Activity

Noise Simulation/Estimation

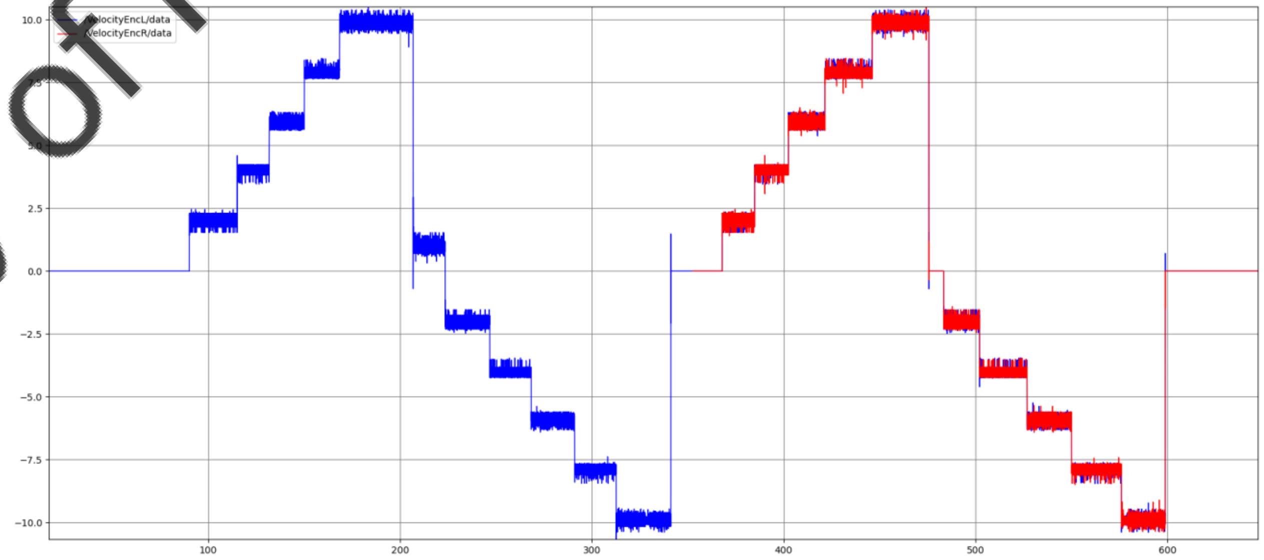
{Learn, Create, Innovate};

Manchester **Robotics**

Activity – Noise Simulation/Estimation

Objective

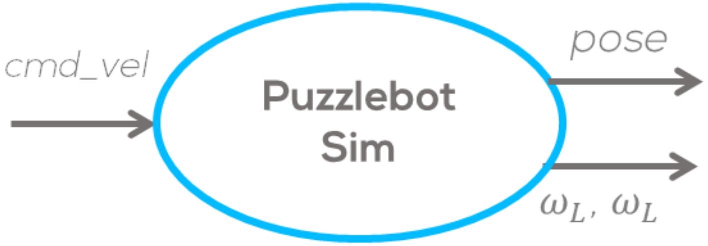
- The following image show the data received by a real Puzzlebot from the encoders.
- The objective is to simulate encoders gaussian noise on the robot's wheels.
- To this end a simple kinematic simulator of the wheels will be developed.



Activity – Noise Simulation/Estimation

Instructions

- This part of the activity consists of creating a node that simulates a kinematic model of the Puzzlebot.
- Simulate a nonholonomic robot (e.g., Puzzlebot) using ROS.
- For this activity, the “pose” is not mandatory, since we will focus on the wheels noise.

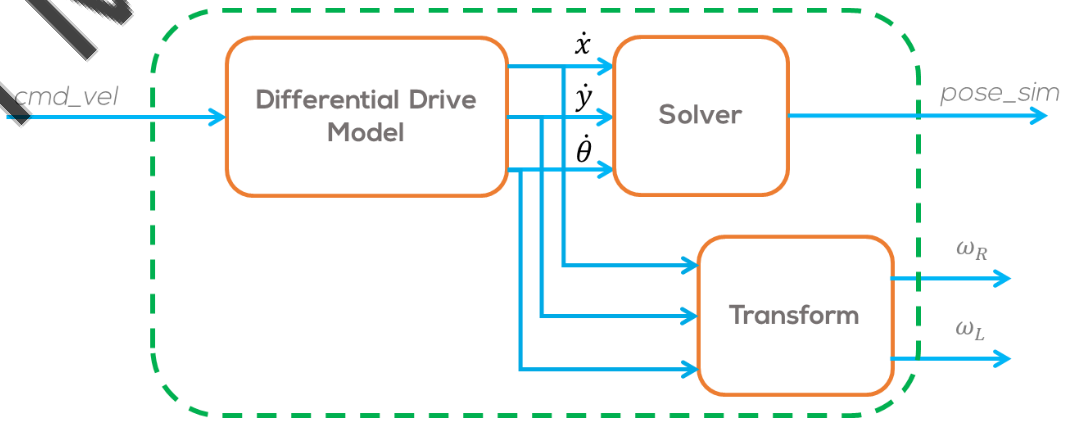


Activity – Noise Simulation/Estimation

- The robot kinematical model is given by:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases}$$

- The name of the package for the simulated node must be *"puzzlebot_sim"*.
- For the input to the robot use *"Twist"* message
    - The topic for commanding the robot must be named *"cmd_vel"*



/puzzlebot_kinematic_model (Node)

Activity – Noise Simulation/Estimation

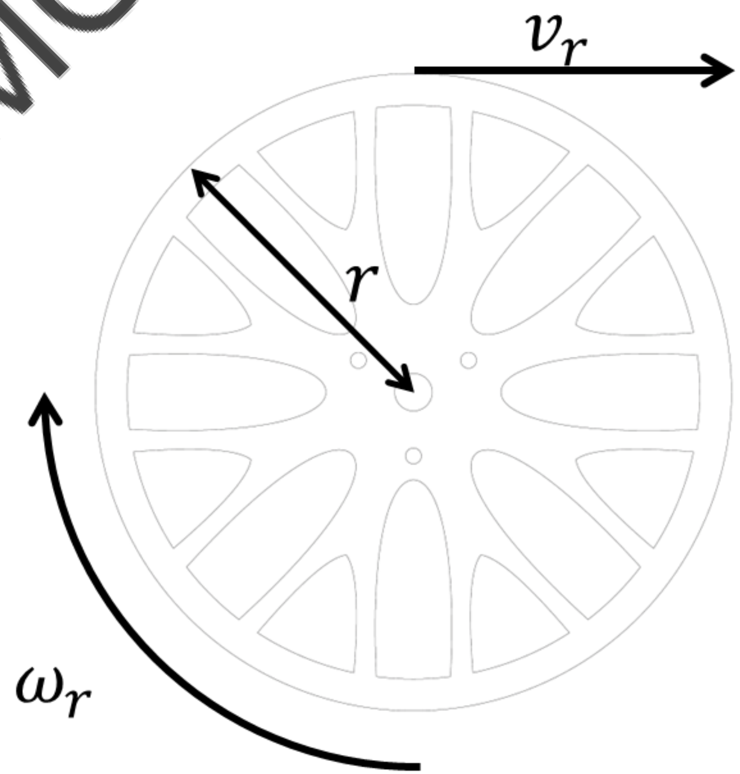
- The wheel's speed must also be published using a *"Float 32" std_msg.*
- The topics for each wheel must be *"wr"* and *"wl"*, for the left and right wheels respectively.

$$Remember:$$

$$v = \frac{v_R + v_L}{2} = r\frac{\omega_R + \omega_L}{2}$$

$$\omega = \frac{v_R - v_L}{l} = r\frac{\omega_R - \omega_L}{l}$$

- Puzzlebot parameters:
  - Radius of the wheel: 5 cm
  - Wheelbase: 19 cm



Activity – Noise Simulation/Estimation

Instructions

- Download the activity template package "puzzlebot_sim" from GitHub or create a package with the following characteristics.

```
ros2 pkg create --build-type ament_python
puzzlebot_sim --node-name puzzlebot_sim --
dependencies rclpy ros2launch python3-numpy
std_msgs geometry_msgs nav_msgs --license Apache-
2.0 --maintainer-name 'Mario Martinez' --
maintainer-email 'mario.mtz@manchester-
robotics.com'
```

Instructions

- In the package "puzzlebot_sim" open the file "puzzlebot_sim.py" on a text editor.

```
$ cd ~/ros2_ws/src/ puzzlebot_sim
$ code .    (for vscode)
```

- The wheel velocities noise will be simulated using the following formula

$$\omega = \omega + rand(0, k_r\ |\omega|)$$

- The code is shown in the nex t slide

```python
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
from std_msgs.msg import Float32
import numpy as np
import transforms3d

class KinematicModelNode(Node):
    def __init__(self):
        super().__init__('kinematic_model')

        #Set the parameters of the system
        self.x = 0.0
        self.y = 0.0
        self.theta = 0.0
        self._l = 0.19
        self._r = 0.05
        self._k_r = 0.016
        self._k_l = 0.016
        self._sample_time = 0.01

        # Velocity inputs
        self.v = 0.0  # Linear velocity (m/s)
        self.omega = 0.0  # Angular velocity (rad/s)

        #Messages to be used
        self.wr = Float32()
        self.wl = Float32()

        # Last update time
        self.last_time = self.get_clock().now()

        # ROS2 Subscribers and Publishers
        self.create_subscription(Twist, 'cmd_vel', self.cmd_vel_callback, 10)
        self.wl_pub = self.create_publisher(Float32, 'wl', 10)
        self.wr_pub = self.create_publisher(Float32, 'wr', 10)

        # Timer to update kinematics at ~100Hz
        self.timer = self.create_timer(self._sample_time, self.update_kinematics)
        self.get_logger().info("Kinematic Model Node Started.")
```

```python
    def cmd_vel_callback(self, msg):
        """ Callback to update velocity commands """
        self.v = msg.linear.x
        self.omega = msg.angular.z

    def update_kinematics(self):
        """ Updates robot position based on real elapsed time """
        # Get current time and compute dt
        current_time = self.get_clock().now()
        dt = (current_time - self.last_time).nanoseconds / 1e9  # Convert to
seconds
        self.last_time = current_time

        if dt > 0:
            # Simulate the encoders data
            omega_r = (self.v + self._l * self.omega / 2.0) / self._r
            omega_l = (self.v - self._l * self.omega / 2.0) / self._r

            #Simulate encoders with added noise
            self.wr.data = omega_r + np.random.normal(0,self._k_r *
np.abs(omega_r))
            self.wl.data = omega_l  + np.random.normal(0,self._k_l *
np.abs(omega_l))

            # Publish new state
            self.publish_wheel_speed()

    def publish_wheel_speed(self):
        self.wl_pub.publish(self.wl)
        self.wr_pub.publish(self.wr)
```

```python
def main(args=None):

    rclpy.init(args=args)

    node = KinematicModelNode()

    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        pass
    finally:
        if rclpy.ok():  # Ensure shutdown is only called once
            rclpy.shutdown()
        node.destroy_node()

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

Instructions

- Save and compile the file

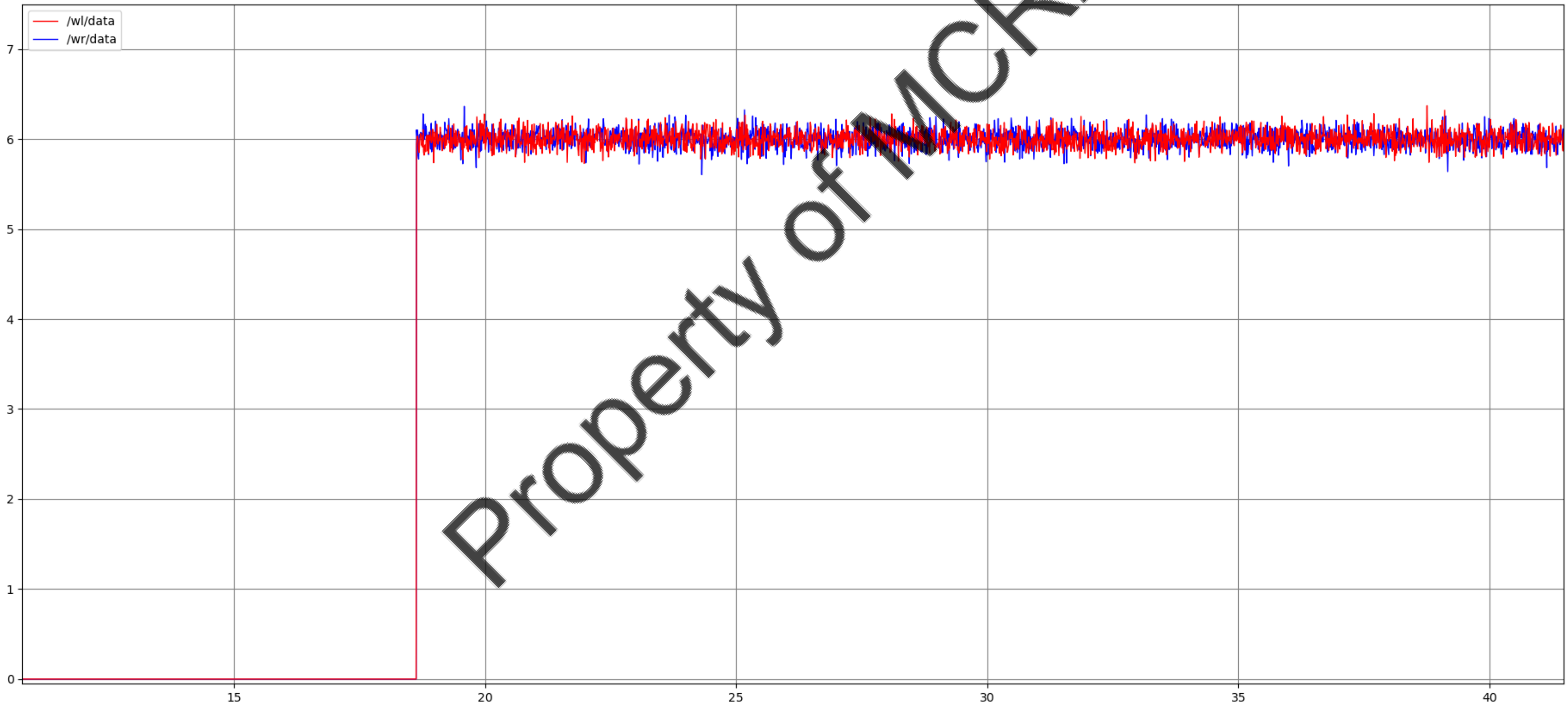
```
$ cd ~/ros2_ws
$ colcon build --packages-select puzzlebot_sim
$ source install/setup.bash
```

- Run the node

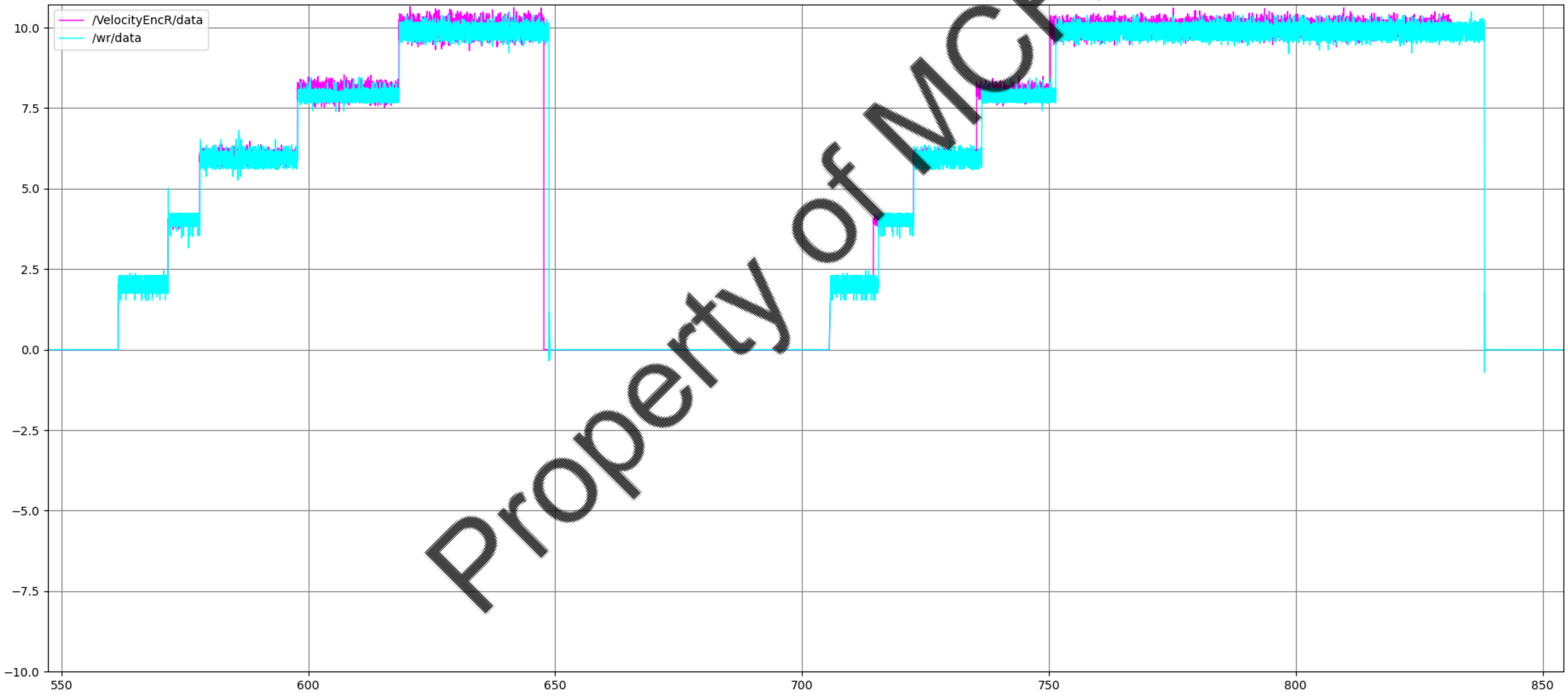
```
$ ros2 run  puzzlebot_sim puzzlebot_sim
```

- Open “rqt_plot” to view the output.
- Compare the results with the real Puzzlebot to calibrate the parameters k_r, k_l

Results



Results


/VelocityEncR/data
/wr/data
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
550
600
650
700
750
800
850