RETO SEMANAL 1. Manchester Robotics

Abraham Ortiz Castro
Dpto. de Ingenierías Tec de Monterrey
Tecnológico de Monterrey
Puebla, México
A01736196@tec.mx

Alan Iván Flores Juárez
Dpto. de Ingenierías Tec de Monterrey
Tecnológico de Monterrey
Puebla, México
a01736001@tec.mx

Jesús Alejandro Gómez Bautista
Dpto. de Ingenierías Tec de Monterrey
Tecnológico de Monterrey
Puebla, México
A01736171@tec.mx

Ulises Hernández Hernández
Dpto. de Ingenierías Tec de Monterrey
Tecnológico de Monterrey
Puebla, México
A01735823@tec.mx

I. Resumen

En este documento se encuentran documentados los procesos llevados a cabo para esta, el primer reto del bloque Implementación de Robótica Inteligente, el cual retoma nuevamente los conceptos de nodos y publicadores vistos en el bloque introductorio a robótica y a ROS. En esta ocasión nos centraremos en el elemento físico del PuzzleBot, el cual nos abre la posibilidad a nuevas complejidades de algoritmos y programas, para conocer sus capacidades es necesario comenzar a entenderlo y exprimir su mayor potencial. Demás está decir que seguiremos trabajando mediante a nodos para cada challenge semanal.

II. Objetivos.

- Desarrollar habilidades en el manejo del framework ROS.
- Incrementar la interacción y la implementación de funciones disponibles en ROS.
- Familiarizarse con el robot PuzzleBot y su interconexión.
- Realizar la primera actividad programada en el robot a través de ROS y micro-ROS.

III. Introducción.

A. Puzzlebot

1) ¿Qué es?: El PuzzleBot, desarrollado por Manchester Robotics es un robot de laboratorio especializado en la educación y en el prototipado, siendo esa la misión del mismo, la del desarrollo de aprendizaje en la robótica a través de este bot. Existen algunas variantes del PuzzelBot que no son más que la misma estructura robótica, con implementaciones de algunas tarjetas o controladores adicionales que le permiten desarrollar nuevas complejidades de algoritmos, así como realizar tareas que los otros modelos no pueden realizar. Entre estas versiones podemos destacar las siguientes:

- PuzzleBot Hacker Edition.
- PuzzleBot Jetson Edition.

El PuzzleBot está compuesto por componentes fuera de serie primordialmente para lograr un precio bajo frente a un gran volumen de producción, con la intención de facilitar el acceso a este producto de desarrollo en robótica a una mayor cantidad de usuarios que adquieran el PuzzleBot completo o en su defecto las partes del mismo de forma independiente. [1]

2) Elementos:

- **Hacker Board:** Actúa como un nodo computacional robusto para los componentes del kit contando con una interfaz de comunicación con unidades externas.
- **NVIDIA Jetson Nano:** Es considerada como la "mente" del PuzzleBot, pues permite la implementación de algoritmos de aprendizaje profundo y visión por computadora.
- **Cámara Raspberry Pi:** Es empleada para la captura de imagen y video que junto al poder de la Jetson pueden procesar algoritmos de visión computacional.
- **Comunicación serial:** Tanto la Hacker Board como la Jetson Nano están comunicadas mediante comunicación serial.

3) Interconexión: El PuzzleBot Jetson Edition es un robot diseñado para que los usuarios implementen algoritmos de bajo nivel en tiempo real en la Hacker Board y algoritmos de alto nivel de como Inteligencia Artificial y visión por computadora en la NVIDIA Jetson Nano. La interconexión entre la Hacker Board y NVIDIA Jetson Nano se realiza a través de comunicación serial.

En el caso del PuzzleBot, la interconexión permite que los diferentes componentes propios del robot se comuniquen y trabajen juntos para realizar tareas complejas. [1]

B. Comunicación SSH

La omunicación SSH en ROS hace referencia a la configuración de múltiples dispositivos para la comunicación entre los mismos a través de ROS. Particularmente, para fines de la práctica nos interesa la comunicación con el robot PuzzleBot a través de SSH, a continuación se explica acerca de los procesos necesarios previos y en el momento de establecer comunicación entre la Jetson Nano y nuestra computadora. [2]

1) Configuraciones previas a la Jetson:

- *Instalación del Sistema Operativo:* Es necesario descargar el image del PuzzleBot e instalarlo en una tarjeta SD desde un computador externo, una vez este listo es necesario incertarla en el compartimento de la Jetson para su funcionamiento.
- *Configuración del Hotspot:* Es necesario emplear una tarjeta de red inalámbrica, ya que la Jetson no cuenta con una. Dentro de la interfaz de la Jetson desde el administrador de Red configuraremos el hotspot, por defecto contará con un nombre y contraseña predeterminado, pero este puede cambiarse. Guarda los cambios para que se apliquen a la Jetson. [2]

2) Configuraciones para la comunicación:

- *Conectar el dispositivo a la red del PuzzleBot:* Una vez el hotspot de la Jetson Nano este configurado, desde nuestra computadora podremos visualizarla como una red disponible, a la cual deberemos conectarnos. Si no configuramos la red de forma personalizada, necesitaremos ingresar la contraseña, la cual es por defecto `Puzzlebot72`.
- *Conectarse vía SSH:* Una vez conectada a la red de la Jetson, desde una ventana de terminal ingresamos el comando `ssh puzzlebot@10.42.0.1`, los últimos dígitos de la línea indican la IP por defecto, si deseas configurarla, recomendamos cambiar el último dígito (altamente recomendable al trabajar en el mismo espacio con otros equipos), después de ello, el comando solicitará una contraseña, la cual es la misma que la de la red, en este caso `Puzzlebot72`. [3]

C. Teleop Twist Keyboard

`teleop_twist_keyboard` se trata de un paquete propio de ROS el cual permite un modo de teleoperación de teclado de manera genérica. Este nodo es el encargado de la traducción de los pulsos sobre las teclas en mensajes. Este paquete basa su funcionamiento en el tipo de mensaje Twist de ROS, los cuales no son más que la forma más común de representar velocidades en un espacio tridimensional. Cada uno de estos contiene componentes principales, y primordialmente contamos con dos de ellos:

- *Linear:* un vector tridimensional que representa las velocidades lineales existentes en las direcciones x, y, z.
- *Angular:* un vector tridimensional que representa las velocidades angulares existentes alrededor de los ejes x, y, z.

[4]

1) Instalación y ejecución: Para instalar el paquete es necesario aplicar el siguiente comando sobre la terminal: `sudo apt- install ros install ros-noetic -teleop-twist-keyboard` Así mismo, para hacer uso del nodo, debemos llamarle mediante el comando en terminal: `rosrun teleop_twist_keyboard teleop_twist_keyboard.py`

2) Controles básicos del robot: Una vez que el nodo se ejecute, podrás usar las teclas del teclado para el control del robot, a continuación se mencionan las entradas (teclas) básicas para el control del robot:

- **i**: activa los motores y con ello las ruedas del robot de forma en que estas giren hacia adelante.
- **,**: activa los motores y con ello las ruedas del robot de forma en que estas giren hacia atrás.
- **k:** detiene a los motores y con ello el movimiento del robot.
- **j:** activa ambos motores, uno lo mantiene el giro "hacia adelante" y el otro lo invierte, de forma que el robot se posiciona en dirección a la derecha.
- **l:** activa ambos motores, uno lo mantiene el giro "hacia adelante" y el otro lo invierte, de forma que el robot se posiciona en dirección a la izquierda.
- **u:** activa ambos motores, uno con una velocidad de giro mayor que otro, de forma que girará hacia la derecha - adelante de manera más amplia en comparación con j.
- **o:** activa ambos motores, uno con una velocidad de giro mayor que otro, de forma que girará hacia la izquierda - adelante de manera más amplia en comparación con l.
- **m:** activa ambos motores, uno con una velocidad de giro mayor que otro, de forma que girará hacia la derecha - atrás de manera más amplia en comparación con j.
- **.:** activa ambos motores, uno con una velocidad de giro mayor que otro, de forma que girará hacia la izquierda - atrás de manera más amplia en comparación con l.
- **q/z:** aumenta (q) y disminuye (z) las velocidades *MÁXIMAS* en un 10
- **w/x:** aumenta (w) y disminuye (x) la velocidad *LINEAL* en un 10%.
- **e/c:** aumenta (e) y disminuye (c) la velocidad *ANGULAR* en un 10%.
- **Cualquier otra tecla:** detenemos el robot.
- **CTRL C:** Salir de los controles

3) Modo Holonómico: Si el robot es de tipo holonómico, es decir, puede desplazarse de manera lateral, se mantiene presionando la tecla *shift* y empleando alguno de los controles básicos del robot. [4]

IV. Solución del problema

Para la solución de este reto, se realizó la creación de un tópico subscriber, que realizara la lectura de las dos velocidades angulares de ambos motores de la Jetson Nano con el objetivo de poder determinar las posiciones tanto en el eje 'x', 'y', así como el ángulo Theta. Para las cuales se utilzaron las ecuaciones cinemática de accionamiento diferencial de un robot diferencial. Las cuales se muestran a continuación:

$$\dot{x} = r(\frac{\omega_R + \omega_L}{2}) * cos(\theta) \tag{1}$$

$$\dot{y} = r(\frac{\omega_R + \omega_L}{2}) * sin(\theta) \tag{2}$$

$$\dot{\theta} = r(\frac{\omega_R - \omega_L}{l}) \tag{3}$$

Con estas ecuaciones y las velocidades angulares de los motores obtenidas de los tópicos `VelocityEncL` y `VelocityEncR` podemos calcular en primera instancia el valor de θ realizando la integración del valor calculado de θ punto, realizando una serie de sumas y tomando una delta de tiempo de una décima de segundo al trabajar con datos discretos, como se muestra en las siguiente sección de código:

```
#Se calcula thetha punto y theta
self.velocidadTheha = self.r*((self.velD-
self.velI)/self.l)
self.theha += self.velocidadTheha*self.periodo_lectura
```

Una vez conociendo el valor de theta, podemos calcular la posición en los ejes 'x' y 'y', como se puede mostrar en la siguiente sección de código, realizando de igual manera las integraciones correspondientes.

```
#Se calcula la velocidad y posición de 'x' y 'y'
    self.velLineal = self.r*((self.velD+self.velI)/2)
    self.posX += self.velLineal*math.cos(self.theha)
    *self.periodo_lectura
    self.posY += self.velLineal*math.sin(self.theha)
    *self.periodo_lectura
```

Una vez calculados estos valores, estos son guardados en un mensaje customizado `msgDato` que es publicado en el tópico `ri_odometria` para su posterior lectura. Cabe mencionar que todo el códgio anterior fue ejecutado dentro del `timer_callback()`.

Otras de las modificaciones realizadas dentro del paquete de ROS2 son la creación de un `launch.py` que inicializa nuestro tópico, asi como ejecutar el `rqt_plot` y `rqt_graph`, así como la cración de un mensaje customizado que contenga las tres varaibles de 'x', 'y' y theta.

V. Resultados

A. Interpretación de resultados

Al finalizar con el código, conexiones y calculos para obtener la posición del puzzlebot, obtenemos como resultado el siguiente video:

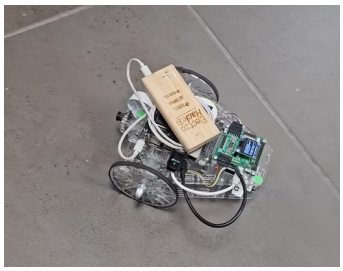

Fig. 1. Video funcionamiento

En el video se puede observar que esta dividido en tres secciones, en la primera es la teleoperación desde el teclado, esto se logro con el paquete de ros2 `teleop_twist_keyboard`, en este con las teclas indicadas por ros2 pudimos mover el puzzlebot hacia enfrente y atras, además de poder hacer giros, en la segunda sección se indican los valores de la posición en 'x','y' y 'theta' y como van aumentando o disminuyendo depende de la dirección del robot, se puede apreciar como en todo momento tenemos la posición del robot que se calculó por medio de la integral de la velocidad, esto se obtuvo con un código al suscribirnos a dos nodos que ya estaban previamente, los cuales daban la velocidad de cada llanta y con esto y ecuaciones matemáticas **??** obtuvimos la posición, por último en la tercera sección se muestra el puzzlebot en el piso y como se desplaza de acuerdo a las teclas presionadas de la computadora.

VI. Conclusiones

Esta actividad se ha podido concluir de manera exitosa debido a que se ha podido mover el robot de manera recta y se han desplegado "bien" los movimientos en el plano del robot. Pero aunque se tiene un comportamiento bueno, no es el mejor que se puede obtener, primero nos pudimos percatar que la Nvidia Jetson Nano estaba ensamblada de manera incorrecta en nuestro robot, generando fricción con los cables afectando así el movimiento en línea recta del robot, para solucionar esto se propone desarmar el robot para después poderlo ensamblar de la manera correcta y evitar el problema. El otro error que se pudo observar fue la precisión de los encoders, debido a que la velocidad que marca del motor derecho e izquierdo aunque son muy similares, no son las mismas, dando así errores muy pequeños al momento de calcular las velocidades lineales, pero con el paso del tiempo esos errores pequeños se van acomulando hasta representar un movimiento errado mayor en algunos ejes; esto se puede solucionar teniendo un pequeño rango de diferencias en los encoders y procesarlos iguales para tener una mejor representación del movimiento en el tiempo. En conclusión, aunque se tuvo un buen resultado, aún se pueden mejorar algunos detalles, los cuales al momento de avanzar con el reto dará un mejor resultado, debido a que se tendrá una mayor precisión en algo tan básico como la odometría del robot.

References

[1] PuzzleBot – Manchester Robotics. (s.f.). link.
[2] La guía para principiantes para configurar un Jetson Nano en JP4.4. (2020, Diciembre 5). ICHI.PRO. link.
[3] Kangalow. (2023, Enero 12). Wi-Fi hotspot setup – NVIDIA Jetson Developer Kits. JetsonHacks. link.
[4] teleop twist keyboard - ROS Wiki. (s.f.). link.
[5] ManchesterRoboticsLtd. (s. f.). GitHub - ManchesterRoboticsLtd/TE3002B-2024: Intelligent Robotics Implementation. GitHub. link.