UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Trabajo de ROS2

Eugenio Ivorra

Índice general

Índice general ii

1 TRABAJO 1

1.1 Introducción 1
1.2 Tarea 0: Mapeo de la Casa (1 Puntos) 1
1.3 Tarea 1: Servicio de Comandos y Cliente de Prueba (3 Puntos) 2
1.4 Tarea 2: Script de Búsqueda del Tesoro (3.5 Puntos) 3
1.5 Entrega del Proyecto 4

A ANEXOS 5

A.1 Manejo de Paquetes Locales Debian 5

1 TRABAJO

1.1 Introducción

Este documento contiene las instrucciones detalladas para el proyecto final de la asignatura, el cual está centrado en el aprendizaje y la aplicación práctica de ROS2, así como en las herramientas asociadas Nav2, Gazebo y Turtlebot3.

1.2 Tarea 0: Mapeo de la Casa (1 Puntos)

1.2.1 Descripción

Se deberá crear un mapa detallado del entorno de la casa utilizando el modelo Turtlebot3 en el simulador Gazebo.

1.2.2 Requisitos

- Inicializar el entorno de simulación con Turtlebot3

```
ros2 launch turtlebot3_gazebo turtlebot3_house.launch.py
```

- Utilizar paquetes de ROS2 para el mapeo automático del entorno (Nav2 y Slam toolbox).
- Asegurarse de que todas las áreas sean exploradas y mapeadas adecuadamente.
- Guardar el mapa generado para su uso posterior.

> Nota: Es posible corregir el mapa generado (.pgm) con una aplicación de edición de imagen tipo GIMP teniendo en cuenta que negro es obstáculo y blanco es paso libre. Es importante que las paredes de la casa estén todas cerradas para que a la hora de navegar el robot no intente salir por algún hueco.

1.3 Tarea 1: Servicio de Comandos y Cliente de Prueba (3 Puntos)

Esta tarea se centra en la creación de un servicio de servidor en ROS2 capaz de manejar comandos específicos para la navegación y actividades del Turtlebot3 y un cliente que permita probar este servicio de forma efectiva.

1.3.1 Servidor de Servicios (2 puntos)

Se desarrollará un servidor de servicios en ROS2 que responderá a comandos de navegación, gestionando tareas de patrullaje y salida del entorno simulado.

Detalles

- **Interfaz de Servicio Personalizada:** Definir una interfaz de servicio personalizada en ROS2 que incluirá la especificación de mensajes de solicitud y respuesta para los comandos del robot.
- **Lógica de los Comandos:** Implementar la lógica necesaria para manejar dos comandos fundamentales:

    1. **Patrullar**: Este comando hará que el Turtlebot3 navegue a través de todas las estancias de la casa. Se deben establecer puntos intermedios para optimizar la ruta de patrullaje y prevenir que el robot se quede atascado.
    2. **GoToExit**: Al ejecutar este comando, el robot deberá navegar hacia la salida de la casa.

- **Retroalimentación de los Comandos:** Después de ejecutar cada comando, el servidor deberá proporcionar una respuesta indicando si el Turtlebot3 logró alcanzar el objetivo propuesto o si encontró algún problema durante la ejecución.

1.3.2 Cliente de Servicios (0.5 puntos)

El cliente de servicios en ROS2 se encargará de enviar solicitudes al servidor y de manejar las respuestas obtenidas.

Funcionalidades

- **Envío de Comandos:** Crear un cliente capaz de enviar los comandos *Patrullar* y *GoToExit* al servidor de servicios.

- **Recepción y Procesamiento de Respuestas:** El cliente recibirá las respuestas del servidor y deberá mostrar el resultado por consola.

1.3.3 Fichero de Launch (0.5 puntos)

Hacer un fichero de Launch que ejecute el nodo servidor y el nodo cliente.

1.4 Tarea 2: Script de Búsqueda del Tesoro (3.5 Puntos)

Programar un script en Python que simule una búsqueda de tesoro con el Turtlebot3 en la simulación. Se proporciona un paquete *busquedaTesoro.deb* que publica en el topic */distanciaTesoro* la distancia entre la pose del robot y el punto donde se encuentra oculto el tesoro cuando el topic */busquedaTesoro* está a True. El topic */distanciaTesoro* publica un vector de 3 valores donde X es la distancia entre el robot y el tesoro en el eje X, Y es la distancia entre el robot y el tesoro en el eje Y y Z es la distancia euclídia entre el robot y el tesoro.

1.4.1 Especificaciones

- Instalar el paquete *busquedaTesoro.deb*. En los anexos se encuentra un manual de instalación.

- Activar la búsqueda estableciendo a True el topic */busquedaTesoro* al inicio del script.

- Ejecutar el comando para iniciar el cálculo de la distancia utilizando el nodo proporcionado por el paquete:

```
ros2 run tesoro_pkg tesoro_nodo
```

- Basándose en la distancia al tesoro, implementar un algoritmo en el script que haga que el robot navegue hasta que encuentre el tesoro.

- La búsqueda será un éxito si el robot encuentra el tesoro en menos de 90 segundos. En caso contrario, se debe considerar un fracaso, detener el robot y establecer el topic */busquedaTesoro* a False.

- Se considera que lo ha encontrado si la distancia total es menor de 0.5 m.

- El script debe proporcionar información por consola sobre el tiempo restante para encontrar el tesoro a lo largo de la ejecución.

1.5 Entrega del Proyecto

La entrega se realizará en el espacio compartido de Poliformat. La fecha máxima de entrega es el día 17 por la mañana. Se puede realizar por parejas en cuyo caso solo lo subirá uno de los miembros.

- Vídeo demostrativo de todas las funcionalidades (10 minutos máximo). 1 Punto
- Memoria descriptiva del trabajo desarrollado (15 hojas máximo). 1 Punto
- Código fuente completo y comentado. 0.5 Punto

A ANEXOS

A.1 Manejo de Paquetes Locales Debian

A.1.1 Instalación

Procederemos a instalar este paquete Debian en nuestra propia máquina utilizando el programa de gestión de paquetes llamado `dpkg`:

```
sudo dpkg --install ros-humble-tesoro-pkg_0.0.0-0jammy_amd64.deb
```

Esto instalará el paquete Debian (incluyendo la descompresión y configuración) en el sistema de archivos del disco duro.

A.1.2 Información de la instalación

Una vez instalado el paquete Debian, estará listo para ser utilizado en nuestro entorno. Para determinar dónde se ha instalado el paquete, primero necesitamos obtener el nombre del paquete:

```
dpkg --info ros-humble-tesoro-pkg_0.0.0-0jammy_amd64.deb
```

Con el nombre en mano, podemos mostrar en qué lugar se instalaron los archivos del paquete `deb`:

```
dpkg -L ros-humble-tesoro-pkg
```

Podemos ver que este paquete se ha instalado en la misma ubicación donde se encuentra la instalación principal de ROS 2.

A.1.3 Desinstalación

Finalmente, si deseamos desinstalar este paquete, podemos hacerlo fácilmente también con `dpkg`:

```
sudo dpkg -P ros-humble-tesoro-pkg
```