SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE
DEPARTMENT OF ROBOTICS AND AUTOMATION ENGINEERING

ACADEMIC YEAR: 202425 SEM: V

ASSIGNMENT NO: 7

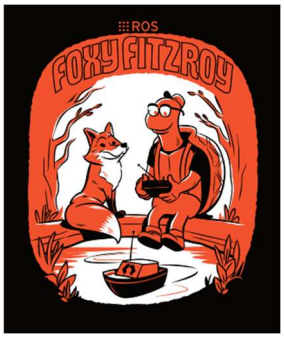
NAME:Akshaj Chainani
PRN: 22070127008
BATCH: RA1

Aim:

1. **Integration of Webots with ROS2.**
2. **Operating a simple wheeled robot in Webots via ROS2.**

Apparatus:

- **Software:**
  - **Oracle VirtualBox**
  - **Linux-based operating system (Ubuntu 20.04)**
  - **ROS2 Foxy Fitzroy**
  - **Webots**
- **Hardware:**
  - **Computer system with required specifications for ROS2**



Theory:

The integration of ROS2 with Webots combines the powerful simulation capabilities of Webots with the flexible architecture of ROS2, allowing developers to create and test robotic applications efficiently. This synergy enables real-time data exchange between simulated robots and ROS2 nodes, facilitating control and monitoring. By conducting experiments in a safe virtual environment, developers can refine algorithms and behaviors, ultimately enhancing the transition from simulation to real-world deployment in robotics.

- **Create the package structure:**
  - **Organize the code in a custom ROS 2 package. Create a new package named "my_package019" from the "src" folder of your ROS 2 workspace. Change the current directory of your terminal to "ros2_ws/src".**
  - **cd my_package019**
    mkdir launch
    mkdir worlds
- **Setup the simulation world:**
  - **A world file containing a robot to launch your simulation and move it inside "my_package019/worlds/".**
  - **A simple robot is already included in this "my_world.wbt" world file.**
- **Edit the "my_robot_driver" plugin:**

- The webots_ros2_driver sub-package automatically creates a ROS 2 interface for most sensors.
- Extend this interface by creating your own custom plugin. This custom plugin is a ROS node equivalent to a robot controller. You can use it to access the Webots robot API and create your own topics and services to control your robot.

- Create the "my_robot.urdf" file:
  - Create a URDF file to declare the MyRobotDriver plugin. This will allow the webots_ros2_driver ROS node to launch the plugin and connect it to the target robot.
  - In the my_package019/resource folder create a text file named "my_robot.urdf".
- Create the launch file:
  - Create the launch file to easily launch the simulation and the ROS controller with a single command.
  - In the my_package019/launch folder create a new text file named "robot_launch.py".

- Edit additional files:
  - Before you can start the launch file, you have to modify the "setup.py" file to include the extra files you added.
  - Open my_package019/setup.py and replace its content.
- Test the code:
  - From a terminal in your ROS 2 workspace run:
    colcon build
    source install/local_setup.bash
    ros2 launch my_package019 robot_launch.py
  - This will launch the simulation.
  - Then, open a second terminal and send a command with:
    ros2 topic pub /cmd_vel geometry_msgs/Twist "linear: { x: 0.1 }"
  - The robot is now moving forward.

Snapshots:

setup.py

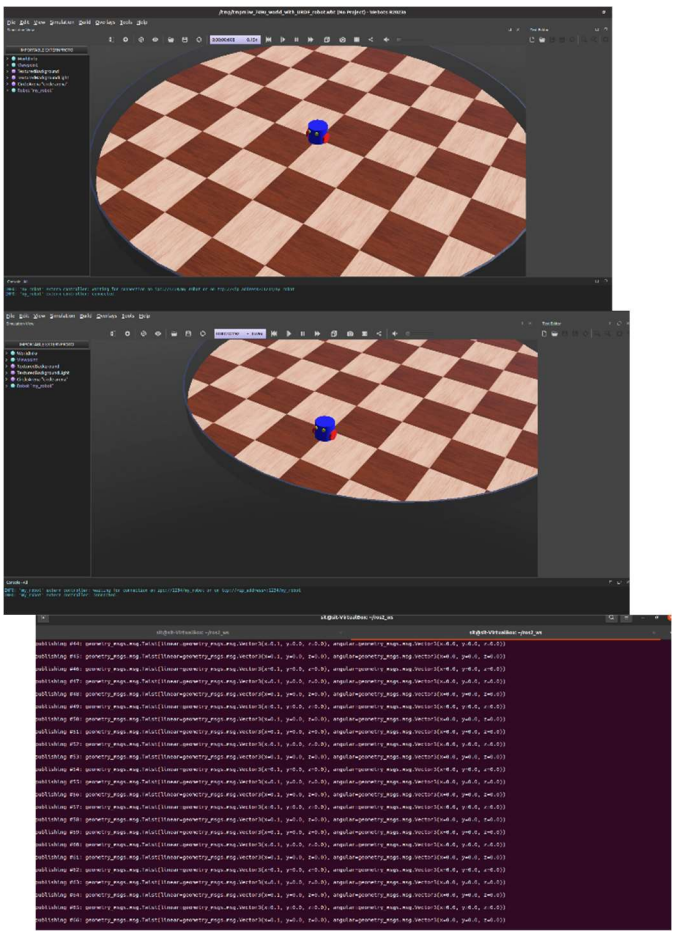
```python
from setuptools import setup

package_name = 'my_package019'
data_files = []
data_files.append(('share/ament_index/resource_index/packages', ['resource/' + package_name]))
data_files.append(('share/' + package_name + '/launch', ['launch/robot_launch.py']))
data_files.append(('share/' + package_name + '/worlds', ['worlds/my_world.wbt']))
data_files.append(('share/' + package_name + '/resource', ['resource/my_robot.urdf']))
data_files.append(('share/' + package_name, ['package.xml']))

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=data_files,
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='user',
    maintainer_email='user.name@mail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'my_robot_driver = my_package019.my_robot_driver:main',
        ],
    },
)
```

setup.py

```python
import os
import pathlib
import launch
from launch_ros.actions import Node
from launch import LaunchDescription
from ament_index_python.packages import get_package_share_directory
from webots_ros2_driver.webots_launcher import WebotsLauncher
from webots_ros2_driver.utils import controller_url_prefix

def generate_launch_description():
    package_dir = get_package_share_directory('my_package019')
    robot_description = pathlib.Path(os.path.join(package_dir, 'resource', 'my_robot.urdf')).read_text()

    webots = WebotsLauncher(
        world=os.path.join(package_dir, 'worlds', 'my_world.wbt')
    )

    my_robot_driver = Node(
        package='webots_ros2_driver',
        executable='driver',
        output='screen',
        additional_env={'WEBOTS_CONTROLLER_URL': controller_url_prefix() + 'my_robot'},
        parameters=[
            {'robot_description': robot_description},
        ]
    )

    return LaunchDescription([
        webots,
        my_robot_driver,
        launch.actions.RegisterEventHandler(
            event_handler=launch.event_handlers.OnProcessExit(
                target_action=webots,
                on_exit=[launch.actions.EmitEvent(event=launch.events.Shutdown())],
            )
        )
    ])
```

setup.py | my_robot.urdf

```xml
<?xml version="1.0" ?>
<robot name="My robot">
    <webots>
        <plugin type="my_package019.my_robot_driver.MyRobotDriver" />
    </webots>
</robot>
```



Learning Outcomes:

- **Package Structure and Organization:**

Students will learn how to create and organize a custom ROS2 package, including setting up essential directories such as launch and worlds, and managing the package structure for effective code organization.

- **Simulation Setup and Customization:**
  Learners will gain practical skills in setting up a simulation world in Webots, editing robot drivers, and creating custom plugins that interface with the Webots robot API, allowing for the control and monitoring of robot behavior in a virtual environment.
- **Testing and Launching Simulations:**
  Students will understand the process of building and launching the ROS2 package, utilizing terminal commands to test the code and control the robot's movement in the simulation, thereby enhancing their ability to transition from simulation to real-world robotic applications.

minal and graphical tools.