# SC627: Assignment 1

Report by: Atharva Mete | 190010012

## Introduction

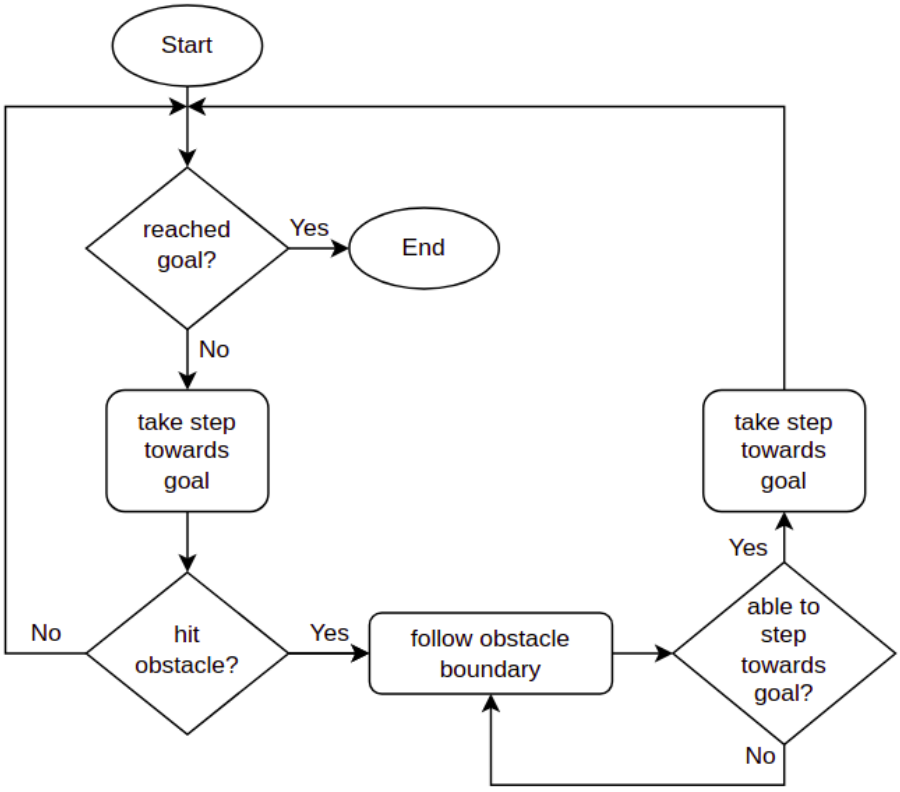
In this assignment, I implemented bug base and bug 1 algorithms. The implementation was done in python and simulated on Gazebo & RVIZ using Turtlebot3

### Task 1: Helper functions:

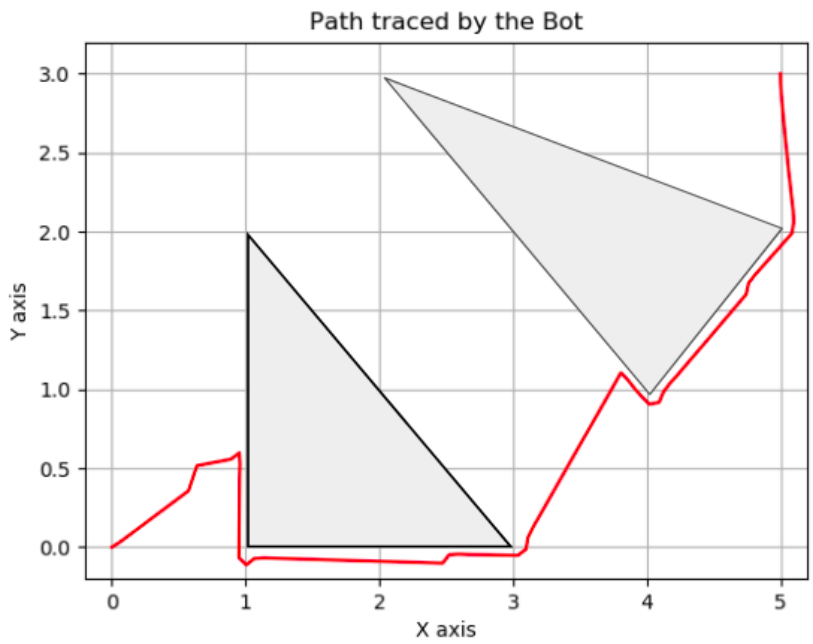
Following helper functions were used to assist the implementation of bug algorithms:
Distance between 2 points, Distance of a point from a line segment, Distance of a point from the nearest point on the polygon, unit tangent vector & unit perpendicular vector.

### Task 2: Bug Base Algorithm:

The following flow chart depicts the working of bug base algorithm:

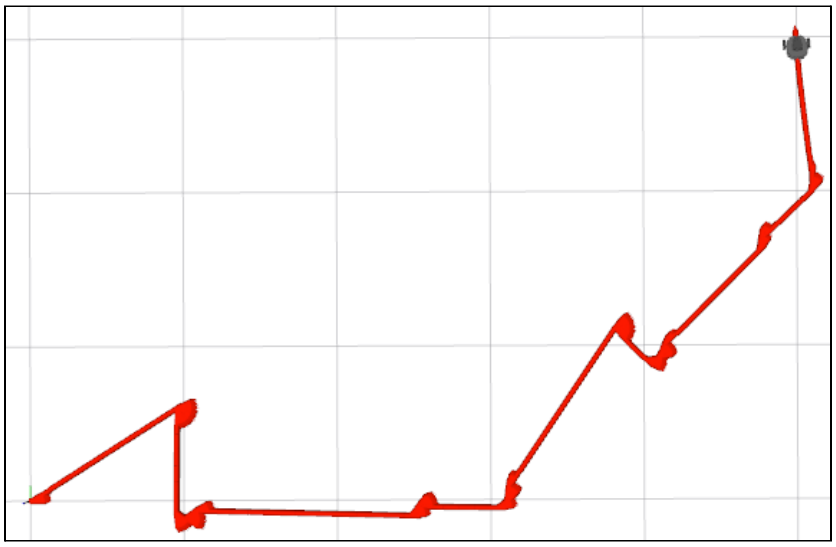


The logic in the above flow chart is used in the algorithm. The following plots show the path traversed by turtlebot:



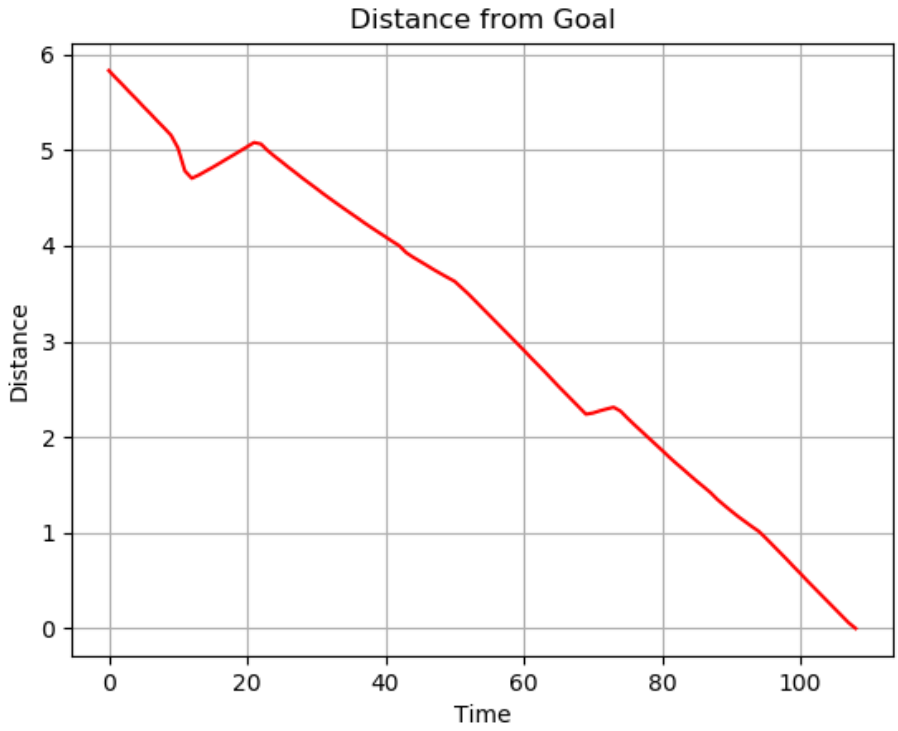

- **Execution Time:** 210.40 sec
- **Total Path Length:** 8.243 m

Following is the path traversed in simulation (RVIZ/Gazebo):



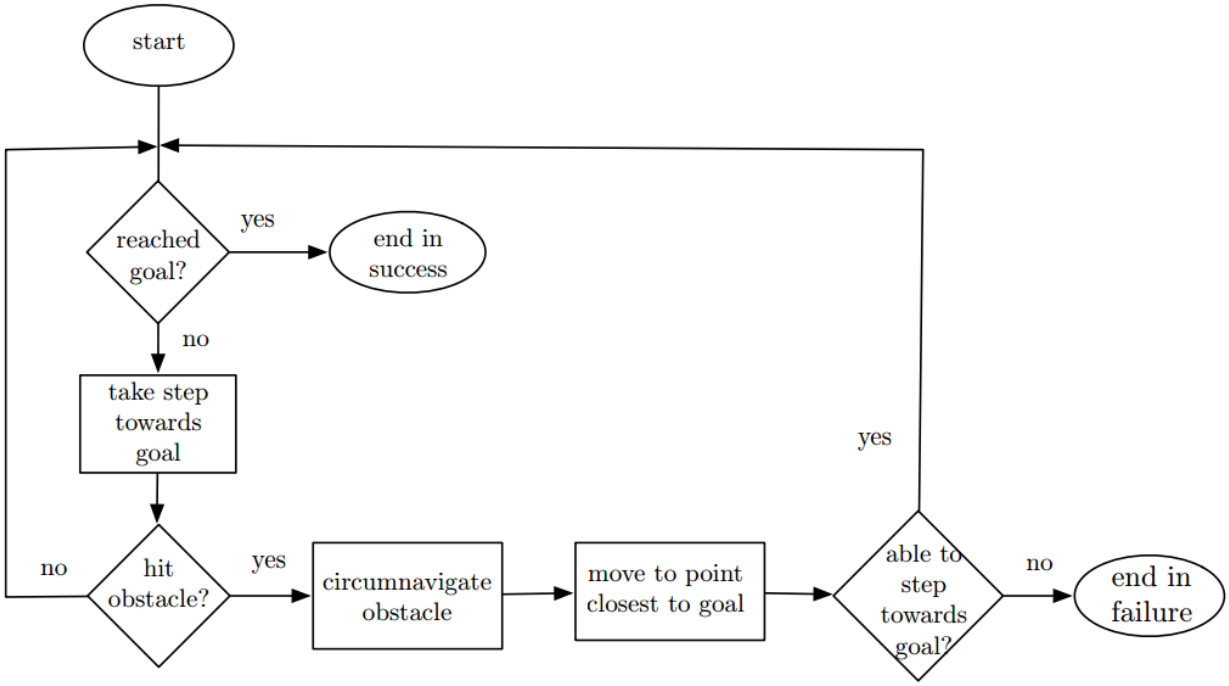
**Note:** On the turns, the distance from the obstacle is sometimes greater than step length, so turtlebot starts to move towards the goal, if there is a free path then it will continue on its path, if there is an obstacle then it will come under step length distance from obstacle & will follow its boundary.

Following is the plot of the distance of the turtlebot from the goal during its motion.

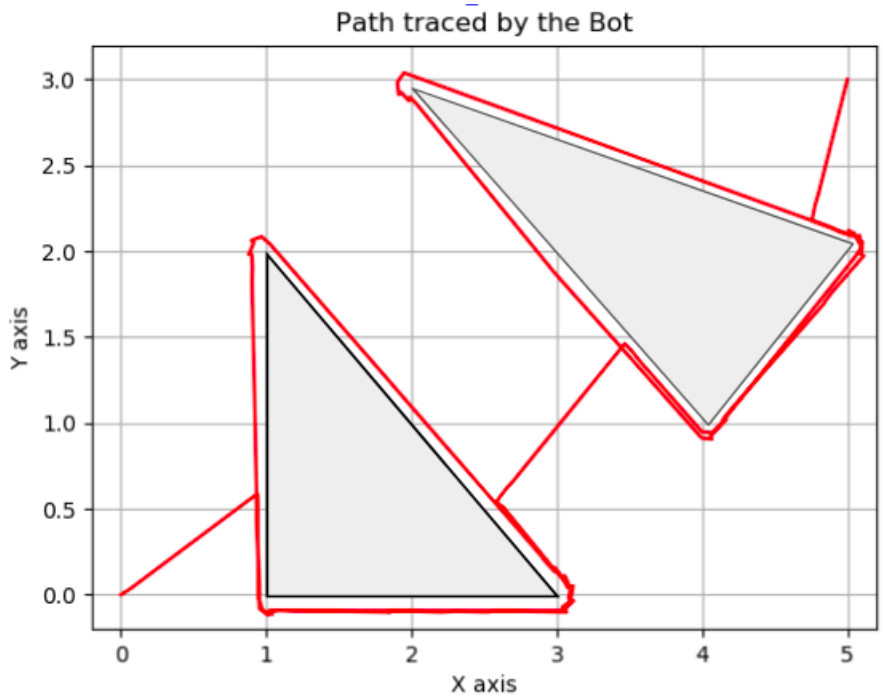



## Task 3: Bug 1 Algorithm

The following flowchart depicts the working of the Bug 1 algorithm:

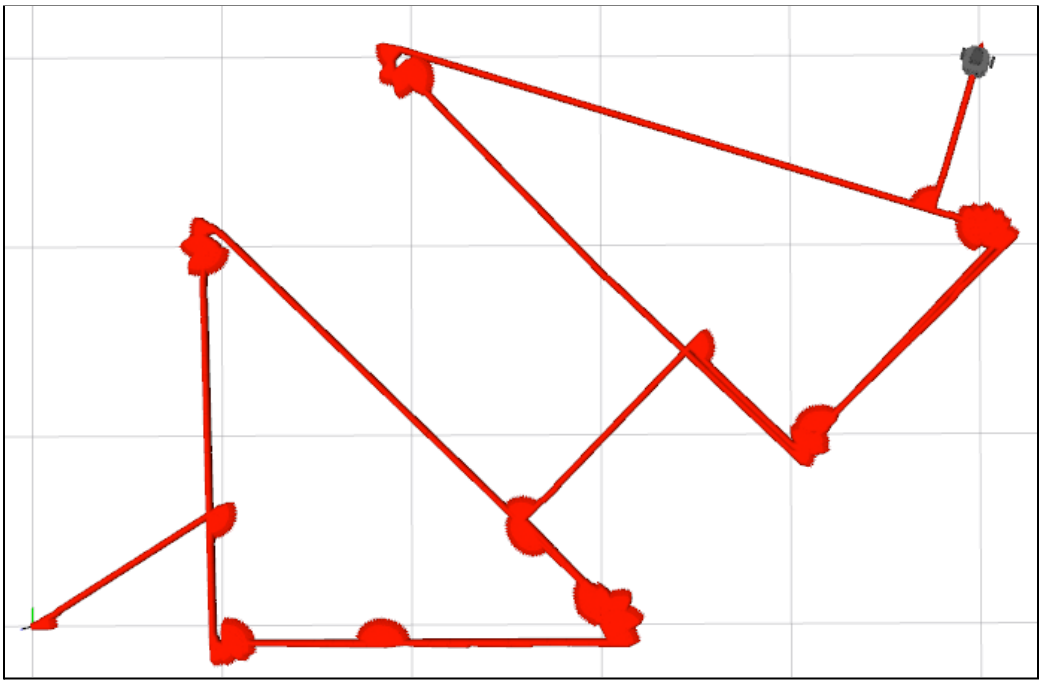


The logic in the above flow chart is used in the algorithm. The following plots show the path traversed by turtlebot:




- **Execution Time:** 820.82 sec
- **Total Path Length:** 25.5813 m

Following is the path traversed in simulation (RVIZ/Gazebo):



**Note:** The problem that I faced initially was that the bot was not following the obstacle boundary perfectly. The was a deviation from the straight path due to inaccuracies and approximations in commander code. The deviation was evident on turns where the radius increases with the turning as it is not a perfectly circular motion.
Thus I made some modifications to the algorithm such that while circumnavigating the obstacle if the distance of the bot from the obstacle is greater than step length then the bot will take half a step length perpendicularly towards the boundary of the obstacle. Thus ensuring that it remains within step length from boundary while circumnavigating, so as to detect the return point.

The following is the plot of the distance of the turtlebot from the goal during its motion: