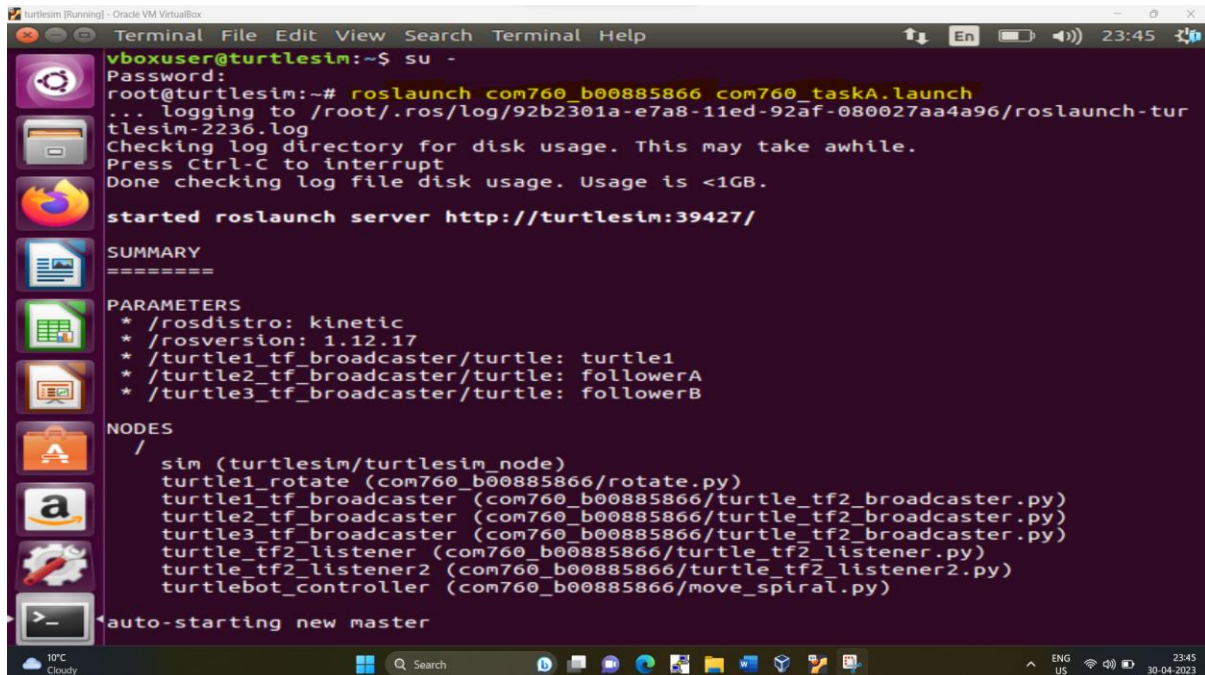


Evidence of successful implementation of turtles where leader is followed by 2 followers in turtlesim simulator ROS Kinetic.

A screenshot after running the following launch file.

`$roslaunch com760_yourBcode com760_taskA.launch`



```
vboxuser@turtlesim:~$ su -
Password:
root@turtlesim:~# roslaunch com760_b00885866 com760_taskA.launch
... logging to /root/.ros/log/92b2301a-e7a8-11ed-92af-080027aa4a96/roslaunch-tur
tlesim-2236.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://turtlesim:39427/

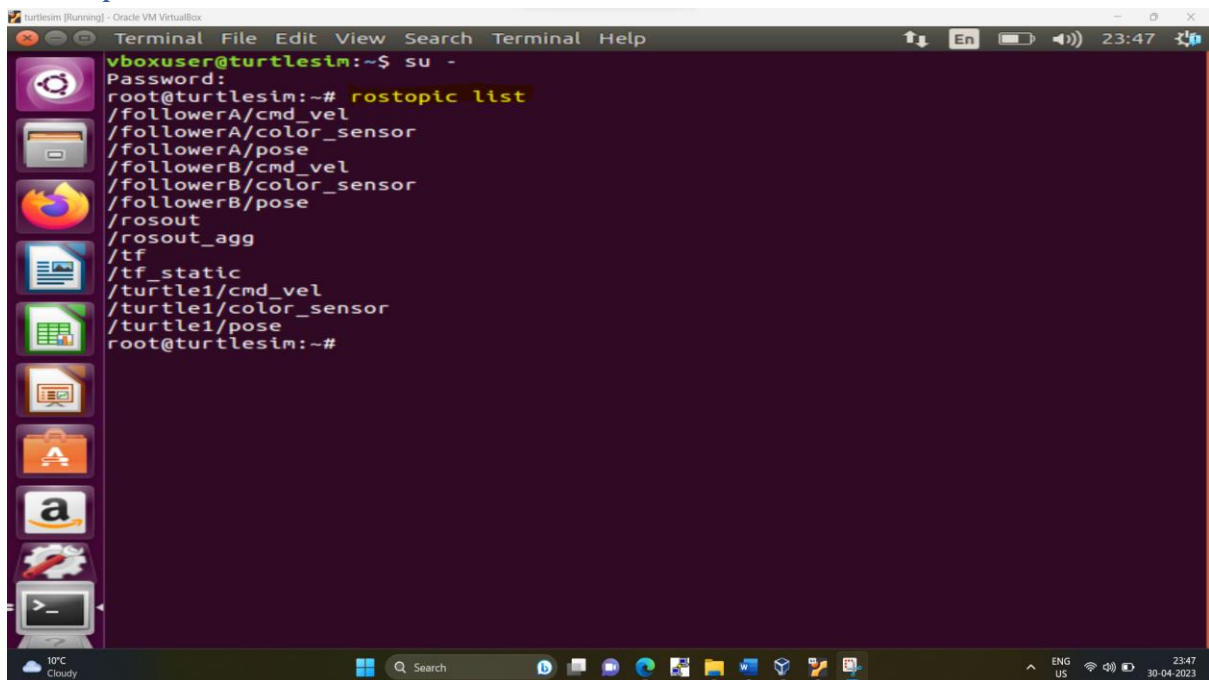
SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.17
* /turtle1_tf_broadcaster/turtle: turtle1
* /turtle2_tf_broadcaster/turtle: followerA
* /turtle3_tf_broadcaster/turtle: followerB

NODES
/
  sim (turtlesim/turtlesim_node)
  turtle1_rotate (com760_b00885866/rotate.py)
  turtle1_tf_broadcaster (com760_b00885866/turtle_tf2_broadcaster.py)
  turtle2_tf_broadcaster (com760_b00885866/turtle_tf2_broadcaster.py)
  turtle3_tf_broadcaster (com760_b00885866/turtle_tf2_broadcaster.py)
  turtle_tf2_listener (com760_b00885866/turtle_tf2_listener.py)
  turtle_tf2_listener2 (com760_b00885866/turtle_tf2_listener2.py)
  turtlebot_controller (com760_b00885866/move_spiral.py)

auto-starting new master
```

A screenshot after running the following command in a separate terminal, in which the topics are published.

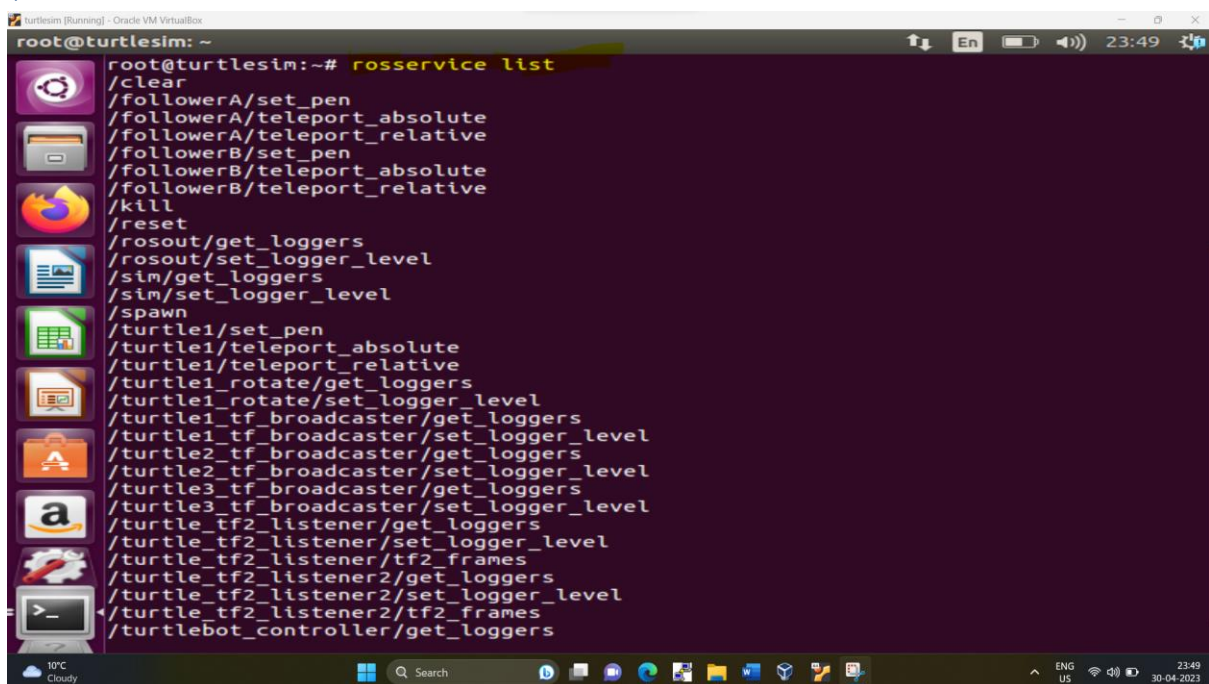
\$rostopic list



```
vboxuser@turtlesim:~$ su -
Password:
root@turtlesim:~# rostopic list
/followerA/cmd_vel
/followerA/color_sensor
/followerA/pose
/followerB/cmd_vel
/followerB/color_sensor
/followerB/pose
/rosout
/rosout_agg
/tf
/tf_static
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
root@turtlesim:~#
```

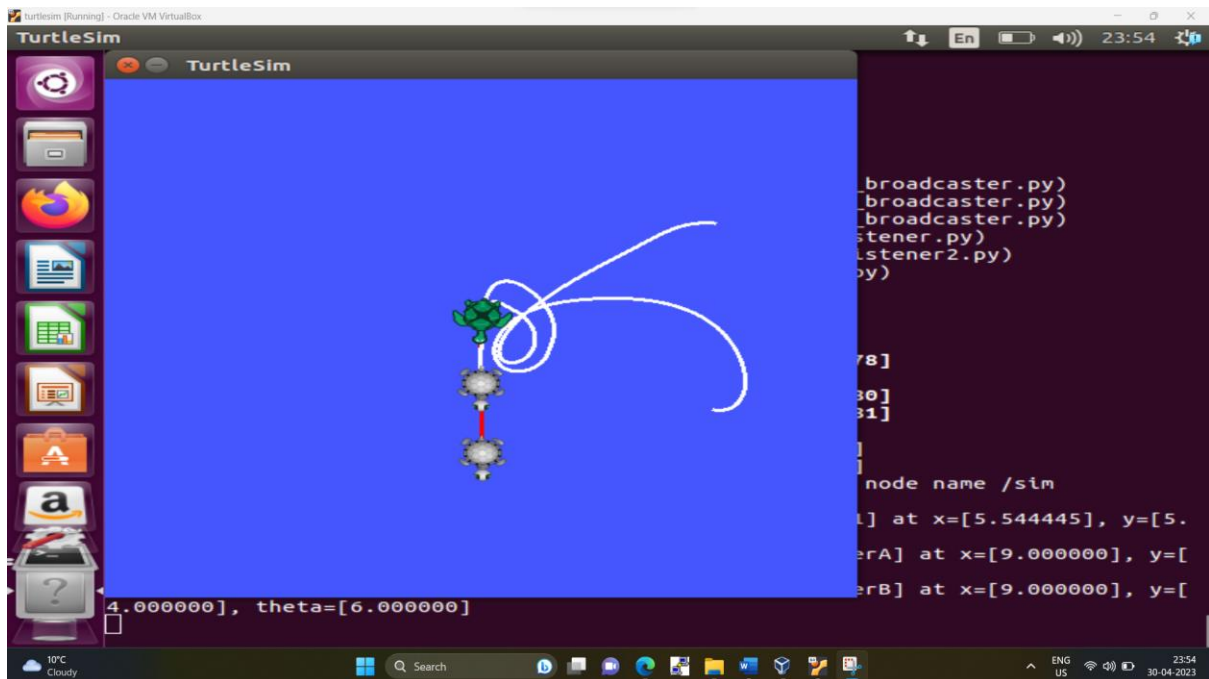
A screenshot containing the services advertised by robot after running the following command in a separate terminal.

\$rosservice list

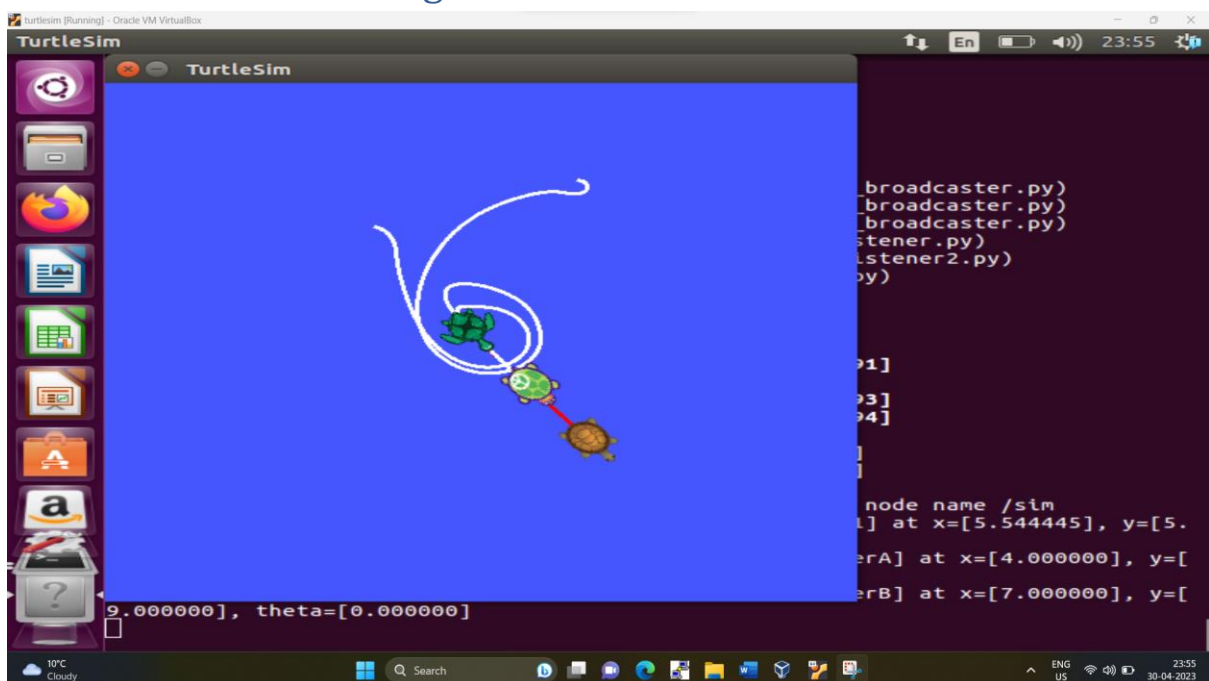


```
root@turtlesim:~# rosservice list
/clear
/followerA/set_pen
/followerA/teleport_absolute
/followerA/teleport_relative
/followerB/set_pen
/followerB/teleport_absolute
/followerB/teleport_relative
/kill
/reset
/rosout/get_loggers
/rosout/set_logger_level
/sim/get_loggers
/sim/set_logger_level
/spawn
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtle1_rotate/get_loggers
/turtle1_rotate/set_logger_level
/turtle1_tf_broadcaster/get_loggers
/turtle1_tf_broadcaster/set_logger_level
/turtle2_tf_broadcaster/get_loggers
/turtle2_tf_broadcaster/set_logger_level
/turtle3_tf_broadcaster/get_loggers
/turtle3_tf_broadcaster/set_logger_level
/turtle_tf2_listener/get_loggers
/turtle_tf2_listener/set_logger_level
/turtle_tf2_listener/tf2_frames
/turtle_tf2_listener2/get_loggers
/turtle_tf2_listener2/set_logger_level
/turtle_tf2_listener2/tf2_frames
/turtlebot_controller/get_loggers
```

A screenshot showing two followers are in the required formation position.



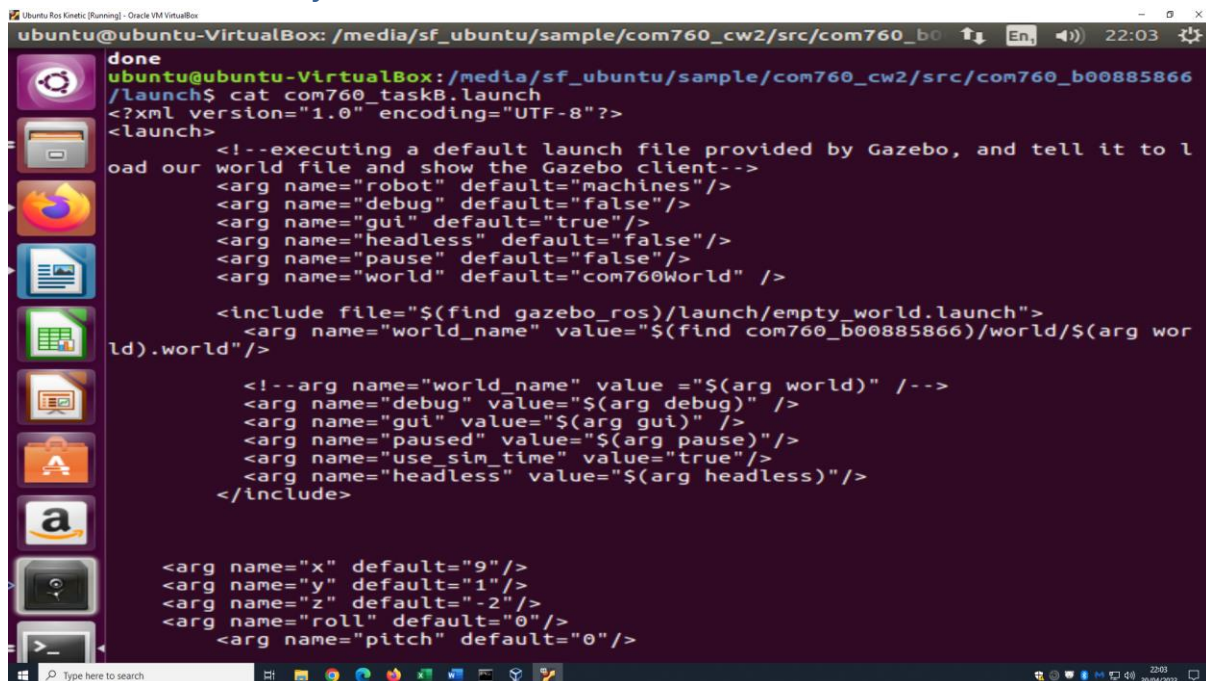
A screenshot showing the leader turtle moves in a random direction with a red pen and the three turtles maintain column formation while moving.



Evidence of successful implementation of robot moving in an environment with the help of laser and reaches docking station by avoiding obstacles in ROS Gazebo.

A screenshot after running the following launch file.

`$roslaunch com760_yourBcode com760_taskB.launch`



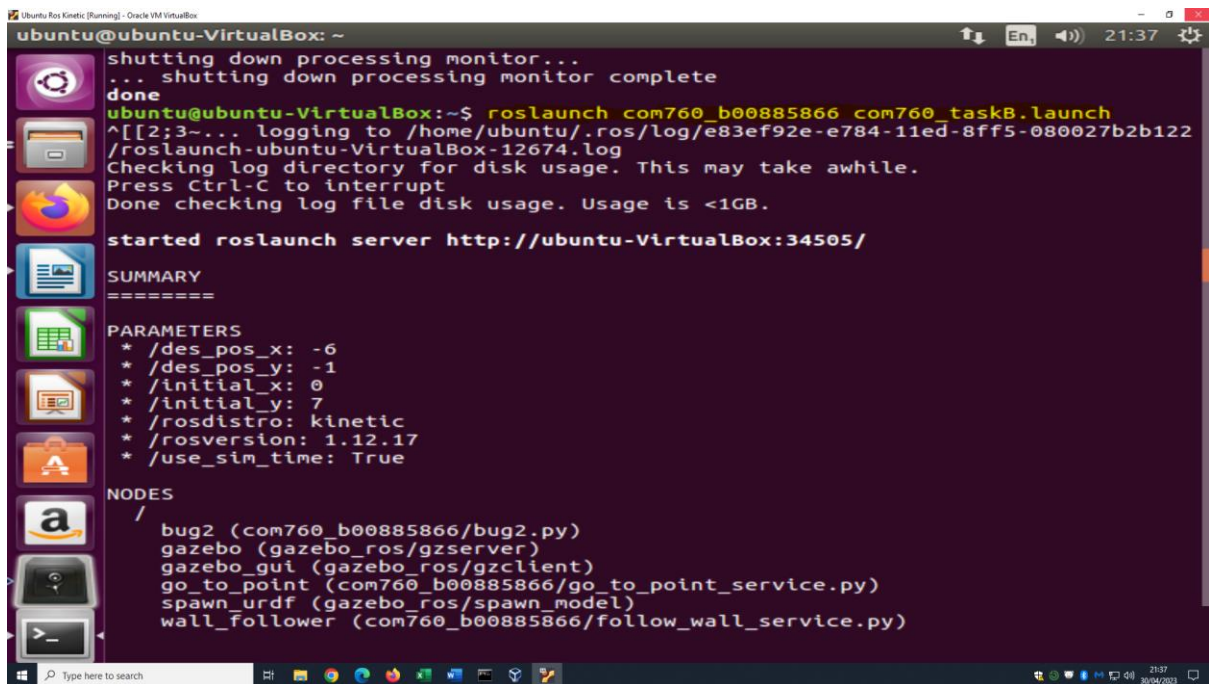
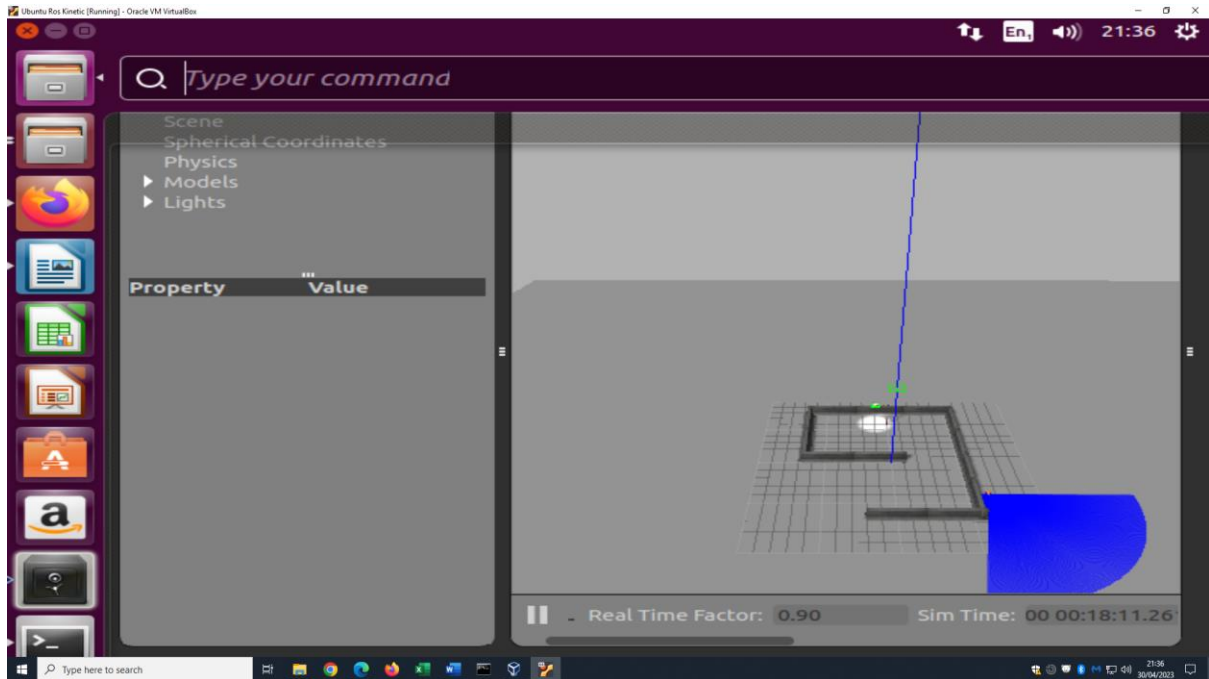
The screenshot shows a terminal window titled 'ubuntu@ubuntu-VirtualBox: /media/sf_ubuntu/sample/com760_cw2/src/com760_b00885866'. The terminal displays the command `cat com760_taskB.launch` and the contents of the launch file. The launch file is an XML document that includes a Gazebo world and sets various arguments for the simulation. The arguments include robot name, debug mode, GUI, headless mode, pause, world name, and initial position/orientation (x, y, z, roll, pitch).

```
done
ubuntu@ubuntu-VirtualBox: /media/sf_ubuntu/sample/com760_cw2/src/com760_b00885866
/launch$ cat com760_taskB.launch
<?xml version="1.0" encoding="UTF-8"?>
<launch>
  <!--executing a default launch file provided by Gazebo, and tell it to l
oad our world file and show the Gazebo client-->
  <arg name="robot" default="machines"/>
  <arg name="debug" default="false"/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="false"/>
  <arg name="pause" default="false"/>
  <arg name="world" default="com760World" />

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find com760_b00885866)/world/$(arg wor
ld).world"/>

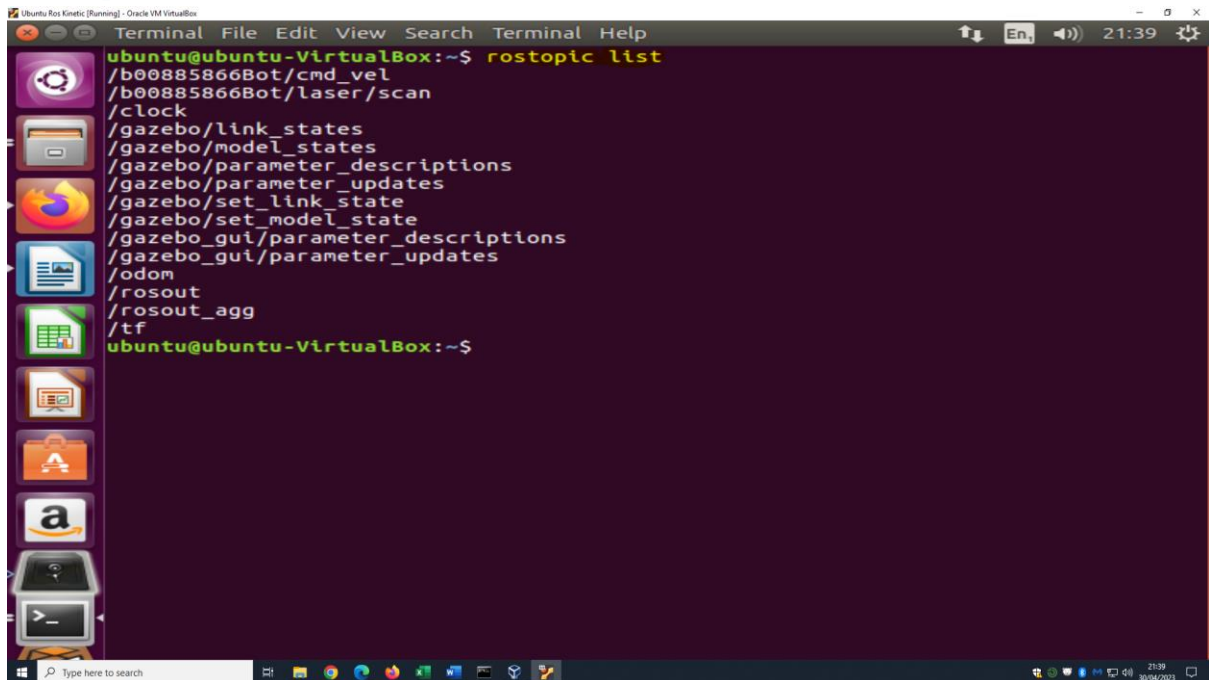
    <!--arg name="world_name" value ="$(arg world)" /-->
    <arg name="debug" value="$(arg debug)" />
    <arg name="gui" value="$(arg gui)" />
    <arg name="paused" value="$(arg pause)" />
    <arg name="use_sim_time" value="true"/>
    <arg name="headless" value="$(arg headless)" />
  </include>

  <arg name="x" default="9"/>
  <arg name="y" default="1"/>
  <arg name="z" default="-2"/>
  <arg name="roll" default="0"/>
  <arg name="pitch" default="0"/>
</launch>
```

A screenshot after running the following command in a separate terminal, in which the topics published by robot.

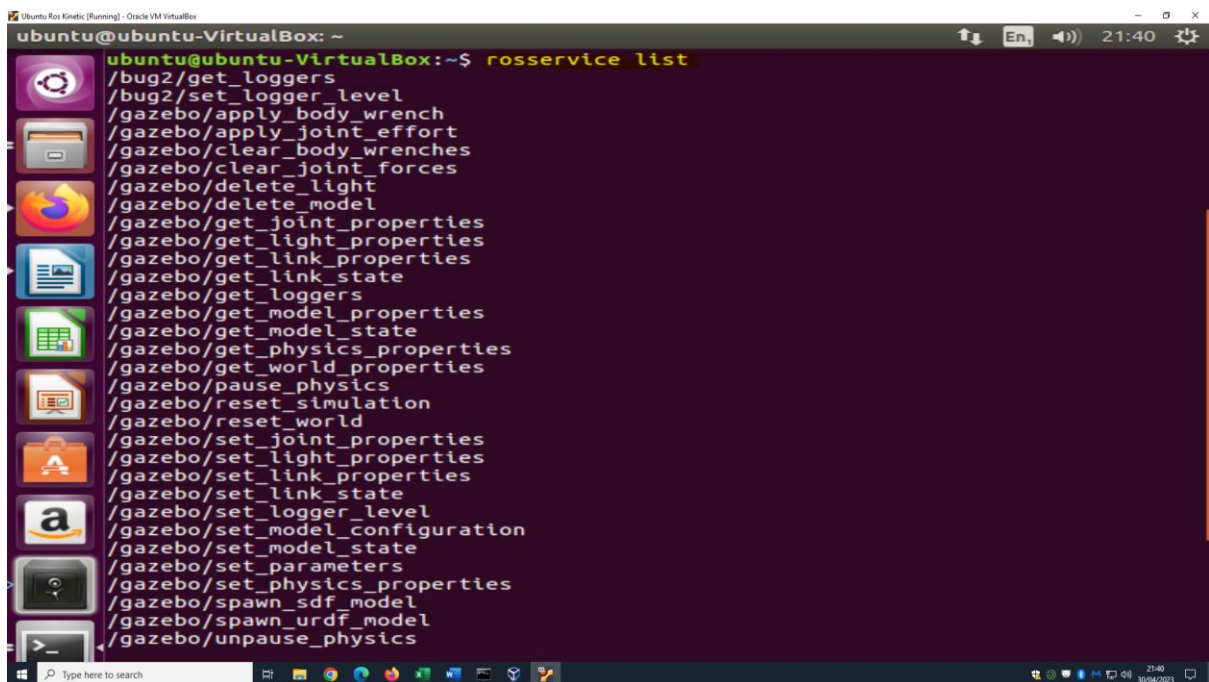
\$rostopic list

A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (21:39, 30/04/2023). The prompt is 'ubuntu@ubuntu-VirtualBox:~\$'. The command 'rostopic list' has been executed, displaying a list of ROS topics published by the robot. The topics are: /b00885866Bot/cmd_vel, /b00885866Bot/laser/scan, /clock, /gazebo/link_states, /gazebo/model_states, /gazebo/parameter_descriptions, /gazebo/parameter_updates, /gazebo/set_link_state, /gazebo/set_model_state, /gazebo_gui/parameter_descriptions, /gazebo_gui/parameter_updates, /odom, /rosout, /rosout_agg, and /tf. The prompt returns to 'ubuntu@ubuntu-VirtualBox:~\$'.

```
ubuntu@ubuntu-VirtualBox:~$ rostopic list
/b00885866Bot/cmd_vel
/b00885866Bot/laser/scan
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/gazebo_gui/parameter_descriptions
/gazebo_gui/parameter_updates
/odom
/rosout
/rosout_agg
/tf
ubuntu@ubuntu-VirtualBox:~$
```

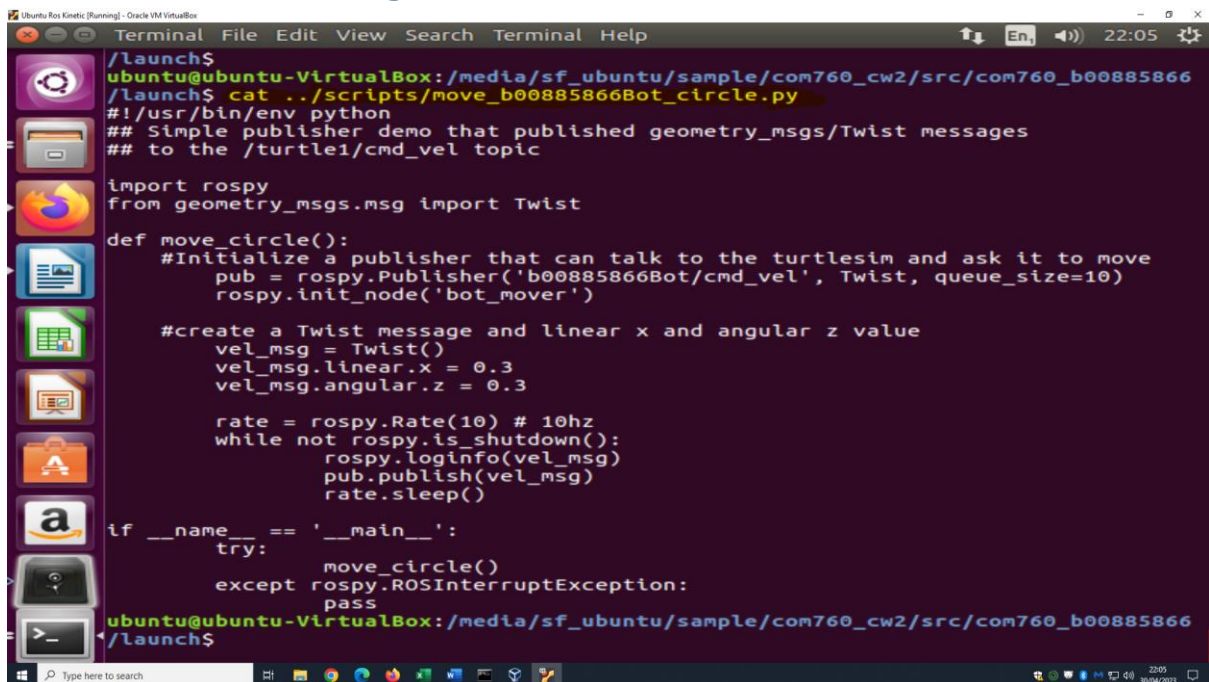
A screenshot containing the services defined for robot after running the following command in a separate terminal.

\$rosservice list

A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (21:40, 30/04/2023). The prompt is 'ubuntu@ubuntu-VirtualBox:~\$'. The command 'rosservice list' has been executed, displaying a list of ROS services defined for the robot. The services are: /bug2/get_loggers, /bug2/set_logger_level, /gazebo/apply_body_wrench, /gazebo/apply_joint_effort, /gazebo/clear_body_wrenches, /gazebo/clear_joint_forces, /gazebo/delete_light, /gazebo/delete_model, /gazebo/get_joint_properties, /gazebo/get_light_properties, /gazebo/get_link_properties, /gazebo/get_link_state, /gazebo/get_loggers, /gazebo/get_model_properties, /gazebo/get_model_state, /gazebo/get_physics_properties, /gazebo/get_world_properties, /gazebo/pause_physics, /gazebo/reset_simulation, /gazebo/reset_world, /gazebo/set_joint_properties, /gazebo/set_light_properties, /gazebo/set_link_properties, /gazebo/set_link_state, /gazebo/set_logger_level, /gazebo/set_model_configuration, /gazebo/set_model_state, /gazebo/set_parameters, /gazebo/set_physics_properties, /gazebo/spawn_sdf_model, /gazebo/spawn_urdf_model, and /gazebo/unpause_physics. The prompt returns to 'ubuntu@ubuntu-VirtualBox:~\$'.

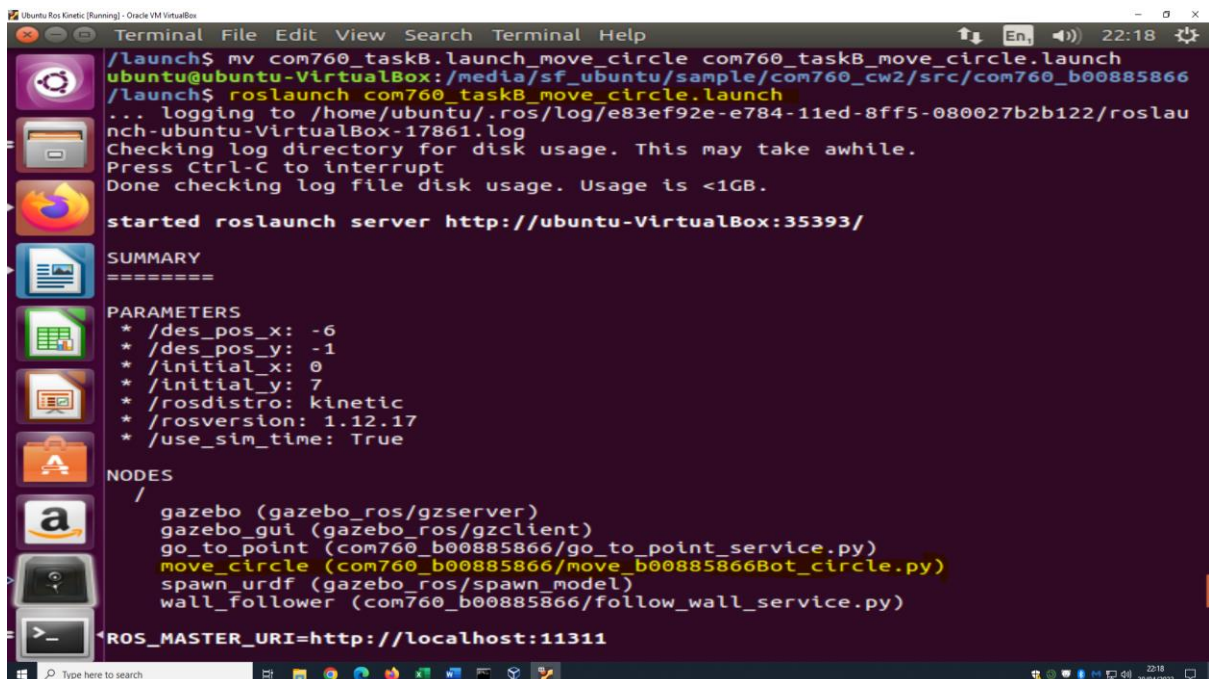
```
ubuntu@ubuntu-VirtualBox:~$ rosservice list
/bug2/get_loggers
/bug2/set_logger_level
/gazebo/apply_body_wrench
/gazebo/apply_joint_effort
/gazebo/clear_body_wrenches
/gazebo/clear_joint_forces
/gazebo/delete_light
/gazebo/delete_model
/gazebo/get_joint_properties
/gazebo/get_light_properties
/gazebo/get_link_properties
/gazebo/get_link_state
/gazebo/get_loggers
/gazebo/get_model_properties
/gazebo/get_model_state
/gazebo/get_physics_properties
/gazebo/get_world_properties
/gazebo/pause_physics
/gazebo/reset_simulation
/gazebo/reset_world
/gazebo/set_joint_properties
/gazebo/set_light_properties
/gazebo/set_link_properties
/gazebo/set_link_state
/gazebo/set_logger_level
/gazebo/set_model_configuration
/gazebo/set_model_state
/gazebo/set_parameters
/gazebo/set_physics_properties
/gazebo/spawn_sdf_model
/gazebo/spawn_urdf_model
/gazebo/unpause_physics
ubuntu@ubuntu-VirtualBox:~$
```

A screenshot showing the robot moves in a circle in Gazebo.



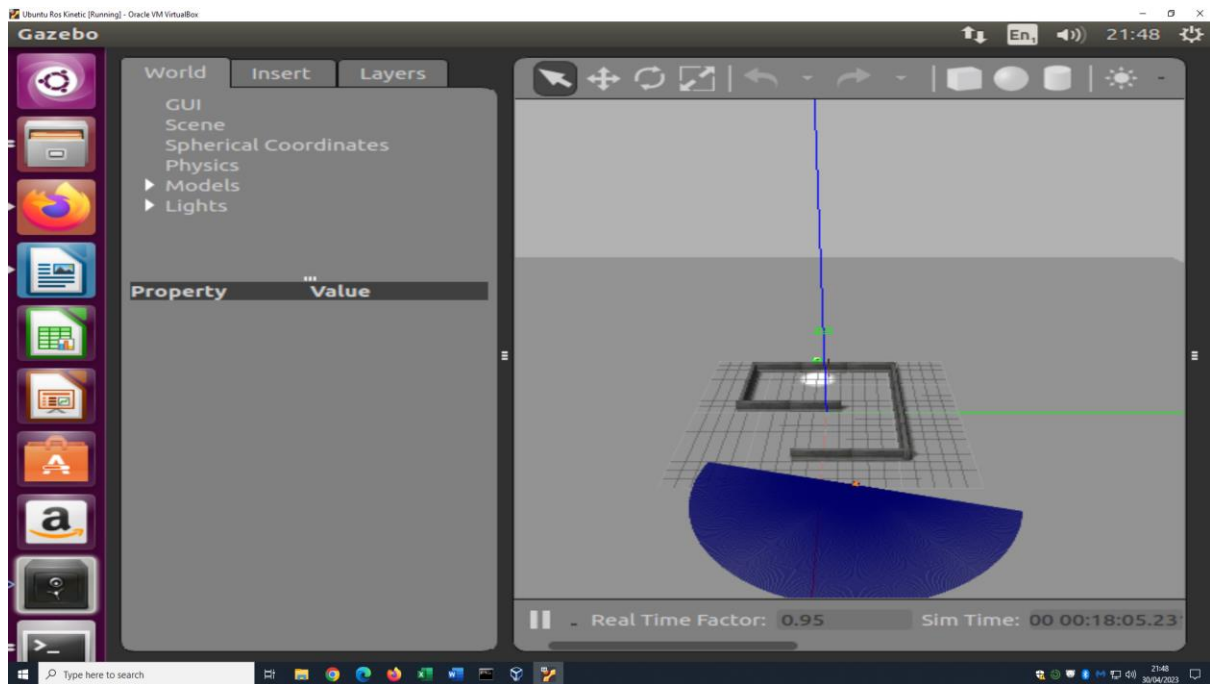
The screenshot shows a terminal window in an Ubuntu VM. The user is in the directory `/media/sf_ubuntu/sample/com760_cw2/src/com760_b00885866`. They run `cat ../scripts/move_b00885866Bot_circle.py` to display the contents of a Python script. The script uses `rospy` to publish `geometry_msgs/Twist` messages to the `/turtle1/cmd_vel` topic, causing a robot to move in a circle. The script defines a `move_circle()` function that sets a linear velocity of 0.3 and an angular velocity of 0.3, then publishes these messages at 10 Hz. The `__main__` block calls `move_circle()` and handles `ROSInterruptException`.

```
/launch$  
ubuntu@ubuntu-VirtualBox:/media/sf_ubuntu/sample/com760_cw2/src/com760_b00885866  
/launch$ cat ../scripts/move_b00885866Bot_circle.py  
#!/usr/bin/env python  
## Simple publisher demo that published geometry_msgs/Twist messages  
## to the /turtle1/cmd_vel topic  
  
import rospy  
from geometry_msgs.msg import Twist  
  
def move_circle():  
    #Initialize a publisher that can talk to the turtlesim and ask it to move  
    pub = rospy.Publisher('b00885866Bot/cmd_vel', Twist, queue_size=10)  
    rospy.init_node('bot_mover')  
  
    #create a Twist message and linear x and angular z value  
    vel_msg = Twist()  
    vel_msg.linear.x = 0.3  
    vel_msg.angular.z = 0.3  
  
    rate = rospy.Rate(10) # 10hz  
    while not rospy.is_shutdown():  
        rospy.loginfo(vel_msg)  
        pub.publish(vel_msg)  
        rate.sleep()  
  
if __name__ == '__main__':  
    try:  
        move_circle()  
    except rospy.ROSInterruptException:  
        pass  
ubuntu@ubuntu-VirtualBox:/media/sf_ubuntu/sample/com760_cw2/src/com760_b00885866  
/launch$
```

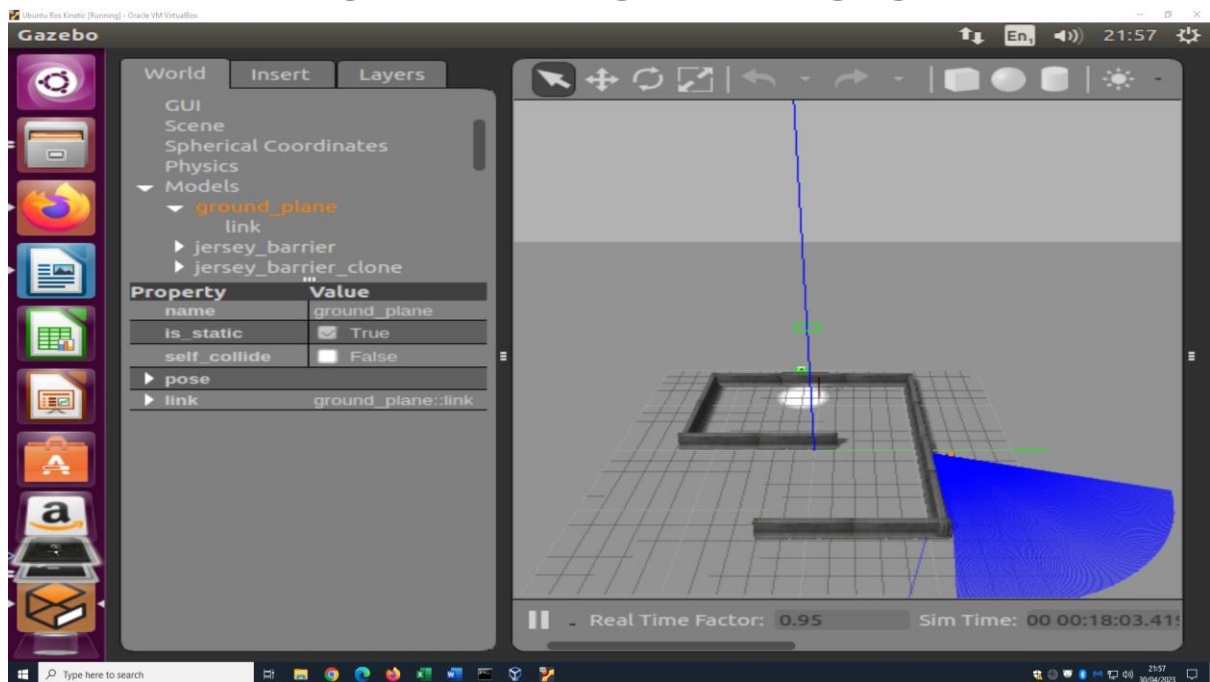


The screenshot shows the same terminal window after running `roslaunch com760_taskB_move_circle com760_taskB_move_circle.launch`. The output shows the launch process, including logging to a file, checking disk usage, and starting the `roslaunch` server. It then displays the launch configuration, including parameters like `/des_pos_x`, `/des_pos_y`, `/initial_x`, `/initial_y`, `/rostdistro`, `/rosversion`, and `/use_sim_time`. The nodes listed are `gazebo`, `gazebo_gui`, `go_to_point`, `move_circle`, `spawn_urdf`, and `wall_follower`. The `ROS_MASTER_URI` is set to `http://localhost:11311`.

```
/launch$ mv com760_taskB.launch_move_circle com760_taskB_move_circle.launch  
ubuntu@ubuntu-VirtualBox:/media/sf_ubuntu/sample/com760_cw2/src/com760_b00885866  
/launch$ roslaunch com760_taskB_move_circle.launch  
... logging to /home/ubuntu/.ros/log/e83ef92e-e784-11ed-8ff5-080027b2b122/roslau  
nch-ubuntu-VirtualBox-17861.log  
Checking log directory for disk usage. This may take awhile.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
started roslaunch server http://ubuntu-VirtualBox:35393/  
  
SUMMARY  
=====  
  
PARAMETERS  
* /des_pos_x: -6  
* /des_pos_y: -1  
* /initial_x: 0  
* /initial_y: 7  
* /rostdistro: kinetic  
* /rosversion: 1.12.17  
* /use_sim_time: True  
  
NODES  
/  
  gazebo (gazebo_ros/gzserver)  
  gazebo_gui (gazebo_ros/gzclient)  
  go_to_point (com760_b00885866/go_to_point_service.py)  
  move_circle (com760_b00885866/move_b00885866Bot_circle.py)  
  spawn_urdf (gazebo_ros/spawn_model)  
  wall_follower (com760_b00885866/follow_wall_service.py)  
  
ROS_MASTER_URI=http://localhost:11311
```

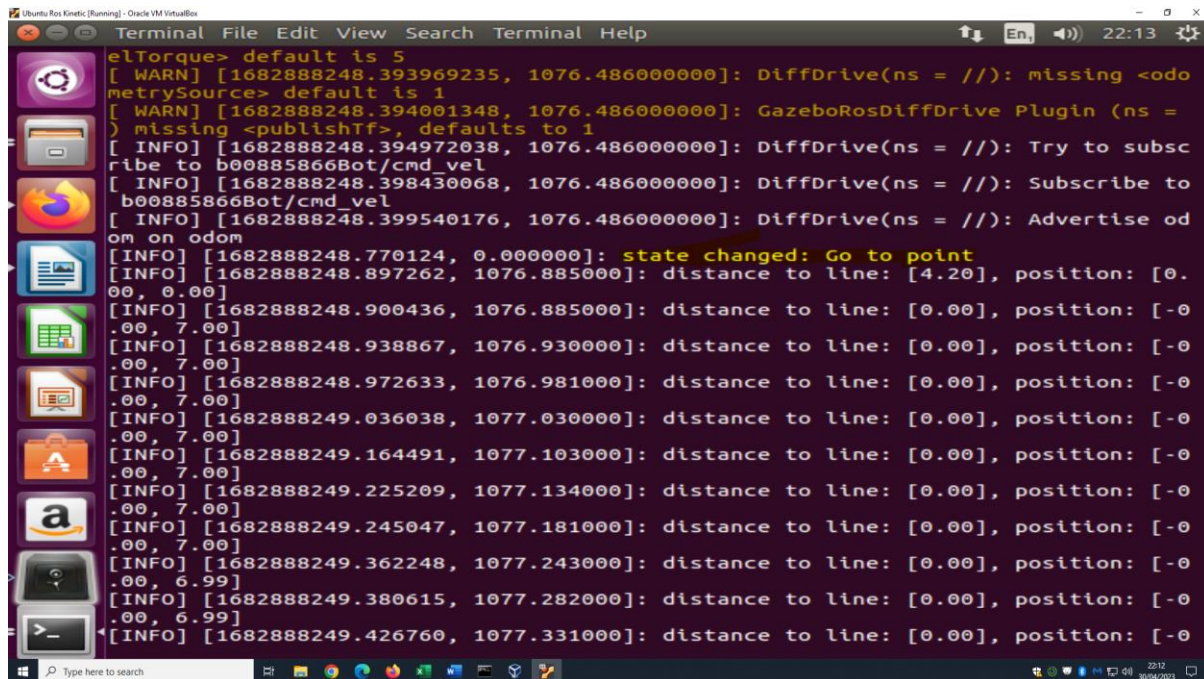



A screen shot demonstrating that the robot begins the journey back to the docking after receiving the homing signal.



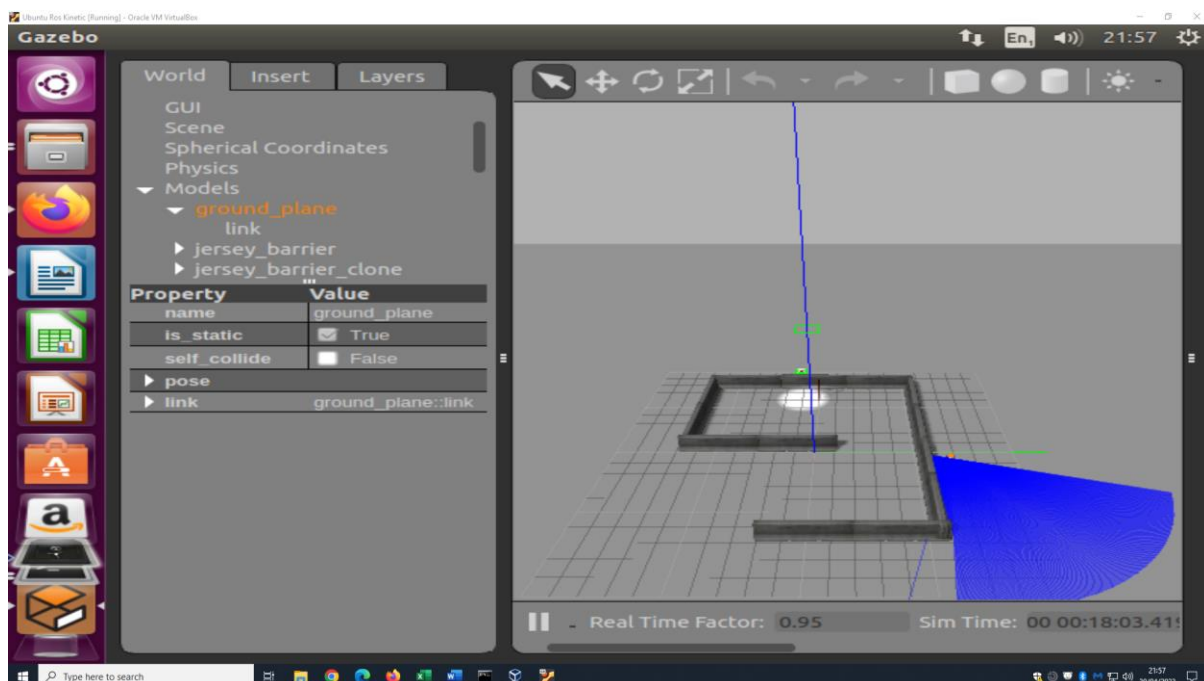
Screenshots demonstrating the successful implementation of Bug 2 algorithm.

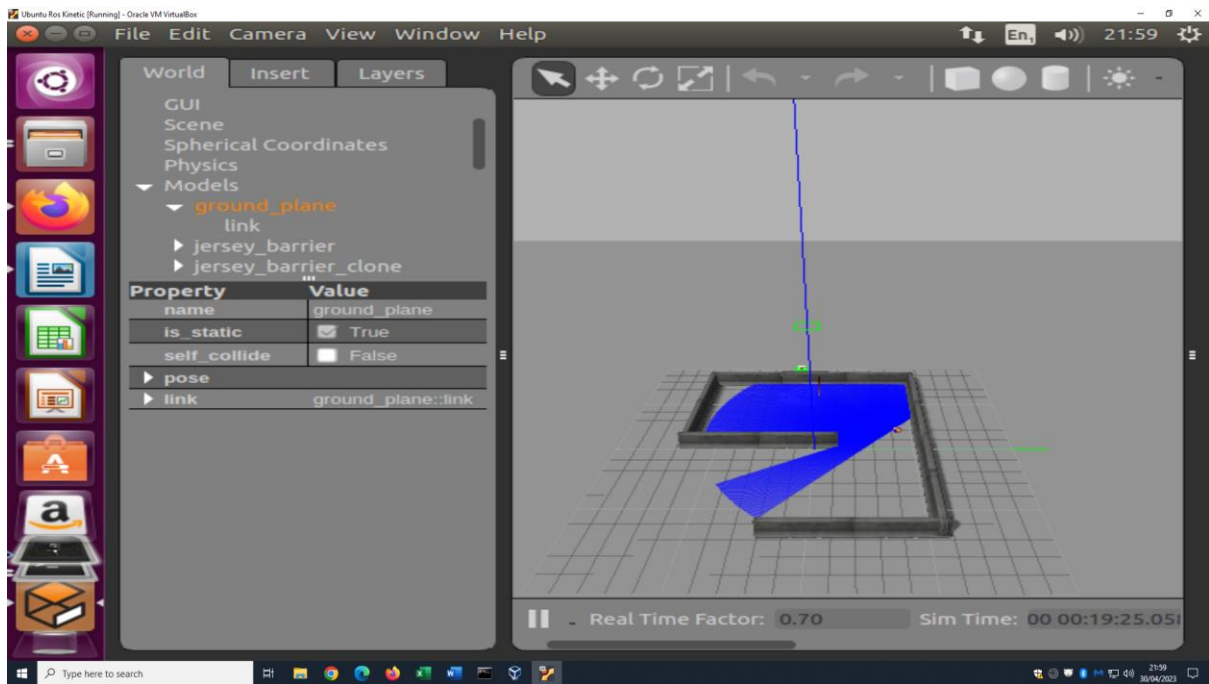
A screenshot showing the robot is in State 0, i.e. Go to point.



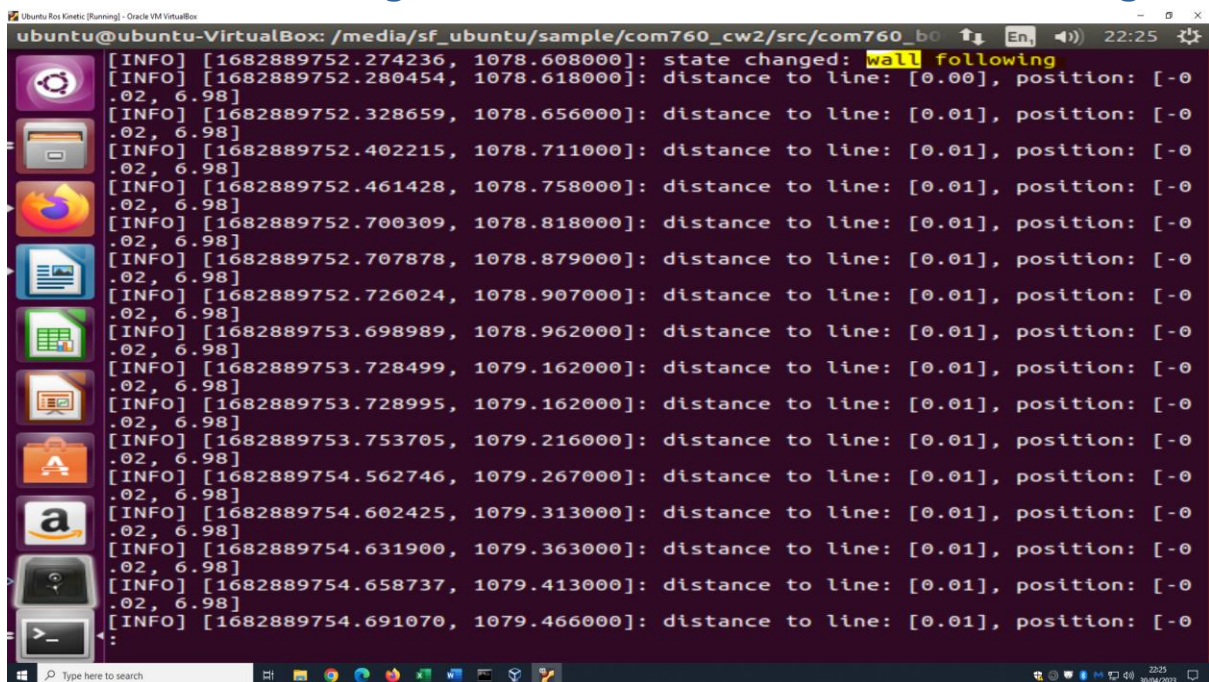
The screenshot shows a terminal window with ROS logs. The logs indicate that the robot is in State 0, "Go to point". The logs show the robot's position and distance to the line at various time steps.

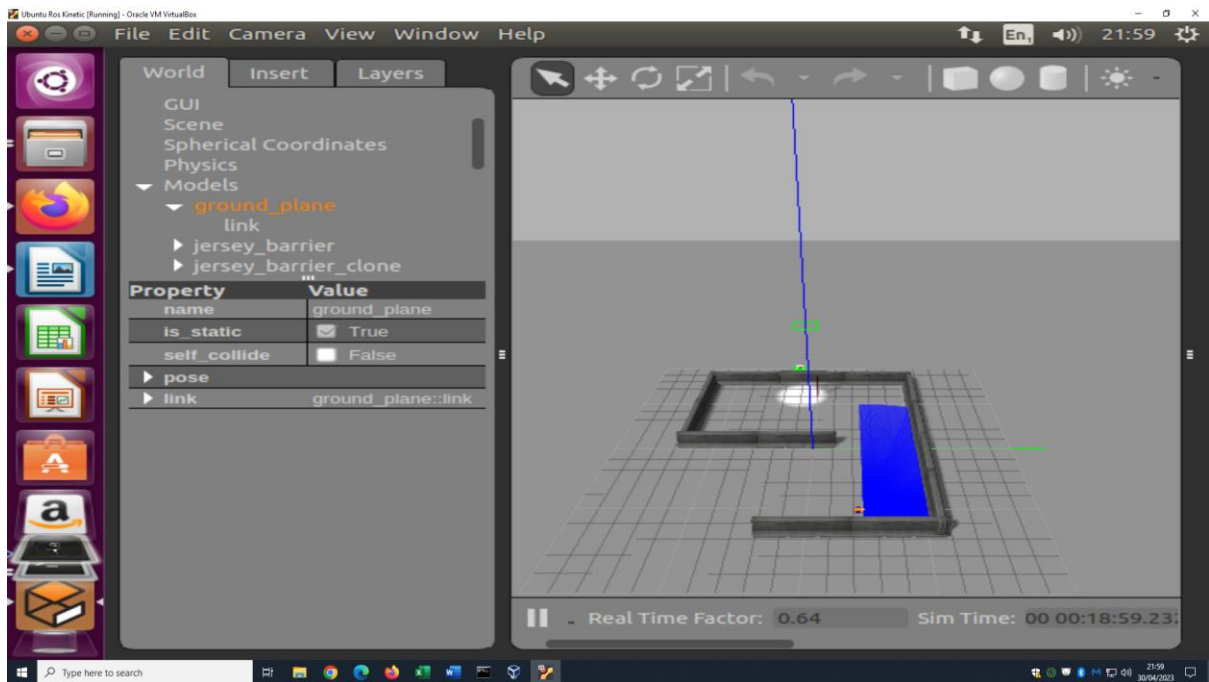
```
elTorque> default is 5
[ WARN] [1682888248.393969235, 1076.486000000]: DiffDrive(ns = //): missing <odom
metrySource> default is 1
[ WARN] [1682888248.394001348, 1076.486000000]: GazeboRosDiffDrive Plugin (ns =
) missing <publishIf>, defaults to 1
[ INFO] [1682888248.394972038, 1076.486000000]: DiffDrive(ns = //): Try to subsc
ribe to b00885866Bot/cmd_vel
[ INFO] [1682888248.398430068, 1076.486000000]: DiffDrive(ns = //): Subscribe to
b00885866Bot/cmd_vel
[ INFO] [1682888248.399540176, 1076.486000000]: DiffDrive(ns = //): Advertise od
om on odom
[ INFO] [1682888248.770124, 0.000000]: state changed: Go to point
[ INFO] [1682888248.897262, 1076.885000]: distance to line: [4.20], position: [0.
00, 0.00]
[ INFO] [1682888248.900436, 1076.885000]: distance to line: [0.00], position: [-0
.00, 7.00]
[ INFO] [1682888248.938867, 1076.930000]: distance to line: [0.00], position: [-0
.00, 7.00]
[ INFO] [1682888248.972633, 1076.981000]: distance to line: [0.00], position: [-0
.00, 7.00]
[ INFO] [1682888249.036038, 1077.030000]: distance to line: [0.00], position: [-0
.00, 7.00]
[ INFO] [1682888249.164491, 1077.103000]: distance to line: [0.00], position: [-0
.00, 7.00]
[ INFO] [1682888249.225209, 1077.134000]: distance to line: [0.00], position: [-0
.00, 7.00]
[ INFO] [1682888249.245047, 1077.181000]: distance to line: [0.00], position: [-0
.00, 7.00]
[ INFO] [1682888249.362248, 1077.243000]: distance to line: [0.00], position: [-0
.00, 6.99]
[ INFO] [1682888249.380615, 1077.282000]: distance to line: [0.00], position: [-0
.00, 6.99]
[ INFO] [1682888249.426760, 1077.331000]: distance to line: [0.00], position: [-0
```





A screenshot showing the robot is in State 1, i.e. Wall following.





A screenshot showing the robot reaches the docking station.

