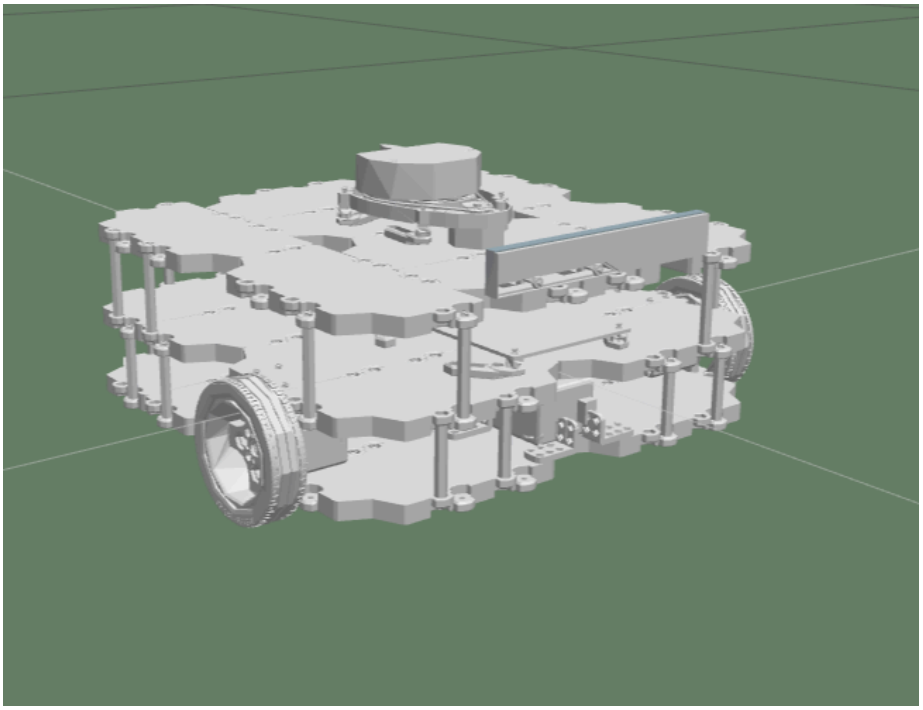
RA: Robòtica Mòbil 2026
Pràctiques

4- Mapeig i Localització en ROS



Dr. Anaís Garrell Zulueta
Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial

CONTINGUTS

1. Què Aprendrem?.....2
2. Visualització del Mapeig en Rviz.....2
2. 1 Guardar la configuració RVIZ.....3
3. SLAM..... 4
3.1. The gmapping package.....4
3.2. Guardar el mapa..... 5
3.3. YALM File..... 5
3.4. Image File (PGM)..... 6
4. Proporcionant el mapa..... 7
5. La comanda per iniciar el node del servidor de mapes (map server)..... 8
6. Requeriments de Hardware.....11
7. Transformacions.....12
8. Crear arxius launch per al node slam_gmapping.....17
9. Visualització de la Localització en RVIZ.....17
Aquesta comanda ja se us obrirà el rviz, però en cas que volgueu obrirlo i visualitzar els topics heu de fer el següent:..... 18
Visualitza Pose Array (Particle Clouds).....18
1. Feu clic al botó "Afegir" sota les visualitzacions i tria la visualització de PoseArray.... 18
2. En les propietats de la visualització, introdueix el nom del topic on es publiquen els núvols de partícules (normalment /particlecloud).....18
3. Per veure la posició del robot, també pots afegir les visualitzacions de RobotModel o TF..... 18
4. L'eix Fix ha de ser "map" per tal de visualitzar adequadament les partícules.....18
10. Localització de Montecarlo (MCL).....21
11. El paquet AMCL..... 21
13. Requeriments del Hardware..... 23
14. Transformacions.....24
15. Creació de l'arxiu launch per al node AMCL.....24
15.1 Paràmetres generals.....24
15.2 Paràmetres del filtre..... 24
15.3 Paràmetres del làser..... 25

1. Què Aprendrem?

- Què significa "Mapping" en la navegació ROS?
- Com funciona el mapeig de ROS?
- Com configurar ROS per fer que el mapeig funcioni amb gairebé qualsevol robot
- Diferents maneres de construir un mapa
- Què significa la localització en la navegació de ROS?
- Com funciona la localització?
- Com realitzem la localització en ROS?

Si heu completat amb èxit la pràctica anterior sabeu que per a realitzar la navegació autònoma, el robot ha de tenir un mapa de l'entorn. El robot utilitzarà aquest mapa per a moltes coses com planificar trajectòries, evitar obstacles, etc. Podeu donar al robot un mapa preconstruït de l'entorn (en el cas poc comú que ja en tingueu un amb el format adequat), o bé podeu construir-ne un. Aquesta segona opció és la més freqüent. Així que en aquesta pràctica, bàsicament recordareu com crear un mapa des de zero.

2. Visualització del Mapeig en Rviz

Com ja sabeu vist, podeu iniciar RViz i afegir visualitzacions per seguir el progrés del mapeig. Per al mapeig, bàsicament necessitareu utilitzar 2 visualitzacions de RViz:

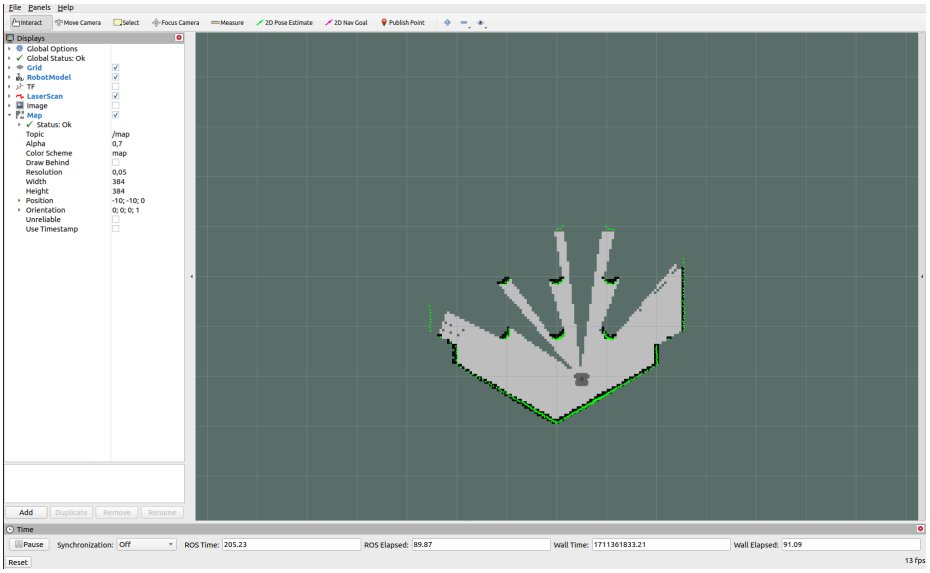
1. Visualitzacions de **LaserScan**
2. Visualitzacions de **Mapes**

NOTA: Hi ha visualitzacions opcionals que potser volgueu afegir a RViz per fer les coses més clares: els **Models dels Robots**. Aquestes visualitzacions et mostraran els models dels robots a través de RViz, permetent-vos veure com es mouen els robots al voltant dels mapes que estan construint. També són útils per detectar errors.

Llenceu les següents comandes en diferents terminals ver a visualitzar l'estat del robot:

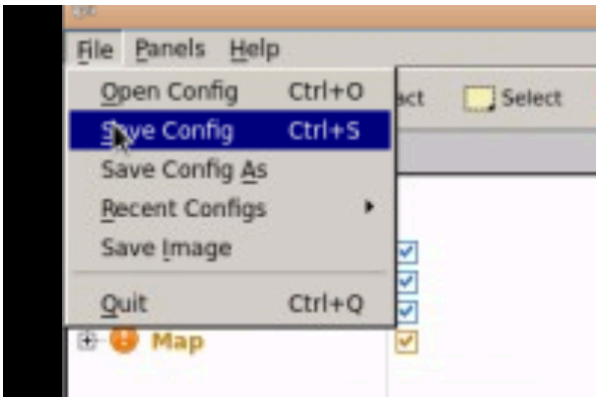
```
T1: roscore
T2: roslaunch turtlebot3_gazebo turtlebot3_world.launch
T3: roslaunch turtlebot3_slam turtlebot3_slam.launch
slam_methods:=gmapping
T4: rosrun rviz rviz
```

Al rviz afegiu els topics necessaris per tal de visualitzar una imatge semblant a la següent:



2. 1 Guardar la configuració RVIZ

Afortunadament, RViz permet desar la configuració actual, així que podreu recuperar-la en qualsevol moment. Per fer-ho, seguiu els següents passos: Aneu a la part superior esquerra de la pantalla de RViz i obriu el menú Fitxer. Seleccioneu l'opció "*Save Config As*".



Ara podreu carregar la vostra configuració desada sempre que ho desitgeu seleccionant l'opció "*Open Config* " al menú "*File*".

Important: Abans de començar aquest procés, assegureu-vos que el RViz estigui configurat correctament per tal de visualitzar el procés de mapeig. No tanqueu els processos llençats anteriorment

```
T5: roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Si aneu navegant per l'entorn podreu crear el mapa, però això ja ho heu fet en pràctiques anteriors.

3. SLAM

Simultaneous Localization and Mapping (SLAM). Aquest és el nom que defineix el problema robòtic de construir un mapa d'un entorn desconegut mentre es fa un seguiment simultani de la ubicació del robot en el mapa que s'està construint. Bàsicament, aquest és el problema que el Mapeig està resolent. A més, la Localització també està implicada, però arribarem allà més endavant. Així que, resumint, necessitem fer SLAM per crear un mapa pel robot.

3.1. The gmapping package

El paquet ROS gmapping és una implementació d'un algorisme SLAM específic anomenat **gmapping**. Això significa que algú ja ha implementat l'algorisme gmapping perquè el pogueu utilitzar dins de ROS, sense haver de programar-lo vosaltres mateix. Així que, si utilitzeu el stack de navegació ROS, només necessiteu saber (i preocupar-vos) com configurar gmapping pel vostre robot específic (que és precisament el que aprendreu ara).

El paquet gmapping conté un Node ROS anomenat **slam_gmapping**, que permet crear un mapa 2D utilitzant les dades del làser i la posició que el robot mòbil proporciona mentre es mou per un entorn. Aquest node bàsicament llegeix les dades del làser i les transformacions del robot, i les converteix en un mapa de graella d'ocupació (OGM).

Així que, bàsicament, el que ja sabeu fer és:

1. Utilitzar un fitxer launch de configuració ja creat (**gmapping_demo.launch**) per iniciar el paquet gmapping amb el robot.
2. Aquest fitxer launch inicia un node **slam_gmapping** (del paquet gmapping). I es mou rel robot per l'entorn.
3. Llavors, el node **slam_gmapping** es subscriu als topics de Làser i de Transformacions (/tf) per obtenir les dades que necessita, i es construeix un mapa.
4. El mapa generat es publica durant tot el procés al topic /map, que és la raó per la qual es pot veure el procés de construcció del mapa amb Rviz (perquè Rviz simplement visualitza topics).

El topic **/map** utilitza un tipus de missatge **nav_msgs/OccupancyGrid**, ja que és un OGM. L'ocupació es representa com un enter en l'interval {0, 100}. Amb 0 que significa completament lliure, 100 que significa completament ocupat, i el valor especial de -1 per completament desconegut.

Ara, potser us podrieu preguntar:

- Què passa si el teu Kobuki no té el làser al centre?
- Què passa si en lloc d'un làser, utilitzes un Kinect?
- Què passa si voleu utilitzar el mapeig amb un robot diferent?

Per tal de respondre les preguntes necessiteu aprendre certs conceptes primer.

3.2. Guardar el mapa

Un altre dels paquets disponibles al stack de navegació ROS és el paquet **map_server**. Aquest paquet proporciona el node **map_saver**, que ens permet accedir a les dades del mapa des d'un servei ROS i desar-lo en un fitxer.

Quan es sol·licita al map_saver que desi el mapa actual, les dades del mapa es guarden en dos fitxers: un és el fitxer YAML, que conté les metadades del mapa i el nom de la imatge, i el segon és la imatge en si mateixa, que conté les dades codificades del mapa de graella d'ocupació.

Recordem que la comanda per a guardar el mapa és:

```
rosrun map_server map_saver -f name_of_map
```

Aquesta comanda obtindrà les dades del mapa del topic map i les escriurà en 2 fitxers, **name_of_map.pgm** i **name_of_map.yaml**.

Nota 1: L'opció -f permet donar un nom personalitzat als fitxers. Per defecte (si no useu l'opció -f), els noms dels fitxers seran map.pgm i map.yaml.

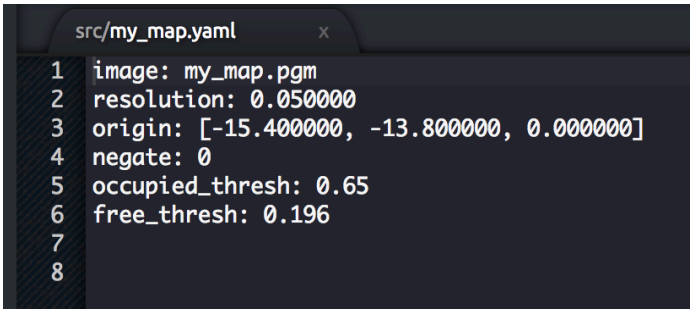
Nota 2: Per poder visualitzar els fitxers generats a través de l'IDE, aquests fitxers han d'estar al directori /home/user/catkin_ws/src. Inicialment, els fitxers es guardaran al directori on executeu la comanda.

Nota 3: Per descarregar els fitxers al voster ordinador fent clic dret sobre ells i seleccionant l'opció 'Descarregar'.

3.3. YALM File

Per tal de visualitzar el fitxer YAML generat en anteriorment, podeu fer una de les següents coses:

1. Obrir-lo a través del Web Shell. Podeu utilitzar, per exemple, l'editor vi per fer-ho escrivint la comanda `vi my_map.yaml`.
2. Descarregar el fitxer i visualitzar-lo al vostre ordinador local amb el vostre propi editor de text.



El fitxer YAML generat contindrà els següents 6 camps:

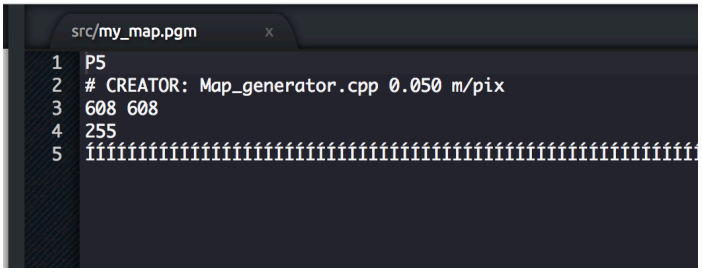
1. **image**: Nom del fitxer que conté la imatge del mapa generat.
2. **resolution**: Resolució del mapa (en metres/píxel).
3. **origin**: Coordenades del píxel inferior esquerre en el mapa. Aquestes coordenades es donen en 2D (x, y). El tercer valor indica la rotació. Si no hi ha cap rotació, el valor serà 0.
4. **occupied_thresh**: Píxels que tenen un valor major que aquest valor seran considerats com a una zona completament ocupada.
5. **free_thresh**: Píxels que tenen un valor menor que aquest valor seran considerats com a una zona completament lliure.
6. **negate**: Inverteix els colors del mapa. Per defecte, el blanc significa completament lliure i el negre significa completament ocupat.

3.4. Image File (PGM)

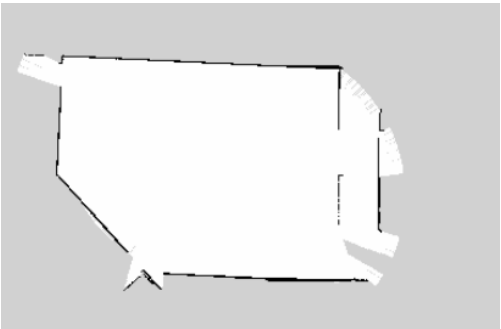
Per tal de visualitzar el fitxer PGM generat anteriorment, podeu fer una de les següents coses:

- Obrir-lo a través de la terminal. Pots utilitzar, per exemple, l'editor vi per fer-ho escrivint la comanda `vi my_map.pgm`.
- Descarregar el fitxer i visualitzar-lo al vostre ordinador local amb el vostre propi editor de text.

Si intenteu visualitzar aquest fitxer amb un editor de text, obtindreu alguna cosa semblant a això:



Això passa perquè aquest no és un fitxer de text, sinó un fitxer PGM (Portable Gray Map). Un PGM és un fitxer que representa una imatge en escala de grisos. Així que, si voleu visualitzar aquest fitxer correctament, haureu d'obrir-lo amb un editor d'imatges. Aleshores, obtindràs alguna cosa com això:



La imatge descriu l'estat d'ocupació de cada cel·la del món en el color del píxel corresponent. **Els píxels més blancs estan lliures, els píxels més negres estan ocupats, i els píxels intermedis són desconeguts.** Les imatges en color i en escala de grisos són acceptades, però la majoria dels mapes són grisos (encara que podrien estar emmagatzemats com si fossin en color). Els llindars en el fitxer YAML s'utilitzen per dividir les tres categories.

Quan es comunica a través de missatges ROS, l'ocupació es representa com un enter en l'interval [0,100], amb 0 que significa completament lliure i 100 que significa completament ocupat, i el valor especial -1 per completament desconegut.

Les dades de la imatge es llegeixen a través de SDL_Image; els formats compatibles varían, depenent del que proporcioni SDL_Image en una plataforma específica. En general, la majoria dels formats d'imatge populars tenen un ampli suport.

NOTA: Pot ser que el editor d'imatges no suporti el format PGM. Si aquest és el cas, podeu provar de convertir-lo a un fitxer PNG per visualitzar-lo. Aquí teniu una eina en línia per fer-ho: https://convertio.co/es/pgm-png/

4. Proporcionant el mapa

A més del node **map_saver**, el paquet map_server també proporciona el **node map_server**. Aquest node llegeix un fitxer de mapa del disc i proporciona el mapa a qualsevol altre node que el sol·liciti a través d'un Servei ROS.

Molts nodes sol·liciten al **map_server** el mapa actual en el qual es mou el robot. Aquesta sol·licitud es fa, per exemple, pel node move_base per obtenir dades d'un mapa i utilitzar-les per realitzar la planificació de rutes, o pel node de localització per saber on es troba el robot en el mapa. Veuràs exemples d'aquest ús en els capítols següents.

El servei a cridar per obtenir el mapa és:

- **static_map (nav_msgs/GetMap)**: Proporciona les dades d'ocupació del mapa a través d'aquest servei.

A més de sol·licitar el mapa a través del servei anterior, hi ha dues temes amb retenció (latched topics) a les quals et pots connectar per obtenir un missatge ROS amb el mapa. Els temes als quals aquest node escriu les dades del mapa són:

- **map_metadata (nav_msgs/MapMetaData):** Proporcional el mapa metadata a través d'aquest topic.
- **map (nav_msgs/OccupancyGrid):** Proporcional el mapa d'ocupació a través d'aquest topic.

NOTA: Quan un tema està amb retenció (latched), vol dir que l'últim missatge publicat a aquest tema es guardarà. Això significa que qualsevol node que escolti aquest tema en el futur rebrà aquest últim missatge, fins i tot si ningú no està publicant en aquest topic. Per especificar que un topic estarà en mode de retenció, només cal establir l'atribut latch a true quan es crea el tema.

5. La comanda per iniciar el node del servidor de mapes (map server)

Per fer un launch del node map_server per a proporcionar informació del mapa donat un arxiu mapa, utilitzeu:

```
rosrun map_server map_server map_file.yaml
```

Recordeu: Heu d'haver creat prèviament el mapa (el fitxer map_file.yaml) amb el node gmapping. No podeu proporcionar un mapa que no hagueu creat!

Nota 1: Si llanceu la comanda des del directori on teniu el fitxer del mapa desat, no cal que especifiqueu la ruta completa al fitxer. Si no esteu en el mateix directori, tingueu en compte que haureu d'especificar la ruta completa al fitxer.

Echo del topic /map_metadata:

```
ubuntu@ip-172-31-44-229:~$ rostopic echo /map_metadata
map_load_time:
  secs: 1483988029
  nsecs: 462798163
resolution: 0.0500000007451
width: 608
height: 608
origin:
  position:
    x: -15.4
    y: -13.8
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.0
    w: 1.0
---
```

Echo del topic /map:

```
ubuntu@ip-172-31-44-229:~$ rostopic echo -n1 /map
header:
  seq: 0
  stamp:
    secs: 1483988029
    nsecs: 462798488
  frame_id: map
info:
  map_load_time:
    secs: 1483988029
    nsecs: 462798163
  resolution: 0.0500000007451
  width: 608
  height: 608
  origin:
    position:
      x: -15.4
      y: -13.8
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
data: [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
```

exercici: Creeu un launch que llenci el node **map_server.** Dins d'aques paquet, creeu un arxiu launch que executi el node map_server que proporcioni el mapa que heu creat.

```
None
<launch>
    <node pkg="map_server" type="map_server" name="map_server"
output="screen" args="/home/user/catkin_ws/src/my_map.yaml">
    </node>
</launch>
```

Executeu el vostre fitxer launch, i comproveu si el mapa s'està proporcionant accedint als topic adequats. Podeu utilitzar la comanda `rostopic list` per veure tots els topics actius, i `rostopic echo /nom_del_topic` per visualitzar els missatges d'un topic específic.

Nota 1: Per tal de proporcionar l'arxiu de mapa, heu d'afegir el path a l'argument del mapa

Echo del topic /map_metadata:

```
ubuntu@ip-172-31-44-229:~$ rostopic echo /map_metadata
map_load_time:
  secs: 1483988029
  nsecs: 462798163
resolution: 0.0500000007451
width: 608
height: 608
origin:
  position:
    x: -15.4
    y: -13.8
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.0
    w: 1.0
---
```

Echo del topic /map:

```
ubuntu@ip-172-31-44-229:~$ rostopic echo -n1 /map
header:
  seq: 0
  stamp:
    secs: 1483988029
    nsecs: 462798488
  frame_id: map
info:
  map_load_time:
    secs: 1483988029
    nsecs: 462798163
  resolution: 0.0500000007451
  width: 608
  height: 608
  origin:
    position:
      x: -15.4
      y: -13.8
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
data: [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
```

Havent mostrat com crear un mapa d'un entorn amb un robot, has de comprendre el següent:

- El mapa que heu creat és un mapa estàtic. Això significa que el mapa sempre romandrà tal com era quan el vau crear. Així que, quan creeu un mapa, captura l'entorn tal com és en el moment exacte en què es realitza el procés de mapeig. Si per qualsevol raó, l'entorn canvia en el futur, aquests canvis no apareixen al mapa, per tant, ja no serà vàlid (o no correspondrà a l'entorn real).

- El mapa que heu creat és un mapa 2D. Això significa que els obstacles que apareixen al mapa no tenen alçada. Així que si, per exemple, si utilitzeu aquest mapa per navegar amb un dron, no serà vàlid. Hi ha paquets que permeten generar mapes 3D. Si esteu interessat en aquest tema, podeu consultar el següent enllaç:

  http://wiki.ros.org/rtabmap_ros/Tutorials/MappingAndNavigationOnTurtlebot

6. Requeriments de Hardware

Ja sabeu que la configuració és MOLT IMPORTANT per tal de construir un mapa adequat. Sense una bona configuració del vostre robot, no obtindreu un bon mapa de l'entorn. I sense un bon mapa de l'entorn, no podreu navegar adequadament.

Així que, per construir un mapa adequat, heu de complir aquests 2 requisits:

1. Proporcionar Bones Dades de Làser.
2. Proporcionar Bones Dades de Odometria.

Després, el node **slam_gmapping** intentarà transformar cada lectura de làser entrant al frame d'odometria.

```
T1: roscore
T2: roslaunch turtlebot3_gazebo turtlebot3_world.launch
T3: rostopic list
```

```
/camera/rgb/image_raw/theor
/clock
/cmd_vel
/cmd_vel_mux/active
/cmd_vel_mux/input/navi
/cmd_vel_mux/input/safety_c
/cmd_vel_mux/input/teleop
/cmd_vel_mux/parameter_desc
/cmd_vel_mux/parameter_upda
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descripti
>_ #1 /gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/joint_states
/kobuki/laser/scan
/mobile_base/commands/veloc
/mobile_base_nodelet_manage
/odom
/rosout
/rosout_agg
/tf
/tf_static
```

T3: rostopic echo /odom

```
user ~ $ rostopic echo -n1 /odom
header:
  seq: 20254
  stamp:
    secs: 675
    nsecs: 283000000
  frame_id: odom_frame
child_frame_id: base_link
pose:
  pose:
    position:
      x: 0.0
      y: 0.0
>_ #1 z: 0.0927561574185
    orientation:
      x: 0.0
      y: 0.0
      z: 0.0
      w: 1.0
  covariance: [1e-05, 0.0, 0.0, 0.
```

rostopic echo /scan:

```
user ~ $ rostopic echo -n1 /kobuki/laser/scan
header:
  seq: 0
  stamp:
    secs: 89
    nsecs: 284000000
  frame_id: laser_sensor_link
angle_min: -1.57079994678
angle_max: 1.57079994678
angle_increment: 0.00436940183863
time_increment: 0.0
scan_time: 0.0
range_min: 0.10000000149
range_max: 30.0
ranges: [29.990480422973633, 29.995317459106445, 30.0, 30.0, 30.0, 30.0, 30.0, 29.99565887451172, 29.992713928222656, 30.0, 30.0, 29.991031
646728516, 30.0, 30.0, 30.0, 29.99979019165039, 30.0, 29.990154266357422, 29.992589950561523, 30.0, 29.992450714111328, 29.99542808532715,
29.991064071655273, 30.0, 30.0, 29.99236297607422, 29.99344253540039, 29.99321174621582, 30.0, 29.99823760986328, 30.0, 30.0, 29.9937229156
49414, 29.999950408935547, 29.993419647216797, 30.0, 29.989418029785156, 30.0, 29.97750473022461, 30.0, 29.988431930541992, 30.0, 29.997327
80456543, 30.0, 29.99301528930664, 30.0, 30.0, 29.980470657348633, 29.99444580078125, 29.987178802490234, 30.0, 30.0, 30.0, 30.0, 30.0, 29.
999492645263672, 30.0, 30.0, 30.0, 29.998613357543945, 29.986387252807617, 30.0, 30.0, 29.995351791381836, 30.0, 29.989456176757812, 30.0,
30.0, 29.998764038085938, 30.0, 29.98979949951172, 30.0, 29.99371337890625, 29.991764068603516, 30.0, 30.0, 29.99911117553711, 30.0, 29.984
121322631836, 30.0, 29.990968704223633, 30.0, 29.99369239807129, 30.0, 29.995807647705078, 29.99005699157715, 30.0, 30.0, 30.0, 30.0, 30.0,
 29.99907875061035, 29.995933532714844, 29.995332717895508, 30.0, 29.99344253540039, 29.993955612182617, 30.0, 29.99266242980957, 30.0, 29.
992557525634766, 30.0, 30.0, 30.0, 29.990720748901367, 30.0, 30.0, 30.0, 30.0, 30.0, 29.989648818969727, 29.999773025512695, 29.99576759338
379, 30.0, 30.0, 30.0, 29.997982025146484, 30.0, 29.986892700195312, 29.984708786010742, 29.99268341064453, 30.0, 30.0, 29.993642807006836,
 29.99467658996582, 30.0, 29.993892669677734, 29.994199752807617, 29.998353958129883, 30.0, 29.978740692138672, 29.988014221191406, 30.0, 3
0.0, 29.999855041503906, 29.983083724975586, 29.988554000854492, 30.0, 29.996627807617188, 29.988346099853516, 30.0, 29.996257781982422, 29
.993988037109375, 30.0, 30.0, 29.996219635009766, 30.0, 29.990314483642578, 30.0, 29.989356994628906, 30.0, 30.0, 30.0, 29.992830276489258,
 29.98858070373535, 29.99994468688965, 29.98659896850586, 30.0, 30.0, 30.0, 30.0, 30.0, 29.992982864379883, 30.0, 30.0, 30.0, 29.99576759338379,
30.0, 30.0, 29.99584197998047, 29.992801666259766, 30.0, 30.0, 30.0, 29.99032974243164, 30.0, 30.0, 30.0, 29.998977661132812, 29.9823570251
46484, 30.0, 30.0, 29.979532248193129, 30.0, 29.991052896118164, 30.0, 30.0, 30.0, 30.0, 29.995889663696629, 30.0, 30.0, 30.0, 30.0, 30.0, 30
```

7. Transformacions

Anteriorment teniu les transformacions del robot configurades correctament, així que podieu confiar que les lectures del làser es transformaran correctament al frame Odom. Però, això sempre serà així? NO, no té perquè ser-ho.

Per tal de poder utilitzar les lectures del làser, necessitem establir una transformació entre el làser i la base del robot, i afegir-la a l'arbre de transformacions. Transformació? Arbre de transformacions?

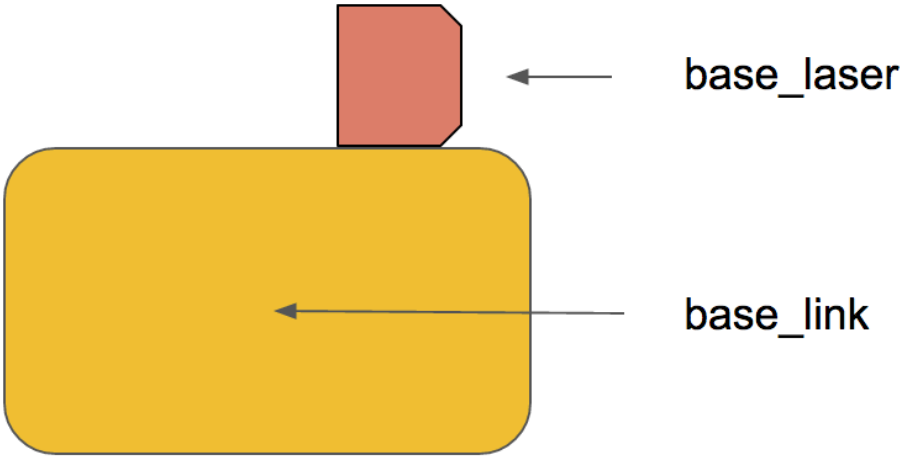
Tenim un robot turtlebot3 amb un làser muntat. Però, per poder utilitzar les dades del làser, necessitem dir-li al robot ON (posició i orientació) aquest làser està muntat al robot. Això és el que es coneix com una transformació entre frames.

Una transformació especifica com les dades expressades en un eix es poden transformar en un eix diferent. Per exemple, si detecteu un obstacle amb el làser a 3 cm davant, això significa que està a 3 cm del làser, però no del centre del robot (que normalment es coneix com **/base_link**). Per saber la distància des del centre del robot, cal transformar els 3 cm de l'eix del **làser** a l'eix **base_link** (que és el que realment necessita saber el sistema de planificació de rutes, quina és la distància des del centre del robot fins a l'obstacle).

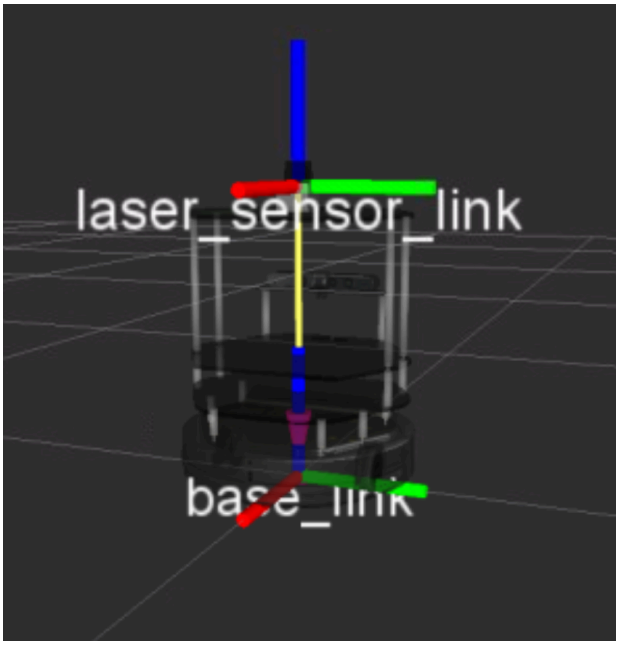
Si no proporcionem aquesta informació al robot, quan el làser detecta un objecte, com pot saber el robot on és aquest objecte? Està davant del robot? Està darrere? Està a la dreta? No hi ha manera que el robot ho pugui saber si no li diem la POSICIÓ i la ORIENTACIÓ del làser respecte al centre del robot. Per fer-ho, hem de fer el següent:

Primer, definirem dos eixos (eixos de coordenades), un al centre del làser i un altre al centre del robot. Per a la navegació, és important que el centre del robot estigui situat al centre de rotació del robot. Anomenarem l'eix del làser com a **base_laser** i l'eix del robot com a **base_link**.

Aquí podeu veure un esquema de com quedaria això:



Per exemple, en el cas del robot turtlebot2 els eixos es veuen així:



Ara, necessitem definir una relació (en termes de posició i orientació) entre l'eix **base_laser** i l'eix **base_link**. Per exemple, sabem que l'eix base_laser està a una distància de 20 cm en l'eix y i 10 cm en l'eix x referent al marc base_link. Llavors haurem de proporcionar aquesta relació al robot. **Aquesta relació entre la posició del làser i la base del robot es coneix a ROS com la TRANSFORMACIÓ entre el làser i el robot.**

Per a què el node **slam_gmapping** funcioni correctament, haureu de proporcionar 2 transformacions:

1. **El frame associat al làser -> base_link:** Normalment un valor fix, transmès periòdicament per un robot_state_publisher, o un tf static_transform_publisher.
2. **base_link -> odom:** Normalment proporcionat pel sistema d'odometria.

Ja que el robot necessita poder accedir a aquesta informació en qualsevol moment, publicarem aquesta informació a un arbre de transformacions. L'arbre de transformacions és com una base de dades on podem trobar informació sobre totes les transformacions entre els diferents marcs (elements) del robot.

Alguns eixos i transformacions poden ser fixos, com ara entre un sensor muntat i la plataforma del robot. Altres poden canviar amb el temps, com les rodes, que giren, o la posició de la **base_footprint** del robot en relació amb la posició inicial de la odometria. Tota aquesta informació es publica en el topic **/tf**, i es llegeix des d'ell.

```
T5: rostopic info /tf
Type: tf2_msgs/TFMessage

Publishers:
 * /tb3_0/turtlebot3_slam_gmapping (http://localhost:41803/)
 * /amcl (http://localhost:34111/)
 * /gazebo (http://localhost:45267/)
 * /robot_state_publisher (http://localhost:35131/)
 * /turtlebot3_slam_gmapping (http://localhost:40889/)

Subscribers:
 * /tb3_0/turtlebot3_slam_gmapping (http://localhost:41803/)
 * /amcl (http://localhost:34111/)
 * /turtlebot3_slam_gmapping (http://localhost:40889/)
 * /rviz (http://localhost:34109/)
```

```
T5: rostopic echo tf -n 10 #show only 10 messages
T5: rosrun rqt_tf_tree rqt_tf_tree #graphical tree view
```

En el nostre cas, Rviz està mostrant la informació relativa al eix /odom tf. Això es configura al panell d'esquerra Displays, sota **GlobalOptions**/**FixedFrame**. El simulador, a més de publicar missatges d'odometria, està publicant aquesta transformació entre els eixos /odom i /base_footprint. Es pot mostrar altra informació, com es veu al panell Display, que es pot habilitar/deshabilitar i configurar, com el model de robot, que consisteix en malla adjunta a marcs tf, l'arbre de tf, i les posicions de l'odometria. Es pot fer pan/rotació/zoom fent clic amb el botó mitjà/esquerre/dret a la finestra de visualització.

Anem a analitzar el tipus de missatge **/tf,** anteriorment hem vist que és un tipus de missatge **/tf2_msgs:**

```
$ roscd tf2_msgs/msg

$ cat TFMessage.msg
```

```
geometry_msgs/TransformStamped[] transforms
```

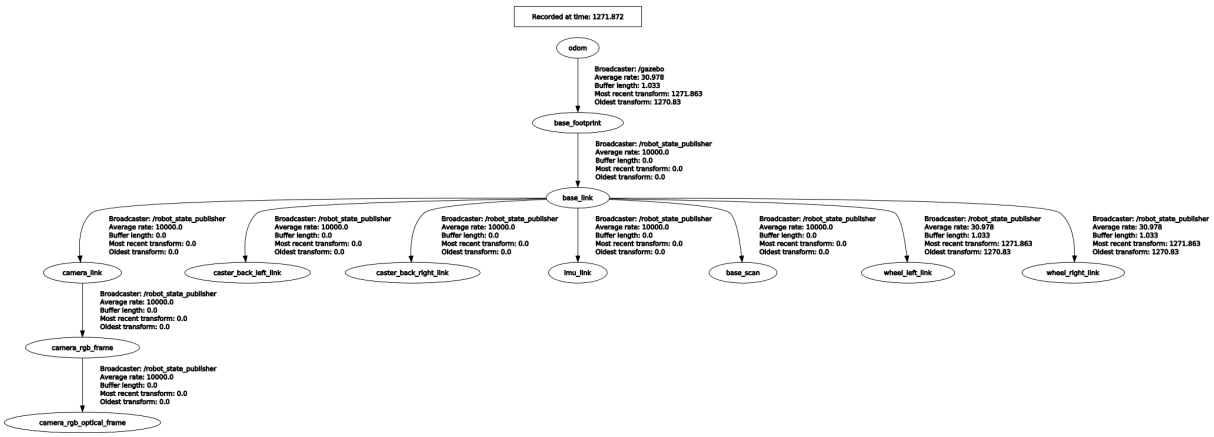
```
$ roscd geometry_msgs/msg/
```

```
$ cat TransformStamped.msg
# This expresses a transform from coordinate frame header.frame_id
# to the coordinate frame child_frame_id
#
# This message is mostly used by the
# <a href="http://wiki.ros.org/tf">tf</a> package.
# See its documentation for more information.

Header header
string child_frame_id # the frame id of the child frame
Transform transform
```

```
$ rostopic echo /tf_static -n1
```

```
$ rosrun rqt_tf_tree rqt_tf_tree
```



Ara, imaginem que acabeu de muntar el làser al vostre robot, de manera que la transformació entre el làser i la base del robot no està configurada. Què podríeu fer? Bàsicament, hi ha 2 maneres de publicar una transformació:

1. Utilitzar un **static_transform_publisher.**
2. Utilitzar un emissor de transformació (**transform broadcaster**).

En aquest curs, utilitzarem el **static_transform_publisher,** ja que és la manera més ràpida. El static_transform_publisher és un node a punt per a ser utilitzat que ens permet publicar

directament una transformació simplement utilitzant la línia de comandes. L'estructura de la comanda és la següent:

```
$ rosrun tf static_transform_publisher x y z yaw pitch roll frame_id
child_frame_id period_in_ms
```

On:

- **x, y, z** son els offsets en meters
- **yaw, pitch, roll** son les rotacions que poden ser usades per a la transformació en qualsevol orientació (en radians)
- **period_in_ms** especifica cada quan s'envia la informació

Nota: Aquí s'han de canviar `x y z yaw pitch roll frame_id child_frame_id period_in_ms` per valors reals, per exemple:

```
$  rosrun tf static_transform_publisher 0 0 0 0 0 0 map odom 100
```

També podeu crear un fitxer de launch que inicii la comanda anterior, especificant els diferents valors de la següent manera:

```
<launch>
  <node pkg="tf" type="static_transform_publisher" name="name_of_node"
      args="x y z yaw pitch roll frame_id child_frame_id period_in_ms">
  </node>
</launch>
```

Nota: Heu de canviar `args="x y z yaw pitch roll frame_id child_frame_id period_in_ms"` per valors reals

Nota2: Normalment no haureu de fer això.

8. Crear arxius launch per al node slam_gmapping

En aquest punt, esteu preparats per crear el fitxer de launch per iniciar el node slam_gmapping. La tasca principal per crear aquest fitxer de llançament, com podeu imaginar, és establir correctament els paràmetres per al node slam_gmapping. Aquest node és altament configurable i té molts paràmetres que es poden canviar per millorar el rendiment del mapeig. Aquests paràmetres es llegiran del servidor de paràmetres de ROS, i es poden establir tant en el fitxer de llançament com en fitxers de paràmetres separats (fitxers YAML). Si no establiu alguns paràmetres, simplement agafarà els valors per defecte. Podeu consultar la llista completa de paràmetres disponibles per al node slam_gmapping aquí: http://wiki.ros.org/gmapping

- **base_frame (default: "base_link")**: Indica el nom de l'eix associat a la base mòbil.
- **map_frame (per defecte: "map"):** Indica el nom de l'eix associat al mapa.

- **odom_frame (per defecte: "odom"):** Indica el nom de l'eix associat al sistema d'odometria.
- **map_update_interval (per defecte: 5.0):** Estableix el temps (en segons) per esperar abans d'actualitzar el mapa.

9. Visualització de la Localització en RVIZ

Com ja heu vist, es pot iniciar RViz i afegir visualitzacions per observar la localització del robot. Per a la localització, bàsicament necessitareu utilitzar 3 elements de RViz:

- Visualització de **LaserScan** (Vist)
- Visualització de **Mapa** (Vist)
- Visualització de **PoseArrayAs**

In [ ]:

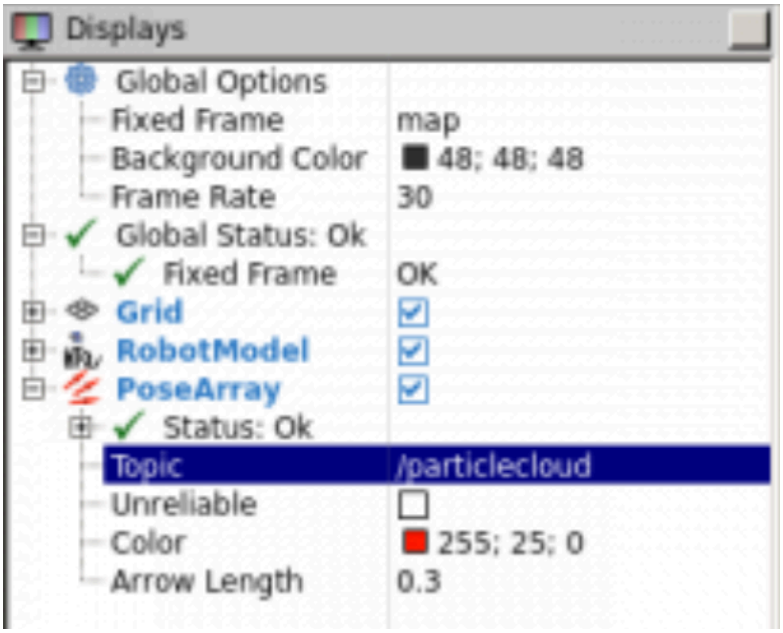
```
T1: roscore
T2: roslaunch turtlebot3_gazebo turtlebot3_world.launch
T3: roslaunch turtlebot3_navigation turtlebot3_navigation.launch
```

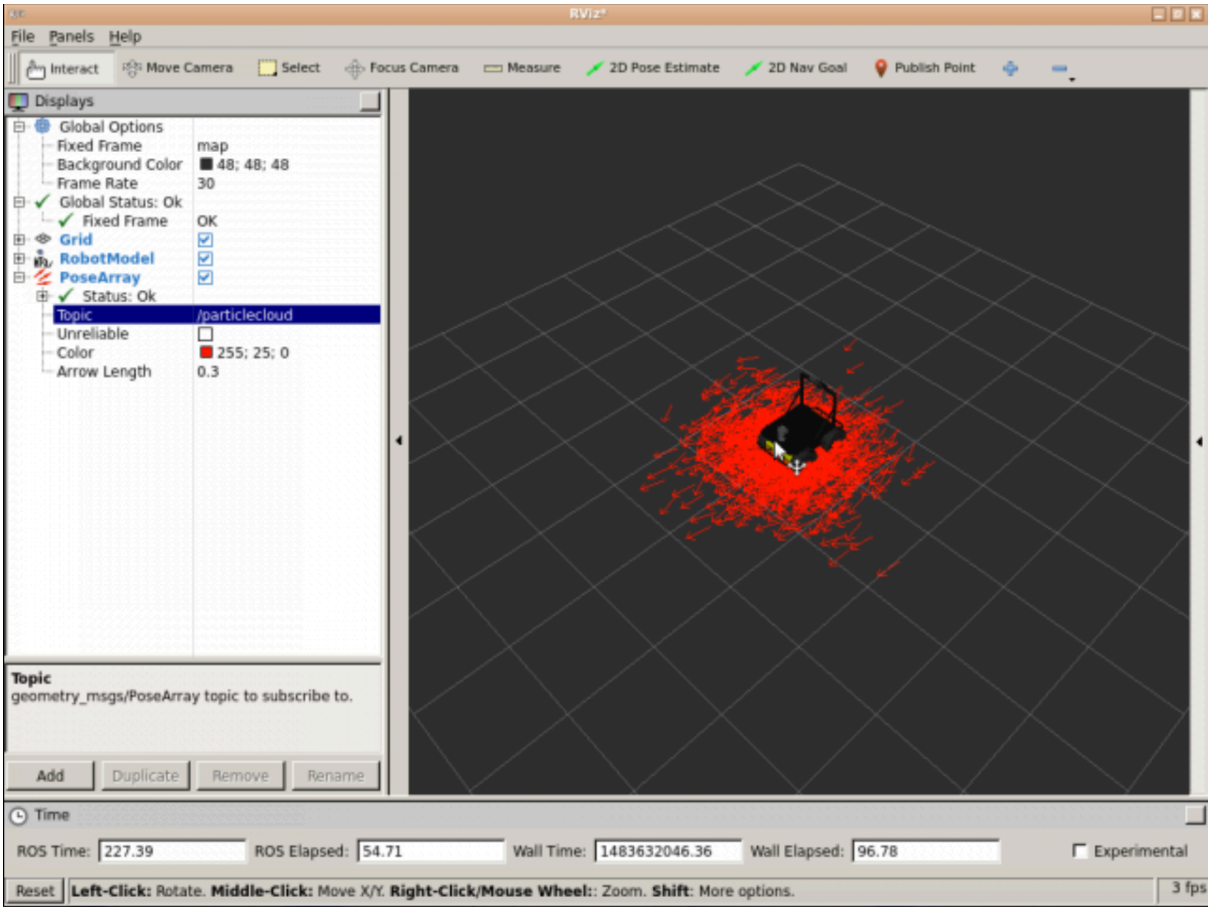
Aquesta comanda ja se us obrirà el rviz, però en cas que volgueu obrirlo i visualitzar els topics heu de fer el següent:

```
T4: rosrun rviz rviz
```

Visualitza Pose Array (Particle Clouds)

1. Feu clic al botó "Afegir" sota les visualitzacions i tria la visualització de **PoseArray**.
2. En les propietats de la visualització, introdueix el nom del topic on es publiquen els núvols de partícules (normalment /**particlecloud**).
3. Per veure la posició del robot, també pots afegir les visualitzacions de **RobotModel** o TF.
4. L'eix Fix ha de ser "map" per tal de visualitzar adequadament les partícules.





Ara que ja sabeu com configurar RViz per visualitzar la localització del robot, posem-nos a treballar!

RECORDEU: Recordeu desar la vostra configuració de RViz per poder-la carregar de nou sempre que ho desitgis. Si no recordes com fer-ho.

A continuació, veurem un exemple de com ROS gestiona el problema de la localització del robot.

a) Afegiu les visualitzacions de **LaserScan** i **Mapa** per visualitzar la posició del robot al món a través de RViz.

b) Utilitzant l'eina de Estimació de Posició 2D, establiu una posició i orientació inicial a RViz per al robot. No cal que sigui exacte, simplement observeu la posició i orientació del robot a la simulació i intenteu ajustar-la de manera similar a RViz.

c) Feu moure el robot per la sala utilitzant el programa de teleoperació amb el teclat.

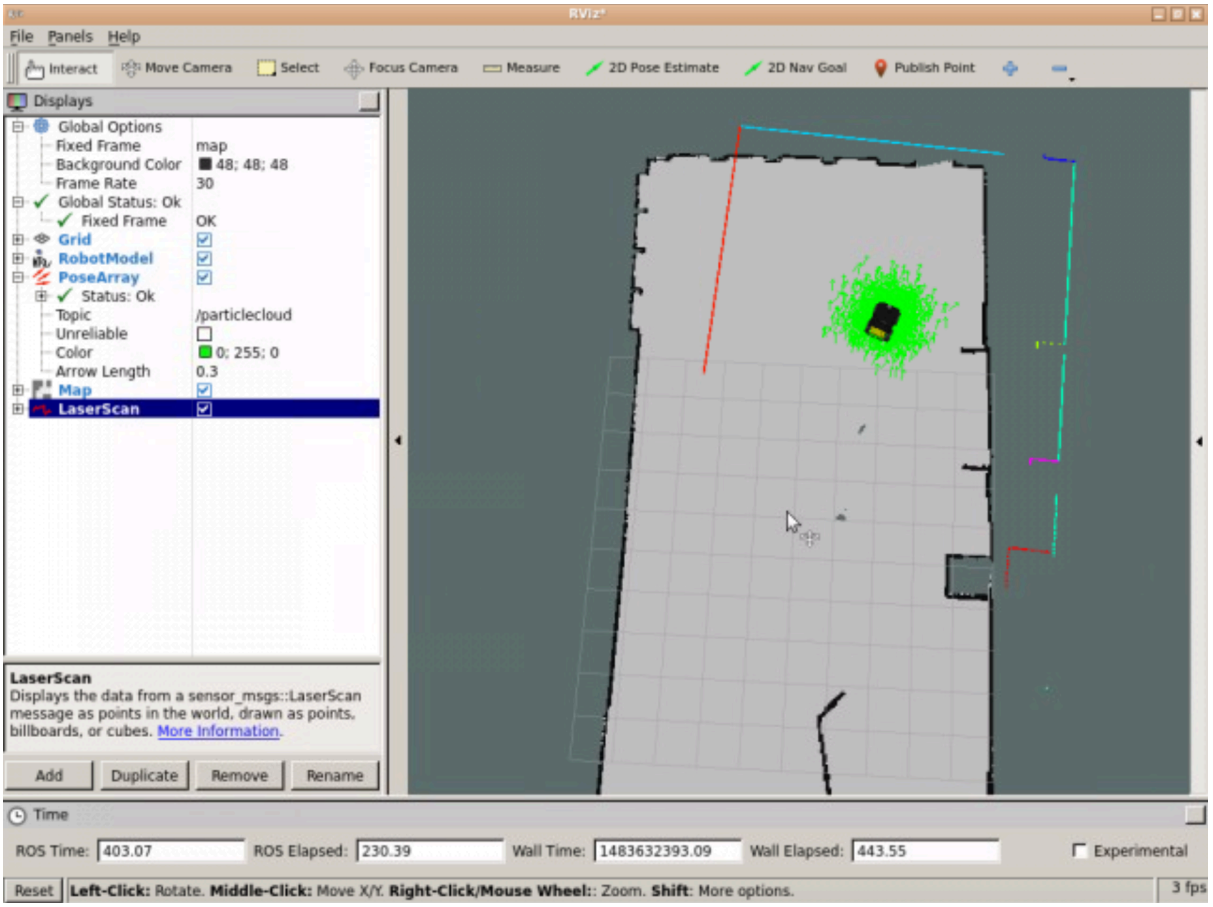
```
T5: roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Nota 1: La localització en ROS es visualitza a través d'elements anomenats partícules (veurem més sobre això més endavant en la unitat).

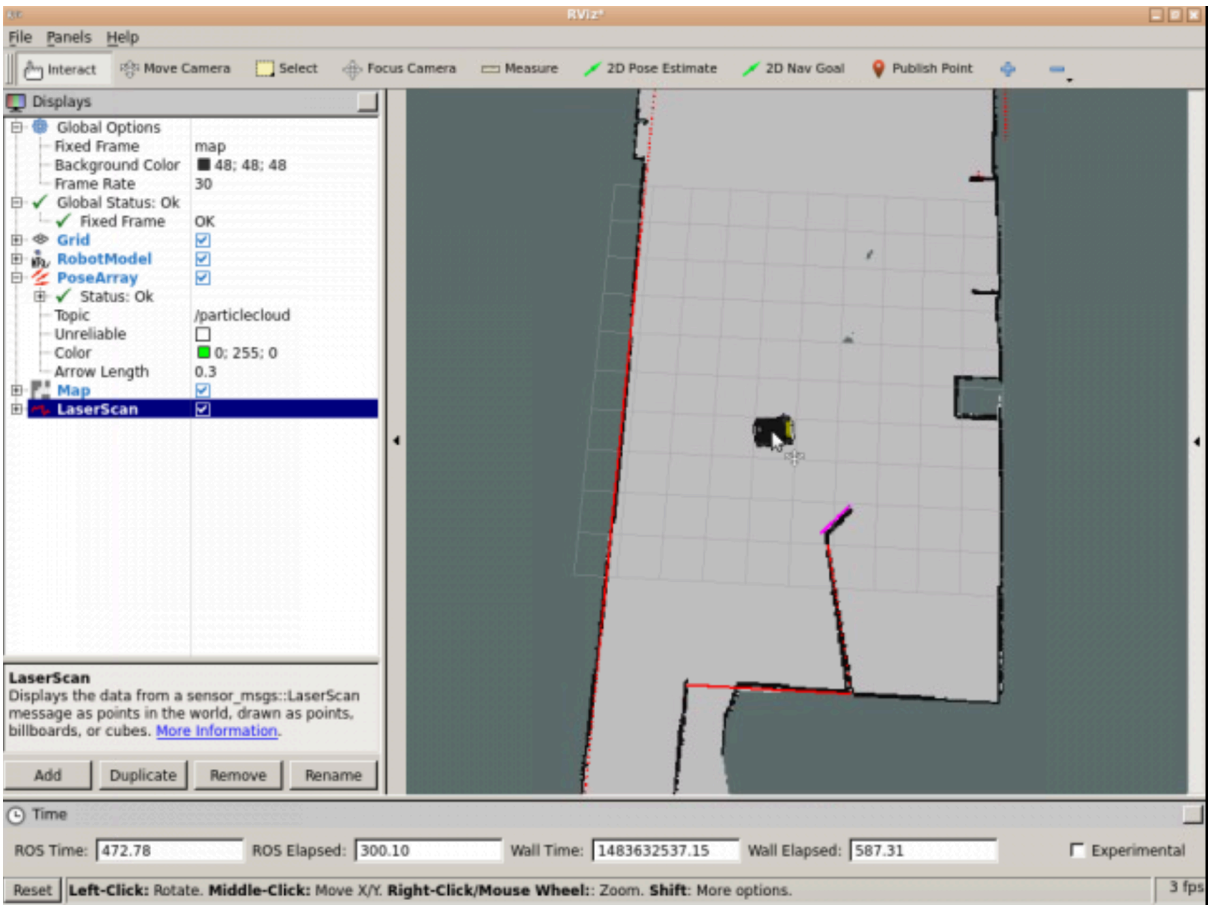
Nota 2: La dispersió del núvol representa la incertesa del sistema de localització sobre la posició del robot.

Nota 3: A mesura que el robot es mou per l'entorn, aquest núvol hauria de reduir-se en mida degut a les dades addicionals de l'escaneig que permeten a amcl refinar la seva estimació de la posició i orientació del robot.

Abans de moure el robot (vosaltres podeu treballar en el món que volgueu);



Després de moure el robot:



D'acord, però... què ha passat exactament? Què eren aquelles fletxes estranyes que estàvem visualitzant a RViz? Primer, introduïm alguns conceptes per entendre millor el que acabes de fer.

10. Localització de Montecarlo (MCL)

Com que el robot pot no sempre moure's com s'espera, genera moltes suposicions aleatòries sobre on es moure a continuació. Aquestes suposicions es coneixen com a partícules. Cada partícula conté una descripció completa d'una possible posició futura. Quan el robot observa l'entorn en què es troba (a través de lectures dels sensors), descarta les partícules que no coincideixen amb aquestes lectures i genera més partícules a prop de les que semblen més probables. D'aquesta manera, al final, la majoria de les partícules convergeixen en la posició més probable en què es troba el robot. Així que com més es mogui, més dades obtindrà dels sensors, per tant, la localització serà més precisa. Aquestes partícules són les fletxes del RViz.

Això es coneix com l'**algoritme de Localització de Montecarlo (MCL), o també com a localització de filtres de partícules.**

11. El paquet AMCL

El paquet AMCL (Localització de Montecarlo Adaptativa) proporciona el node amcl, que utilitza el sistema MCL per seguir la localització d'un robot que es mou en un espai 2D. Aquest node s'abona a les dades del làser, el mapa basat en el làser i les transformacions del robot, i publica la seva posició estimada al mapa. En iniciar-se, el node amcl inicialitza el seu filtre de partícules segons els paràmetres proporcionats.

NOTA: Com haureu pogut notar, per a denominar aquest paquet de ROS (i node), s'ha afegit la paraula "Adaptativa" a l'algoritme de Localització de Montecarlo. Això és perquè, en aquest node, podrem configurar (adaptar) alguns dels paràmetres que s'utilitzen en aquest algoritme. Aprendràs més sobre aquests paràmetres més endavant en aquest capítol.

Recopilant informació:

- Primer, vau iniciar un node amcl utilitzant el fitxer de llançament preconfigurat.
- En segon lloc, vau configurar una posició inicial utilitzant l'eina d'Estimació de Posició 2D (que va publicar aquesta posició al topic /initialpose).
- Després, vau començar a moure el robot per la sala, i el node amcl va començar a llegir les dades publicades en el topic del làser (/scan), el topic del mapa (/map) i el topic de la transformació (/tf), i va publicar la posició estimada on es trobava el robot en els temes /amcl_pose i /particlecloud.
- Finalment, a través de RViz, vau accedir a les dades que publicava aquest node en el tema /particlecloud, per la qual cosa vas poder visualitzar-les, gràcies al núvol de "fletxes", que indicaven la posició més probable en què es trobava el robot i la seva orientació.

Ara potser us pregunteu... però d'on obté aquest node el mapa? I aquesta és una gran pregunta! Anteriorment, vau aprendre sobre el node **map_server**, que permet proporcionar les dades d'un mapa des del fitxer del mapa.

Bàsicament, el que estem fent aquí és cridar aquesta funcionalitat per proporcionar les dades del mapa al node amcl

L'output del topic /amcl_pose:

```
ubuntu@ip-172-31-44-229:/home/user/catkin_ws/src/summit/summit_xl_common/summit_xl_navigation/launch$ rostopic echo -n1 /amcl_pose
header:
  seq: 0
  stamp:
    secs: 232
    nsecs: 502000000
  frame_id: map
pose:
  pose:
    position:
      x: -0.023827886945
      y: -0.0305145982103
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: -0.00486628383324
      w: 0.999988159571
  covariance: [0.22456029407813477, -1.306187293989016 2e-05, 0.0, 0.0, 0.0, 0.0, -1.306187293989016 2e-05, 0.17435311007245433, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.06808637427489805]
---
```

13. Requeriments del Hardware

Com hem vist en el tema del mapeig, la configuració també és molt important per localitzar correctament el robot a l'entorn.
Per aconseguir una localització adequada del robot, cal complir tres requisits bàsics:

- Proporcionar **Dades de Làser** de Qualitat
- Proporcionar **Dades d'Odometria** de Qualitat
- Proporcionar **Dades del Mapa** Basat en Làser de Qualitat

Assegureu-vos que el teu robot estigui publicant aquestes dades i identifiqueu tant el topic com el tipus de missatge que utilitza cada tema per publicar les dades.

Feu una ullada a un o més dels missatges que es van publicar en aquest tema per comprovar la seva estructura.

Nota 1: Tingueu en compte que per poder visualitzar aquests topics, el node amcl s'ha de llançar.

```
/imu/data/yaw/parameter_updates
/initialpose
/joint_states
/joy_teleop/cmd_vel
/map
/map_metadata
/navsat/fix
/navsat/fix/position/parameter_descriptions
/navsat/fix/position/parameter_updates
/navsat/fix/status/parameter_descriptions
/navsat/fix/status/parameter_updates
/navsat/fix/velocity/parameter_descriptions
/navsat/fix/velocity/parameter_updates
/navsat/vel
/odometry/filtered
/particlecloud
/platform_control/cmd_vel
/rosout
/rosout_agg
/scan
/set_pose
/tf
/tf_static
/twist_marker_server/cmd_vel
/twist_marker_server/feedback
```

14. Transformacions

Com també vam veure al capítol anterior (Mapejament), hem de publicar una transformació correcta entre l'eix del làser i la base de l'eix del robot. Això és bastant evident, ja que com ja heu après, el robot utilitza les lectures del làser per recalcular constantment la seva localització.

Més específicament, el node amcl té aquests 2 requisits pel que fa a les transformacions del robot:

- L'amcl transforma les lectures del làser entrants l'eix de l'odometria (~odom_frame_id). Per tant, ha d'haver un camí a través de l'arbre tf des del marc en què es publiquen les lectures del làser fins al marc de l'odometria.
- L'amcl busca la transformació entre l'eix del làser i l'eix de base (~base_frame_id), i la bloqueja per sempre. Per tant, amcl no pot gestionar un làser que es mogui respecte a la base.

Un cop més, vosaltres no us heu de preocupar d'aquest aspecte, però tingueu això en compte si algun dia dissenyeu el vostre robot.

15. Creació de l’arxiu launch per al node AMCL

Com heu vist en el Mapejament, també necessiteu tenir un fitxer de launch per iniciar el node amcl. Aquest node també és altament personalitzable i podem configurar molts paràmetres per millorar el seu rendiment. Aquests paràmetres es poden establir tant en el fitxer de launch com en un fitxer de paràmetres separat (fitxer YAML). Podeu veure una llista completa de tots els paràmetres que té aquest node aquí: http://wiki.ros.org/amcl

Si aneu al path següent, podreu trobar el launch on estan alguns d’aquests paràmetres:

/catkin_ws/src/turtlebot3/turtlebot3_navigation/launch

Fem una ullada a alguns dels més importants:

15.1 Paràmetres generals

- **odom_model_type (default: "diff")**: Col·loca el model d'odometria a utilitzar. Pot ser "diff," "omni," "diff-corrected," o "omni-corrected."
- **odom_frame_id** (per defecte: "odom"): Indica l'eix associat amb l'odometria.
- **base_frame_id** (per defecte: "base_link"): Indica l’eix associat amb la base del robot.
- **global_frame_id** (per defecte: "map"): Indica el nom de l’eix de coordenades publicat pel sistema de localització.
- **use_map_topic** (per defecte: false): Indica si el node obté les dades del mapa del topic o de la crida a servei.

15.2 Paràmetres del filtre

Aquests paràmetres permetran configurar la manera com el filtre de partícules realitza les seves operacions.

- **min_particles** (per defecte: 100): Estableix el nombre mínim permès de partícules pel filtre.
- **max_particles** (per defecte: 5000): Estableix el nombre màxim permès de partícules pel filtre.
- **kld_err** (per defecte: 0.01): Estableix l'error màxim permès entre la distribució real i la distribució estimada.
- **update_min_d** (per defecte: 0.2): Estableix la distància lineal (en metres) que el robot ha de moure's per realitzar una actualització del filtre.
- **update_min_a** (per defecte: π/6.0): Estableix la distància angular (en radians) que el robot ha de girar per realitzar una actualització del filtre.
- **resample_interval** (per defecte: 2): Estableix el nombre d'actualitzacions del filtre requerides abans de la re-mostratge.
- **transform_tolerance** (per defecte: 0.1): Temps (en segons) amb el qual s'endarrereix la transformació publicada, per indicar que aquesta transformació és vàlida en el futur.

- **gui_publish_rate** (per defecte: -1.0): Taxa màxima (en Hz) a la qual s'envien les exploracions i les rutes per visualització. Si aquest valor és -1.0, aquesta funció està desactivada.

15.3 Paràmetres del làser

Aquests paràmetres permetran configurar la manera com el node amcl interactua amb el làser.

- **laser_min_range** (per defecte: -1.0): Distància mínima de l'escaneig que es considerarà; -1.0 farà servir la distància mínima reportada pel làser.
- **laser_max_range** (per defecte: -1.0): Distància màxima de l'escaneig que es considerarà; -1.0 farà servir la distància màxima reportada pel làser.
- **laser_max_beams** (per defecte: 30): Quantitat de raigs espaiats uniformement en cada escaneig que es farà servir en actualitzar el filtre.