Resumen completo del Sprint 1 con explicaciones

1️⃣ Conceptos básicos de ROS 2

1.1 Qué es un workspace

- Un **workspace** es una carpeta donde desarrollas tus paquetes y nodos ROS 2.
- Contiene el código fuente, archivos de configuración y se puede compilar para generar los scripts ejecutables.
- En nuestro caso:

~/universidad_agusti/tercero/proyecto/sprint_1/prii3_ws

1.2 Qué es un paquete

- Un **paquete** es la unidad mínima de código en ROS 2.
- Contiene:
    - Nodos (scripts que hacen algo)
    - Dependencias
    - Archivos de configuración (package.xml, setup.py)
- En este Sprint tenemos el paquete **prii3_turtlesim** que controlará turtlesim.

1.3 Qué es un nodo

- Un **nodo** es un programa que hace **una tarea concreta** dentro del sistema ROS 2.
- Los nodos pueden:
    - Publicar mensajes en un **topic** (como mover la tortuga)
    - Suscribirse a topics
    - Llamar o atender **servicios**
    - Ejecutar acciones
- En este Sprint, nuestro nodo **prii3_turtlesim_node.py** mueve automáticamente la tortuga para dibujar un número.

1.4 Qué es un topic

- Un **topic** es un canal de comunicación donde los nodos publican y suscriben mensajes.

- Ejemplo: turtle1/cmd_vel es un topic donde publicamos **velocidades lineales y angulares** para mover la tortuga.

2️⃣ Preparación del entorno

Source de ROS 2 Foxy

source /opt/ros/foxy/setup.bash

Actualizar repositorios e instalar turtlesim

sudo apt update

sudo apt install ros-foxy-turtlesim

Verificar ejecutables del paquete

ros2 pkg executables turtlesim

Esto asegura que ROS 2 Foxy y turtlesim están correctamente instalados.

3️⃣ Crear workspace y paquete Python

Crear workspace

mkdir -p ~/universidad_agusti/tercero/proyecto/sprint_1/prii3_ws

cd ~/universidad_agusti/tercero/proyecto/sprint_1/prii3_ws

Crear paquete Python con dependencias rclpy y turtlesim

ros2 pkg create prii3_turtlesim --build-type ament_python --dependencies rclpy turtlesim --destination-directory .

rclpy: librería Python de ROS 2 para crear nodos.
turtlesim: paquete que contiene el simulador de la tortuga.

4️⃣ Crear nodo Python

4.1 Archivo del nodo

Archivo: prii3_turtlesim/prii3_turtlesim/prii3_turtlesim_node.py

#!/usr/bin/env python3

import rclpy

from rclpy.node import Node

```
from geometry_msgs.msg import Twist


# Clase que define un nodo
class TurtleDrawer(Node):
    def __init__(self):
        super().__init__('turtle_drawer')  # Nombre del nodo
        self.pub = self.create_publisher(Twist, 'turtle1/cmd_vel', 10)
        self.timer = self.create_timer(0.5, self.move)  # Ejecuta move() cada 0.5s
        self.step = 0


    def move(self):
        msg = Twist()
        # Movimiento de la tortuga (dibujar número)
        if self.step < 20:
            msg.linear.x = 1.0
            msg.angular.z = 1.0
        elif self.step < 40:
            msg.linear.x = 1.0
            msg.angular.z = -1.0
        else:
            msg.linear.x = msg.angular.z = 0.0
            self.get_logger().info('Dibujo terminado')
            self.timer.cancel()
        self.pub.publish(msg)
        self.step += 1


def main():
    rclpy.init()        # Inicializa ROS 2
    node = TurtleDrawer()  # Crea nodo
    rclpy.spin(node)      # Mantiene nodo activo
    node.destroy_node()    # Limpieza
```

```
    rclpy.shutdown()      # Cierra ROS 2


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

- **Twist**: mensaje que indica velocidad lineal y angular.
- **create_timer**: ejecuta periódicamente la función move.

4.2 Hacer ejecutable

chmod +x prii3_turtlesim/prii3_turtlesim/prii3_turtlesim_node.py

---

4.3 Registrar nodo en setup.py

Archivo: prii3_turtlesim/setup.py

```
entry_points={
    'console_scripts': [
        'drawer = prii3_turtlesim.prii3_turtlesim_node:main',
    ],
},
```

Esto permite ejecutar el nodo con:

ros2 run prii3_turtlesim drawer

5️⃣ Compilar workspace

cd ~/universidad_agusti/tercero/proyecto/sprint_1/prii3_ws

colcon build

source install/setup.bash

Opcional para no sourcear siempre:

echo "source ~/universidad_agusti/tercero/proyecto/sprint_1/prii3_ws/install/setup.bash" >> ~/.bashrc

source ~/.bashrc

6️⃣ Probar turtlesim y nodo

- Terminal 1:

ros2 run turtlesim turtlesim_node

- Terminal 2:

ros2 run prii3_turtlesim drawer

La tortuga debe moverse automáticamente dibujando el número de tu grupo.

7️⃣ VS Code (opcional)

- Abrir workspace: code ~/universidad_agusti/tercero/proyecto/sprint_1/prii3_ws
- Configurar Python, instalar extensiones **Python** y **ROS**.
- Terminal integrada para compilar y ejecutar nodos sin salir del editor.
- Depuración con launch.json para ejecutar nodo directamente.

8️⃣ Control de versiones con Git

cd ~/universidad_agusti/tercero/proyecto/sprint_1/prii3_ws

git init

git add .

git commit -m "Workspace con nodo prii3_turtlesim"

git remote add origin <URL-del-repo>

git branch -M main

git push -u origin main

9️⃣ Qué se entrega en el Sprint

- **PBI 1.1:** ROS 2 Foxy y turtlesim instalados y configurados.
- **PBI 1.2:** Workspace prii3_ws con paquete prii3_turtlesim.
- **PBI 1.3:** Nodo Python prii3_turtlesim_node.py que dibuja automáticamente el número del grupo, código gestionado con Git.