Lab 3: Navigation and Obstacle Avoidance

Objective

To design a software system that allows a (main) thread of execution to command it to drive the robot to an absolute position on the field while avoiding obstacles, by use of the odometer and an ultrasonic sensor.

Method

- Using the navigation tutorial provided, create a class that extends `Thread` (or, if you want to be very resourceful, a class that implements `TimerListener`) to control the robot's motors to drive to a specified point on the field, given in Cartesian coordinates. Your class should at least implement the following public methods:

  - `void travelTo(double x, double y)`
    This method causes the robot to travel to the absolute field location (x, y). This method should continuously call `turnTo(double theta)` and then set the motor speed to forward(straight). This will make sure that your heading is updated until you reach your exact goal. (This method will poll the odometer for information)

  - `void turnTo(double theta)`
    This method causes the robot to turn (on point) to the absolute heading `theta`. This method should turn a *MINIMAL* angle to it's target.

  - `boolean isNavigating()`
    This method returns true if another thread has called `travelTo()` or `turnTo()` and the method has yet to return; false otherwise.

- Adjust the parameters of your controller, if any, to minimize the distance between the desired destination and the final position of the robot, while simultaneously minimizing the number of oscillations made by the robot around its destination before stopping.

- Write a program to travel to the waypoints (60, 30), (30, 30), (30, 60), and (60, 0) in that order. This will the path you demonstrate to the TA for the first half of the demo. Test this program ten (10) times, and record the position of the robot a) as reported by the odometer, and b) as measured on the field.

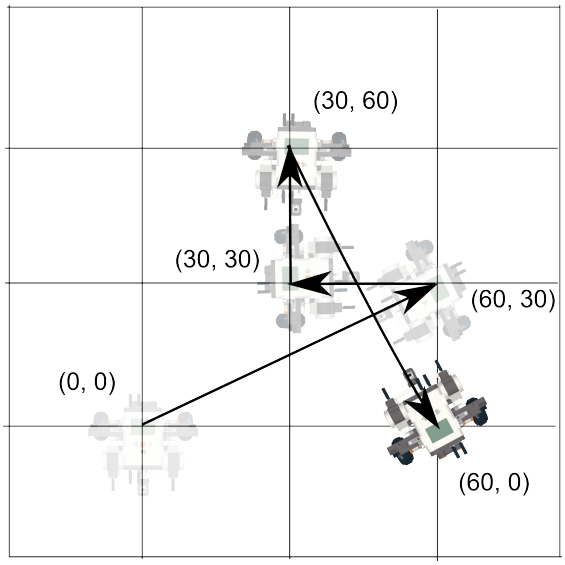

Figure 1: Robot path (to be navigated) for first part of lab.

- Modify your code to use the ultrasonic sensor to detect an obstacle (a cinder block) and avoid collision with it. You may mount the ultrasonic sensor as you wish, and may use your wall follower code if desired. Additionally, you may create another class to perform this function.

- Write a program to, with obstacle avoidance, travel to the waypoints (0, 60) and (60, 0) in that order. To demonstrate the effectiveness of your obstacle avoidance, the TA will first run your robot's program to travel that path without obstacles, then place a single cinder block somewhere along the path (though not at a waypoint), and run your robot's program again.

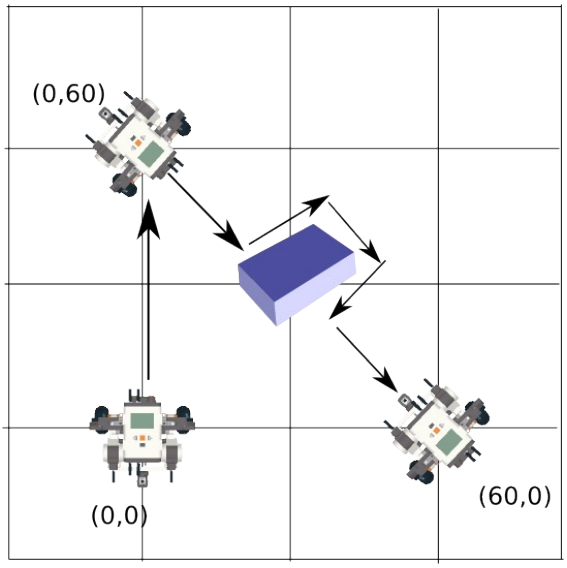

Figure 2: Robot path (to be navigated) for second part of

lab, with an obstacle avoidance example.

Data

- One table, containing the positions of the robot reported by your odometer and measured by you from the first part of the lab, as well as the error between each result, and the mean and standard deviations of the x errors and y errors. (i.e. A total of six columns: odometer x, odometer y, actual x, actual y, x error, y error).

Data Analysis

- Show your work for the computation of mean and standard deviation of the errors from the first part of the lab.
- Are the errors present as a result of the odometer or the navigator? Give reasons to back up your claim.

Observations and Conclusion

- In three to four sentences, explain the operation of your controller(s) for navigation. How accurately does it move the robot to its destination? How quickly does it *settle* (stop oscillating) on its destination? You do not need to provide a quantitative analysis.
- How would increasing the speed of the robot affect the accuracy of your navigation? What is the main source of error in navigation (and odometry)?

Further Improvements

- What steps can be taken to reduce the errors in navigation and odometry? In four (4) to six (6) sentences, Identify at least one hardware and one software solution, and provide an explanation as to why they would work.

To Submit

- One document in .pdf format containing the lab report