Departamento de Ingeniería Eléctrica y Electrónica
Robótica
Facultad de Ingeniería
Universidad de los Andes

Taller II

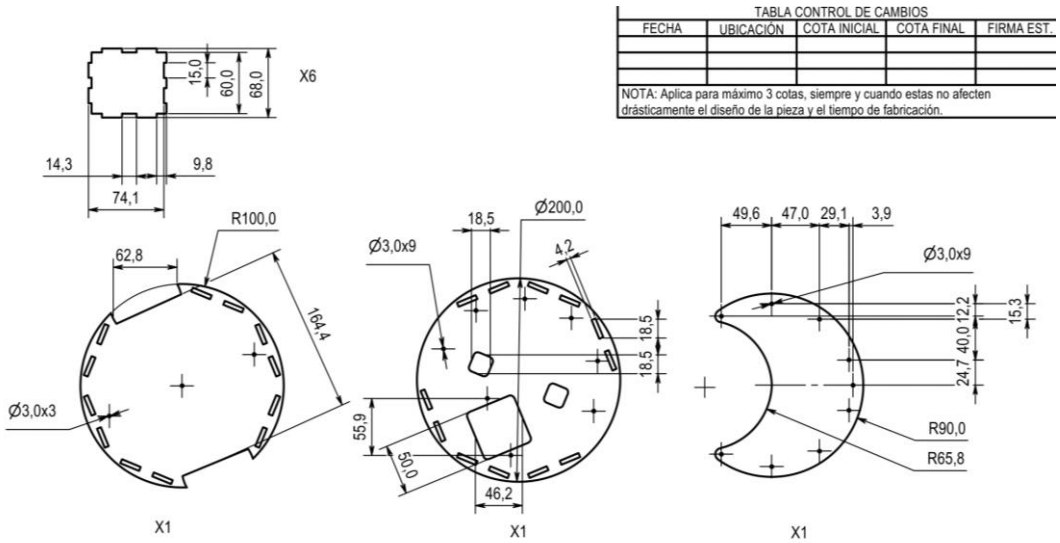
Grupo 11
Alvarez Vanegas Daniel Alejandro (201911320)
González Gutiérrez Cesar Luis (201912673)
Ortiz Almanza David Santiago (201913600)
Pinzón Roncancio Juan Sebastián (201915773)

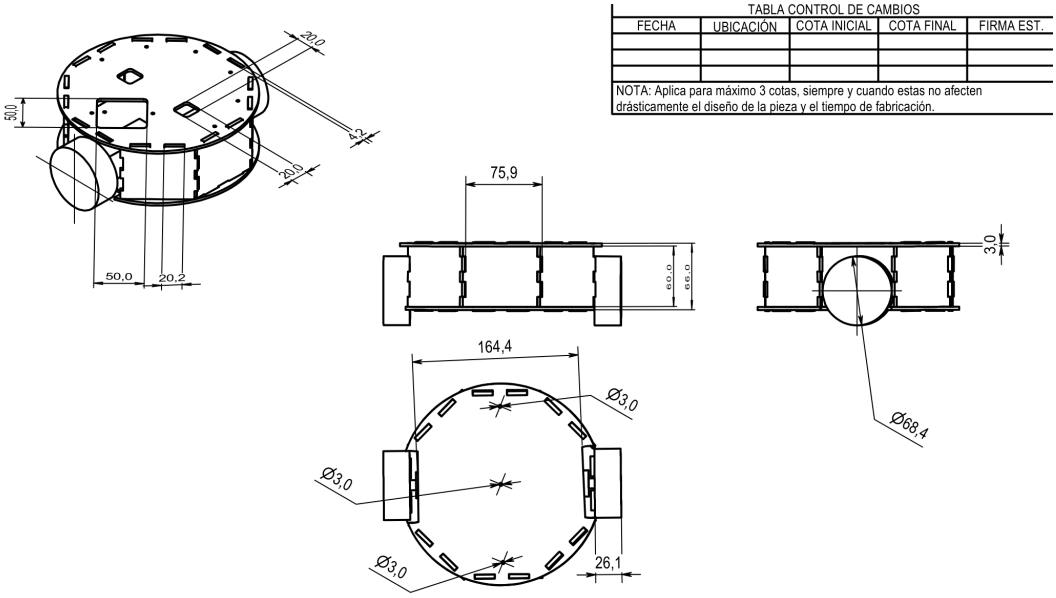
I. Diseño de robot diferencial

a. Lista de materiales

- 1X Raspberry Pi 4 link
- 1X Arduino UNO link
- 1X Puente H L298N link
- 1X Convertidor de voltaje LM2596 link
- 2X Rueda loca link
- 2X Llantas link
- 2X Motor con enconder link
- 2X Batería Lipo Turnigy 7.4V 2200mAh link
- 20X Jumpers macho-macho link
- 1X mini protoboard link
- Lámina MDF Tamaño 1/2 Pliego 3mm link

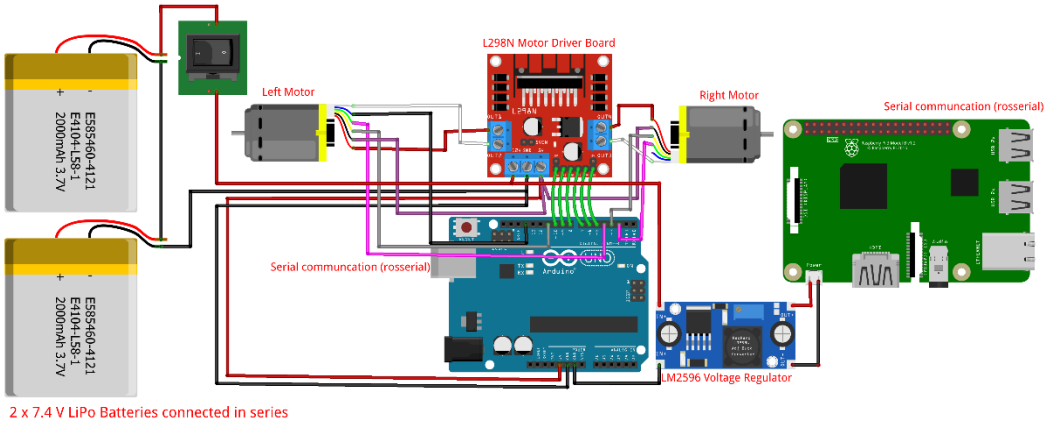
b. Plano mecánico





Para cumplir los objetivos del taller se diseñó un robot diferencial, utilizando dos ruedas a los lados del robot y dos ruedas locas en la parte inferior que sirven como soporte. Como se puede ver en la imagen de los planos, para cumplir con las restricciones de diseño el diámetro del chasis del robot mide 20cm. Además, pensando en la incorporación del módulo de manipulación se dejo el espacio cuadrado grande que se puede observar en la tapa superior del chasis para incorporar el brazo del robot.

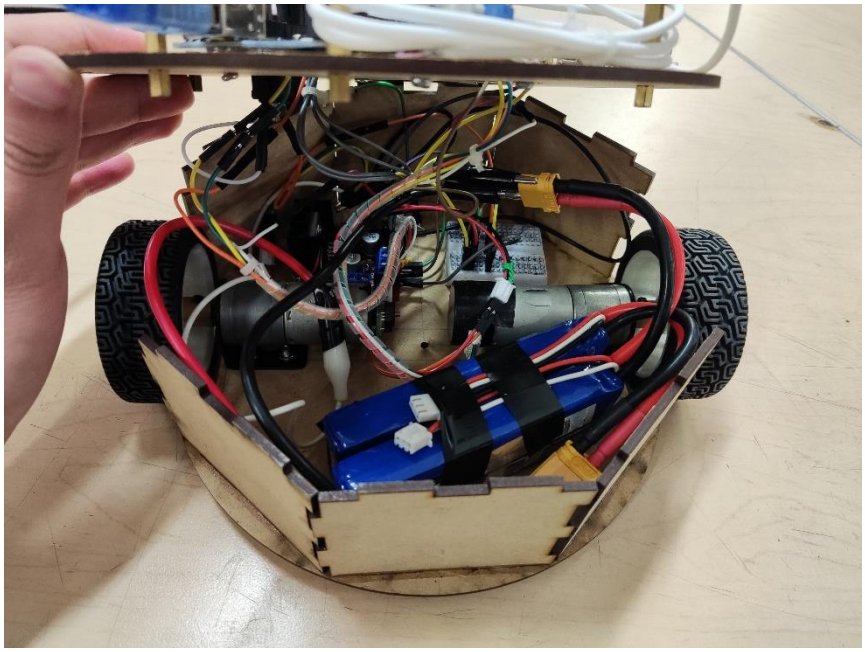
c. Plano electrónico

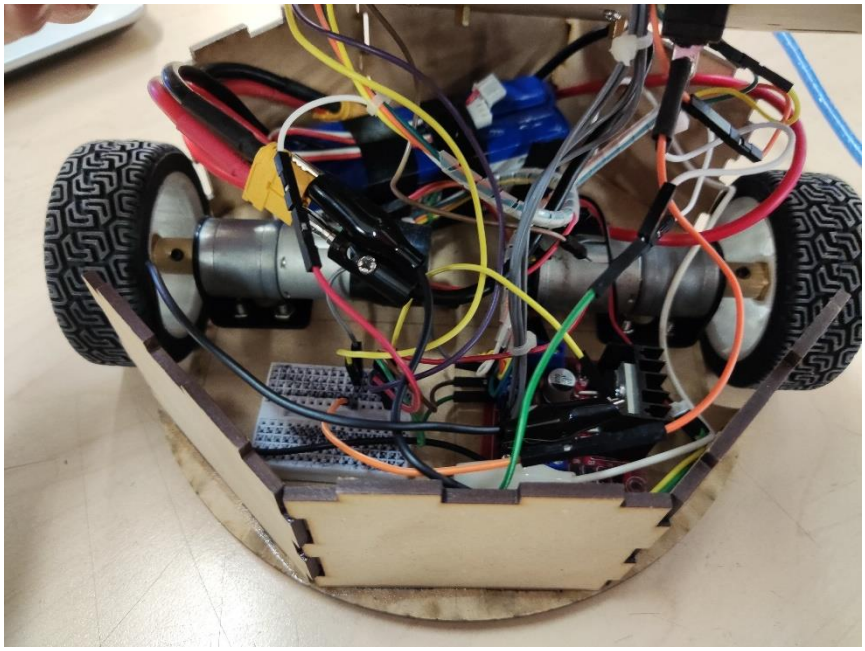


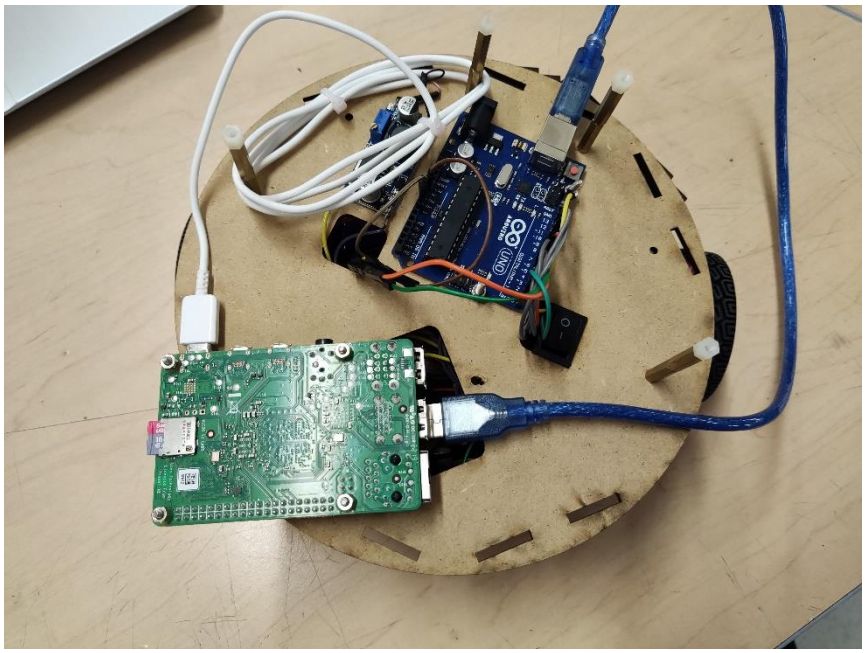
Para el diseño electrónico del robot se utilizó la estructura que se puede observar en la imagen. Se utilizaron dos baterías de 7.4V para proporcionar la potencia suficiente al circuito compuesto por la Raspberry pi 4, el puente

H y los motores con sus respectivos encoders. Se utiliza el controlador de voltaje LM2596 para regular la entrada de las Raspberry a 5V. El Arduino es alimentado por medio de la conexión USB con la Raspberry.

d. Integración parte mecánica y electrónica

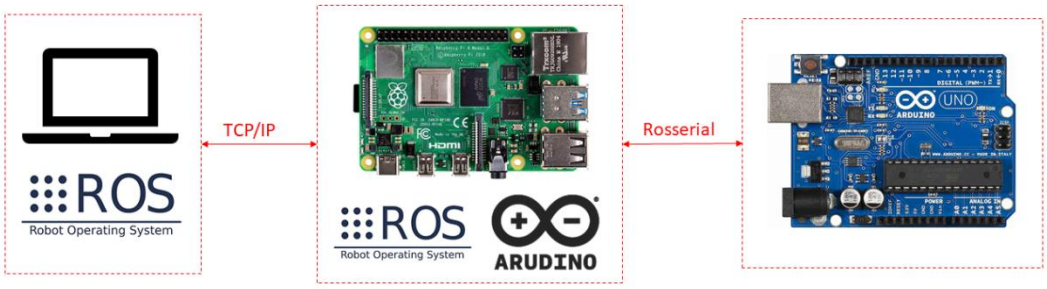






Como se puede observar en las imágenes, se integró la parte electrónica con la mecánica de acuerdo con los planos. En el interior del chasis se pusieron las baterías, el puente H, una protoboard para realizar las conexiones y los motores con sus respectivos encoders. En la parte superior del robot se coloco el Arduino UNO, la Raspberri pi 4, el regulador de voltaje y un interruptor de apagado y encendido. En la parte inferir del robot se colocaron las ruedas locas para dar el soporte necesario a la hora de moverse.

e. Funcionamiento del robot



El robot funciona por medio de las conexiones observadas en la imagen. Por medio de un computador con ROS se conecta la Raspberry Pi 4, la cual también tiene ROS instalado para poder hacer una correcta conexión. Luego la Raspberry se conecta con el Arduino UNO por medio de rossserial, lo que convierte al Arduino en un nodo de ROS para poder enviar las señales al driver y se muevan los motores según se desee desde el computador.

Para conocer la posición del robot se implemento un nodo de odometria el cual publica la posición en base a los encoders. La información de los encoders también es utilizada para realizar el control PID que permite controlar la velocidad de las ruedas dependiendo la trayectoria que se realice.

II. Solución taller

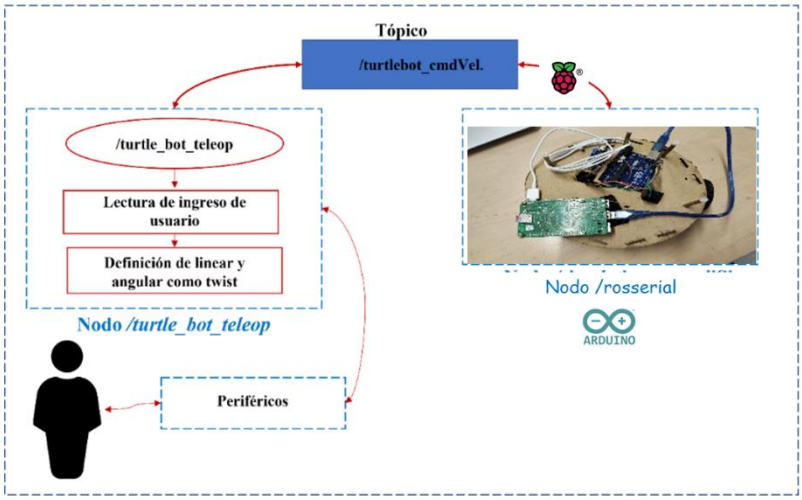
a. Punto 1

Para este punto se solicitaba publicar información de movimiento del robot, esto con ayuda del tópico /cmdVel, al cual le debíamos publicar información dependiendo de la entrada del usuario por teclado y con una velocidad leída por consola. Este requerimiento funcional se cumplió a través del desarrollo de un nodo basado en el uso de la librería pynput para leer las acciones del usuario y dependiendo de estas se publicaba un mensaje de tipo Twist con una velocidad linear y angular dependiendo de cada tecla. Específicamente a través de las teclas: 'Q', 'W', 'S', 'E' se logran 4 movimientos independientes para el sistema, teniendo rotación horaria, antihoraria, así como un movimiento positivo y negativo en el eje y del sistema de referencia inercial del robot.

Aspectos para tener en cuenta:

- El robot se mantiene en estado estático al no presionar ninguna tecla.
- El nodo tiene ingreso de parámetros de velocidad lineal y angular.
- Se poseen 4 movimientos independientes.

El diagrama que describe la solución obtenida se muestra a continuación:



El funcionamiento de este punto se puede observar en el siguiente link: https://youtu.be/DwLabG7IgKA

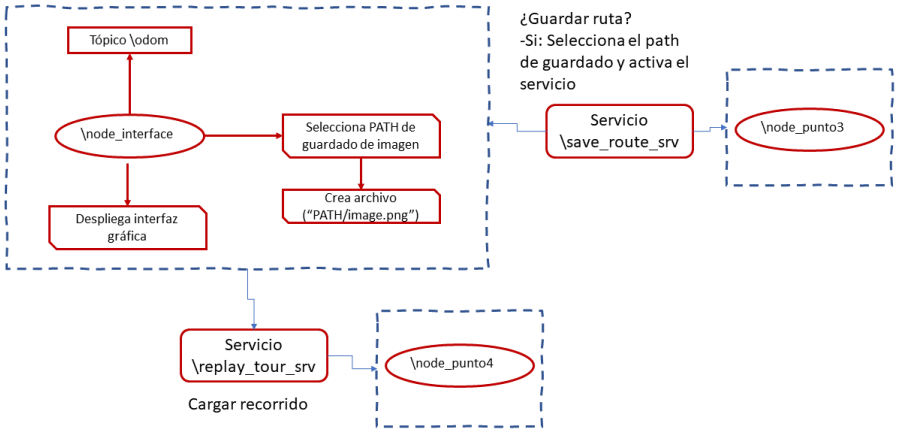
b. Punto 2

En este inciso se requería implementar una interfaz gráfica que permitiese reflejar la posición en tiempo real del Robot y el trayecto -con referencia en el marco global- que este ha recorrido desde el momento que se inicia la simulación. Para ello, se construyó el nodo /turtle_bot_interface el cual debía cumplir con dos funcionalidades de forma simultánea, la primera, la construcción y despliegue de la interfaz gráfica; la segunda, la suscripción al tópico de \turtlebot_position.

Despliegue de la interfaz gráfica:
El despliegue de la interfaz se realizó en el hilo principal del programa, para su construcción se empleó la librería de Tkinter a partir de la cual se crearon todos los elementos gráficos y se implementó la función de trazo de línea, la cual trazaba un segmento de recta a partir de 2 puntos de referencia.

Suscripción al tópico y función de callback
Para lograr hacer seguimiento de la posición del TurtelBot en el marco global, se decidió suscribir el nodo al tópico \turtlebot_position, sin embargo, como se nombró previamente, esta suscripción se implementó bajo un Thread o hilo a parte del principal, de forma de que la función de callback, correspondiente a la suscripción, corriese de forma paralela al despliegue de interfaz.

Posteriormente, se definió la función de callback respectiva, de forma general, lo que realizaba era acceder primero a los valores msg.linear.x y msg.linear.y que correspondían a los valores de posición en tiempo real del robot.



El funcionamiento de este punto se puede observar en el siguiente link: https://youtu.be/lycMS3Wc8zw

c. Punto 3

Para este punto se solicitaba que se le preguntará al usuario si quería guardar el recorrido en robot y en caso afirmativo se guardara en un archivo .txt (en una ruta especificada por el usuario) la secuencia de acciones que realizó el usuario durante el recorrido. Para ofrecer esta funcionalidad se modificó el nodo /turtle_bot_teleop y el nodo /turtle_bot_interface.

Modificaciones al nodo turtle_bot_interface:

- Se implementó que apenas se ejecuta el nodo se mostrará una ventana emergente que pregunta al usuario si se quiere guardar o no el recorrido.

- En caso de que el usuario quiera guardar el recorrido, debe seleccionar la ruta y el nombre deseado. Esta información se almacena en una variable de tipo string.

- En caso de que el usuario haya seleccionado que no desea guardar el recorrido, entonces, la variable de la ruta se establece como una cadena vacía.

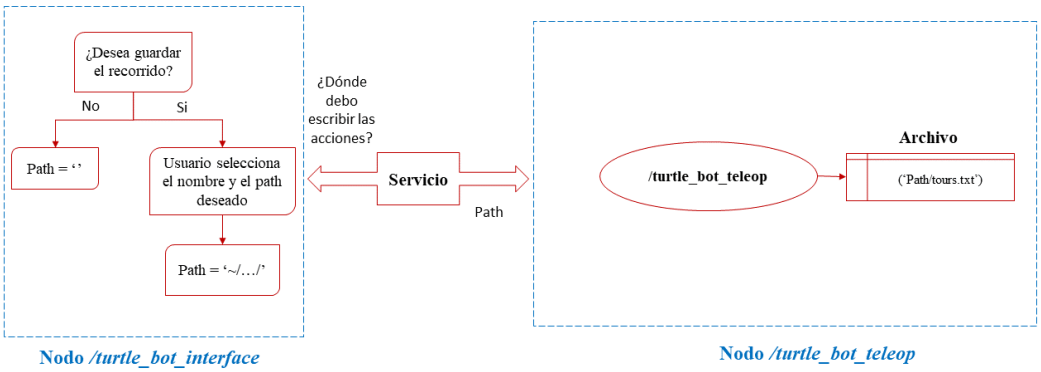
Por último, se implementó un servicio para que el nodo /turtle_bot_teleop pudiera acceder a la información de la variable anteriormente guardara pasara saber si debe guardar o no el recorrido.

Modificaciones al nodo turtle_bot_teleop:

- Se implementó un cliente del servicio ofrecido por el nodo turtle_bot_interface para poder acceder a la información de si se debe guardar o no el recorrido y en dado caso saber en qué ruta.

- Por último, se agregó a los callbacks de presionar y soltar que cuando se llamaran se guardara la tecla oprimida.

A continuación, se muestra un diagrama que resume de manera gráfica la solución implementada.



El funcionamiento de este punto se puede observar en el siguiente link: https://youtu.be/LS_OdON8NsE

d. Punto 4

Para este punto se solicitaba que a partir de un archivo .txt de un recorrido guardado, se reproduzca la secuencia de acciones del robot. Para replicar las acciones realizadas por el usuario el recorrido guardado se realizó el siguiente proceso:

Se observa que tecla presiono el usuario en el archivo ‘tours.txt’. Después de esto se pueden presentar dos casos, si se presionó la tecla “Q” o “E” se da la instrucción al robot de rotar en la dirección correspondiente a la de la tecla presionada la cantidad de veces que se repite en el archivo ‘tours.txt’ con la frecuencia dada como parámetro.

Para ofrecer esta funcionalidad se modificó el nodo /turtle_bot_teleop y se creó el nodo turtle_bot_player.

Modificaciones al nodo turtle_bot_interface:

- Se implemento un botón en la interfaz que permite cargar el recorrido a reproducir, el usuario puede indicar la carpeta en donde se encuentran los archivos .txt asociados al recorrido.

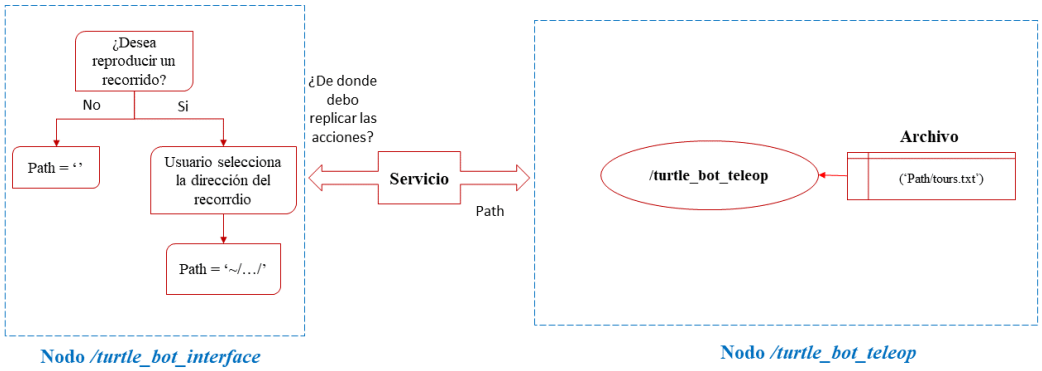
- Se implementó un cliente del servicio ofrecido por el nodo turtle_bot_player para invocarlo desde la interfaz.

Creación del nodo /turtle_bot_player:

- Se implementó un Publisher al tópico / cmdVel para publicar mensajes de tipo Twist con velocidades lineales y angulares según la tecla leída en el archivo tours.txt.

- Se implementó un servicio que ejecuta la funcionalidad descrita anteriormente.

A continuación, se muestra un esquemático que resume la solución implementada:



El funcionamiento de este punto se puede observar en el siguiente link: https://youtu.be/AfQ7cBpw8JI