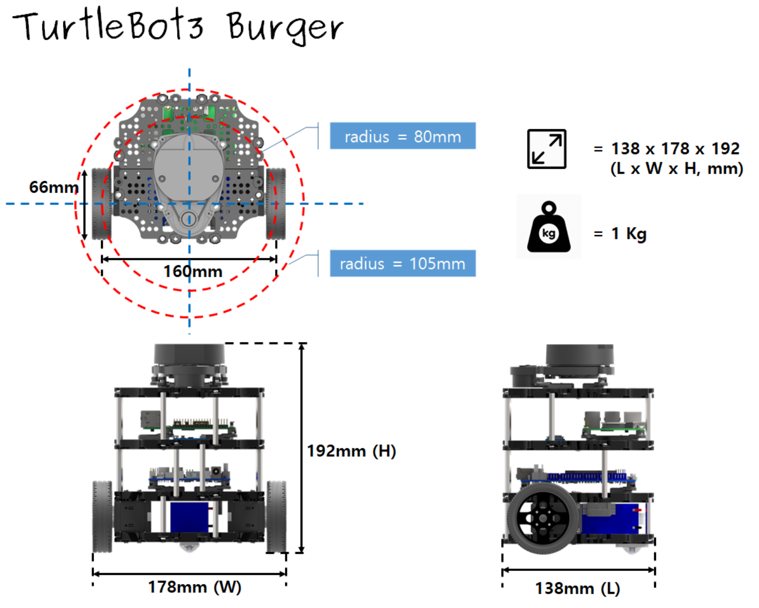
Department of Electrical and Computer Engineering
University of Illinois at Chicago

ECE 452 Individual Project 3 Date: 04/09/2020

Due date: 04/17/2020

This project is a continuation of Project 2. You will implement wall following algorithm with Turtlebot3 using 2D **LaserScan** sensor. **LaserScan** sensor is used to detect any obstacles in the planar environment in 360 degree. It is located on the top of the robot, see fig 1.



(a) Dimension

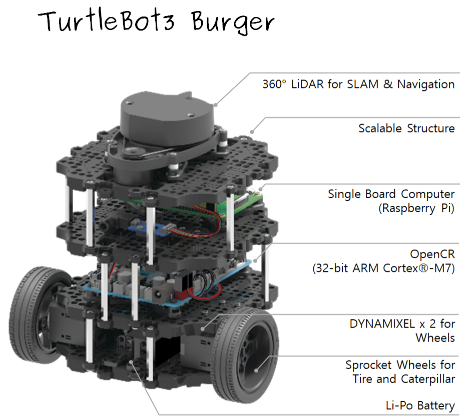

TurtleBot3 Burger
360° LiDAR for SLAM & Navigation
Scalable Structure
Single Board Computer (Raspberry Pi)
OpenCR (32-bit ARM Cortex®-M7)
DYNAMIXEL x 2 for Wheels
Sprocket Wheels for Tire and Caterpillar
Li-Po Battery
(b) Components

Figure 1: Overview of Turtlebot3 Burger

1 Environment setup

1. Similar to project 2, create `project_3` package in catkin workspace:

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg project_3 rospy
$ cd ~/catkin_ws
$ catkin_make
$ mkdir -p ~/catkin_ws/src/project_3/scripts
```

   As usual, download the starter code `follow_wall.py` from Blackboard and copy it to `scripts` directory.

2. Install `wall_following` world. Download the **wall_following.zip** file from blackboard. In the terminal window navigate to `Downloads` directory, and install the world following these steps:

```
$ cd ~/Downloads
$ sudo apt-get install unzip
$ unzip wall_following.zip -d wall_following
$ cd wall_following
$ ./setup.sh
```

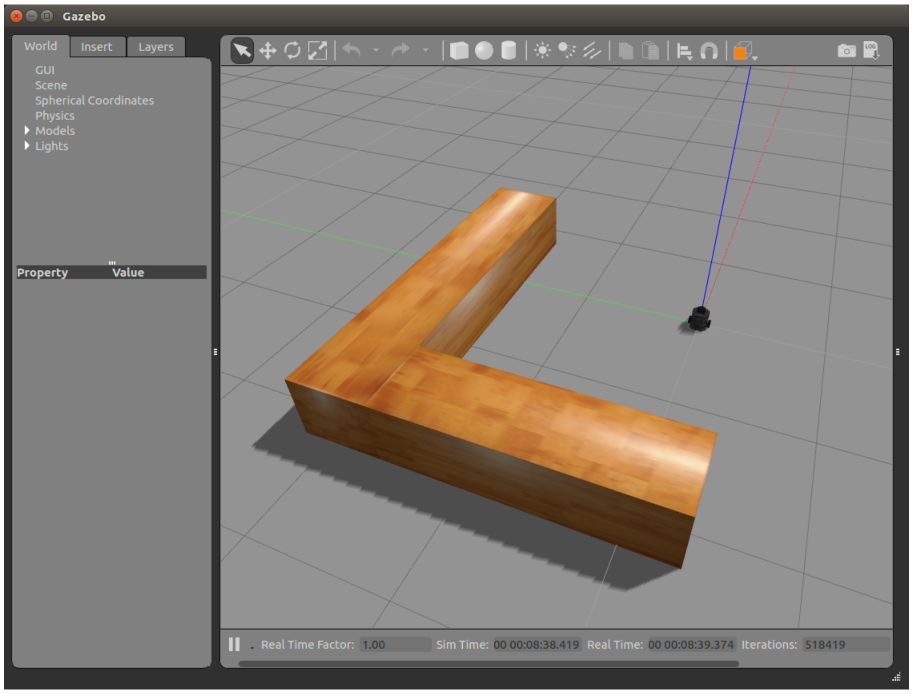


Figure 2: Wall following world

2 The Task

Using the **laserScan** sensor information implement wall following algorithm. Run simulations in the Turtlebot wall following world (see figure 2). Launch the world typing following command:

```
$ roslaunch turtlebot3_gazebo turtlebot3_wall_following.launch
```

Similarly, use `rosrun` to execute wall following code in new terminal.

```
$ rosrun project_3 follow_wall.py
```

To implement this task, first command the robot to go somewhere behind the wall (using similar idea in project 2 to execute the motion). Then decide actions of the robot accordingly based on the readings of laser scan signals from `\scan` topic. Robot should circle around the obstacle infinitely.

`\scan` topic gives raw laser data `msg` described in here. Specifically, `msg.ranges` is a vector of scanning measurements describing relative distance between the obstacle and robot. In the starter code, `msg.ranges` is already parsed for you with minimum distance of the obstacle in different regions like front, right, left etc.

You were given a starter code in `follow_wall.py` as a reference. But you are free to go with your own way of solving this problem.

3 The Report

To submit the report describing your work, follow these steps:

1. Record a screen video of your robot performing the task. Upload the video to YouTube. The name of the video should read **ECE452_Spring-2020_Proj-03_Surname**.
2. Please submit your report to Blackboard.
   (a) On the first page of the report (cover page), please give the **link** to your video.
   (b) Explain the difficulties you faced.
   (c) The report should contain your well-commented code.
3. Upload `follow_wall.py` to Blackboard. If you have additional files upload them as well.

On Plagiarism

IT IS IMPORTANT THAT YOU THOROUGHLY UNDERSTAND HOW TO WORK WITH THE ROBOT. COPYING THE CODE FROM ANOTHER GROUP OR OTHER SOURCE IS THEREFORE NOT ALLOWED. IT WILL BE CONSIDERED PLAGIARISM, RESULTING IN A FAILING GRADE AND FURTHER POSSIBLE DISCIPLINARY ACTION.

4 Appendix

4.1 Robot Model

Choose the body frame T so that its origin is at the midpoint between the two wheels on the common axis of the two wheels. The x-axis should point in the direction of the robot motion while the y-axis is perpendicular to the x-axis, coincident with the axes of the wheels. The following figure shows the spatial frame S and the body frame T.

Note that the robot "lives" in SE(2) space (on a plane), not SE(3). That is, using x, y and θ one can describe the robot's configuration. However, you can utilize the standard SE(3) representation and set:

$$z = 0 \quad \text{and} \quad \omega = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

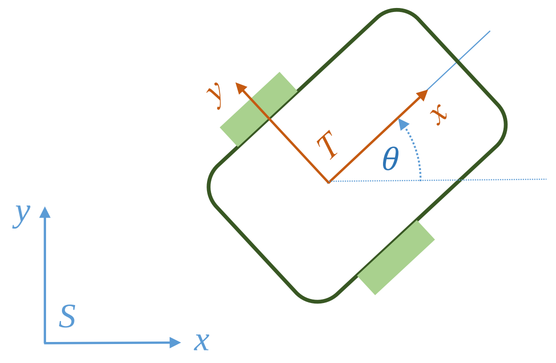


Figure 3: Robot model