Taller 2-Robótica

Brayan Felipe Rubiano Enciso - *bf.rubiano@uniandes.edu.co*
Daniel Martinez - *ds.martinezc@uniandes.edu.co*
Danilo Fernandez Sarria - *d.fernandez@uniandes.edu.co*
Universidad de los Andes
Departamento de Ingeniería Eléctrica y Electrónica

Febrero 2022

1. Introducción

En este taller se elaborará la conexión entre ros y el robot que se diseñó con el fin de poder ejercer control sobre él. Para ello se utilizará una tarjeta raspberry pi 4 en la cual se elaborará todos los códigos en lenguaje *python*. El control de los motores se efectuará a partir de una placa arduino UNO la cual funcionará como nodo que recibirá las instrucciones del robot. A continuación se muestran los objetivos:

2. Objetivos

1. Cree un nodo en ROS, llamado /robot_teleop, que permita a un usuario controlar por teclado su robot. Es decir, que las velocidades que son publicadas al robot de forma lineal y angular (en los tópicos respectivos) coincidan de forma natural con la distribución de teclas básicas (Como mínimo el robot debe tener cuatro movimientos diferentes). Aclaración: Si no se presiona ninguna tecla el robot se debe quedar quieto. El usuario debe definir el valor de la velocidad lineal y angular, en el marco de referencial local, con la que se mueve el robot al iniciar el nodo. No se admitirá tener que modificar el código para actualizar la velocidad. La velocidad lineal y angular del robot debe ser enviada a través del tópico /robot_cmdVel.

2. Cree un nodo de ROS, llamado /robot_interface que permita visualizar la posición en tiempo real de su robot, a través de una interfaz gráfica. Es decir, el programa deberá ir mostrando en la pantalla una gráfica donde se representa la posición del robot en el marco global de referencia en tiempo real y muestre el camino recorrido por el mismo desde donde inició. La interfaz debe contar con el espacio para asignarle un nombre a la gráfica y poderlo guardar en el directorio deseado, al finalizar el recorrido.

3. Complemente el nodo /robot_teleop y el nodo /robot_interface para que al iniciar el nodo se le pregunte al usuario en la interfaz si desea guardar el recorrido del robot. En caso que la respuesta sea afirmativa, debe guardar en un archivo .txt la secuencia de acciones que realizó el usuario durante el recorrido del robot. El nombre del archivo debe ser preguntado al usuario y capturado desde la interfaz.

4. Cree un nodo de ROS, llamado /robot_player, que a partir de un archivo .txt de un recorrido guardado, reproduzca la secuencia de acciones del robot. La partida a reproducir debe ser solicitada a partir de un servicio y dicha llamada al servicio ser realizada directamente desde la interfaz creada anteriormente. El nodo /robot_player debe proveer el servicio de recibir el nombre de la partida a reproducir.

3. Implementación

Se creó un paquete de nombre taller_2 en el cual se incluyen las carpetas de scripts y srv que son las que se utilizaron en este taller. Dentro de la carpeta de scripts se encuentran los documentos teleop.py, interface.py, encoders.py y player.py. Estos son los tres nodos de ROS que se utilizaron. Siempre que se vaya a correr alguno de los nodos, es necesario primero ejecutar el comando *roscore* luego dirigirse a la carpeta catkin_ws y ejecutar los comandos *catkin_make y source devel/setup.bash*.

Se ensambló un robot diferencial el cual va a repetir las instrucciones dadas.

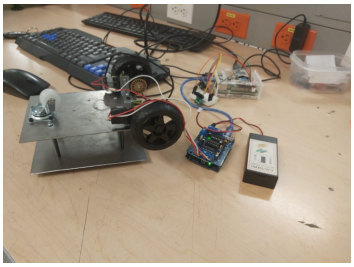

Figura 1: Robot ensamblado

Los tópicos a los cuales se va a conectar el robot son los siguientes
$/\mathbf{cmd}_v el = Encargado de enviar las instrucciones del robot, es de tipo std.msg/String$
$= Es la posición de referencia que se tiene del robot, este manda un mensaje$ **geometry_msgs/Twist**:

msg.linear.y = Es la posición en y en el marco de referencia global.

msg.linear.x = Es la posición en x en el marco de referencia global.

msg.angular.z = Es la posición angular en el marco de referencia global.

Grafo de Ros

Se identificaron dos nodos, uno cuando el nodo de teleop.py o player.py está encendido y otro cuando lo hace el nodo encoder.py e interfaz.py A continuación se muestran los grafos.

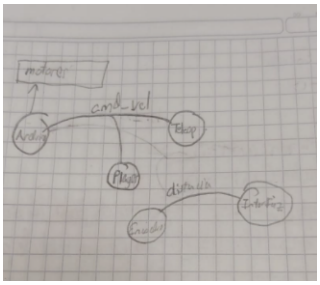

Figura 2: Grafo cuando se encuentra encendido el nodo de teleop. Se puede observar que el nodo de turtlebot envía por el tópico de cmd$_V elalcualestásuscritoelnododeinterfacequeseencargadelservicioydelagráfica.Estopermiteelcontr$

Se puede observar que en ambos casos, el nodo de interfaz se mantiene encendido sucribiéndose al tópico encoders.py. Esto le permite graficar en tiempo real la posición del robot.

3.1. Diseño Mecánico

Se utilizaron los siguiente elementos:

- Raspberry pi 4
- Dos motores y dos ruedas
- Una rueda loca
- Baterías de 3.4V
- Arduino UNO
- Módulo puente H L293d

El chasis del robot es de aluminio.

Punto 1

Para el primer punto se parte del robot construido. Se debe de crear un nodo en python el cual permita que el usuario pueda controlar el robot a partir del arduino. El arduino se encarga de encender los motores dependiendo de la dirección a la que se quiera ir.

Para realizar esta implementación se decidió usar la libreria **pynput** la cual nos permite realizar la importación del teclado por medio de un **key listener**. Con esto en mente, se busca usar 2 funciones que permitan mover el robot al oprimir alguna tecla, y si no se oprime ninguna el robot debe quedarse totalmente quieto.
Estas funciones son:

onpress = Este metodo guarda la tecla presionada como un comando.

En este caso al presionar una tecla se envía un String, este permitirá que al tener la tecla seleccionada el robot realice el movimiento que se le pide.

onrelease = Este método hace que al soltar una tecla, se coloque un comando específico. En este caso, es el nombre del comando en mayúscula ej: 'W'.

En este caso se cancela el vector al enviar el comando 'W' y el robot se queda completamente quieto.

En el metodo onpress se usó las teclas "w", .ª", "sƶ "d", permitiendo mover al robot hacia arriba, izquierda, atrás y derecha respectivamente. Y en el método onrelease se usó las teclas "W", "S", .ᴬƶ "D"las cuales permiten que el robot se quede quieto. En el nodo teleop.py se envía información a través del topico cmd_vel.

El movimiento de las ruedas a partir del arduino es a partir de la librería AFMotor.h la cual activa o desactiva los motores a partir de los comandos BACKWARDS, FORWARD o RELEASE.

- Adelante: w
- Atrás: s
- Derecha: d
- Izquierda: a
- Comandos 'W', 'A', 'S', 'D': [0,0], stop.

En este punto también se utilizó una tasa de 10hz con el comando rospy.rate().

Estos comandos tipo Twist se guardan en una variable llamada "msg" la cual se publica en el tópico cmd_vel. Para esto, se utilizaron Strings que se envían a dicho tópico para que el arduino lea el tópico y tome la decisión de cuáles motores mover. El ancho de la cola se estableció en 10.

Diseño del código

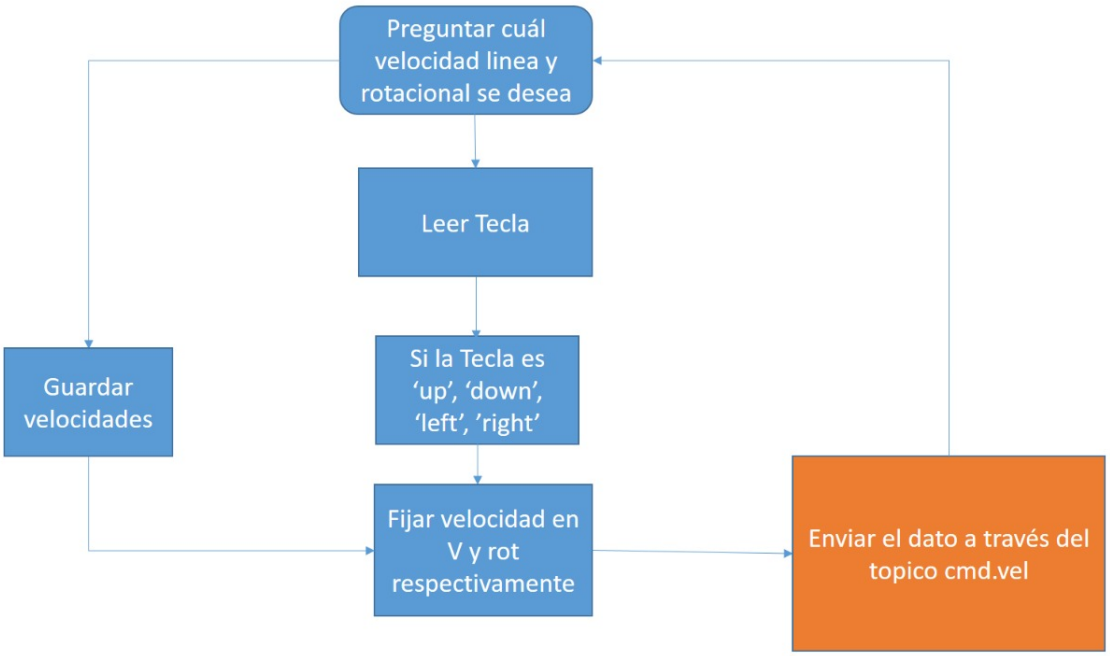

Figura 3: Diseño del codigo punto 1

Punto 2

Para la realización del punto 2 se pide crar un nodo llamado interface el cual permite visualizar la posición del robot mostrada en tiempo real. Para esto se busca graficar las coordenadas en x & y para visualizar la posición y el recorrido del robot. Inicialmente se usa la libreria de **Tkinter** en la cual se crea una pestaña tk(). El nodo interface se suscribe al nodo position para poder recibir datos de la posición del robot. El tópico de la posición del robot publica datos x, y & z. Las primeras 2 corresponden a la posición, mientras que z corresponde a la posición angular del robot así que no se tendrá en cuenta en esta actividad. Estos datos de x & y se utilizan generar un scatter dentro de la pestaña siguiendo así la poición del robot. Todo esto es posible gracias a la implementación de la librería **matplotlib**.

Para este punto se creo un nodo encoder el cual a partir de la velocidad linea de las ruedas del robot utiliza un modelo para predecir la posición del robot. Este nodo lee la velocidad de cada rueda a partir del uso de encoders que van conectados a la raspberry los cuales son leidos mediante la libreria GPIO.

Diseño del código

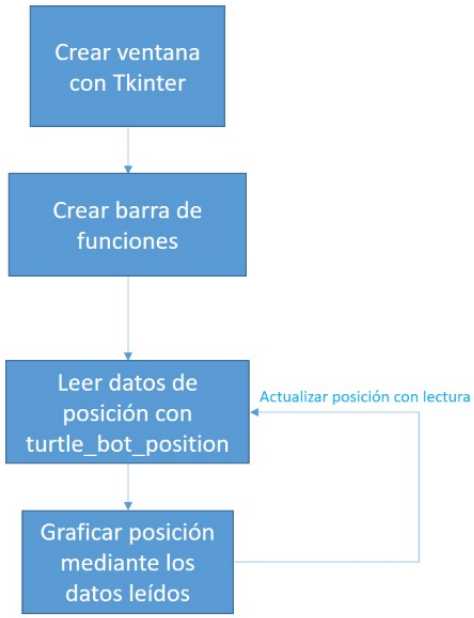

Figura 4: Diseño del codigo punto 2

Punto 3

En este punto se busca completar el nodo de interfaz, teleop y player para que se le pregunte al usuario si desea guardar el recorrido del robot y poder asignarle un nombre a ese archivo.txt. Así que inicialmente se ha de modificar el nodo teleop para que genere una pregunta al usuario cuya respuesta tiene que ser un booleano ("¿Desea guardar el recorrido?"). Para esto, se importó la libreria *OS*. Se tienen if dentro de los ifs de los comandos los cuales permiten que si al responder con True a la pregunta anterior se guarde el comando generado por la tecla en un txt. Su nombre es ("pasos.txt") todo esto dentro de un directorio especifico que se busca con la librería. Se modificó que al presionar la barra espaciadora se imprimiera la palabra "pausa". Esto es con el fin de que en el momento que se desee hacer alguna lectura, exista una marca para finalizar el recorrido del robot.

Diseño del código

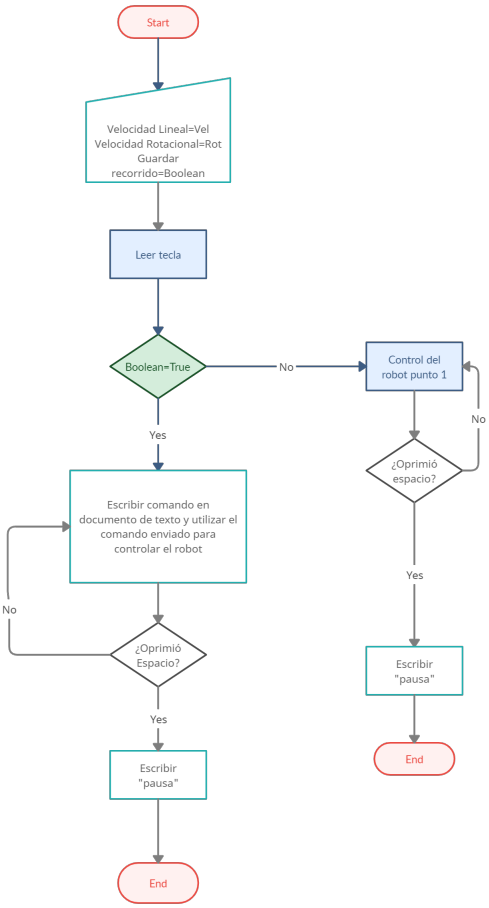

Figura 5: Diseño del codigo 3

Punto 4

Para este ultimo punto se busca que a partir de un archivo de texto se reproduzca la secuencia de acciones del robot. Así que, inicialmente se agregó un botón reproducir el cual abre una nueva ventana donde se escribe el nombre del documento para luego se reproducido. Esta nueva ventana contiene un cuadro de texto y en esta el usuario escribe el nombre del archivo .txt. Se utilizó la librería OS la que permite buscar el archivo en la ruta que es necesitada, en este caso, los archivos de pasos del robot se encuentran en la carpeta de scripts.

Después de modificar la interfaz, se creó un servicio con una entrada y una salida de tipo string. La entrada corresponde al nombre del archivo a seleccionar y la salida corresponde a la ruta completa del archivo y su ubicación en el equipo. Para leer el archivo se implementó un for con el comando readline() que lee linea a linea el archivo .txt.

Cada una de estas lineas se introduce en un comando if el cual ejecuta los comandos de tipo "Up/n" "Down/n", "Left/n", Right/n"(esto es debido a que en el archivo de publicación se le agregó el "/n" para que los comandos quedaran separados linea por linea. Estos comandos, dependiendo de cual sea, publican datos tipo String con el fin de enviarlos al arduino y que este analice los pasos a seguir para enviarle las instrucciones a los motores.

Diseño del código

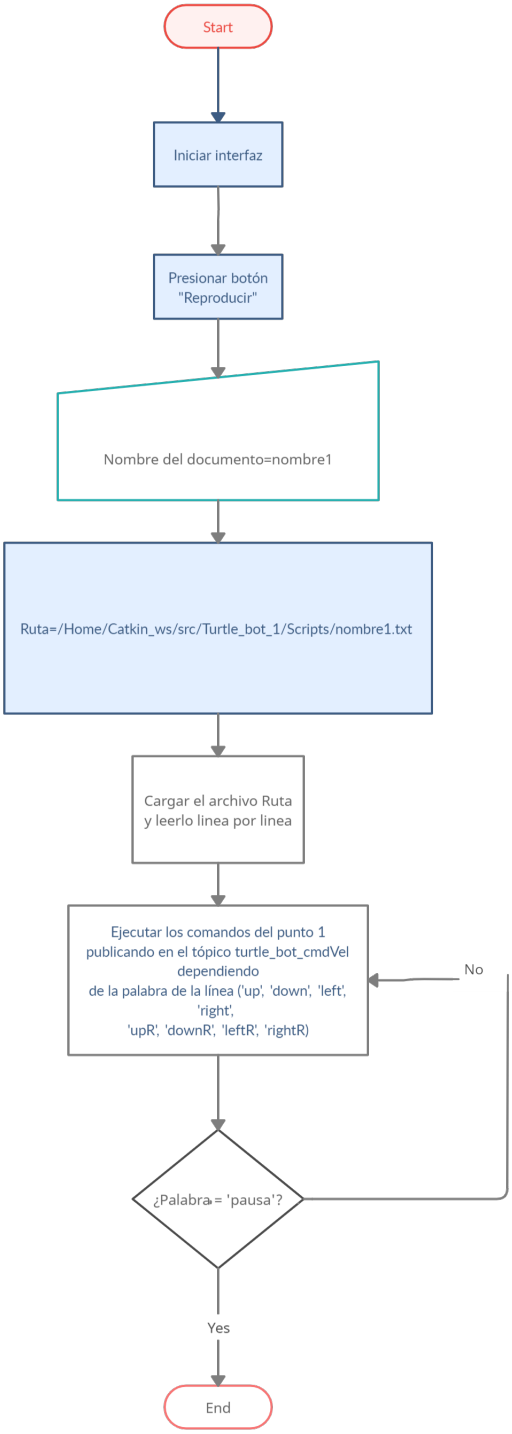

Figura 6: Diseño del codigo 4