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

ACADEMIC YEAR: 202425 SEM: V

ASSIGNMENT NO: 8

NAME:Akshaj Chainani
PRN: 22070127008
BATCH: RA1

Experiment 8: Controlling Epuck Robot in Webots via ROS

Aim:
To control the Epuck robot in the Webots simulation environment using ROS2 and explore its capabilities through various sensor and actuator interfaces.

Apparatus:
Computer with Webots simulation software
ROS2 (Robot Operating System 2) installed
Epuck ROS2 API (`webots_ros2_epuck` package for simulated Epuck robot)

Brief Theory:

Epuck Robot Overview:
The Epuck is a small, versatile robot designed for educational purposes. Equipped with multiple sensors (infrared, light, ToF, camera, IMU, ground sensors) and actuators (motors, LEDs), it is wellsuited for control experiments and SLAM (Simultaneous Localization and Mapping) in the ROS2 ecosystem.

ROS2 Integration with Webots:
The Webots ROS2 package provides an interface for simulating Epuck's sensors and actuators, allowing commands and data to flow between Webots and ROS2. This experiment will use topics such as `/cmd_vel` for velocity control, `/odom` for odometry, and `/scan` for sensor data, as well as visualizing data in RViz.

Basic Commands:

1. Launching the Epuck in Webots
   Launching Webots with ROS2 Epuck:
   ```bash
   ros2 launch webots_ros2_epuck robot_launch.py
   ```
   Launch with RViz visualization:
   ```bash
   ros2 launch webots_ros2_epuck robot_launch.py rviz:=true
   ```

2. Sensors and Data Acquisition:
   Infrared Sensors (Proximity):
```bash
ros2 topic echo /ps1
```
   Distance Measurement (ToF sensor):
```bash
ros2 topic echo /scan
```
   Light Sensors:
```bash
ros2 topic echo /ls1
```

3. Controlling LEDs:
   Binary LED Control:
```bash
ros2 topic pub /led0 std_msgs/Int32 '{ "data": 1 }'
```
   RGB LED Control:
```bash
ros2 topic pub /led1 std_msgs/Int32 '{ "data": 0xFF0000 }'
```

4. Velocity Control (Differential Drive):
   Moving forward:
```bash
ros2 topic pub /cmd_vel geometry_msgs/Twist "linear:
  x: 0.1
  angular:
  z: 0.0"
```
   Turning:
```bash
ros2 topic pub /cmd_vel geometry_msgs/Twist "linear:
  x: 0.0
  angular:
  z: 0.5"
```

5. Odometry and Position Tracking:
   Accessing odometry data:
```bash
ros2 topic echo /odom
```

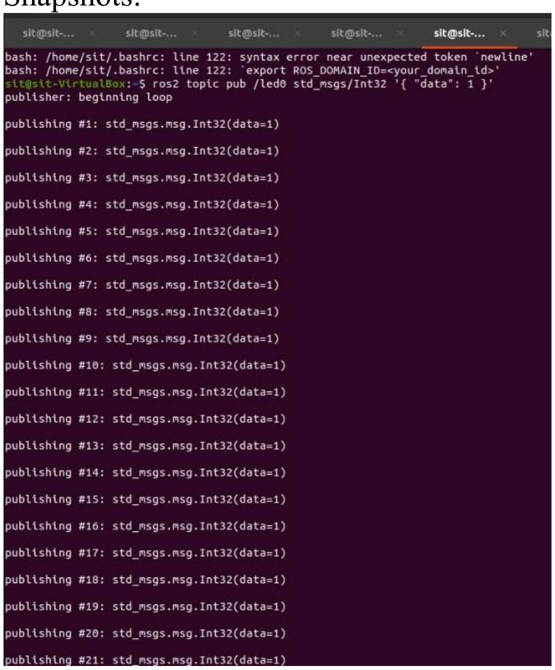
6. Camera Visualization:
   Display camera data in RQT:
```bash
```

ros2 run rqt_image_view rqt_image_view
```
Select `/camera` for viewing.

7. IMU Sensor Data:
Accessing accelerometer and gyroscope data:
```bash
ros2 topic echo /imu
```

8. Navigation and Map Saving:
Set goal position for navigation:
```bash
ros2 topic pub 1 /goal_pose geometry_msgs/PoseStamped "pose: {position: {x: 0.3, y: 0.0}, orientation: {w: 1.0}}"
```
Save generated map:
```bash
ros2 run nav2_map_server map_saver_cli f $HOME/Pictures/map
```

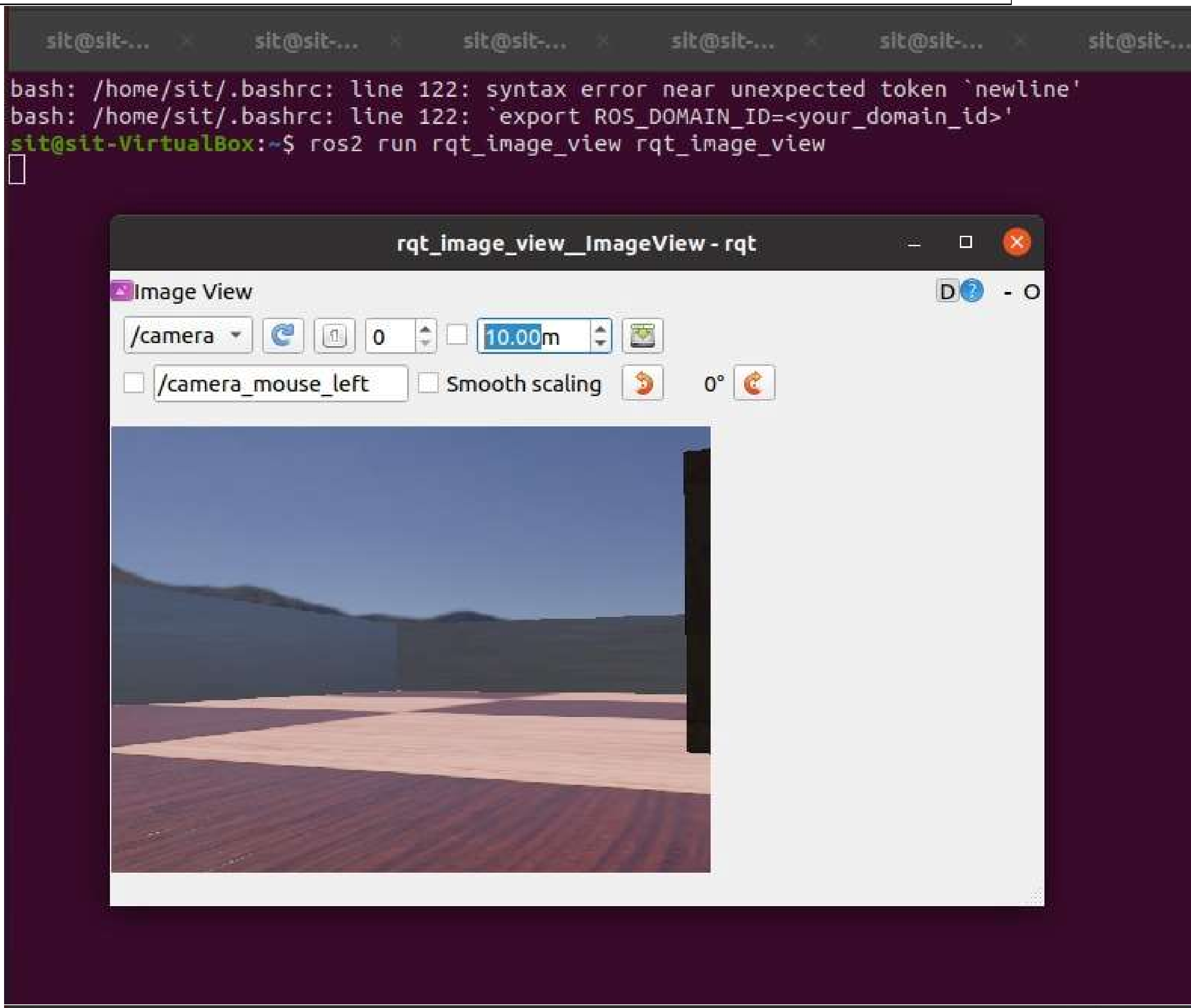
Snapshots:

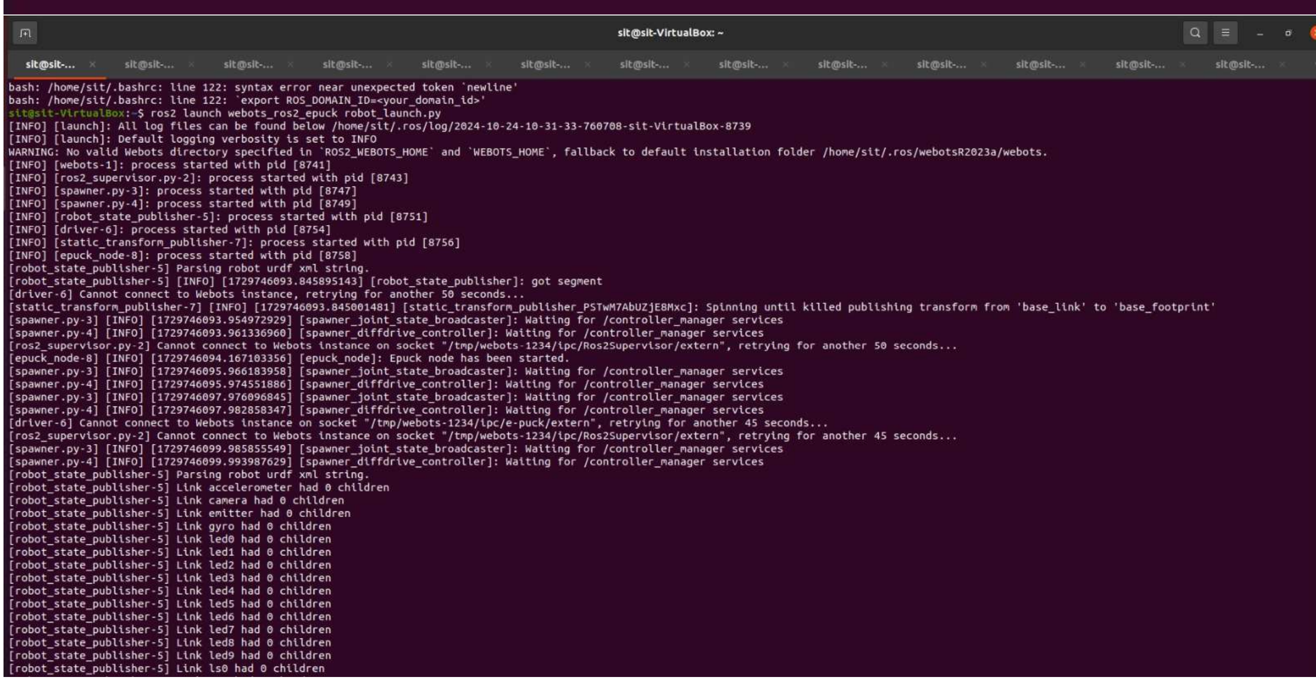

```

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

॥वसुधैव कुटुम्बकम्॥

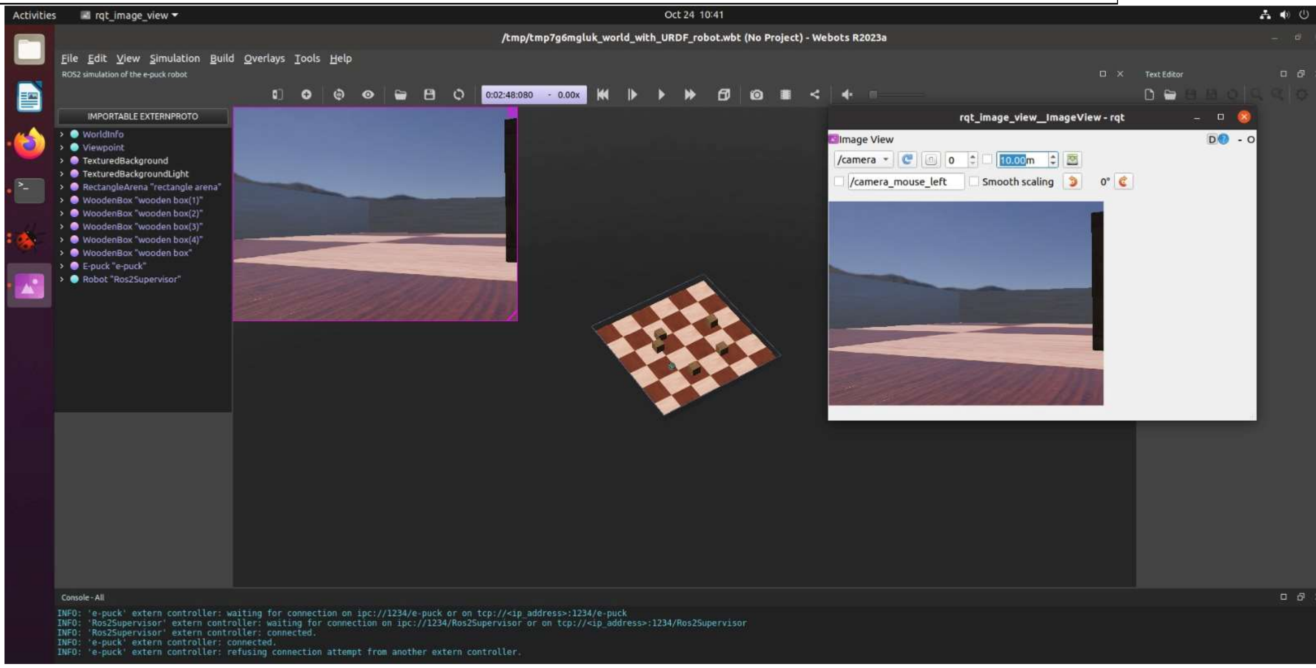
ACADEMIC YEAR: 202425 SEM: V





```
    nanosec: 820000000
  frame_id: ps1
radiation_type: 1
field_of_view: 0.10000000149011612
min_range: 0.0
max_range: 0.06657209992408752
range: 0.06909874081611633
---
header:
  stamp:
    sec: 167
    nanosec: 840000000
  frame_id: ps1
radiation_type: 1
field_of_view: 0.10000000149011612
min_range: 0.0
max_range: 0.06657209992408752
range: 0.0699426606297493
---
header:
  stamp:
    sec: 167
    nanosec: 860000000
  frame_id: ps1
radiation_type: 1
field_of_view: 0.10000000149011612
min_range: 0.0
max_range: 0.06657209992408752
range: 0.06959757208824158
---
header:
  stamp:
    sec: 167
    nanosec: 880000000
  frame_id: ps1
radiation_type: 1
field_of_view: 0.10000000149011612
min_range: 0.0
max_range: 0.06657209992408752
range: 0.07021138817071915
---
header:
  stamp:
    sec: 167
    nanosec: 900000000
  frame_id: ps1
```

```
publishing #709: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #710: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #711: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #712: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #713: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #714: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #715: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #716: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #717: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #718: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #719: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #720: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #721: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #722: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #723: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #724: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #725: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #726: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #727: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #728: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #729: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #730: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
publishing #731: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.1, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
```



Learning Outcomes:
 Understand the ROS2 framework for controlling the Epuck robot in Webots.
 Gain experience in using ROS2 commands to interact with Epuck sensors and actuators.
 Learn to visualize data from sensors such as infrared, camera, and IMU in RViz.
 Understand basic navigation techniques and mapping using ROS2.