# Lab 3 - Robot Operating System: Part B

## 3.1 Run your code with a new world map

In this lab, you will run a robot in a new world map described in CE801-Summer Assignment.

**Task 3.1** Build your node and run it with a new world map in Gazebo.

Download *rosbot_sim_room_12.world* from the Moodle and copy it to the following directory

```
~/ros_workspace/src/rosbot_description/src/rosbot_gazebo/worlds
```

Using *gedit* to open the launch file *rosbot_world.launch* as follows,

```
cd ~/ros_workspace/src/rosbot_description/src/rosbot_gazebo/launch/
gedit rosbot_world.launch
```

Find the *“world_name”,* change its value to *rosbot_sim_room_12.world*, *i.e.*

```
<launch>
   <arg name="world" default="empty"/>
   <arg name="paused" default="false"/>
   <arg name="use_sim_time" default="true"/>
   <arg name="gui" default="true"/>
   <arg name="headless" default="false"/>
   <arg name="debug" default="false"/>
   <include file="$(find gazebo_ros)/launch/empty_world.launch">
   <arg name="world_name" value="$(find rosbot_gazebo)/worlds/rosbot_sim_room_12.world"/>
   <arg name="paused" value="$(arg paused)"/>
   <arg name="use_sim_time" value="$(arg use_sim_time)"/>
   <arg name="gui" value="$(arg gui)"/>
   <arg name="headless" value="$(arg headless)"/>
   <arg name="debug" value="$(arg debug)"/>
  </include>
</launch>
```

Therefore, your program knows where the simulation map is.

Now, you can use the following command to create *tutorial_rosbot.launch*:

```
touch ~/ros_workspace/src/tutorial_pkg/launch/tutorial_rosbot.launch
```

and then using gedit to open

```
gedit tutorial_rosbot.launch
```

type the following commands into

```
<launch>
   <param name="/use_sim_time" value="true"/>
   <!-- Launch  rosbot world map -->
```

```
    <arg name="use_gazebo" default="true"/>
    <include unless="$(arg use_gazebo)" file="$(find astra_launch)/launch/astra.launch"/>
    <include if="$(arg use_gazebo)" file="$(find rosbot_description)/launch/rosbot.launch"/>
   <!-- Launch RobotMove node -->
    <node pkg="tutorial_pkg" type="tutorial_pkg_node" name="RobotMove" output="screen">
    </node>
</launch>
```

Now, you can use the following command to launch it:

```
source ~/ros_workspace/devel/setup.sh
roslaunch tutorial_pkg tutorial_rosbot.launch
```

You will see ROSbot moves along vertically in the Gazebo simulator window.

If you want to stop it, you may hold “Ctl^C” using your keyboard.

## 3.2 Change the robot moving direction in Gazebo

Now, you will learn how to change the robot moving direction from the vertical motion in Task 3.1 to the horizontal motion in Task 3.2.

**Task 3.2**: Go to the following directory:

```
cd ~/ros_workspace/src/rosbot_description/src/rosbot_description/launch
```

Using *gedit* to open the launch file *rosbot_gazebo.launch*

```
gedit rosbot_gazebo.launch
```

Then, you should the following commands to replace its contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<launch>
    <!-- Robot initial pose -->
    <arg name="x" default="0"/>
    <arg name="y" default="0"/>
    <arg name="z" default="0"/>
    <arg name="roll" default="0"/>
    <arg name="pitch" default="0"/>
   <arg name="yaw" default="1.5708"/>
   <!-- Robot head angle, default towards x axis. -->
   <rosparam command="load" file="$(find joint_state_controller)
        /joint_state_controller.yaml" />
   <node name="joint_state_controller_spawner" pkg="controller_manager"
        type="spawner" output="screen" args="joint_state_controller" />
   <param name="robot_description" command="$(find xacro)/xacro.py '$(find
        rosbot_description)/urdf/rosbot.xacro'"/>
   <node name="rosbot_spawn" pkg="gazebo_ros" type="spawn_model" output
      ="screen" args="-urdf -param robot_description -model rosbot -x $(arg x)
```

```
        -y $(arg y) -z $(arg z) -R $(arg roll) -P $(arg pitch) -Y $(arg yaw)" />
    <node name="robot_state_publisher" pkg="robot_state_publisher" type
        ="state_publisher"/>
</launch>
```

Now, you can open a terminal and then use the following command to launch it:

```
source ~/ros_workspace/devel/setup.sh
roslaunch tutorial_pkg tutorial_rosbot.launch
```

You can see that ROSbot moves horizontally in the Gazebo simulator window.

If you want to stop it, you may type “Ctl^C” using your keyboard.

## 3.3 Useful ROS tools

Here are some very useful tools in ROS, which are intended to examine nodes and topics.

*rosnode*

It is a command for examining which nodes are registered in the system and also checking their statuses, which can be used as below:

| | |
|---|---|
| *rosnode ping* | -- test connectivity to node |
| *rosnode list* | -- list active nodes |
| *rosnode info* | -- print information about node |
| *rosnode machine* | -- list nodes running on a particular machine or list machines |
| *rosnode kill* | -- kill a running node |
| *rosnode cleanup* | -- purge registration information of unreachable nodes |

Detailed info could be found in ROS documentation.

*rostopic*

It is a command for examining which topics are already being published and subscribed, checking details of the selected topic or reading messages being sent in it. Its format is:

| | |
|---|---|
| *rostopic bw* | -- display bandwidth used by topic |
| *rostopic delay* | -- display delay of topic from timestamp in header |
| *rostopic echo* | -- print messages to screen |
| *rostopic find* | -- find topics by type |
| *rostopic hz* | -- display publishing rate of topic |
| *rostopic info* | -- print information about active topic |
| *rostopic list* | -- list active topics |
| *rostopic pub* | -- publish data to topic |
| *rostopic type* | -- print topic or field type |

Detailed info could be found in ROS documentation.

*rqt_graph*

It is a graphical tool for visualizing the data flow across different nodes in the system.

```
rqt_graph
```

**Task 3.3** Use *rosnode*, *rostopic* and *rqt_graph* tools to examine system and check how data is passed between nodes.

Answer:

**In terminal 1**: you can use the following command to launch *tutorial_pkg*

```
source ~/ros_workspace/devel/setup.sh
roslaunch tutorial_pkg tutorial_rosbot.launch
```

You can see that ROSbot moves horizontally in the Gazebo simulator window

**In terminal 2**: you can use the following command to see active nodes

```
rosnode list
```

Using the following command to list active topics*:*

```
rostopic list
```

Then using the following graphical tool for visualization of data flow across nodes.

```
rqt_graph
```

Then you should see a GUI graph that shows data flow among ROS nodes and topics.

## 3.4 Velocity message

You can open a terminal and typing the following command:

```
rosmsg show geometry_msgs/Twist
```

You will have the speed message definition as bellow:

```
geometry_msgs/Vector3 linear
float64 x
float64 y
 float64 z
geometry_msgs/Vector3 angular
float64 x
float64 y
float64 z
```

which could be also used for setting the desired linear and angular velocities for the robot.

**Task 3.4:** Suppose you have been given the *moveForward* function,

```
void RobotMove::moveForward(double forwardSpeed){
    geometry_msgs::Twist msg; //The default constructor will set all commands to 0
    msg.linear.x = forwardSpeed; //Drive forward at a given speed along the x-axis.
    commandPub.publish(msg);
}
```

and the *moveRight* function.

```
void RobotMove::moveRight(double turn_right_speed){
    geometry_msgs::Twist msg;
    msg.angular.z = turn_right_speed;
    commandPub.publish(msg);
}
```

Define a new function, *moveForwardRight,* that publishes the forward speed and turn-right speed at the same time.

Answer:

```
void RobotMove::moveForwardRight(double forwardSpeed, double
turn_right_speed){
    //move forward and right at the same time
    geometry_msgs::Twist msg;
    msg.linear.x = forwardSpeed;
    msg.angular.z = turn_right_speed;
    commandPub.publish(msg);
}
```

## References

ROS Wiki

*http://wiki.ros.org/*

Installation

*http://wiki.ros.org/ROS/Installation*

Tutorials

*http://wiki.ros.org/ROS/Tutorials*

Available packages

*http://www.ros.org/browse/*

ROS Cheat Sheet

*https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/*
*https://kapeli.com/cheat_sheets/ROS.docset/Contents/Resources/Documents/index*

ROS Best Practices

*https://github.com/leggedrobotics/ros_best_practices/wiki*

ROS Package Template

*https://github.com/leggedrobotics/ros_best_practices/tree/master/ros_package_template*