Introducción a SLAM

Simultaneous Localization And Mapping (SLAM) es un problema computacional que consiste en construir un mapa de un entorno desconocido, mientras el agente o robot navega dentro de este entorno.

Crear robot y un mundo vacío

Creamos un workspace llamado labo_robotica

```
mkdir labo_robotica
cd labo_robotica
mkdir src
catkin_make
source devel/setup.bash
```

Creamos un paquete llamado description. Trabajaremos con este único paquete

```
cd src
catkin_create_pkg description urdf
```

Para esta dirigida usaremos los archivos de: https://github.com/Franz04Rony/ros-robot
Entramos al paquete description, clonamos el repositorio y movemos todas las carpetas del repositorio un nivel arriba

```
cd description
git clone https://github.com/Franz04Rony/ros-robot.git
cd ros-robot
mv launch urdf world ../
```

```
catkin_make
```

Podemos comprobar si funciona correctamente ejecutando Rviz y Gazebo

```
roslaunch description rviz.launch
```

```
roslaunch description gazebo.launch
```

Agregar el plugin Laser sensor scanner al robot

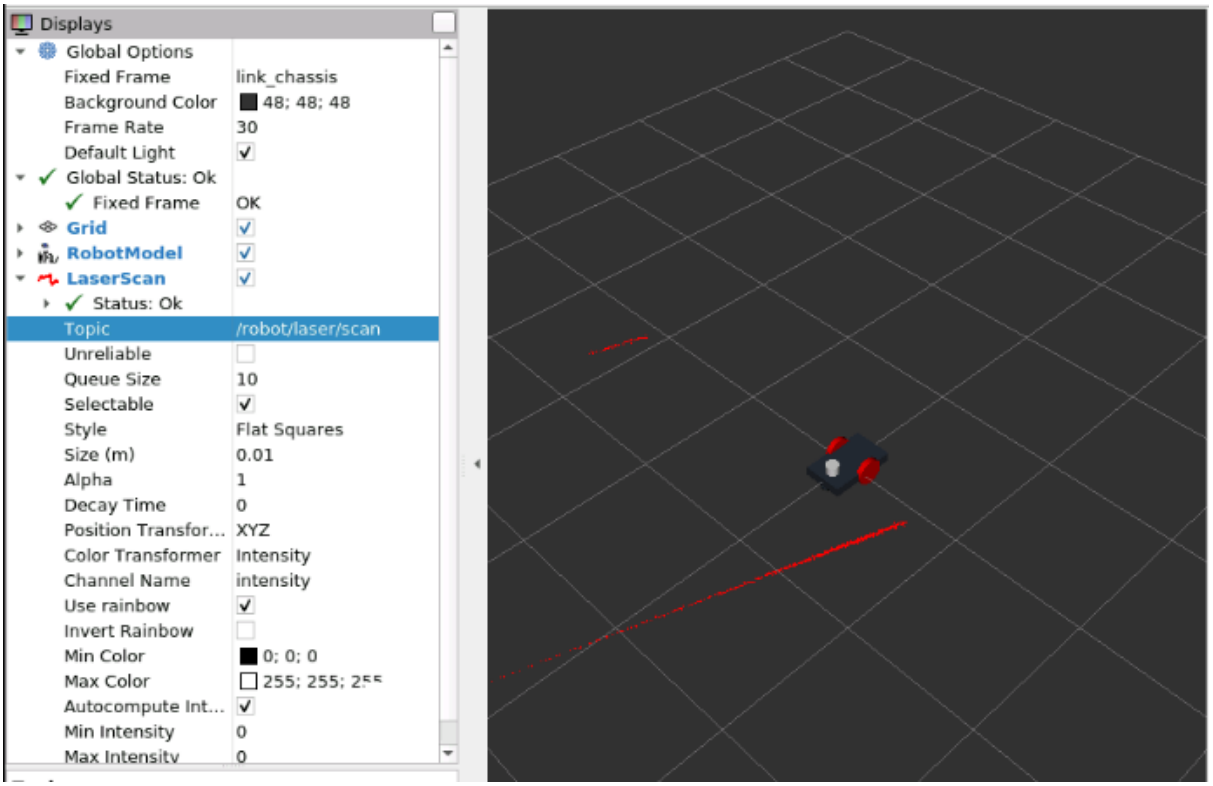
Agregar en el archivo urdf/robot.gazebo:

```
<gazebo reference="sensor_laser">
    <sensor type="ray" name="head_hokuyo_sensor">
      <pose>0 0 0 0 0 0</pose>
```

```
      <visualize>false</visualize>
      <update_rate>20</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>720</samples>
            <resolution>1</resolution>
            <min_angle>-1.570796</min_angle>
            <max_angle>1.570796</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.10</min>
          <max>10.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
        <topicName>/robot/laser/scan</topicName>
        <frameName>sensor_laser</frameName>
      </plugin>
    </sensor>
</gazebo>
```

Para comprobar el funcionamiento del sensor laser, volvemos a ejecutar Rviz y Gazebo.

```
roslaunch description rviz.launch
roslaunch description gazebo.launch
```



OBSERVACION: Rviz es solamente un visualizador, mientras que Gazebo es un simulador. Por lo que en Gazebo se puede realizar cambios a tiempo real del entorno.

Realizar SLAM sobre un entorno (mundo)

Cambiar en el archivo launch/gazebo.launch (linea 18). Aquí definiremos qué mundo usaremos

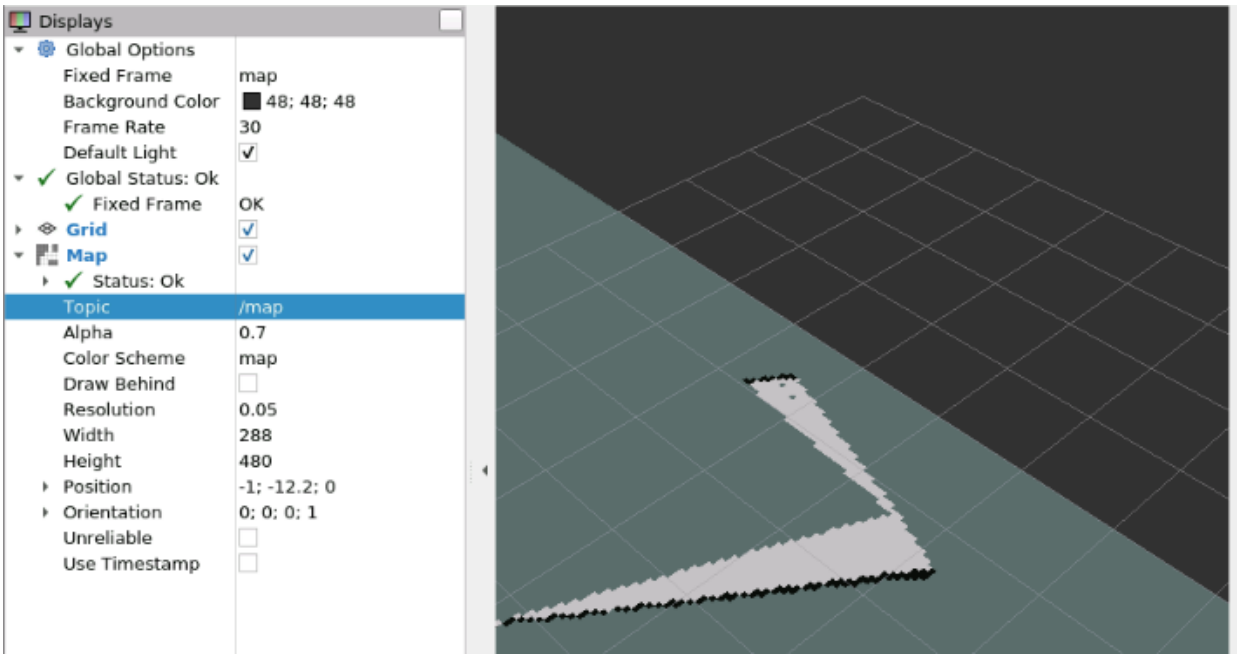
```
<arg name="world" default="$(find description)/world/world1.world" />
```

Dentro de la carpeta launch creamos un archivo llamado gmapping.launch, y dentro agregamos:

```
<launch>
  <arg name="scan_topic"  default="/robot/laser/scan" />
  <arg name="base_frame"  default="link_chassis"/>
  <arg name="odom_frame"  default="odom"/>

  <node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher"></node>

  <node pkg="rviz" type="rviz" name="rviz"></node>

  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
    <param name="base_frame" value="$(arg base_frame)"/>
    <param name="odom_frame" value="$(arg odom_frame)"/>
    <param name="map_update_interval" value="5.0"/>
    <param name="maxUrange" value="6.0"/>
    <param name="maxRange" value="8.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>
    <param name="minimumScore" value="200"/>
    <param name="srr" value="0.01"/>
    <param name="srt" value="0.02"/>
    <param name="str" value="0.01"/>
    <param name="stt" value="0.02"/>
    <param name="linearUpdate" value="0.5"/>
    <param name="angularUpdate" value="0.436"/>
    <param name="temporalUpdate" value="-1.0"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="80"/>
    <param name="xmin" value="-1.0"/>
    <param name="ymin" value="-1.0"/>
    <param name="xmax" value="1.0"/>
    <param name="ymax" value="1.0"/>

    <param name="delta" value="0.05"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>
    <remap from="scan" to="$(arg scan_topic)"/>
```

```
  </node>
</launch>
```

```
catkin_make
roslaunch description gazebo.launch
roslaunch description gmapping.launch
```



Para moverse a través del mundo, abrimos un nuevo terminal y usamos el comando:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```

Guardar el mapa generado

En caso no se tenga el paquete map-server. Se puede instalar con:

```
sudo apt-get install ros-noetic-map-server
```

Dentro del workspace description creamos la carpeta maps. Dentro de la carpeta maps guardamos nuestro mapa con el nombre "Mapa1":

```
rosrun map_server map_saver -f Mapa1
```

Separar el modelo del robot y SLAM en paquetes distintos

Una buena práctica en ROS es separar en paquetes distintos cada función del robot

Creamos un paquete llamado mapping

```
cd ~/labo_robotica/src
catkin_create_pkg mapping urdf
```

En el paquete mapping nos interesa tener solo el archivo que llama al nodo de gmapping, este es el gmapping.launch. Dentro de este paquete creamos una carpeta launch

```
cd mapping
mkdir launch
```

Movemos solamente el archivo gmapping.launch del paquete description al paquete mapping, dentro de la carpeta launch, y luego ponemos:

```
cd ~/labo_robotica
catkin_make
```

Probamos que funcione correctamente

```
roslaunch description gazebo.launch
roslaunch mapping gmapping.launch
```

Localizar el robot en el mapa guardado

Se utilizará AMCL(Adaptive Monte Carlo Localization). AMCL implementa el map_server para tomar un mapa estático y localizar el robot dentro de él usando un Localizador Monte-Carlo Adaptativo

Dentro del paquete mapping, creamos el archivo launch/amcl.launch y colocamos:

```
<launch>

  <!-- Map Server Arguments -->

  <arg name="map_file" default="$(find description)/maps/Mapa1.yaml"/>
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)" />
  <node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher"></node>
  <node pkg="rviz" type="rviz" name="rviz"></node>

  <!-- Arguments -->
  <arg name="scan_topic"     default="/robot/laser/scan"/>
  <arg name="base_frame"  default="link_chassis"/>
  <arg name="odom_frame"  default="odom"/>
  <arg name="initial_pose_x" default="0.0"/>
  <arg name="initial_pose_y" default="0.0"/>
  <arg name="initial_pose_a" default="0.0"/>

  <!-- AMCL -->
  <node pkg="amcl" type="amcl" name="amcl">

    <param name="min_particles"             value="500"/>
    <param name="max_particles"             value="3000"/>
    <param name="kld_err"                   value="0.02"/>
    <param name="update_min_d"              value="0.20"/>
    <param name="update_min_a"              value="0.20"/>
    <param name="resample_interval"         value="1"/>
    <param name="transform_tolerance"       value="0.5"/>
    <param name="recovery_alpha_slow"       value="0.00"/>
```

```
    <param name="recovery_alpha_fast"       value="0.00"/>
    <param name="initial_pose_x"            value="$(arg initial_pose_x)"/>
    <param name="initial_pose_y"            value="$(arg initial_pose_y)"/>
    <param name="initial_pose_a"            value="$(arg initial_pose_a)"/>
    <param name="gui_publish_rate"          value="50.0"/>
    <remap from="scan"                      to="$(arg scan_topic)"/>
    <param name="laser_max_range"           value="3.5"/>
    <param name="laser_max_beams"           value="180"/>
    <param name="laser_z_hit"               value="0.5"/>
    <param name="laser_z_short"             value="0.05"/>
    <param name="laser_z_max"               value="0.05"/>
    <param name="laser_z_rand"              value="0.5"/>
    <param name="laser_sigma_hit"           value="0.2"/>
    <param name="laser_lambda_short"        value="0.1"/>
    <param name="laser_likelihood_max_dist" value="2.0"/>
    <param name="laser_model_type"          value="likelihood_field"/>
    <param name="odom_model_type"           value="diff"/>
    <param name="odom_alpha1"               value="0.1"/>
    <param name="odom_alpha2"               value="0.1"/>
    <param name="odom_alpha3"               value="0.1"/>
    <param name="odom_alpha4"               value="0.1"/>
    <param name="odom_frame_id"             value="$(arg odom_frame)"/>
    <param name="base_frame_id"             value="$(arg base_frame)"/>

  </node>
</launch>


cd ~/labo_robotica
catkin_make


roslaunch description gazebo.launch
roslaunch mapping amcl.launch
```

Visualizar nodos y tópicos

Podemos visualizar los nodos y tópicos activos con:

```
rqt_graph
```

Problema propuesto

Conseguir el mapa de un entorno utilizando SLAM. Para ello se tiene que crear un workspace llamado problema1, y dentro del workspace dos paquetes, uno llamado description (irá todo lo relacionado al robot), y otro llamado slam (donde se correrá el nodo de gmapping). Guardar el mapa generado y localizar el robot en este mapa utilizando AMCL.
Entorno a usar: https://github.com/Franz04Rony/ros-robot/blob/main/world/world2.world