**Title:** Simulation Exercises with ROS and TurtleBot3 burger

**Name/Surname:** Burhan Mohayaddin

**Course:** Industrial Robotics

**Date:** 01/14/2021

## Table of Contents

Introduction....1
Task 1....2
Spawning TurtleBot3....2
Task 2....2
Autonomous Mapping of Environment....2
Instructions to run the code....2
Task3....3
Navigate Environment by Reaching Given Points....3
Instructions to run the code....3
Conclusion....3

# Introduction

This report presents broad explanation alongside solution to the laboratory tasks that have to be implemented on ROS and simulated on Gazebo.

# Task 1

## Spawning TurtleBot3

In this task TurtleBot3 robot has to be spawned to the gazebo_house environment. Moreover, the coordinates (x and y) have to be specified in order to specify the initial position of the robot. So, in order to launch the robot “roslaunch” command of ROS can be utilized. The command is:

- $ roslaunch turtlebot3_tasks task1.launch

This command basically, launches “task1.launch” launch file inside “turtlebot3_tasks”. Additionally, initial x and y positions have been provided inside launch file and they can be changed.

# Task 2

## Autonomous Mapping of Environment

In the second task, the robot should autonomously map the environment without having previous knowledge about it. It can be implemented through several ways which has its own advantages and disadvantages, while in this report it has been implemented using simple wall following algorithm. So, it has three states and it switches between these states:

- find the wall
- turn left
- follow the wall

In this case robot right tracks the wall. It continually read its laser sensor and based on its left, right and front sensor readings it decides whether there is wall in front, left or right and changes state based on that. So, in order to do that robot subscribes to '/scan' topic in order to get laser distance information. After that, some regions of laser data are extracted in order to be utilized. This is done in clbk_laser function. The decision of which action has to be taken is done in take_action function. In order to move the robot, velocity commands have been published to the '/cmd_vel' topic.

The disadvantages of this code is that it is not possible to start to track the environment from any position. So, if there is no wall nearby it will try to randomly move in order to search for a wall which can make it far from the interested environment. Additionally because, it just follows the wall there can be some regions that are out of mapping depending on the environment.

In order to do mapping, slam method will be utilized which means “Simultaneous localization and mapping”. It will be launched by launch file which will start both Gazebo and Rviz which will be used to analyze sensor reading and viewing created map.

## Instructions to run the code

- $ roslaunch turtlebot3_tasks robot_gazebo_and_slam.launch
- $ cd ~/uni_ws/src/turtlebot3_tasks/scripts
- open new terminal and source the workspace
- $ python2.7 turtlebot3_follow_wall.py
- open new terminal
- $ rosrun map_server map_saver -f ~/map

# Task3

## Navigate Environment by Reaching Given Points

In this task, the robot has to go to the destination points that are provided through a text file inside the map that has been mapped before. In order to navigate the robot inside map, “turtlebot3_navigation.launch” launch file will be utilized which is provided by ROS. The navigation is realized through ROS navigation stack which is based on ROS actions. In order to configure, run and interact with navigation stack. It is implemented through SimpleActionServer which takes as an input geometry_msgs/PoseStamped message. SimpleActionClient is used in order to communicate with it. move_base node tries to go to the given point and obtain desired position by utilizing global and local motion planner while avoiding obstacles.

Desired pose is given inside .txt file and here the angles are in degree which will be converted to radians then. Goal points can be changed from goal.txt file.

## Instructions to run the code

- $ roslaunch turtlebot3_tasks task3.launch
- open new terminal and source the workspace
- $ cd ~/uni_ws/src/turtlebot3_tasks/scripts
- $ python2.7 set_nav_goals.py

# Conclusion

In conclusion, in these laboratory tasks turtlebot3 robot has been implemented both in order to map the environment autonomously and then navigate through the environment by reaching destination points.