



Tecnológico de Monterrey

Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

Fundamentación de Robótica (Gpo 101)

Reporte: Reto semanal 1 Manchester Robotics

Alumno

José Diego Tomé Guardado A01733345

Pamela Hernández Montero A01736368

Victor Manuel Vázquez Morales A01736352

Fernando Estrada Silva A01736094

Fecha de entrega

Jueves 11 de Abril de 2024

RESUMEN

Este trabajo se centra en el manejo del framework ROS y micro-ROS, así como en el análisis y la interacción con el robot Puzzlebot para comprender la interconexión de sus componentes y su funcionamiento físico. Dentro de este, se describen varios aspectos clave, incluyendo el incremento del grado de interacción y la implementación de funciones disponibles.

Adicionalmente, se implementó la odometría para estimar la posición y orientación del robot basándose en la información de sus sensores de movimiento. Además, se detalla el funcionamiento del robot teleoperado, que implica configurar el Puzzlebot Jetson, conectarlo a través de una red WiFi, ejecutar el agente micro-ROS y utilizar un dispositivo externo para controlarlo remotamente mediante comandos de teleoperación.

OBJETIVOS

El objetivo general de este proyecto es implementar la teleoperación en el robot Puzzlebot utilizando los frameworks ROS y micro-ROS, con el fin de facilitar una mayor interactividad y funcionalidad en el control remoto del robot. Entre nuestros objetivos particulares se encuentran:

1. Explorar y familiarizarse con el framework ROS y sus herramientas para el desarrollo de aplicaciones robóticas.
2. Investigar las funcionalidades disponibles en ROS y micro-ROS con el fin de incrementar el grado de interacción y mejorar el rendimiento de los sistemas robóticos.
3. Analizar la estructura y el funcionamiento del robot Puzzlebot, comprendiendo la interconexión de sus componentes y sus capacidades físicas.
4. Experimentar con la operación remota del Puzzlebot utilizando el ROS teleop twist keyboard package, configurando adecuadamente la conexión a través de la red WiFi y ejecutando el agente micro-ROS para establecer la comunicación efectiva con el robot.
5. Implementar la odometría para estimar la posición y orientación del robot basándose en la información de sus sensores de movimiento.

INTRODUCCIÓN

En el contexto del desarrollo y operación de sistemas robóticos, la implementación efectiva de frameworks como ROS (Robot Operating System) y micro-ROS ha adquirido una relevancia creciente debido a su capacidad para facilitar el diseño, control y teleoperación de robots en diversos entornos. Dentro de este marco, el presente trabajo se centra en la implementación de la teleoperación en el robot Puzzlebot utilizando dichos frameworks.

Como se menciona en la página de Manchester Robotics el PuzzleBot^[1] de Manchester Robotics tiene la misión de globalizar el acceso a herramientas de aprendizaje y desarrollo para robótica, debido a que se presenta como "un laboratorio en un robot" diseñado para la educación y la creación de prototipos en el ámbito de la robótica.

El 'cerebro' de PuzzleBot es representado por la Hacker Board_[2]. Esta placa actúa como un nodo computacional robusto para los componentes del kit de inicio, como motores y sensores, y como interfaz de comunicación para computadoras externas, como portátiles, teléfonos, NVIDIA Jetson Nano y Raspberry Pi.

Por otro lado el Puzzlebot edición Jetson_[3] es un robot móvil modular diseñado para aplicaciones de inteligencia artificial y visión por computadora debido a que incorpora una serie de componentes que colaboran para proporcionar funcionalidades avanzadas y flexibilidad en el desarrollo y control del robot.

En su núcleo está equipado con una placa NVIDIA Jetson Nano, una plataforma de computación potente y especializada diseñada específicamente para aplicaciones de IA y visión por computadora. Esta placa actúa como el cerebro del robot, encargándose del procesamiento de datos y la ejecución de algoritmos de control y visión.

Entre los componentes físicos del Puzzlebot representados en la *figura 1* se incluyen, motores servo para el movimiento, así como sensores GPIO que ofrecen información sobre el entorno. También integra una cámara Raspberry Pi para la captura de imágenes y la realización de tareas de visión por computadora. Por otro lado, la comunicación del Puzzlebot se realiza mediante un módulo Wi-Fi/Bluetooth integrado, que permite la conexión a redes inalámbricas para la comunicación con dispositivos externos y la transmisión de datos.

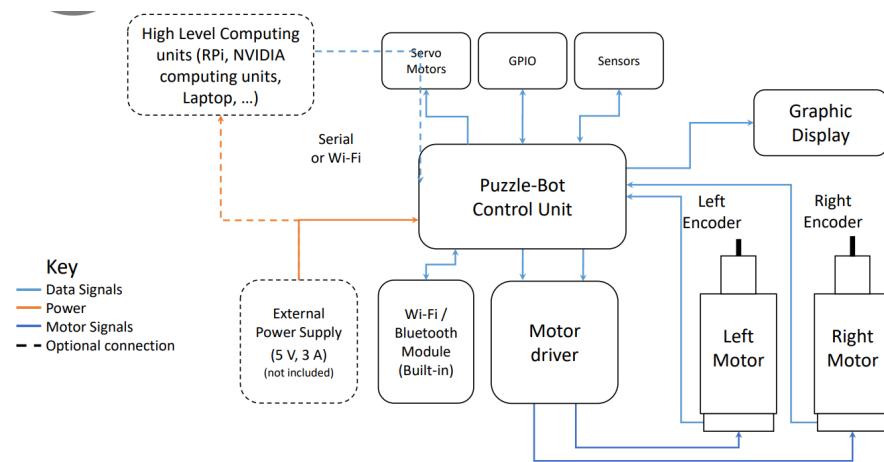


Figura 1. Diagrama de bloques de la unidad de control del Puzzlebot_[2].

Para establecer la comunicación y control remoto del Puzzlebot, se emplea la comunicación SSH (Secure Shell)_[4], un protocolo que permite la conexión segura a dispositivos remotos a través de una red, facilitando así la configuración y la interacción con el robot desde un terminal externo. Este método asegura que la comunicación entre el terminal externo y el Puzzlebot se realice de manera segura, protegiendo los datos y garantizando la integridad de la conexión.

Además, se utilizará el paquete Teleop Twist Keyboard por rohbotics para ROS_[5], que proporciona una interfaz sencilla para controlar el movimiento del robot utilizando el teclado

de un ordenador externo. Esta herramienta permitirá a los usuarios teleoperar el Puzzlebot de manera intuitiva y eficiente, utilizando comandos simples del teclado para controlar su movimiento en tiempo real.

Adicionalmente, se implementó la odometría^[6] para estimar la posición y orientación del robot basándose en la información de sus sensores de movimiento. Esta técnica permite realizar un seguimiento preciso de los desplazamientos del Puzzlebot, calculando su posición y orientación en función de los datos proporcionados por los encoders en las ruedas y otros sensores de movimiento. La odometría mejora la precisión en la estimación de la posición del Puzzlebot durante la teleoperación, lo que contribuye a una mayor eficiencia en su control y desplazamiento, especialmente en entornos dinámicos y cambiantes.

SOLUCIÓN DEL PROBLEMA

Antes de iniciar con la elaboración del reto, es indispensable contar con ciertos requerimientos de software y hardware. El protocolo de comunicación entre el Puzzlebot y un ordenador externo será remoto, por lo que es de suma importancia dar a conocer los siguientes aspectos.

Comunicación remota a Hackerboard

Inicialmente, esta configuración ha sido utilizada para probar el funcionamiento del Puzzlebot mediante comunicación serial a través de un ordenador externo y el robot. Utilizando directamente el firmware de la hackerboard, se controlarán aspectos como velocidades angulares y lineales de los motores utilizando ya sea wifi o serial.

Una vez que se haya instalado el firmware correctamente *FirmwareBin*, debe abrirse la ubicación del archivo en la terminal y ejecutar los siguientes comandos:

Para dar permisos a Ubuntu de ejecutar directamente el archivo y posteriormente ejecutarlo:

```
chmod +x FirmwareFlash.sh  
./FirmwareFlash.sh
```

A continuación, el sistema solicitará cambiar la SSID y contraseña de la red predeterminada, en este caso se han configurado como: *LecheroBot* y *Puzzlebot72*.

Una vez configurada, en la pantalla OLED integrada en la hackerboard, se mostrará el nombre asignado al robot y su dirección IP correspondiente. Para acceder a la red creada, deberá conectarse de la misma forma en que se conecta a una red de internet, buscando el nombre de esta junto con su contraseña.

Red Hotspot Jetson-nano

Antes de poder crear una red hotspot para la jetson, evidentemente es necesario cargar el sistema operativo a la placa única. Para ello, se necesitará la imagen del sistema operativo

(jetson_2gb_ubuntu) flasheada en una tarjeta micro SD de al menos 60GB de almacenamiento. Para realizar el flasheo, se utilizará el software Balena Etcher.

Conexión remota ordenador externo-jetson

Ahora tendremos qué configurar un punto de acceso utilizando nuestra NVIDIA Jetson Nano para poder tener correctamente la comunicación del dispositivo con una red inalámbrica.

Como primer paso necesitaremos conectar nuestra Jetson Nano a un monitor para poder manejar todo el programa que será similar a Ubuntu.

En la esquina inferior derecha observaremos y tendremos que dar click a las conexiones de redes donde veremos una pantalla emergente que se muestra en la *figura 2* donde se observan las redes disponibles. Daremos click en el símbolo de + para crear nuestra propia red.

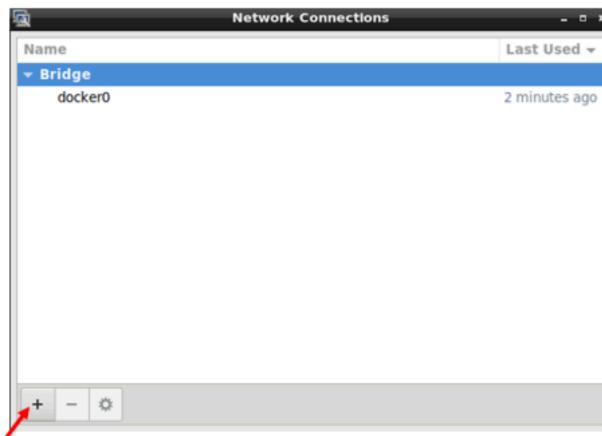


Figura 2. Ventana emergente para crear una red

En la *figura 3* podemos observar que ahora vamos a tener que personalizar el nombre de la red y el SSID, y importante qué seleccionemos el Hotspot en mode.

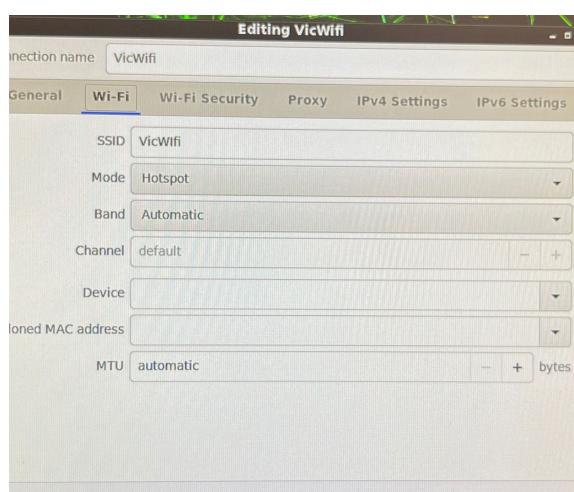


Figura 3. Personalizar los parametros del Hotspot

En la pestaña de Wi-Fi Security como se ve en la *figura 4* tendremos que seleccionar la seguridad que nosotros prefiramos y también establecer una contraseña. Además de que en la Configuración de IPV4 como se observa en la *figura 5* necesitaremos establecer una ip fija y el segmento IP de la red con su submáscara y la gateway.

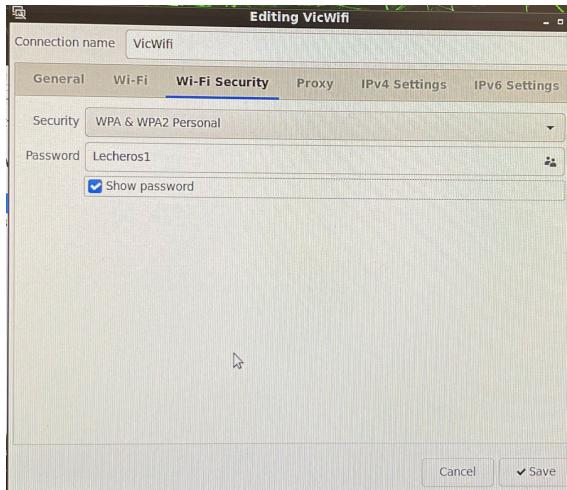


Figura 4. Establecer protocolo de seguridad

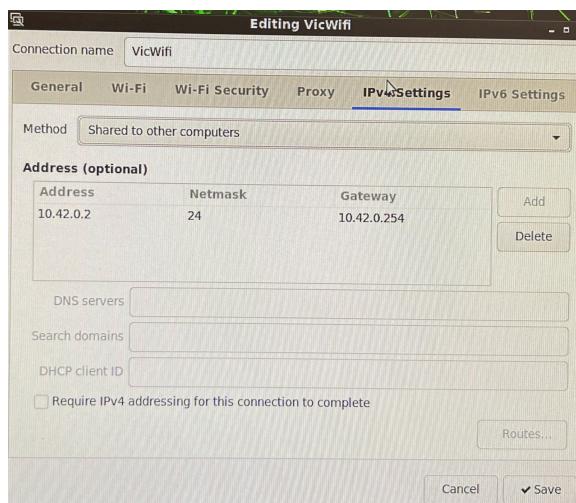


Figura 5. Establecer el segmento IP del Hotspot

Finalmente necesitamos dar click en el botón de save para poder guardar todo los pasos que ya hicimos y tener la red ya configurada completamente para que el punto de acceso ya se encuentre activo.

Conexión del Hotspot en computadora

Para poder conectar ya el punto de acceso que hemos creado, solo necesitamos en Ubuntu abrir las redes que se encuentran disponibles, es necesario tomar en cuenta que debemos de tener un módulo Wi-Fi en el robot para poder acceder al Hotspot. En el listado de las redes debe de aparecer la que ya configuraste con el nombre que le pusimos, en este caso **VicWifi** como se ve en la *figura 6* y pondremos la contraseña que igualmente ya la asignamos anteriormente.

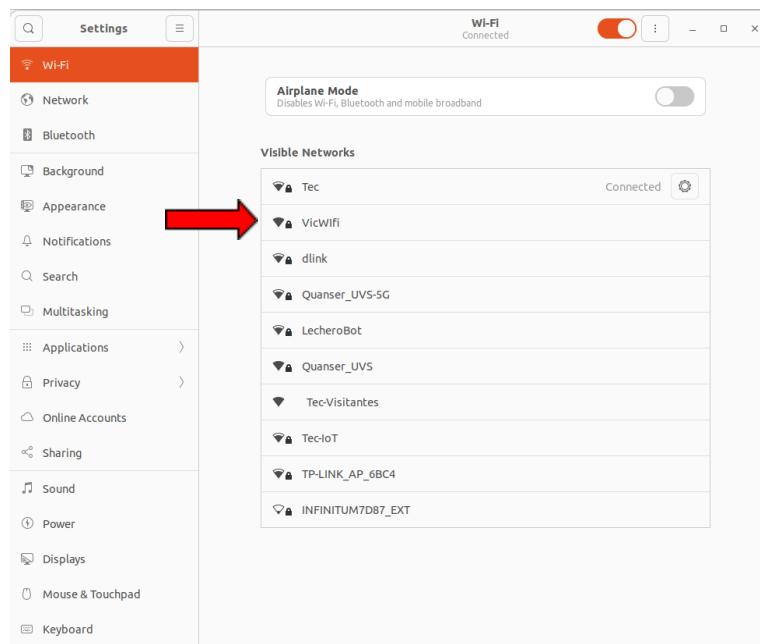


Figura 6. Conexión del Hotspot creado en la Jetson

Conexión Jetson nano con ordenador externo via ssh

Una vez que se ha implementado la red local para la jetson nano, es posible iniciar la comunicación entre dispositivos de manera remota. Para ello, se utilizará el ya mencionado protocolo SSH.

Antes de iniciar el control, es necesario conectarse al hotspot creado a la Jetson, utilizando la contraseña configurada previamente. Posteriormente, debe ingresarse el siguiente comando en la terminal para indicar dicho protocolo:

```
ssh puzzlebot@10.42.0.2
```

A continuación, el sistema solicitará ingresar la contraseña, sin embargo, debe ingresar la contraseña a la red del robot, no la de wi-fi (evitar confusiones entre redes), la cual para este caso se mantiene como Puzzlebot72.

Accediendo al robot, ahora debe habilitarse el puerto serial del mismo, para ello antes debe sersecoarse que la jetson está conectada mediante comunicación serial a la hackerboard. El comando a ingresar es:

```
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyUSB0
```

Finalmente, en otra terminal, para tener acceso a parámetros como dirección y velocidad de los motores, se ingresa el siguiente comando:

```
ros2 run teleop_twist_keyboard
```

Como resultado, se desplegará esta información:

```
puzzlebot@jetson: ~ ferestrada@fernando: ~
ferestrada@fernando: $ ros2 run teleop_twist_keyboard teleop_twist_keyboard
This node takes keypresses from the keyboard and publishes them
as Twist/TwistStamped messages. It works best with a US keyboard layout.
-----
Moving around:
  u    i    o
  j    k    l
  m    ,    .

For Holonomic mode (strafing), hold down the shift key:
-----
  U    I    O
  J    K    L
  M    <    >

t : up (+z)
o : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

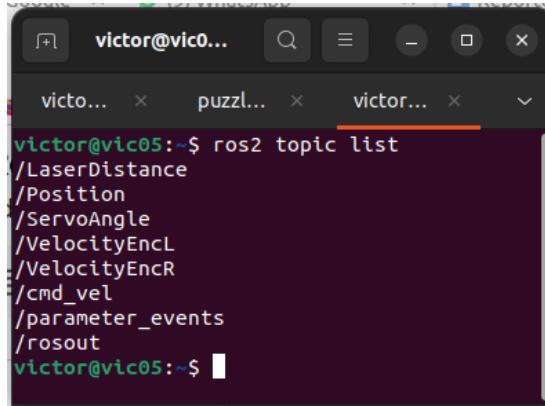
currently:      speed 0.5      turn 1.0
```

Figura 7. Interfaz para controlar Puzzlebot

Siguiendo sus instrucciones el robot podrá moverse hacia adelante, atrás, rotar, cambiar de velocidad y detenerse.

Implementación básica de odometría

Ahora que hemos logrado realizar la configuración de la Jetson Nano y del Puzzlebot para realizar el manejo del robot de manera teleoperada procederemos a realizar una implementación básica de odometría. Para implementar esto, haremos uso de los tópicos generados al ejecutar el agente microROS correspondiente al del Puzzlebot. Este agente, que permite el manejo de manera teleoperada del robot, también genera algunos tópicos que nos resultan útiles para realizar la implementación de odometría.



```
victor@vic05:~$ ros2 topic list
/LaserDistance
/Position
/ServoAngle
/VelocityEncL
/VelocityEncR
/cmd_vel
/parameter_events
/rosout
victor@vic05:~$
```

Figura 8. Lista de tópicos creados al ejecutar el agente micro ROS

Fórmulas matemáticas

En este caso en específico, haremos uso de los tópicos VelocityEncL y VelocityEncR, que vienen siendo las velocidades lineales de cada uno de los motores del Puzzlebot. Basándonos en la información proporcionada por Manchester Robotics en la sesión previa, realizaremos la implementación de odometría para el robot, partiendo del siguiente diagrama:

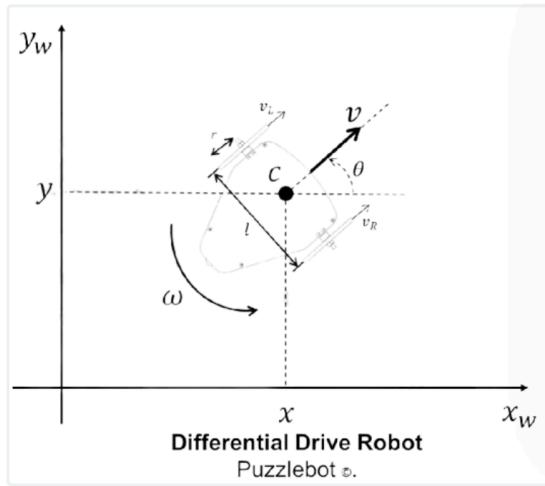


Figura 9. Representación de la orientación del Puzzlebot en 2D

Ahora bien, por medio de un análisis matemático se puede llegar a ciertas fórmulas específicas que nos serán útiles para conocer la posición de nuestro robot más adelante:

$$\begin{cases} \dot{x} = v \cdot \cos\theta \\ \dot{y} = v \cdot \sin\theta \\ \dot{\theta} = \omega \end{cases} \quad [1]$$

$$v = \left(\frac{v_R + v_L}{2} \right) \quad [2]$$

$$\omega = \left(\frac{v_R - v_L}{l} \right) \quad [3]$$

En dónde:

- v_R = Velocidad lineal de la rueda derecha
- v_L = Velocidad lineal de la rueda izquierda
- l = Distancia entre las dos ruedas

Ahora bien, implementando una integral a estos valores de velocidad, podemos obtener la posición 2D de nuestro robot: x, y, θ . Es importante mencionar que tal y como se observa en la *figura 9* y en las ecuaciones previamente listadas, necesitamos tener en consideración ciertas medidas del robot, tales como l y r . A continuación, explicaremos la implementación en código de este análisis matemático.

Código de ROS

Previo a comenzar con la programación en sí, comenzaremos por crear el paquete y el nodo correspondiente. Para ello, usando la terminal de Ubuntu, creamos una nueva carpeta *Week1_IA/src* y creamos el nodo utilizando el siguiente comando:

```
ros2 pkg create --build-type ament_cmake --license Apache-2.0 --node-name mi_nodo mi_paquete --dependencies rclpy std_msgs geometry_msgs
```

Tal y como se puede observar en el comando previamente mencionado, nuestro paquete dependerá de *rclpy*, *std_msgs* y *geometry_msgs*. Dicho esto, comenzamos con la programación del nodo *odometry_node*:

1. Importamos las librerías correspondientes para el manejo de nodos, así como el tipo de dato *Pose2D* de la librería *geometry_msgs* para el cálculo de la posición:

```
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Pose2D
import rclpy.qos
from std_msgs.msg import Float32
```

2. Creamos suscripciones a los tópicos correspondientes a la velocidad de cada una de las ruedas, asociando dicha suscripción con una función: *read_left* para la rueda izquierda y *read_right* para la rueda derecha. Dentro de estas funciones, únicamente realizaremos una copia de los datos a una variable propia del nodo:

```
self.left_speed_subscriber = self.create_subscription(Float32,
'VelocityEncL', self.read_left, rclpy.qos.qos_profile_sensor_data)

def read_left(self, msg):
```

```
    self.left = msg
```

3. Creamos un nuevo publisher, el cuál se encargará de publicar la posición cada 0.1 segundo, es decir, a una frecuencia de 10 Hz.

```
self.position_publisher = self.create_publisher(Pose2D, 'Position', 10)

timer_period = 0.1

self.timer = self.create_timer(timer_period, self.position_callback)
```

4. Dentro de la función position callback, realizamos los cálculos correspondientes de las fórmulas previamente mencionadas para el cálculo de velocidad y, realizando una integral, obtendremos la posición. Este dato calculado se publica a través del publisher de la posición:

```
def position_callback(self):

    #Calculamos la velocidad para x, y y theta

    self.xp = ((self.right.data+self.left.data)/2)*np.cos(self.theta)
    self.yp = ((self.right.data+self.left.data)/2)*np.sin(self.theta)
    self.thetap = (self.right.data - self.left.data)/self.l

    #Integramos para obtener la posición

    self.x += self.xp
    self.y += self.yp
    self.theta += self.thetap

    self.position_publisher.publish(self.position)
```

Recordemos que matemáticamente la integral es equivalente a una suma (área bajo la curva).

5. Finalmente, dentro de nuestro main realizamos la inicialización del nodo:

```
#Inicialización del nodo

def main(args=None):

    rclpy.init(args=args)

    m_p = My_publisher()

    rclpy.spin(m_p)

    m_p.destroy_node()

    rclpy.shutdown()
```

```
if __name__ == '__main__':
    main()
```

A continuación, mostraremos la manera de ejecutar este nodo y los resultados obtenidos.

RESULTADOS

Para poder observar la implementación de esta implementación básica de odometría debemos realizar los pasos listados a continuación (previamente mencionados de manera más detallada en secciones previas del documento).

1. Conectarse al hotspot configurado para la Jetson Nano.
2. En una nueva terminal, enlazarse con la Jetson Nano escribiendo lo siguiente en terminal.
3. Posteriormente, deberemos ejecutar el correspondiente agente de micro-ros para que la hackerboard conectada a la Jetson comience a funcionar.
4. Una vez realizados los pasos anteriores, es posible comenzar a manipular o manejar nuestro puzzlebot de manera teleoperada.
5. En este punto, todos los nodos y tópicos necesarios para el funcionamiento de nuestro programa estarán en funcionamiento, por lo que, es momento de iniciar el nodo que hemos creado para esta actividad. Para ello, abriremos una nueva terminal y nos moveremos al directorio en donde estamos y ejecutaremos el nodo creado:

```
ros2 run odometry odometry
```

A continuación, podemos observar un video demostrativo con su respectiva explicación del funcionamiento del código en conjunto con el movimiento del robot:

Vídeo del funcionamiento:

https://drive.google.com/file/d/1vXFyKFU9GtLIDw_WFWMm6KtZHK96tyq/view?usp=drive_link

Primera prueba

En principio tuvimos algunas complicaciones para poder empezar a tomar nuestro eje de referencia para el movimiento del Puzzlebot. No obstante, logramos establecer un margen sobre el que actuaría el robot. Movimos el robot hacia el frente mirándonos correctamente los valores en positivo y después hacia atrás en Y para observar cómo cambia a altos números negativos. Finalmente hicimos varios giros para también ver cómo es que theta estaba cambiando a los giros del Puzzlebot y observamos que en ocasiones podía marcar un valor más alto a lo esperado para el ángulo, esto debido a que en ocasiones el motor se mantiene girando pero la rueda, al no estar perfectamente ajustada, no gira.

Segunda prueba

En esta prueba, optamos por un recorrido más extenso pero con movimientos más precisos, lo que resultó en mejores valores para (x, y), otorgando así un sentido más coherente con la posición consecutiva del Puzzlebot. Los giros se ejecutaron de manera más efectiva, ya que los valores estaban siendo marcados correctamente. Al aumentar el número de giros, estos seguían manteniendo su coherencia, mientras que en los tramos de avance casi recto, observamos que los ángulos apenas variaron, lo que se tradujo en una mayor precisión en esta prueba

CONCLUSIONES

Durante el desarrollo del reto pudimos lograr un avance significativo en la implementación de la teleoperación en el robot Puzzlebot con la ayuda de ROS y micro-ROS, en general la mayoría de los objetivos planteados al inicio fueron logrados satisfactoriamente, pudimos tener una gran familiarización con el entorno del movimiento del robot y la configuración de la Jetson para poder generar un punto de acceso denominado como Hotspot. Se analizó correctamente la estructura y el funcionamiento completo del Puzzlebot, comprendiendo la interconexión de sus componentes y las capacidades físicas.

También tuvimos de gran manera la experimentación con la operación remota del Puzzlebot utilizando el ROS teleop twist keyboard y una gran implementación de la odometría para estimar correctamente la posición y orientación del robot. Tuvimos ciertos inconvenientes en la realización del reto como lo fue tener una mala conexión del módulo Wi-Fi debido a que nuestro Puzzlebot no quería prenderlo, posteriormente tuvimos un inconveniente con el hotspot ya que lo teníamos mal configurado y creado desde la Jetson y tuvimos que crear otro punto de acceso para que por fin ya estuviera en funcionamiento. Ya como parte física, la pila que teníamos estaba consumiendo muy rápido su batería por lo que no daba mucho tiempo de hacer pruebas o de que estuviera encendida.

A pesar de tener estos inconvenientes, pudimos lograr los objetivos de la mejor manera y finalmente tener una resolución del reto adecuada. Como posible mejor principal podríamos tener una mejor optimización de la odometría con la automatización del proceso de configuración de redes o proporcionando más scripts para facilitar la conexión remota del Puzzlebot. Podríamos integrar también un sistema de localización que fuera más preciso como poder implementar unos sensores de posición para tener una mejor estimación de la posición de robot. Estas mejoras pueden ser totalmente potenciales para poder contribuir a tener una mayor eficiencia en el control remoto y una localización más exacta.

REFERENCIAS

1. Manchester Robotics. *Puzzlebot: a revolution in robotics*. Manchester-robotics. Recuperado el 10 de Abril del 2024 de <https://manchester-robotics.com/puzzlebot/>
2. Manchester Robotics. *Hacker board*. Manchester-robotics. Recuperado el 10 de Abril del 2024 de <https://manchester-robotics.com/puzzlebot/hacker-board/>
3. Manchester Robotics. *Puzzlebot Jetson Edition*. Manchester-robotics. Recuperado el 10 de Abril del 2024 de <https://manchester-robotics.com/puzzlebot/puzzlebot-jetson-edition/>
4. Zúñiga F. (17 de Agosto del 2022). *SSH: qué es y cómo funciona este protocolo*. Airsys. <https://www.arsys.es/blog/ssh#:~:text=SSH%20son%20las%20siglas%20de,como%20v%C3%ADA%20para%20las%20comunicaciones>.
5. rohbotics. *ros2_teleop_keyboard*. Recuperado el 10 de Abril del 2024 de https://github.com/rohbotics/ros2_teleop_keyboard
6. Stefal. (última edición 2019-10-29). Publicación de información de odometría a través de ROS. Ros.org. <https://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom>