**Readme**

**Turtlesim:**

**Part (a):**

The solution is to use spawn service to show two turtle bots in single turtlesim window.

A launch file of name ***two\_turtles.launch*** has been created.

**Pre-requisites:**

* ros
* turtlesim package

**Steps:**

1. Create the workspace catkin\_ws and navigate to catkin\_ws/src
2. Download the package ‘follower’, and place it in catkin\_ws/src
3. Run the following commands in catkin\_ws

$ catkin\_make

$ source devel/setup.bash

1. Run the following command to launch the file two\_turtles.launch

$ roslaunch follower two\_turtles.launch

You will see a turtlesim window with two turtles i.e turtle1 and turtle2.

1. Use arrow keys to move the turtle1.

**Part (b):**

For the part (b), a ros package named ***‘follower'*** is provided. This package has ***'launch'*** folder, and ***'nodes'*** folder. In the launch folder ***'turtle\_follower.launch'*** is the corresponding launch file for this task.

The ***‘nodes***’ folder contains ***'tf\_braodcaster.py'*** which broadcasts the transformation between ***'world'*** and ***'turtle1'***, and fixed transformation between ***'turtle1'*** frame and reference frame i.e ***'ref\_frame'.*** The turtle2 follows turtle1 by following the ***'ref\_frame',*** maintaining the distance of 0.8 meters.

There is another file ***'dist\_cal.py'***, which calculates the Eucladian Distance between turtle1 and turtle2.

**Steps:**

1. Run the following commands in catkin\_ws

$ catkin\_make

$ source devel/setup.bash

1. Run the following command to launch the ‘***turtle\_follower.launch’***

$ roslaunch follower turtle\_follower.launch

You will see a turtlesim window with two turtles i.e turtle1 and turtle2.

1. Use arrow keys to move the turtle1, turtle2 will follow turtle1 keeping the distance of 0.8 meters.
2. To check the distance between turtle1 and turtle2, open new terminal and run the following:

$ rosrun follower dist\_cal.py

**NOTE:** Please run ***source devel/setup.bash***, berfor executing roslaunch or rosrun commands

**Navigation\_Stack-Simulation:**

***Part (a):***

This part of the assignment was done by using the tutorial link that was provided. The environment was set up successfully, and the siimulation was running fine. In this part we created gmaps of two different gazebo environments. These maps are available in the ***'maps'*** folder in the package ‘***turtlebot3\_gazebo***’

After creating the map, 'turtlebot3\_navigation' was used to navigate to the specific point provided by rviz ***'2D Nav Goal'*** tool. The screenshots for this task are provided in the screenshots folder. Furthermore, google drive link to the screen recorded videos is also provided in a ***'Link\_to\_Videos.txt'***.

**Pre-requisites:**

The following packages needs to be installed:

* turtlebot3
* turtlebot3\_msgs
* turtlebot3\_navigation
* turtlebot3\_teleop

After installing these packages, turtlebot3\_simulatioins repository can be cloned by running: (optional)

$ git clone -b melodic-devel <https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git>

You will have to replace the launch files, node files and maps in turtlebot3\_gazebo package with the files provided in ***Navigation\_Stack-Simulation/turtlebot3\_gazebo.ss***

However, in the scope of this assignment, downloading our ‘***turtlebot3\_gazebo’*** package located in Question#2 folder is recommended.

**Steps: (optional)**

1. Open the new terminal and run the following command:

$ cd catkin\_ws/src

1. Download the package ***‘turtlebot3\_gazebo’*** provided in the Question#2 folder, and copy it to ***catkin\_ws/src***

**(OR)**

clone the following package in **‘catkin\_ws/src’**:

1. In the root directory, run the following command to launch the gazebo environment:

$ export TURTLEBOT3\_MODEL=waffle\_pi

$ roslaunch turtlebot3\_gazebo turtlebot3\_house.launch

1. In the new terminal, run the following commands for gmapping:

$ export TURTLEBOT3\_MODEL=waffle\_pi

$ roslaunch turtlebot3\_slam turtlebot3\_slam.launch slam\_methods:=gmapping

1. In the new terminal, run the following commands to launch the teleop node:

$ export TURTLEBOT3\_MODEL=waffle\_pi

$ roslaunch turtlebot3\_teleop turtlebot3\_teleop\_key.launch

1. Control the turtle boat and explore the environment to generate gmap of the environment
2. Save the map in the ***turtlebot3\_gazebo/maps*** folder by running the following command

$ rosrun map\_server\_server -f ~/map

1. After saving the map, run the following commands to launch the navigation:

$ export TURTLEBOT3\_MODEL=waffle\_pi

$ roslaunch turtlebot3\_navigation turtlebot3\_navigation.launch map\_file:=$<path to map file>/<map\_name>.yaml

(map\_name refers to the name of the map you saved)

1. In the rviz use 2D Pose Estimate tool to provide the initial pose of the robot, then run teleop to move the robot too and fro for better location estimation.
2. Use 2D Nav Goal tool to point the targeted location and direction of the robot. The robot will plan the path and will navigate to that goal location.

**NOTE:** The above steps are to complete the part(a) of the question and are optional. This process has been done and our package ‘***turtlebot3\_gazebo’*** have the gmaps of two environments. (refer to the provided screenshots or videos)

**Part(b):**

For this part we have created a cpp file 'goal\_publisher.cpp' which subscribes to the topic ***'/goal\_location',*** and based on the integer received, it publishes the pre-selected goal point (x, y, w) located on the map to the topic ***'/move\_base/goal'***. The navigation tool automatically plans the path and makes the robot navigate to that goal point on the map by avoiding obstacles. I have used gmap of house environment for this task.

The goal locations (1, 2, or 3) can be published as integers on topic ***'/goal\_location'*** using publish command on terminal or using some publisher node. I have written a 'publish\_goal\_number.cpp' for the convenience. This cpp file creates a ***'pulisher\_node'*** which prompts the user to enter any number from (1 to 3), to navigate to the corresponding goal location, and integer 0 for exit. User input is then published on the ***rostopic '/goal\_location'***, which is subscribed by the ***'goal\_publisher\_node'*** in the 'goal\_publisher.cpp' file.

Furthermore, a single launch file '***combined.launch'*** has been created to launch both gazebo simulation and rviz visualization.

**Steps:**

1) Download the package **‘turtlebot3\_gazebo’**, and place it in the ***'src'*** folder.

2) In **catkin\_ws/src** folder run **catkin\_make,** and **source devel/setup.bash**

3) Launch the gazebo and rviz simulation using **'combined.launch'** by typing the following command to the terminal:

$ export TURTLEBOT3\_MODEL = waffle\_pi

$ roslaunch turtlebot3\_gazebo combined.launch map\_file:=$<path\_to\_maps>/map\_home.yaml

4) In the rviz, use **'2D Pose Estimate'** tool to provide the estimate of initial pose.

5) Run the teleop command and move the robot to and fro, for better location estimation (optional).

6) Run the ***'goal\_publisher\_node'*** by running the following command in the terminal:

$ rosrun turtlebot3\_gazebo goal\_publisher\_node

7) In the separate terminal, run the **'publisher\_node'** to prompt the goal location (1, 2, or 3) from user

$ rosrun turtlebot3\_gazebo publisher\_node

8) Prompt the goal location

9) While the robot is navigating to the provided goal location, give it other goal location and the robot will move towards the updated goal location (See Videos for reference).

10) Echo the topics to check the data being published. ss

$ rostopic echo/goal\_location

$rostopic/move\_base/goal