Solution Services Quiz

Solution Quiz

Services Quiz

For the **services_quiz_srv** package:

Service Interface File: Turn.srv

In [ ]:

```
string direction                # Direction to turn (right or left)
float64 angular_velocity        # Angular Velocity (in rad/s)
int32 time                      # Duration of the turn (in seconds)
---
bool success                    # Did it achieve it?
```

END Service Interface File: Turn.srv

Make File: CMakeLists.txt

In [ ]:

```
cmake_minimum_required(VERSION 3.5)
project(services_quiz_srv)

# Default to C99
if(NOT CMAKE_C_STANDARD)
  set(CMAKE_C_STANDARD 99)
endif()

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(std_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)
# uncomment the following section in order to fill in
# further dependencies manually.
# find_package(<dependency> REQUIRED)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

rosidl_generate_interfaces(${PROJECT_NAME}
  "srv/Turn.srv"
)

ament_package()
```

END Make File: CMakeLists.txt

For the **services_quiz** package:

Launch File: services_quiz_server.launch.py

In [ ]:

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


def generate_launch_description():
    return LaunchDescription([
        Node(
            package='services_quiz',
            executable='services_quiz_server',
            name='services_quiz_server_node',
            output='screen'),
    ])
```

END Launch File: services_quiz_server.launch.py

Python File: server.py

In [ ]:

```
import rclpy
from rclpy.node import Node
from services_quiz_srv.srv import Turn
from geometry_msgs.msg import Twist
import time


class QuizService(Node):

    def __init__(self):

        super().__init__('service_moving')

        self.srv = self.create_service(
            Turn, 'turn', self.srv_callback)

        self.publisher_ = self.create_publisher(Twist, 'cmd_vel', 10)
        self.twist = Twist()

    def srv_callback(self, request, response):

        if request.direction == "right":
            self.twist.angular.z = -request.angular_velocity
        else:
            self.twist.angular.z = request.angular_velocity

        i = 0
        # loop to publish the velocity estimate, current_distance = velocity * (t1 - t0)
        while (i <= request.time):

            self.publisher_.publish(self.twist)
            i += 0.1
            time.sleep(0.1)

        # set velocity to zero to stop the robot
        self.twist.angular.z = 0.0
        self.publisher_.publish(self.twist)

        self.get_logger().info("Turned robot " + request.direction +
                               " for " + str(request.time) + " seconds")

        response.success = True

        return response


def main(args=None):

    rclpy.init(args=args)

    quiz_service = QuizService()

    rclpy.spin(quiz_service)

    rclpy.shutdown()


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

END Python File: server.py

Launch File: services_quiz_client.launch.py

In [ ]:

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


def generate_launch_description():
    return LaunchDescription([
        Node(
            package='services_quiz',
            executable='services_quiz_client',
            name='services_quiz_client_node',
            output='screen'),
    ])
```

END Launch File: services_quiz_client.launch.py

Python File: client.py

In [ ]:

```
import rclpy
from rclpy.node import Node
from services_quiz_srv.srv import Turn


class QuizClient(Node):

    def __init__(self):

        super().__init__('stop_client')

        self.client = self.create_client(Turn, 'turn')

        while not self.client.wait_for_service(timeout_sec=1.0):
            self.get_logger().info('service not available, waiting again...')

        self.req = Turn.Request()


    def send_request(self):

        self.req.direction = 'right'
        self.req.angular_velocity = 0.2
        self.req.time = 10
        self.future = self.client.call_async(self.req)


def main(args=None):
    rclpy.init(args=args)
    client = QuizClient()
    client.send_request()

    while rclpy.ok():

        rclpy.spin_once(client)
        if client.future.done():
            try:
                response = client.future.result()
            except Exception as e:
                client.get_logger().info(
                    'Service call failed %r' % (e,))
            else:
                if response is True:
                    client.get_logger().info(
                    'Robot turned correctly' )
                else:
                    client.get_logger().info(
                    'Something went wrong' )
            break

    client.destroy_node()
    rclpy.shutdown()


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

END Python File: client.py

Setup File: setup.py

In [ ]:

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

package_name = 'services_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': [
            'services_quiz_server = services_quiz.server:main',
            'services_quiz_client = services_quiz.client:main'
        ],
    },
)
```

END Setup File: setup.py

Execute in Shell #1

In [ ]:

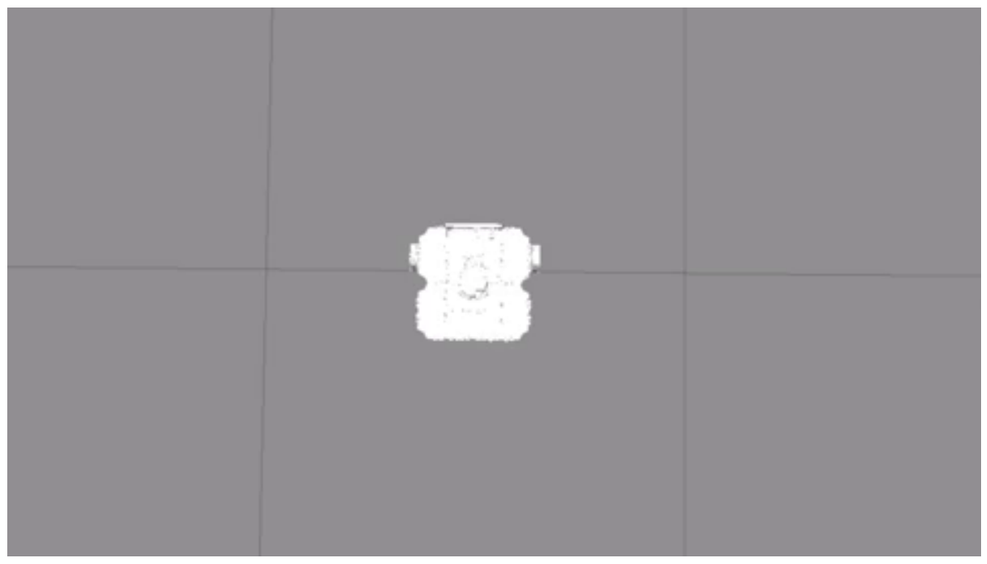
```
ros2 launch services_quiz services_quiz_server.launch.py
```

Execute in Shell #2

In [ ]:

```
ros2 launch services_quiz services_quiz_client.launch.py
```

You will get something like this:



In [ ]: