Solution Topics Quiz

Solution Quiz

Topics Quiz

Launch File: topics_quiz.launch.py

In [ ]:

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='topics_quiz',
            executable='topics_quiz_node',
            name='topics_quiz_node',
            output='screen'),
    ])
```

END Launch File: topics_quiz.launch.py

Python File: move_with_odom.py

```
In [ ]:
```

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


class TopicsQuiz(Node):

    def __init__(self):

        super().__init__('exercise31')

        self.publisher_ = self.create_publisher(Twist, 'cmd_vel', 10)
        self.subscriber = self.create_subscription(
            Odometry, '/odom', self.odom_callback, 10)

        self.timer_period = 0.2
        self.odom_x = 0
        self.odom_y = 0
        self.odom_yaw = 0
        self.motion_state = 'FORWARD_1'

        self.cmd = Twist()
        self.timer = self.create_timer(self.timer_period, self.motion)

    def euler_from_quaternion(self, quaternion):
        """
        Converts quaternion (w in last place) to euler roll, pitch, yaw
        quaternion = [x, y, z, w]
        Bellow should be replaced when porting for ROS 2 Python tf_conversions is done.
        """
        x = quaternion[0]
        y = quaternion[1]
        z = quaternion[2]
        w = quaternion[3]

        sinr_cosp = 2 * (w * x + y * z)
        cosr_cosp = 1 - 2 * (x * x + y * y)
        roll = np.arctan2(sinr_cosp, cosr_cosp)

        sinp = 2 * (w * y - z * x)
        pitch = np.arcsin(sinp)

        siny_cosp = 2 * (w * z + x * y)
        cosy_cosp = 1 - 2 * (y * y + z * z)
        yaw = np.arctan2(siny_cosp, cosy_cosp)

        return roll, pitch, yaw

    def odom_callback(self, msg):
        # Save the frontal laser scan info at 0°
        self.odom_x = msg.pose.pose.position.x
        self.odom_y = msg.pose.pose.position.y
        orientation_q = msg.pose.pose.orientation
        orientation_list = [orientation_q.x,
                            orientation_q.y, orientation_q.z, orientation_q.w]
        (roll, pitch, self.odom_yaw) = self.euler_from_quaternion(orientation_list)

    def motion(self):
        # print the data
        self.get_logger().info('Current X: "%s"' % str(self.odom_x))
        self.get_logger().info('Current Y: "%s"' % str(self.odom_y))
        self.get_logger().info('Current YAW: "%s"' % str(self.odom_yaw))
        # Logic of move
        if self.motion_state == 'FORWARD_1':
            if self.odom_x < 1.2:
                self.cmd.linear.x = 0.3
            else:
                self.motion_state = 'TURN'

        elif self.motion_state == 'TURN':
            if self.odom_yaw < 1.57:
                self.cmd.linear.x = 0.0
                self.cmd.angular.z = 0.2
            else:
                self.motion_state = 'FORWARD_2'

        if self.motion_state == 'FORWARD_2':
            if self.odom_y < 1:
                self.cmd.linear.x = 0.3
                self.cmd.angular.z = 0.0
            else:
                self.motion_state = 'END'

        if self.motion_state == 'END':
            self.cmd.linear.x = 0.0
            self.cmd.angular.z = 0.0

        self.publisher_.publish(self.cmd)


def main(args=None):

    rclpy.init(args=args)
    topics_quiz = TopicsQuiz()
    rclpy.spin(topics_quiz)
    topics_quiz.destroy_node()
    rclpy.shutdown()


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

END Python File: move_with_odom.py

Setup File: setup.py

In [ ]:

```
from setuptools import setup
import os
from glob import glob

package_name = 'topics_quiz'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launch/*.launch.py'))
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='user',
    maintainer_email='user@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'topics_quiz_node = topics_quiz.move_with_odom:main'
        ],
    },
)
```

END Setup File: setup.py

So, in the above Python script, we are generating a very simple logic:

- The robot **moves forward** until the X coordinates of the odometry is higher than 1.2 meters.
- Then, it rotates 90º (1.57 radians) to the left
- Finally, the robot **moves forward** until the Y coordinates of the odometry is higher than 1 meter.

So, at the end, you will get something like this: