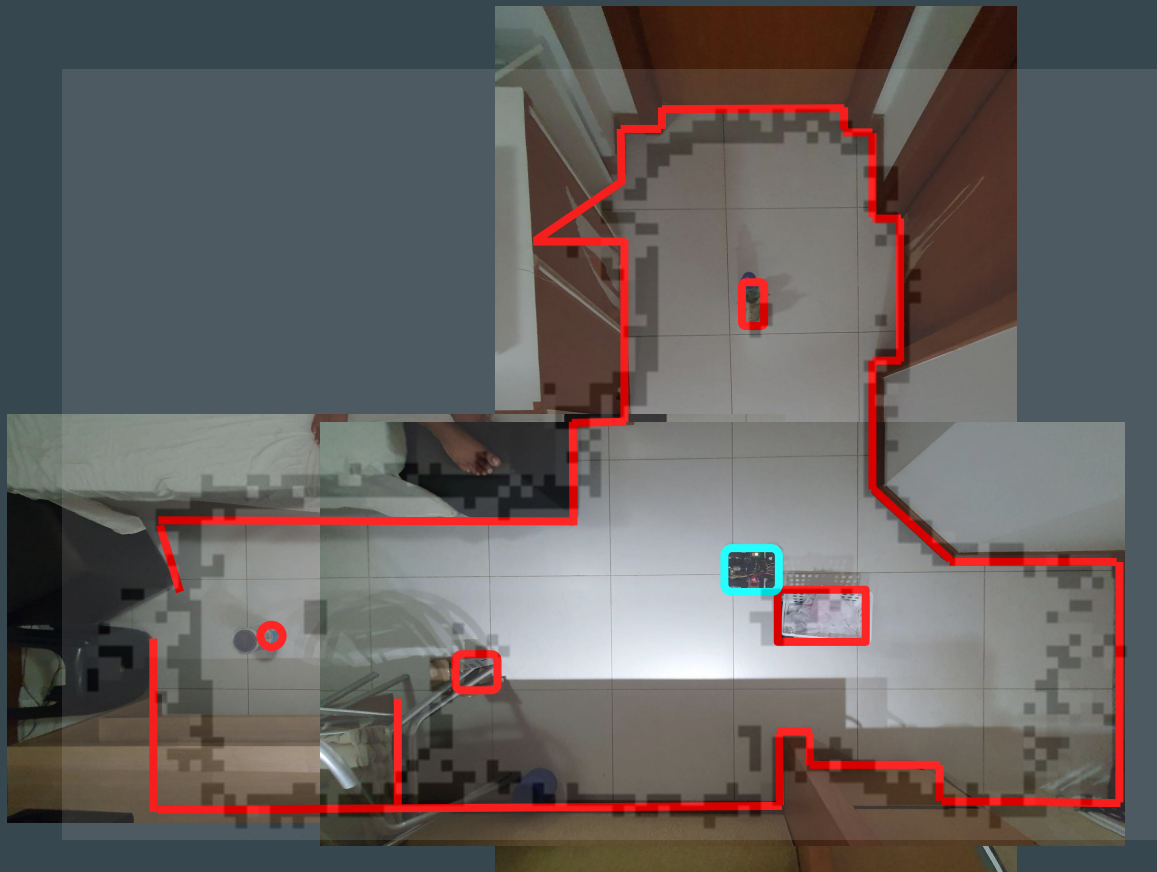
# Resultados

## Operação

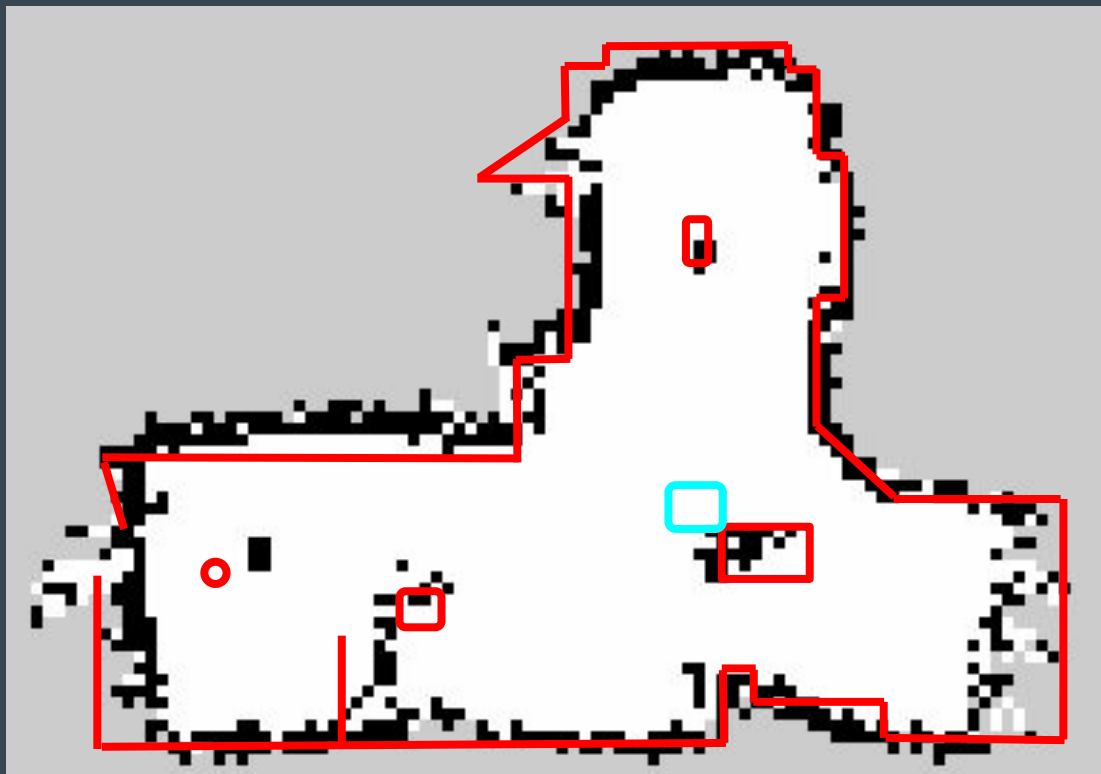
- Movimentação lenta
  - 75mm/s
- Pouco esterçamento
  - $\pm 6^\circ$



# Resultados



# Resultados



# Referências

CRUZ Júnior, G. P. Investigação de técnicas lidar slam para um dispositivo robótico de inspeção de ambientes confinados. *Revista da SBA*, v. 2, n. 1, 2021.

DANTAS Junior, J. S. C. Slam de um robô móvel terrestre utilizando sensor lidar e odometria com filtro de kalman estendido. *Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza*, 2023.

SOARES, L. P. Projeto e construção de um robô móvel car-like equipado com lidar 2d. *Curso de Graduação em Engenharia Elétrica da Universidade Federal da Bahia – Universidade de Salvador*, 2023.

VELOSO, P. H. O. Desenvolvimento de um protótipo para veículos autônomos com localização e mapeamento simultâneos. *Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Formiga*., 2019.

# Teste em sala de aula

