Róbotica Laboratorio 1

Martín Galván Castro
Dpto. Ingeniería de Sistemas y Computación
Universidad de los Andes
Bogota, Colombia
ms.galvan@uniandes.edu.co

Santiago Sinisterra Arias
Dpto. Ingeniería Electrónica
Universidad de los Andes
Bogotá, Colombia
s.sinisterra@uniandes.edu.co

María Reales Camacho
Dpto. Ingeniería Electrónica
Universidad de los Andes
Bogotá, Colombia
m.realesc@uniandes.edu.co

I. Instrucciones de Uso

A. Dependencias

Antes de instalar el archivo para hacer uso de los nodos, es recomendable instalar y configurar las dependencias del proyecto. En este caso, solo se utilizan dos librerías. La primera siendo matplotlib. Y la segunda siendo pynput. A continuación se hablara más a fondo sobre estas.

1) Matplotlib: Matplotlib es una librería para la creación de visualizaciones y gráficas en python. Esta se necesita para poder utilizar de manera correcta el nodo interfaz.

Para instalar esta librería, basta con abrir una terminal de comandos y poner el comando ”pip install matplotlib”. Alternativamente, se puede hacer el mismo procedimiento con el package manager de preferencia. El mismo comando para las distribuciones Debian y Ubuntu seria ”sudo apt-get install python3-matplotlib.

2) Pynput: Pynput es una librería que permite el control y el monitoreo de dispositivos de entrada. En especifico, para monitoriar entradas del teclado. Esta es la librería usada por el nodo de teleoperación. Para su instalación, el mismo proceso para instalar es usado. ”pip install pynput” o el equivalente en el manejador de paquetes de su preferencia. Sin embargo, se debe hacer un procedimiento adicional para que funcione correctamente.

Pynput solo corre correctamente con X. X se refiere a un servidor de visualización. estos son programas responsables de gestionar y coordinar la salida y entrada de varios componentes. En este caso, la entrada del teclado de la maquina virtual. La maquina virtual con la que se trabajo opera por defecto con otro servidor de visualización llamado Wayland. Para desactivarlo, hay que seguir las siguientes instrucciones:

1) Encontrar el archivo de configuración del sistema operativo. En Ubuntu esta es /etc/gdm3/custom.conf
2) Realizar el comando sudo nano sobre este archivo
3) Buscar la linea comentada que dice ”WaylandEnable=false”
4) Descomentarla y guardar el archivo
5) Reiniciar

B. Instalación del paquete

Se puede instalar el trabajo realizado de dos distintas formas. La primera, es clonar el repositorio en el que se trabajo a partir de git. Y la segunda es usando el zip. Este taller no pretende ser una guia de como clonar repositorios, asi que se recomienda utilizar el zip y descomprimirlo en su área de trabajo.
Despues, llamar el comando ”colcon build” para construir los paquetes. Esto genererara todo lo necesario para empezar a correr los archivos, pero antes de correr algo, hay que llamar el comando source en ”install/setup.bash” en todas las terminales de comando en las que se este trabajando.
Por ultimo, para realizar las llamadas a los nodos e inicarlos hay tres entry point:

1) turtle_bot_teleop
2) turtle_bot_interface
3) turtle_bot_player

II. Funcionamiento

A continuación, se explicara como funcionan los 3 nodos desarrollados para el cumplimiento del taller.

A. turtle_bot_teleop

Para la operación de este nodo, fue necesario implementar threads que se ejecutan a la par del hilo de ejecución principal, y 2 recursos que son compartidos por todos los hilos. Los hilos que se ejecutan en paralelo se encargan de monitoriar el estado de 5 teclas. Estas son las teclas w, a, s, d y la tecla ”Esc”. Se incluye esta ultima como mecanismo para cerrar la ejecución del nodo.

1) Recursos compartidos: Los recursos compartidos mencionados anteriormente se refieren a 2 variables sobre las cuales todos los threads tienen acceso. La primera variable es un booleano que se utiliza como mecanismo para marcar el final de la ejecución del nodo. La segunda variable es un diccionario, cuyas llaves representan cada una de las

teclas que se usan para controlar al robot, y sus valores son booleanos que indican si esta siendo presionada. De esta forma, cuando se presiona una tecla, el hilo de ejecución, o ”Listener” encargado de revisar el estado de esa tecla, accederá al diccionario, y cambiara el valor del booleano. De igual manera, cuando se deje de presionar la tecla, vuelve a acceder al diccionario y cambia el valor del booleano a False.

2) Nodo: Al iniciar un nodo, se crea un publicador que publica en el tópico ”/turtlebot_cmdVel”, y se crea un timer, el cual termina llamando una función para enviar la velocidad al turtle bot.

El comando que envía la velocidad al robot empieza creando un objeto ”Twist” con velocidades 0 en lineal y revisa el diccionario compartido, y revisa que teclas son presionadas. En caso de que una tecla es presionada, modifica el objeto twist. Si presional las teclas ”w” o ”s” termina sumando o restando la velocidad linear x. Y si presiona las teclas ”a” o ”d”, termina sumando o restando las velocidades angulares. Después, publica el objeto twist en el topico.

3) Ejecución del programa: El nodo principal, o main, empieza preguntando por la velocidad lineal y linear para enviar al robot. Despues de eso, llama una función que crea todos los Listeners y los inicia. Después, crea el nodo de ROS 2 y empieza un ciclo while que pregunta continuamente el valor del booleano de continuar. Dentro de este ciclo, invoca el comando spin del nodo una vez. Al salir del ciclo, cierra el nodo y termina la ejecución del programa.

B. turtle_bot_interface

Este nodo se termina encargando de múltiples tareas. Esta encargado de graficar en tiempo real, se encarga de guardar la entrada del teleop en un archivo .txt, y también funciona como un cliente para enviar el nombre de un archivo .txt a turtle_bot_player para que replique sus contenidos. Sin embargo, para la ejecución de este solo se terminaron creando dos nodos. Uno encargado de continuamente llamar al comando ”spin” del nodo, y el principal que es en donde se crea y ejecuta la interfaz. Adicionalmente, se tienen dos variables compartidas en los dos hilos, que consisten en listas que van a contener los vectores de posición del robot.

1) Interfaz: En la interfaz, se puede observar, el espacio de una grafica de matplotlib, una barra de utilidades de matplotlib, y multiples botones. La primera linea de botones que aparece se utilizan para el control de la grafica. Un botón para empezar la grafica, otro para detenerla, y otro para limpiar el contenido de la grafica. La segunda linea de botones son botones que se utilizan para interactuar con turtle_bot_player. Creando un cliente, y enviando un archivo.
La forma en la que la grafica opera, es que cuando se llama al boton ”start”, se empieza a graficar lo que se reciba del tópico ”/turtlebot_position”. La forma en la que esto se logra se va a explicar en la sección del nodo.
Por otro lado, el control del cliente, se basa con 2 botones. Uno para crear el cliente, el cual, accede al nodo que se va a crear, y crea un cliente en este. El segundo botón, al presionarlo, en caso de que el cliente halla sido creado exitosamente, va a pedir un archivo. Al seleccionarlo, va a enviar una request al servicio enviándole el nombre del archivo.

2) Nodo: El nodo se subscribe a dos posibles tópicos, si se va a capturar la entrada del nodo teleop, se suscribe a este. Pero siempre se va a suscribir al topico de posición. Cuando recibe un mensaje del tópico de posición, llama a los dos vectores que son compartidos, y adjunta la posición en x y en y a estos vectores. Como esto se hace de manera sincrónica, la interfaz solo tiene que leer estos vectores y graficarlos para mostrar un cambio en la posición.
por otro lado, cuando el nodo guarda la entrada del nodo turtle_bot_interface, cuando recibe mensajes del topico de velocidad, escribe una nueva linea en el archivo en donde se guardara esta entrada. En esta linea, se escriben dos números, separados por una coma. En donde el primer numero es la velocidad lienar en x y el segundo la velocidad angular.

3) Ejecución del programa: Al ejecutar el programa, primero aparece una ventana preguntando si se desea guardar la entrada del teleop. Si se dice que si, otra ventana preguntara el lugar y nombre del archivo sobre el cual se va a guardar la entrada. Posteriormente, se inicia la interfaz, en donde se pueden usar los botones para acceder a las funcionalidades anteriormente mencionadas de la interfaz.

C. turtle_bot_player

El ultimo nodo es el mas sencillo. Opera con un unico hilo de ejecución, y consiste en un servicio y un publicador. Cuando el nodo recibe el mensaje del cliente de la interfaz, busca el archivo, lo lee, y por cada linea en esta, publica la velocidad linear y angular al topico de velocidad.

III. PUNTO 1

En la figuras se puede denotar el correcto funcionamiento del teclado para ingresar los movimientos requeridos en el tópico deseado.

En la figura 7 se refleja que se puede iniciar con éxito la función al ejecutar en una terminal el nodo de $/turtle\_bot\_teleop$ e ingresar la velocidad lineal y angular a aplicar.

Pues, como demostrado en la figura 8 utilizando las teclas *wasd* en la terminal donde se inicializa se permite cambiar la velocidad, lo que se ve reflejado en la terminal de coppelia, ya que se demuestra que en esta se registran los valores de velocidad acorde a lo que se ingresa en el teclado, y si no se ingresa algún valor como se detiene el robot.
De igual manera se denota como al tener un listener para cada una de las teclas, permite que se realicen movimientos con un incremento en la velocidad linear y angular que requieren el

registro de dos de las teclas al mismo tiempo, y se evidencian como el incremento de velocidad en dos diferentes ejes.

Con respecto a la gráfica rqt de la figura 9 anterior se puede reafirmar el correcto uso del nodo y el tópico al establecer la relación de publisher hacia el tópico correspondiente.

Con las figuras anteriores se considera que es un nodo eficiente al no solo registrar las teclas esperadas con las correspondientes velocidades, pero tambien al permitir múltiples entradas al mismo tiempo, y poder actualizar la velocidad acorde a cualquier combinación de inputs, inclusive cancelando la velocidad en el momento de registrar dos posiciones opuestas.

IV. Punto 2

En la figura 10 se encuentra como, inicializando el nodo de $/turtle\_bot\_interface$ se encuentra la interfaz con todas las opciones necesarias.

En la figura 11 se encuentran presentes los botones de start, stop y clear, con los que se puede registrar en el rango de tiempo deseado la posición, y por consecuencia el recorrido, del robot. Start inicializando el registro, Stop deteniéndolo, y Clear borrando el registro presente. En la misma interfaz se encuentra un plano donde se gráfica el recorrido registrado por el usuario en tiempo real.

De forma consecuente, en la figura 12con el uso del botón de guardado se visualiza la ventana emergente para guardar la imagen del recorrido con el directorio y nombre deseado por el usuario.

Con respecto a la gráfica rqt de la figura 13, se demuestra que el nodo de la interfaz se encuentra suscrito al tópico de posición para permitir la visualización tiempo real de donde se encuentra el robot.

Analizando las figuras anteriores se puede considerar que la interfaz es óptima al permitir con el uso de pocos botones el visualizar en tiempo real un recorrido y guardar una ejecución como una imágen, además de permitir múltiples funcionalidades opcionales de visualización, y manejar posibles errores de ejecución tales como oprimir start cuando ya haya iniciado un registro y demás situaciones.

V. Punto 3

En las imágenes de las figuras 14 y 15 se encuentra como, al momento de ejecutar el nodo de la interfaz, sale una ventana emergente que pregunta si se desea guardar el recorrido del robot. En caso de negarse sigue directo a la interfaz, y al aceptarse se pregunta por el nombre del archivo donde se registrara el recorrido y el directorio donde se desea guardar.

Al acceder al archivo txt una vez finalizado el recorrido, como visto en la figura 16 se encuentra la secuencia de velocidades del recorrido realizado con respecto a como se registraron en el tiempo de ejecución.

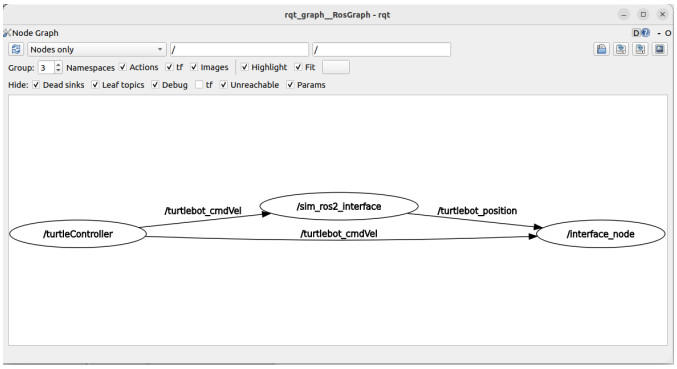

Fig. 1: Grafo rqt de guardado de recorrido

En el rqt de la figura 1 se evidencia como la interfaz se suscribe al tópico de la velocidad para poder registrar en el documento la velocidad del recorrido.

Considerando lo visto anteriormente, se puede analizar que el guardado del recorrido es conveniente y ligero, al utilizar solamente un archivo txt con las velocidades registradas, permite que el archivo sea ligero y conserve los datos de una forma compacta y directa. De igual manera al implementar una ventana emergente, permite que el registro de los datos sea simple para el usuario.

VI. Punto 4

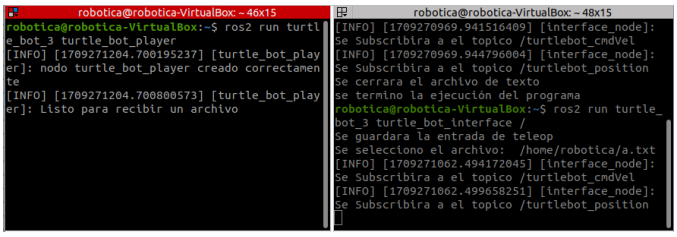

Fig. 2: Inicialización del player

Al inicializar el nodo de $/turtle\_bot\_player$ se habilita la posibilidad de reproducir cualquier recorrido guardado anteriormente.

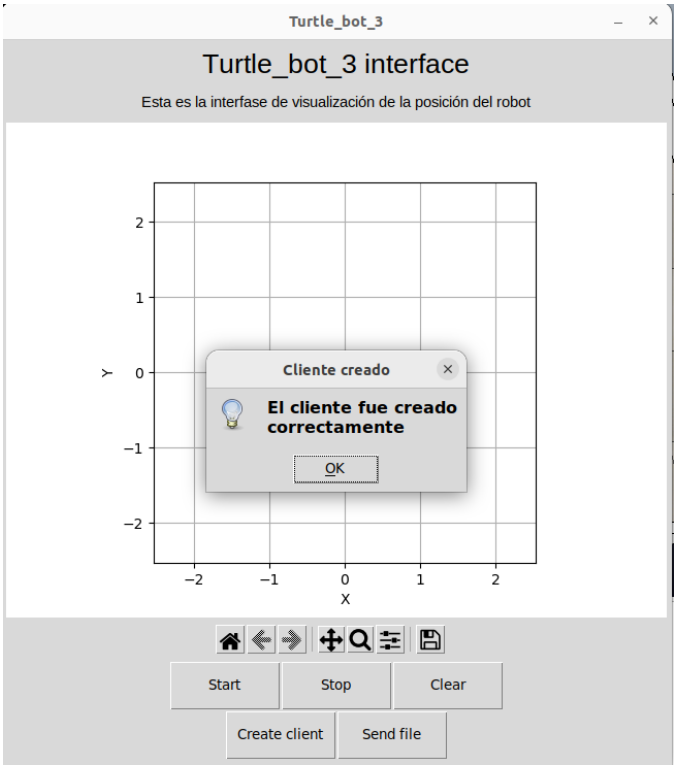

Fig. 3: Inicialización del cliente

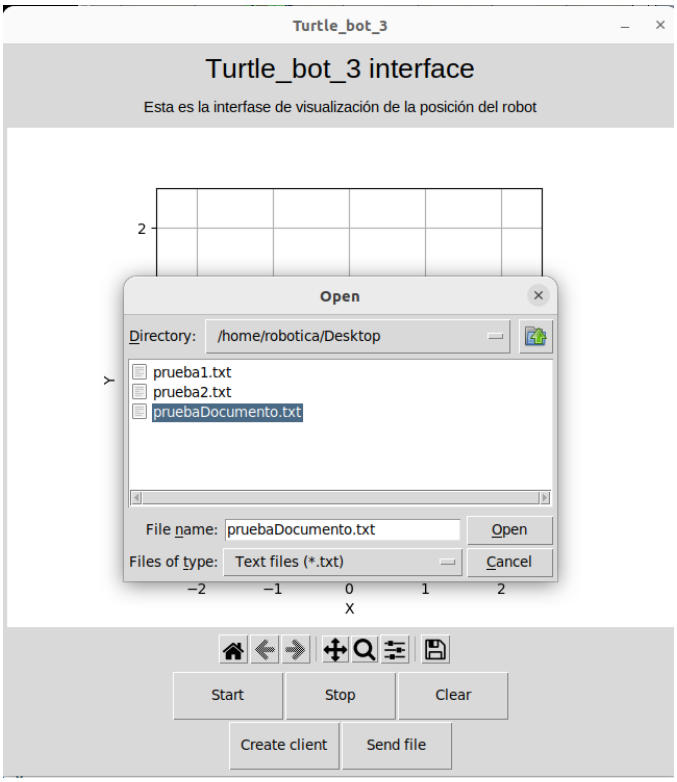

Fig. 4: Selección de recorrido a reproducir

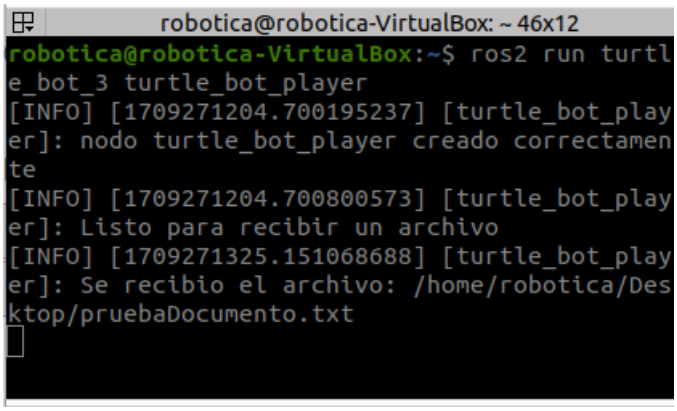

Fig. 5: Ejecución de recorrido guardado

Con el servicio ejecutándose, al crear el cliente e ngresar el archivo que se busca reproducir, se publican las velocidades del recorrido en secuencia, siendo fiel al tiempo de ejecución al registrar inclusive los tiempos donde no hubo cambios de velocidad. El servicio vuelve a estar disponible una vez se finaliza el recorrido ejecutado.

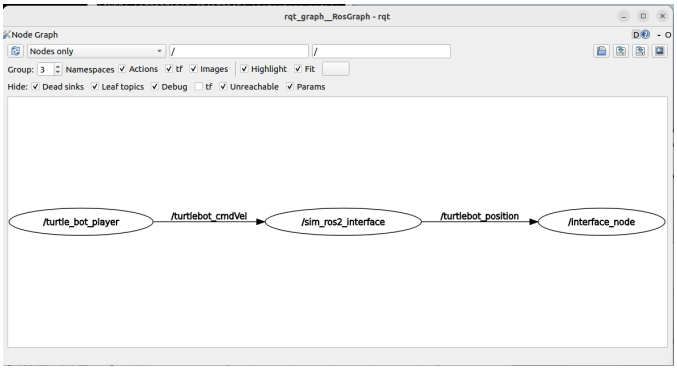

Fig. 6: Grafo Rqt del player

Con el servicio ejecutándose, se encuentra que la relación del nodo del player con el tópico es adecuado al este ser un publisher en el tópico de velocidades que registra las velocidades del recorrido que se busca reproducir.

Evaluando lo anterior, se puede considerar que por medio de la ventana emergente, el servicio manda de forma facilitada al player el nombre del archivo y que en la terminal, al mostrar el nombre del archivo se puede rectificar que se seleccionó el archivo deseado. Así mismo se considera que es eficiente al realizar el recorrido sin problema alguno y con una cantidad baja de pasos.

VII. Conclusiones

Concluyendo respecto a los resultados del taller, se puede considerar que la implementación realizada es exitosa al implementar todas las funcionalidades esperadas sin errores presentes. Así mismo se encuentra un valor agregado al considerar las funcionalidades opcionales como registro individual de las entradas del teclado, y opciones extra de visualización en la interfaz. De forma paralela se puede resaltar el manejo

de errores en los nodos, lo cual permite una ejecución limpia y minimiza los posibles errores generados por el usuario, en general resultando en una implementación funcional y completa.
Entre las dificultades presentadas, se resalta el uso de matplotlib con otros hilos, lo cual se resolvió al descubrir que funciona correctamente solamente al ejecutarse en el hilo principal. Así mismo el entendmiento de la distribución de responsabilidades entre los nodos, lo cual se resolvió al establecer con seguridad los trabajos a realizar y tener estos elementos pre-establecidos por cada nodo en el momento de realizar el programa.

Por último las consideraciones que toca tener con respecto a ejecutar múltiples funciones al mismo tiempo, siendo estos casos respecto el player y el capturar movimiento de la interfaz, y el teleop y el player. Pues podría resultar en que se manden los mensajes al mismo tiempo. Esta dificultad se solucionó plenamente con las instrucciones de uso indicando de forma clara que se debe evitar lo anteriormente mencionado para una correcta ejecución.

Con todo lo anterior en mente, se pudo poner en practica el uso de ROS, nodos, tópicos e hilos, conjunto a librerías externas, para poder, por medio de publishers y listeners, ingresar y desplegar la información deseada por medio de multiples plataformas, sea ya ingresandola con el teclado o un archivo txt, o desplegandola en una aplicación de un tercero, o una interfaz realizada con librerías de python.

References

[1] pynput Package Documentation. [En línea]. Disponible en: https://pynput.readthedocs.io/en/latest/

[2] Matplotlib 3.8.3 documentation. [En línea]. Disponible en: https://matplotlib.org/stable/index.html

VIII. Anexos

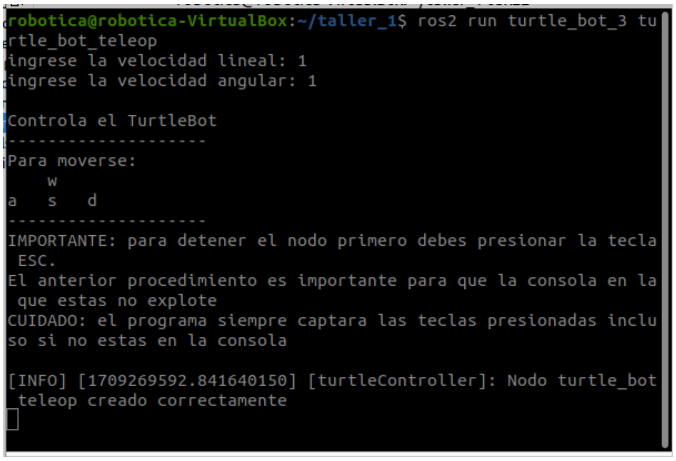

Fig. 7: Inicialización del teleop

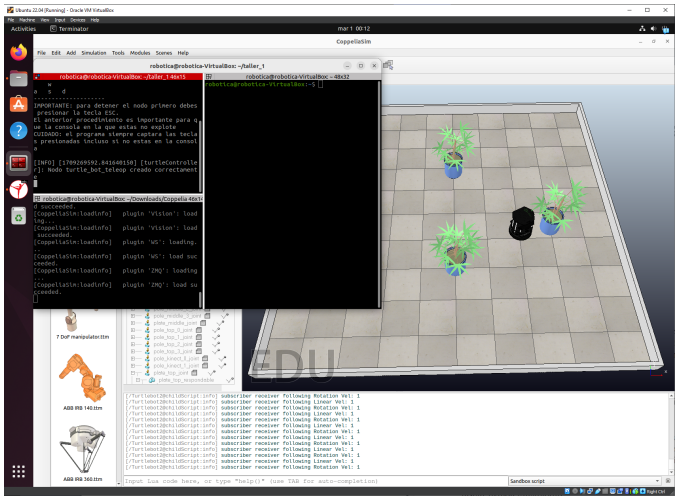

Fig. 8: Ejecución del teleop

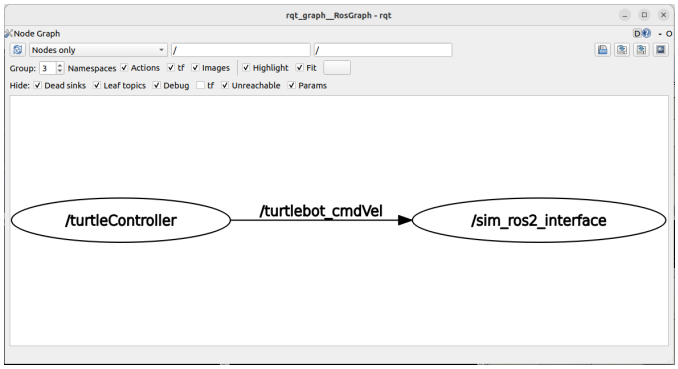

Fig. 9: Grafo RQT del ingreso del teclado

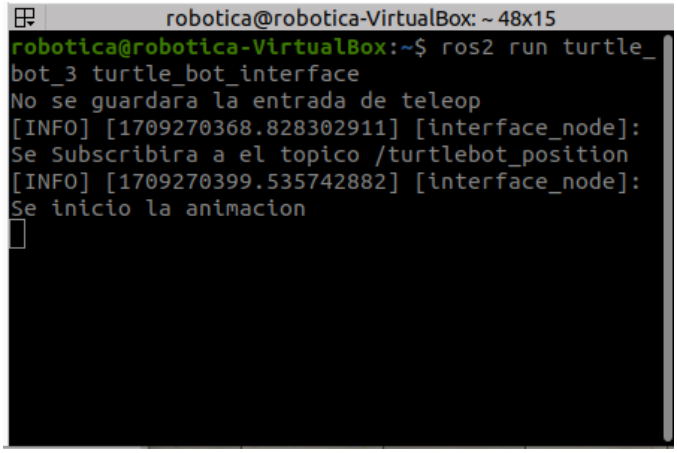

Fig. 10: Inicialización del punto 2

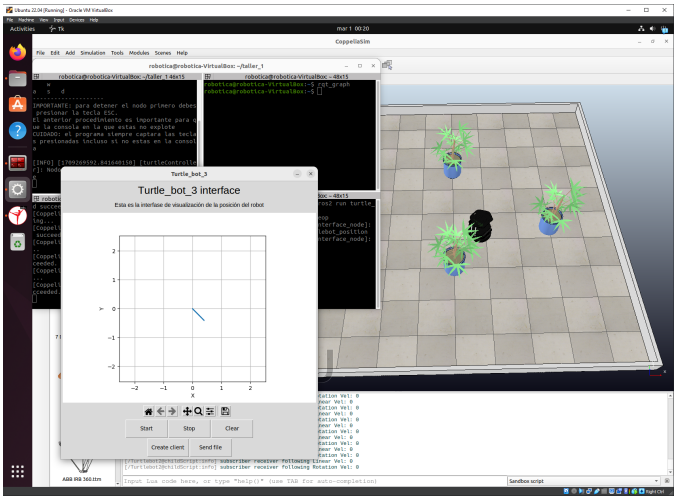

Fig. 11: Interfaz implementada

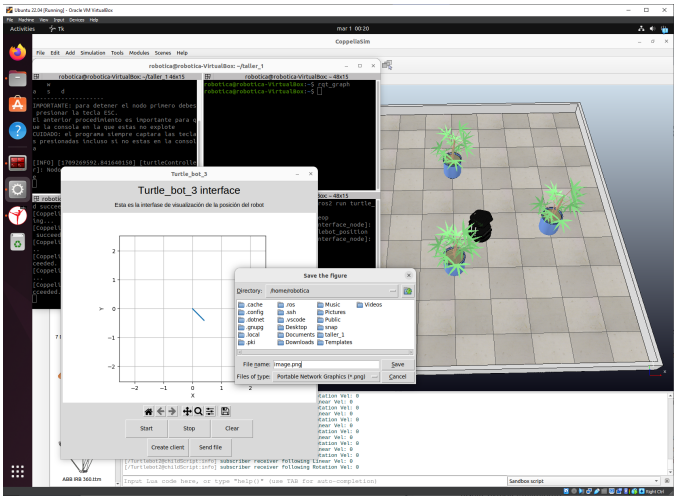

Fig. 12: Ventana emergente de guardado de imagen del recorrido

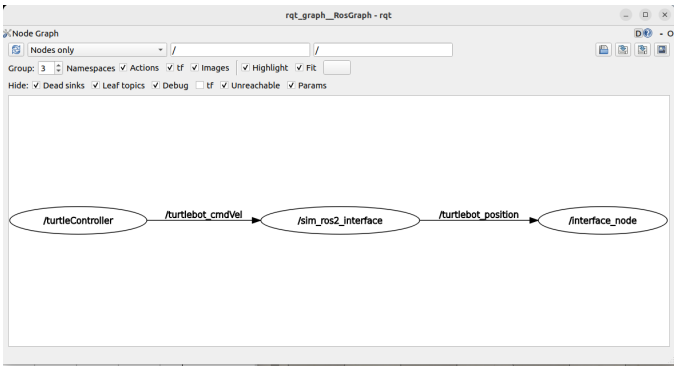

Fig. 13: Grafo Rqt de la inferfaz

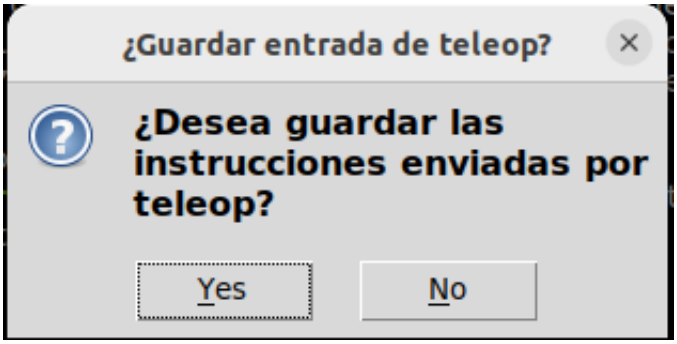

Fig. 14: Ventana Emergente de opción

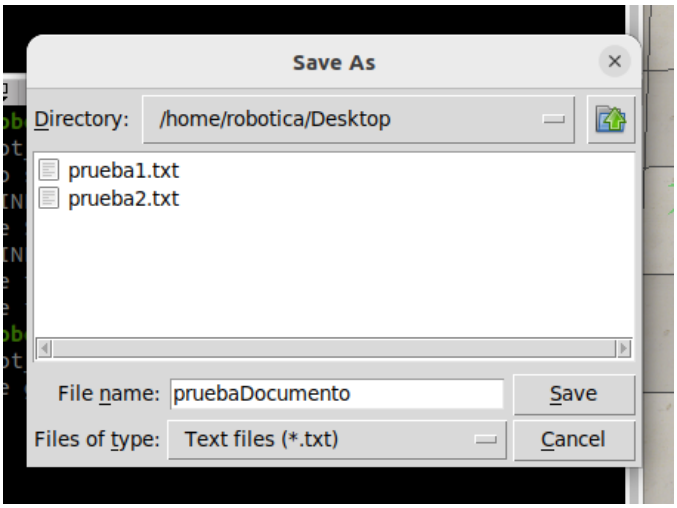

Fig. 15: Ventana Emergente de guardado

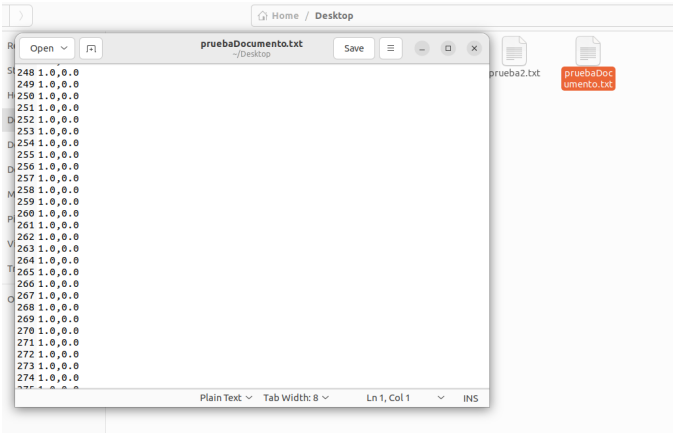

Fig. 16: Archivo txt de recorrido registrado

Repositorio GitHub: https://github.com/msgalvan1207/taller_1_Robotica.git

Videos OneDrive: https://uniandes-my.sharepoint.com/:f:/g/personal/m_realesc_uniandes_edu_co/EvahzlO20p1Fj4VKH31ooRwBvazV-WsozLqizglkoSszWw?e=g2MxDm