Launch Guide: Katana UGV Control System

Introduction

This guide provides instructions for launching the Katana UGV control system components. The system consists of:

- ESP32 Bridge: Handles communication between ROS2 and ESP32
- Xbox Controller: Provides teleoperation capabilities
- GUI: Visualizes controller inputs and UGV status
- RViz: Optional visualization of UGV state

Build Instructions

First, set the required file permissions (Control Station):

```
# Option 1: Make all Python files executable at once
find ~/katana_ws/src -name "*.py" -exec chmod +x {} \;

# Option 2: Make specific files executable
chmod +x ~/katana_ws/src/gui/gui/control_visual.py
chmod +x ~/katana_ws/src/gui/setup.py
chmod +x ~/katana_ws/src/gui/launch/remote_control.launch.py
chmod +x ~/katana_ws/src/remote_control/remote_control/xbox_control.py
chmod +x ~/katana_ws/src/remote_control/setup.py
```

If you need to build the workspace (UGV):

```
cd ~/seb_ws
source /opt/ros/humble/setup.bash
colcon build --symlink-install
```

For control station workspace:

```
cd ~/katana_ws
source /opt/ros/humble/setup.bash
colcon build --symlink-install
```

Basic Control Setup

Terminal Window 1: ESP32 Bridge Node (On UGV)

```
cd ~/seb_ws
source install/setup.bash
ros2 run esp32_bridge esp32_bridge_node
```

Terminal Window 2: Xbox Controller (On Control Station)

```
cd ~/katana_ws
source install/setup.bash
ros2 run remote_control xbox_control
```

Terminal Window 3: Control Visualization (On Control Station)

```
cd ~/katana_ws
source install/setup.bash
ros2 run gui control_visual
```

Combined Launch Option

Alternatively, launch all control station components with:

```
cd ~/katana_ws
source install/setup.bash
ros2 launch gui remote_control.launch.py
```

Optional: RViz Visualization

- Purpose: Visualize UGV state and sensor data

Launch RViz:

```
rviz2
```

Verification

To verify the system is working correctly:

- Check Xbox controller input:
  - Topic: `/joy`
  - Should update with controller movement
- Check movement commands:
  - Topic: `/cmd_vel`
  - Should reflect controller input
- Check encoder feedback:
  - Topic: `/wheel_encoders`
  - Should update with motor movement

Check Topics:

```
ros2 topic list
ros2 topic echo /joy
ros2 topic echo /cmd_vel
ros2 topic echo /wheel_encoders
```

Troubleshooting

If the system isn't working:

- Check if Xbox controller is detected:

```
ls /dev/input/js0
```

- Verify ROS2 network setup between UGV and Control Station:

```
ros2 node list
ros2 topic list
```

- Check ESP32 serial connection on UGV:

```
ls /dev/ttyUSB0
```

- Verify GUI is updating with controller input
- Check for any error messages in terminal outputs

Controls

Xbox Controller mapping:

- Left Stick: Forward/Backward movement
- Right Stick: Left/Right rotation
- RB Button: Turbo mode (2x speed)
- B Button: Emergency stop