Reto semanal 1. Manchester Robotics

Implementación de Robótica Inteligente

Integrantes:

A017314050 | Eleazar Olivas Gaspar

A01735696 | Azul Nahomi Machorro Arreola

A01732584 | Angel Estrada Centeno

A01735692 | Arick Morelos del Campo

Profesores:

Rigoberto Cerino Jimenez

Alfredo Garcia Suarez

Juan Manuel Ahuactzin Larios

Cesar Torres Huitzil

Abril 2024

Resumen
A lo largo de este reporte, detallaremos los pasos necesarios para configurar un Puzzlebot de manera que pueda ser teleoperado desde una computadora que utiliza micro-ROS. Comenzaremos presentando los componentes que forman parte del Puzzlebot, proporcionando una visión general de sus elementos constitutivos y su función en el sistema. Estos componentes incluyen los motores, las ruedas, los sensores, los actuadores y el módulo de comunicación para conectividad con el ordenador.

Una vez comprendida la estructura y funcionalidades del Puzzlebot, pasaremos a describir el proceso de configuración de cada uno de estos elementos para lograr la teleoperación. Esto incluye el establecimiento de la comunicación entre el ordenador y el robot utilizando micro-ROS, que requiere la instalación y configuración del software necesario en ambos dispositivos. Se detallará cómo crear y ejecutar nodos micro-ROS en la computadora para enviar comandos de control al Puzzlebot, como movimientos hacia adelante, atrás, a la izquierda o a la derecha, utilizando diferentes tipos de entradas de datos.

Además, veremos cómo configurar los nodos micro-ROS en el Puzzlebot para recibir y procesar los comandos de teleoperación. Esto abarca la asignación de comandos a las acciones correspondientes en el robot, como el control de motores para cambiar la dirección y velocidad de movimiento.

Por último, cubriremos las pruebas del sistema teleoperado, que son fundamentales para asegurar que el Puzzlebot responda de manera precisa y fiable a los comandos de la computadora.

Objetivos
Para alcanzar los propósitos de este reto debemos conocer el hardware y software del Puzzlebot para comprender sus componentes, funciones y su integración con ROS, para esta integración debemos aprender la configuración de comunicación adecuada, así como también a través de los nodos podemos manejar diferentes aspectos del robot.

Perfeccionamos nuestras habilidades de programación con ROS para la interacción de control hacia el Puzzlebot, también practicaremos la publicación y suscripción a tópicos ROS.

Introducción
Durante el desarrollo de proyectos anteriores con ROS (Robot Operating System), se nos mostró las capacidades de la plataforma para la comunicación de información a través de nodos, tópicos, mensajes, servicios y parámetros. Todo esto para el manejo de un motor con sus respectivos drivers.

Ahora para el desarrollo de este reto se nos permitirá ver los alcances que tiene esta plataforma aplicada a un modelo de robot más completo, como puede ser el caso del Puzzle Bot, que contiene una NVIDIA Jetson Nano, Hacker Board, RaspBerry Pi Camera y dos motores con motoreductor. A continuación explicaremos un poco sobre cada componentes:

Puzzlebot es un robot educativo diseñado para enseñar y practicar conceptos de robótica y programación utilizando el framework ROS y micro-ROS. Los componentes principales del Puzzlebot incluyen:

- Chasis y ruedas: El chasis es la estructura que sostiene todos los componentes del robot, y las ruedas permiten el movimiento del robot. Puede tener dos ruedas motrices y una rueda loca para estabilidad.
- Motores: Los motores controlan el movimiento de las ruedas, permitiendo que el robot avance, retroceda y gire.
- Sensores: Puzzlebot puede estar equipado con varios sensores, como ultrasonidos para detección de obstáculos, infrarrojos, de línea o incluso cámaras, dependiendo del modelo.
- Controlador principal: El cerebro del robot, que puede ser una placa de desarrollo o una microcontroladora compatible con ROS o micro-ROS.
- Módulos de comunicación: Estos permiten que el robot se comunique con un ordenador o dispositivo móvil. Puede ser Wi-Fi, Bluetooth o radiofrecuencia.
- Batería: Proporciona la energía necesaria para el funcionamiento de los distintos componentes del robot.
- La interconexión de estos elementos se realiza a través de conexiones físicas como cables y placas de circuitos, y se gestiona por software mediante ROS o micro-ROS, lo que permite controlar y coordinar las distintas funciones del robot.

En cuanto a la conexión SSH (Secure Shell) es un protocolo de red seguro utilizado para iniciar sesión de forma remota en un servidor o dispositivo desde otro dispositivo. En el contexto de Puzzlebot, SSH puede usarse para conectarse de forma remota a la placa controladora del robot desde un ordenador. El funcionamiento de la comunicación SSH consiste en:

- Conexión: Se establece una conexión entre el ordenador y el robot utilizando la dirección IP del robot y un cliente SSH (como OpenSSH).
- Autenticación: El cliente SSH se autentica en el robot mediante un nombre de usuario y contraseña, o preferentemente mediante claves públicas y privadas para mayor seguridad.
- Terminal remoto: Una vez autenticado, el usuario tiene acceso a un terminal remoto en el robot, lo que le permite ejecutar comandos de control, supervisar procesos y modificar configuraciones.
- Cifrado: Todas las comunicaciones a través de SSH están cifradas para proteger la confidencialidad e integridad de los datos.

Teleop Twist Keyboard es una herramienta disponible en ROS que permite controlar un robot móvil utilizando el teclado del ordenador. Su funcionamiento se basa en:

- Instalación y configuración: El paquete se instala en el entorno de ROS del ordenador y se configura para conectarse a los tópicos de control del robot.
- Uso: Al ejecutar el comando correspondiente, la herramienta muestra instrucciones en la terminal sobre cómo usar las teclas para controlar el robot.
- Envío de comandos: Al presionar las teclas indicadas, el programa publica mensajes de tipo Twist en un tópico específico de ROS, como /cmd _vel.
- Control del robot: El robot escucha en el tópico /cmd _vel y responde a los mensajes Twist ajustando su velocidad lineal y angular según las entradas del teclado.

Con todas estas característica que posee el Puzzle Bot, nos da un gran abanico de posibilidades para configurar este robot, pero para esta primer instancia, empezaremos con las configuración del robot para que podamos teleoperarlo con una PC por medio de ROS

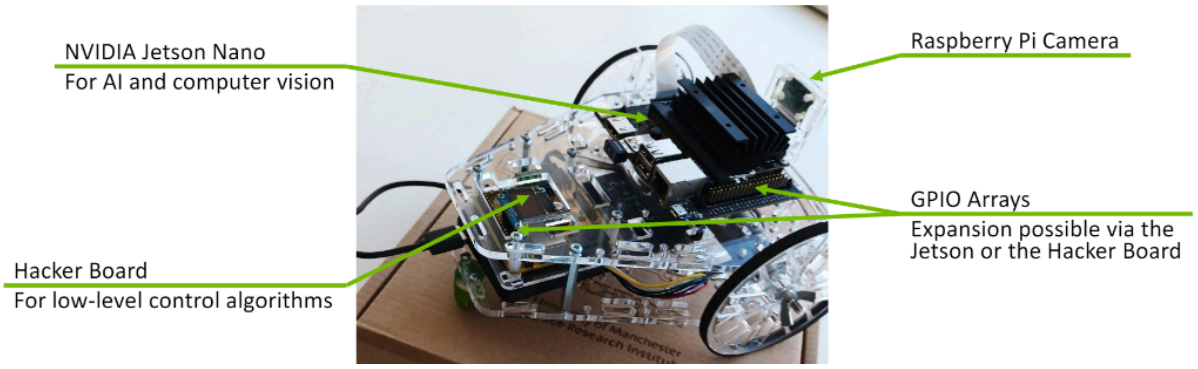

Fig.1. Componentes del Puzzle Bot

Solución del problema

Para la solución del Problema se dividió el proceso en 2 partes, la primera para la implementación del Puzzlebot así como la instalación de recursos necesarios para el funcionamiento del mismo y la configuración de redes necesarias para la conexión del robot, y la segunda siendo el desarrollo del nodo que da la posición del robot en un formato [x,t,**θ**].

-Configuración del robot.

Para la configuración del robot se empezó por la por hacer flashear una tarjeta micro sd con ubuntu 20.04 el cual se estará usando desde la placa Jetson Nano. Este proceso se hizo haciendo uso de herramientas como balena etcher, el cual teniendo una imagen el software se encarga de flashear dicha imagen dentro de la microSD. Mientras que se estaba haciendo el proceso de flasheado en la SD en paralelo se tuvo que hacer lo mismo con la Hacker Board para descargar el firmware apropiado para el uso del Puzzle Bot Esto se hizo por medio de archivos que se otorgaron por el profesor en clase.

-Configuración de redes:

Dentro del sistema hay 2 redes, de las cuales una es levantada por la Hacker Board, y otra por un hot-spot que se conecta a la Jetson Nano.

1) Hacker Board: dentro de la hacker board se inicia una red llama Puzzle bota Azul con el fin de tener redes distintas a otros equipos. Esta se hizo una vez a lo largo de la solución del reto con el fin de verificar que los componentes del robot funcionan de forma correcta.
2) Jetson Nano: En la Jetson se maneja una red de nombre JetsonAzul a la cual se le asignó la siguiente IP 10.42.0.69.

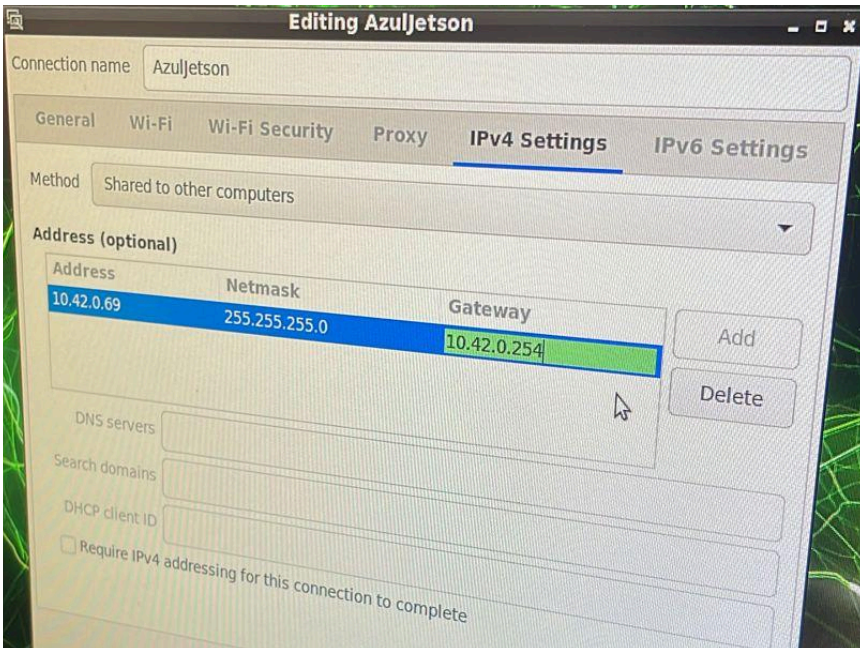

Fig.2. Configuración de la IPv4, Netmask y Gateway de la jetson.

Esta IP es la que se usa para conseguir una conexión remota con el robot, para el desarrollo del nodo y el manejo del Puzzlebot.

-Manejo del robot:

Para el manejo del robot se empieza por hacer la conexión remota. Esto se hace desde terminal y haciendo uso de la IP anteriormente presentada.

```
ssh puzzlebot@10.42.0.69
```

Desde el terminal se ejecuta la anterior línea de comando con el fin de hacer la conexión remota.

Posteriormente se inicia el agente de ros2

```
ros2 run micro_ros_agent micro_ros_agent serial --dev /dev/ttyUDB0
```

Una vez iniciado en agente de ros podemos empezar a correr funciones de ros dentro del Puzzlebot.

Para el manejo del mismo se usa el siguiente comando.

ros2 run teleop_twist_keyboard teleop_twist_keyboard

Este comando ejecuta un nodo de ros2 el cual no deja hacer uso del robot desde el teclado en una terminal. (Teleop_Twist_Keyboard - ROS Wiki, s. f.)

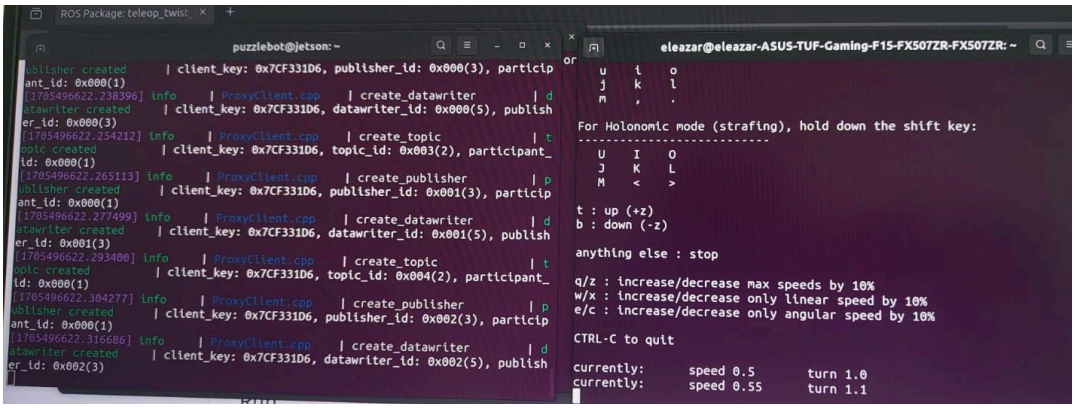

Fig.3. ejecución del comando para la teleoperación

-Nodo ros2:

```python
import rclpy
from rclpy.node import Node
from std_msgs.msg import Float32
from geometry_msgs.msg import Pose2D
import numpy as np

#Definición de la clase
class Velocity(Node):

    #Constructor
    def __init__(self):

        #Parametros velocidad
        self.thp = 0.0
        self.xp = 0.0
        self.yp = 0.0

        #Parametros posicion
        self.th = 0.0
        self.x = 0.0
        self.y = 0.0

        #Parámetros físicos
        self.r = 0.19
        self.l = 0.05

        #Velocidades angulares de encoders
        self.velR = 0
        self.velL = 0
```

Fig.4. Importación de librerías a utilizar, creación de la clase y declaración de variables globales

```python
        #Definición del tópico
        super().__init__('Azul_Velocity')

        #Creación de suscriptores para leer datos de encoders R L
        self.subR = self.create_subscription(Float32, 'VelocityEncR', self.listener_callbackR,
        self.subL = self.create_subscription(Float32, 'VelocityEncL', self.listener_callbackL,

        #Confirmación de la creación del nodo
        self.get_logger().info('Velocity node successfully initialized!!!')

        #Definición del periodo de tiempo
        timer_period = 0.5

        #Timer para operaciones
        self.timer = self.create_timer(timer_period, self.timer_callback)

        #Creación del tópico publicador: Azul_Velocity
        self.publisher = self.create_publisher(Pose2D, 'Azul_Velocity', 10)

        #Tipo de mensaje
        self.msg = Float32()

```

Fig.5.Creación de publicadores, suscriptores, y timer

```python
    #Lectura del encoder Velocity R
    def listener_callbackR(self, msg):

        #Guarda el dato
        self.velR = msg.data

    #mLectura del encoder Velocity L
    def listener_callbackL(self, msg):

        #Guarda el dato
        self.velL = msg.data

    #Método del timer
    def timer_callback(self):

        #Fórmulas para obtener las velocidades
        self.thp = self.r * ((self.velR - self.velL)/self.l)
        self.th += self.thp * 0.5

        self.xp = self.r * ((self.velR + self.velL)/2) * np.cos(self.th)
        self.x += self.xp * 0.5

        self.yp = self.r * ((self.velR + self.velL)/2) * np.sin(self.th)
        self.y += self.yp * 0.5

        #Publicación del mensaje en formato Pose2D
        pose_msg = Pose2D()
        pose_msg.x = self.x
        pose_msg.y = self.y
        pose_msg.theta = self.th
        self.publisher.publish(pose_msg)
```

Fig.6. Lectura, procesamiento y publicación de los datos

```python
#Main Fnc
def main(args=None):
    #Inicialiation for rclpy
    rclpy.init(args=args)
    #create node
    m_p = Velocity()
    #Spin method for publisher calback
    rclpy.spin(m_p)
    #Destoy node
    m_p.destroy_node()
    #rclpy shutdown
    rclpy.shutdown()

#main call method
if __name__ == 'main':
    main()
```

Fig.7. Función main del código

Implementación del nodo desde el Puzzle Bot:

Una vez completado el nodo desde Ubuntu solo queda el paso de hacer la implementación en la Jetson Nano. Para este paso solo se tiene que hacer conexión con la jetson con el comando ssh mencionado previamente, con el fin de posteriormente moverse a la carpeta de /ros2_ws/src/ para crear el nodo desde la jetson. Este último paso se logra con el siguiente comando.

ros2 pkg create --build-type ament_python --license Apache-2.0 azul_puzzlebot --node-name pos_puzzlebot_azul --dependencies std_msgs geometry_msgs

Después de haber creado el nodo solo hace falta abrir el nodo desde un editor de código, y pegar el código anteriormente realizado, y correrlo con el comando siguiente.

ros2 run azul_puzzlebot pos_pozzlebot_azul

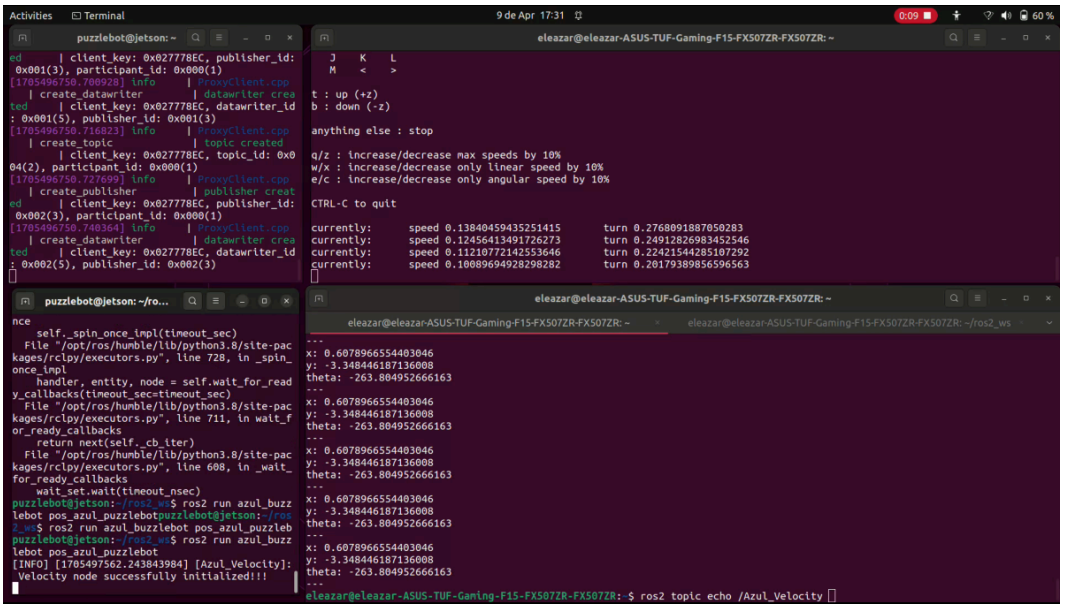

Fig.8. Implementación del nodo desde el Puzzle Bot

Solución del reto:

Para ver los resultados que obtuvimos solo es necesario hacer en conjunto los pasos previamente discutidos, y correr de forma simultánea el agente de ros, el nodo que nos deja controlar el robot, y el nodo que nos da la posición del robot.

Una vez realizado este paso se pudo observar que se estaba ejecutando de forma correcta lo nodos, pero al ver los datos obtenidos por el nodo de posición podemos fundamentar que el error obtenido es obtenido por la condición el la que el robot se encuentra; esto se trata a fondo el la sección de resultados.

Resultados

Dentro de los resultados obtenidos en el desarrollo del reto semanal encontramos buenas respuestas así como errores o detalles sobre los cuales se tendrá que trabajar más adelante. siento estos los siguientes:

- Funcionamiento del nodo: Al empezar analizar los resultados del nodo para saber la posición del robot nos pudimos dar cuenta de que había un error el cuál se debió a variables externas en el sistema las cuales eran mayormente provocadas por la condición de las llantas. Al analizar cómo es que una de lar ruedas se movía con una velocidad mayor, se propuso que la unión del motor a la rueda no se encontraba segura; lo cuál provoca un deslizamiento de la misma.
- Desplazamiento del robot: Uno de los detalles que más llamó la atención es que debido al mal cuidado de las llantas del robot el desplazamiento de esto no es como se esperaba. Debido a que al mandar la señal para que el robot se moviera hacia enfrente, parece haber un deslizamiento en la rueda, lo que provoca que el Puzzlebot se desviara de su trayectoria. El anterior fenómeno era apreciable a simple vista, y al analizar los resultados del nodo que se generó para saber su posición.

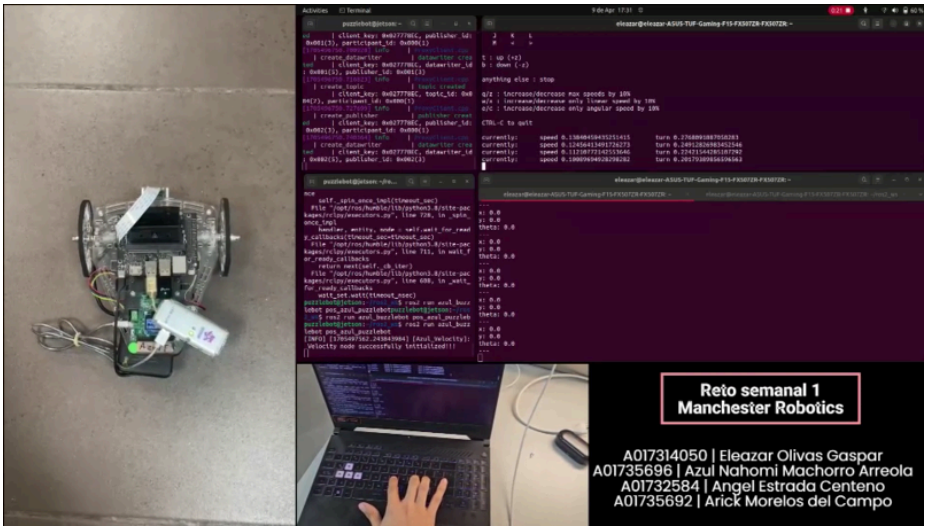

Fig.8. Captura del vídeo demostrativo de los resultados

Siendo los anteriores los puntos focales más importantes en el planteamiento y desarrollo de retos futuros se busca el encontrar soluciones en paralelo a las propuestas en clase. Por otro lado se puede ver el funcionamiento del nodo y el Puzzle Bot dentro del video adjunto en la sección de anexos.

Conclusiones

En conclusión, este trabajo ha demostrado con éxito cómo configurar un Puzzlebot para teleoperación utilizando una computadora con micro-ROS. A lo largo del proceso, hemos cumplido con los propósitos y objetivos planteados al inicio, proporcionándoles una experiencia práctica en el manejo de ROS y micro-ROS para controlar un robot físico. Asimismo hemos adquirido un conocimiento sólido de los componentes del Puzzlebot y su papel en el sistema, así como su interconexión para lograr la teleoperación efectiva.

La integración del Puzzlebot con micro-ROS se logró con éxito, incluyendo la comunicación entre el robot y la computadora. Durante el proceso, se realizaron pruebas de funcionamiento para asegurar que el Puzzlebot respondiera de manera precisa a los comandos de la computadora, los resultados obtenidos muestran un sistema teleoperado eficiente, capaz de responder a diferentes entradas de datos y realizar movimientos precisos según las instrucciones.

Bibliografía o referencias

1. *PuzzleBot Jetson Edition – Manchester Robotics*. (s. f.).

   https://manchester-robotics.com/puzzlebot/puzzlebot-jetson-edition/

2. *teleop_twist_keyboard - ROS Wiki*. (s. f.). https://wiki.ros.org/teleop_twist_keyboard

3.

Anexos:

1)Video de funcionamiento y resultados:

https://drive.google.com/file/d/1z0DwcNvS3-ZxXB4I_WUAL3miwOXGwxpk/view?usp=sharing