PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC-2685 ROBÓTICA MÓVIL

PROYECTO FINAL: INTRODUCCIÓN A ROS2 NAV2

Fecha de Presentación: viernes 11 de julio de 2025

Descripción del proyecto

Ha sido un largo semestre!. A través de él, hemos logrado programar los TurtleBots virtuales para darles instrucciones precisas de movimiento (control de bajo nivel), efectuar localización en un mapa conocido (robótica probabilística) y hacer que explore el ambiente (comportamientos reactivos). Ahora que ya comprendemos la teoría que sustenta a los sistemas de navegación, en esta oportunidad utilizaremos esta tecnología desde una perspectiva más aplicada. Para ello, haremos uso de un conjunto de paquetes de ROS que comunmente son conocidos bajo el nombre de *ROS2 Nav2* [1].

Actividad 1: Preparación del Simulador

Para configurar el stack de navegación de ROS2, comenzaremos creando un nuevo paquete denominado *proyecto_grupo_x_2024*, donde *"x"*, deberá ser reemplazado por su número de grupo.

Dentro del paquete recién creado, ustede deberá construir un archivo launch que inicialice los siguientes componentes:

Simulador de TurtleBot

Para simular las dinámicas de movimiento del TurtleBot, usted deberá levantar el nodo *kobuki_simulator* proporcionado por el paquete *very_simple_robot_simulator* [2], tal como se ve a continuación:

```
<node pkg="very_simple_robot_simulator" exec="kobuki_simulator.py" name="kobuki_simulator"
     output="screen" />
```

Simulador de LIDAR

De manera similar al caso anterior, para simular las lecturas realizadas por un LIDAR usted deberá levantar el nodo *lidar_simulator* incluido dentro del paquete *very_simple_robot_simulator* [2], como se muestra a continuación:

```
<node pkg="very_simple_robot_simulator" exec="lidar_simulator" name="lidar_simulator" />
```

Dentro de la declaración del tag <node>, configure los siguientes parámetros utilizando el tag <param>:

effective_hfov = 181.0
view_depth = 20.0

Para mayor información sobre como utilizar el tag <param>, véase [3].

Simulador de ambiente (World State)

Para simular el ambiente donde se desplazará nuestro robot, utilizaremos el nodo *world_state_gui* incluido dentro del paquete *very_simple_robot_simulator* [2]. Se recomienda que al lanzar este nodo, se cargue inmediatamente el mapa del entorno proporcionado para este proyecto (mapa_bodega.yaml, mapa_bodega.pgm), tal como se ejemplifica a continuación:

```
<node pkg="very_simple_robot_simulator" exec="world_state_gui.py" name="world_state_gui"
    output="screen" >
  <param name="map_file" value="$(find-pkg-share proyecto_grupo_x_2024)/mapa_bodega.yaml"
    />
</node>
```

Pregunta 1: ¿Qué dimensiones métricas tiene el mapa proporcionado?. Para realizar este cálculo, revise la información contenida en el archivo "mapa_bodega.yaml" y en su presentación explique como llegó al resultado.

Pregunta 2: ¿ Qué significa el campo *"origin"* del archivo "mapa_bodega.yaml" ?

Transformación de Coordenadas

En general, los robots están compuestos por una base móvil más una serie de sensores que los ayudan a percibir su entorno y a localizarse dentro de él. Estos sensores pueden estar ubicados en posiciones y ángulos arbitrarios sobre la base del robot, y por lo tanto, para conocer la posición del robot dentro de un ambiente, primero debemos **transformar** las coordenadas del sensor a las coordenadas del centro del robot.

Para poder realizar estas transformaciones, primero es necesario asignar un sistema de coordenadas a cada entidad relevante para el problema (base móvil, lidar, odometría, ambiente/mapa, etc.), y luego, definir una función matemática (transformación) que permita relacionar espacialmente cada uno de estos sistemas de coordenadas. A modo de ejemplo, en la figura 1 se muestra un robot móvil junto a los sistemas de coordenadas más comunes que podremos encontrar en navegación 2D.

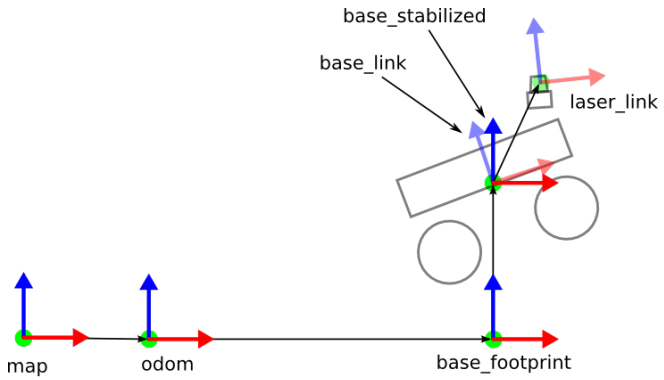

Figure 1: Definición de sistemas de coordenadas para un robot móvil.

Según lo anterior, para poder deducir la pose del centro del robot a partir de las mediciones del Lidar, tendremos que informar a ROS acerca de la diferencia de distancia y ángulo que existe entre el sistema de coordenadas de la base móvil y el sistema de coordenadas del Lidar. Para ello, utilizaremos el nodo de transformación *tf2_ros* de ROS2 [4]. Éste recibirá como parámetro de entrada la relación que existe entre la base móvil y nuestro Lidar mediante un vector del tipo: (x, y, z, yaw, pitch, roll), tal como se ejemplifica a continuación:

```
<node pkg="tf2_ros" exec="static_transform_publisher" name="base_link_to_laser" args="--x
    x --y y --z z --roll roll --pitch pitch --yaw yaw --frame-id base_link --child-frame-
    id laser" />
```

Pregunta: En el caso particular del simulador, consideraremos que el Lidar se encuentra ubicado justo en el centro del robot. Según esto, ¿ Qué valor debería tomar el vector de transfomación a publicar por tf2_ros (x, y, z, yaw, pitch, roll)?.

Visualizador de ROS (RViz)

Cuando trabajamos con robots móviles, ya sea en etapa de configuración o simplemente en operación normal, es de suma utilidad visualizar como van evolucionando las varibales del sistema mientras el robot se va desplazando en el ambiente. Para el caso de sistemas de navegación, en general necesitaremos visualizar: mapa, robot dentro del mapa, partículas generadas por el localizador, información entregada por los sensores, planeamiento de rutas, etc.

Para poder realizar esta tarea de manera sencilla, ROS2 provee el visualizador *RViz2* [5]. Este visualizador permite leer directamente de los tópicos del sistema y convertir los datos recibidos en información visual.

En caso de no contar con este módulo en su computador, puede instalarlo mediante el siguiente comando:

```
sudo apt install ros-humble-rviz2
```

Agregue el nodo *rviz* a su archivo launch de la siguiente manera:

```
<node pkg="rviz2" exec="rviz2" name="rviz" output="screen" />
```

Una vez que alcance este punto, pruebe que puede levantar su archivo launch de forma correcta. Al ejecutar su lanzador, deberá desplegarse tanto la interfaz de *RViz2* como la del simulador *World State*. Una vez inicidos los nodos, tenga las siguientes consideraciones:

y que puede agregar el tópico *world_map* en *RViz2*. Al ejecutar su lanzador, deberá desplegarse tanto la interfaz de *RViz2* como la del simulador *World State*. Luego de agregar el tópico *world_map*, realice las siguientes configuraciones en la interfaz gráfica de *RViz2*:

- Establezca el sistema de coordenadas fijo (fixed frame) como *"world_map"*.
- Agregue el tópico *"world_map"* para su visualización.
- Dentro del tópico *"world_map"*, cambie el parámetro *"Topic → Durability Policy"* a *"Transient Local"*.

Luego de los pasos anteriores, usted deberá poder visualizar el mapa en *RViz*.

Teleoperación de Robot

En determinadas ocasiones, es útil contar con una interfaz que permita a un humano controlar los movimientos del robot de forma manual. En esta oportunidad utilizaremos el paquete *teleop_twist_keyboard*, el cual nos permitirá enviar comandos de velocidad al robot a través del teclado. Para instalar dicho paquete, ejecute el siguiente comando:

```
sudo apt install ros-humble-teleop-twist-keyboard
```

Para poder levantar el nodo que permite controlar el robot, usted deberá ejecutar el archivo launch creado en los puntos anteriores, y en una segunda terminal, ejecutar el siguiente comando:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Una vez iniciado este nodo, sigas las instrucciones desplegadas en la pantalla para mover el robot.

Alternativamente, si usted tiene un joystick disponible, puede intentar utilizar el paquete *teleop_twist_joy* e iniciar el nodo de la siguiente forma:

```
ros2 run teleop_twist_joy teleop_node
```

Reporte

Como resultado de esta actividad:

- Conteste las preguntas planteadas más arriba.
- Muestre una visualización del mapa en *RViz*.

Actividad 2: Mapping

Mapeo con Slam Toolbox

Una vez que haya probado el correcto funcionamiento de los nodos detallados en la Actividad 1, utilizaremos el paquete *slam_toolbox* para generar el mapa del ambiente *"real"* dado por *Word State*. Para instalar dicho paquete, ejecute el siguiente comando en la terminal:

```
sudo apt install ros-humble-slam-toolbox
```

Posteriormente, siga los siguientes pasos para realizar la construcción del mapa:

- Levante el archivo launch generado hasta ahora, el cual debería contener los nodos del simulador (kobuki_simulator, lidar_simulator, world_state), tranformación (tf2_ros) y Rviz2.
- En una segunda terminal, levante el nodo *"sync_slam_toolbox_node"* utilizando el siguiente comando:

```
ros2 run slam_toolbox sync_slam_toolbox_node --ros-args -p odom_frame:=odom --ros-
    args -p base_frame:=base_link --ros-args -p map_frame:=map --ros-args -p
    scan_topic:=/scan --ros-args -p map_update_interval:=1.0 --ros-args -p
    max_laser_range:=5.0 --ros-args -p minimum_travel_distance:=0.1 --ros-args -p
    use_scan_matching:=true --ros-args -p minimum_travel_heading:=1.57 --ros-args -p
     do_loop_closing:=true
```

- En la interfaz gráfica de *RViz*, establezca el sistema de coordenadas fijo (fixed frame) como *"map"*.
- Agregue el tópico *"map"* para su visualización.
- En una tercera terminal, inicialice el nodo de teleoperación (teleop_twist_keyboard). A través de esa ventana, usted deberá comandar los movimientos del robot. Opcionalmente, usted puede usar un joystick en caso de contar con uno.
- Haga que el robot recorra todo el ambiente, y a través de rviz, asegúrese de que el mapa se va construyendo de manera adecuada.
- Una vez que el mapa está completamente construido, guardelo en un rachivo a través del siguiente comando:

```
ros2 run nav2_map_server map_saver_cli -f my_first_slam_map
```

Pregunta 1: ¿Que archivos se generan luego de ejecutar el nodo map_saver_cli?
Pregunta 2: ¿Cuál es la resolución del mapa generado ([m/pix])?
Pregunta 3: ¿Cómo podría generar el mismo mapa pero con otra resolución?
Pregunta 4: ¿Por qué el sistema de coordenadas fijo debe establecerse como *map* y no como *world_map*?

Reporte

Como resultado de esta actividad:

- Conteste las preguntas planteadas más arriba.
- Presente mapa generado para el ambiente completo.

Actividad 3: Localization

Localizador Monte Carlo Adaptivo (AMCL)

En el Laboratorio 3, usted programó un algoritmo de localización basado en la técnica *Monte-Carlo Localization* (o Filtro de Partículas), el cual permite localizar al robot dentro de un ambiente utilizando un mapa de ocupación. En esta oportunidad, utilizaremos el algoritmo *Adaptive Monte Carlo Localization* (AMCL) [6] que se encuentra implementado en ROS2.

Pregunta 1: ¿Cuál es la principal diferencia entre el algoritmo AMCL y el MCL estudiado en clases?.

Para instalar el módulo de ROS2 que contiene AMCL usted deberá ejecutar el siguiente comando:

```
sudo apt install ros-humble-nav2-amcl
```

Para utilizar este algoritmo, debemos inicializar el nodo *amcl* dentro de nuestro archivo launch de la siguiente forma:

```
<node pkg="nav2_amcl" exec="amcl" name="amcl" />
```

Sin embargo, este nodo posee una gran variedad de parámetros que pueden ser configurados a través del tag <param>. En esta oportunidad, configuraremos nuestro nodo *amcl* con los siguientes parámetros:

global_frame_id: world_map base_frame_id: base_link odom_frame_id: odom robot_model_type: nav2_amcl::DifferentialMotionModel use_map_topic: true map_topic: world_map laser_max_beams: 181 min_particles: 250

Una vez agregado y configurado el nodo *amcl* en su archivo launch, ejecútelo para verificar que todo levanta correctamente.

Pregunta 2: Investigue cuál es la utilidad de los parámetros del cuadro anterior y explíquelos brevemente en su presentación.

Activación de AMCL

Para que el nodo *AMCL* pueda ser utilizado luego de ser inicializado, usted deberá activarlo a través del nodo *lifecycle_bringup*. Este nodo es parte del paquete *nav2_util*, el cual debe ser previamente instalado utilizando el siguiente comando:

```
sudo apt install ros-humble-nav2_util
```

Una vez instaldo el paquete *nav2_util* levante su archivo launch (con el nodo *amcl* en su interior), y en otra terminal, ejecute el siguiente comando para activarlo:

```
ros2 run nav2_util lifecycle_bringup amcl
```

Si la activación es correcta, usted podrá ver un mensaje indicando la activación de *amcl* en la terminal donde levantó su archivo launch.

Establecimiento de pose inicial (initialpose)

Como ya mencionamos, el nodo *AMCL* implementa el algoritmo de localización basándose en un filtro de partículas. A diferencia de la implementación que realizamos para el laboratorio 3, *AMCL* comienza lanzando partículas solamente en una vecindad del robot. De esta forma, puede converger en un tiempo razonable y con una buena precisión, independientemente del tamaño del mapa.

Según lo anterior, luego de levantar y activar el nodo *AMCL*, debemos publicar la pose aproximada donde se encuentra el robot al momento de comenzar el algoritmo. Para publicar la pose inicial aproximada, RViz nos proporciona el botón *"2D Pose Estimate"*, el cual permite establecer la pose inicial del robot sobre el mapa a través de una flecha.

Reporte

Como resultado de esta actividad:

- Conteste las preguntas planteadas más arriba.
- Visualice la pose localizada del robot en *RViz*, agregando el tópico *amcl_pose*.
- ¿Que simboliza el cículo que identifica al robot en el mapa?
- Explique la relación entre la posición del robot en interfaz de simulador "World State" y posición en Rviz. ¿Ambas poses deberían ser siempre exactamente iguales?.
- ¿Cómo podría establecer una pose inicial por defecto en AMCL ? (ver documentacion [6])

Actividad 4: Navigation2

Stack de Navegación de ROS2

Una vez que tenemos información de los sensores, mapa del entorno, transformación de coordenadas y localización del robot, solo nos falta definir un punto objetivo y calcular la mejor ruta hasta él.

Para todas estas tareas (y más!) utilizaremos el stack de navegación de ROS que se encuentra implementado por el paquete *nav2* [1]. Para instalar los módulos que utilizaremos en esta sección, ejecute el siguiente comando:

```
sudo apt install ros-humble-navigation2 ros-humble-nav2-bringup
```

Debido a la alta complejidad de *nav2*, este paquete posee un gran número de parámetros de configuración que no alcanzaremos a analizar con detalle en el presente proyecto. Sin embargo, junto a este documento usted encontrará un archivo de configuración que definirán los parámetros que necesitamos para nuestra actividad: *nav2_params.yaml*.

Para mantener un orden, crearemos el directorio *"params"* dentro del paquete del proyecto, y almacenaremos el archivo *nav2_params.yaml* dentro de él. Haga los cambios necesarios dentro de su archivo *setup.py* de manera tal que el archivo de configuración sea accesible en tiempo de ejecución.

A continuación, incluiremos el sistema de navegación de *nav2* a nuestro archivo launch. Para ellos, incluya el siguiente tag dentro de su archivo:

```
<include file="$(find-pkg-share nav2_bringup)/launch/navigation_launch.py" >
  <arg name="params_file" value="$(find-pkg-share proyecto_grupo_test_2024)/nav2_params.
  yaml" />
</include>
```

Finalmente:

- Ejecute su archivo launch.
- Active el nodo AMCL.
- Agregue todos los tópicos que desee a *RViz* (map, amcl_pose, scan, path, etc.).

- Establezca la pose inicial de su robot.
- Envíe a su robot a una pose deseada mediante el botón *"2D Nav Goal"*.

It's Alive !!!

Demostración

Para la demostración, usted deberá:

1. Levantar todos los nodos antes señalados mediante el lanzamiento de un archivo launch confeccionado por usted. Este archivo launch deberá configurar *RViz* de manera tal que todos los tópicos de interés sean visibles inmediatamente.
2. Una vez levantado el conjunto de nodos, ustede deberá visualizar al menos:
    - Mapa
    - Pose del robot proporcionada por AMCL
    - Lecturas del Lidar
    - Ruta global planeada (global_path).
3. Enviar el robot a una pose arbitraria (no trivial).

Pregunta 1: ¿ Las lecturas del Lidar visualizadas son coherentes con la geometría del mapa?. En caso de que no lo sean, ¿por qué ocurren las discrepacias?.

References

[1] https://docs.nav2.org/

[2] https://github.com/gasevi/very_simple_robot_simulator

[3] http://wiki.ros.org/roslaunch/XML/param

[4] https://docs.ros.org/en/humble/Concepts/Intermediate/About-Tf2.html

[5] https://docs.ros.org/en/humble/Tutorials/Intermediate/RViz/RViz-User-Guide/RViz-User-Guide.html

[6] https://docs.nav2.org/configuration/packages/configuring-amcl.html