# Design Principles and Methods - Navigation Lab Report

Harley Wiltzer (260690006)
Juliette Regimbal (260657238)

October 13, 2016

## 1 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.

## 2 Method

- Using the nagivation tutorial provided, create a class that extends `Thread` (or, if you want to be very resourceful, a class that implements `TimeListener`) 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 its 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. Test this program ten (10) times, and record the position the robot a) as reported by the odometer, and b) as measured on the field.
- 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 trun 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.

# 3 Data

# 4 Data Analysis

# 5 Observations and Conclusion

# 6 Further Improvements

Figure 1: Final Position of Robot (cm) - Part 1

| Trial No. | x-odometer | y-odometer | x-actual | y-actual | x-error | y-error |
|---|---|---|---|---|---|---|
| 1 | 60.92 | -1.80 | 61.8 | 0.4 | -0.88 | -2.20 |
| 2 | 60.84 | -1.52 | 63.0 | -0.2 | -2.16 | -1.32 |
| 3 | 60.61 | -0.87 | 61.9 | 0.5 | -1.29 | -1.31 |
| 4 | 60.88 | -1.51 | 62.2 | -0.3 | -1.32 | -1.21 |
| 5 | 60.94 | -1.66 | 61.5 | -0.7 | -0.56 | -0.96 |
| 6 | 60.86 | -1.58 | 62.4 | -0.4 | -1.54 | -1.18 |
| 7 | 59.92 | 0.20 | 61.0 | 1.6 | -1.08 | -1.40 |
| 8 | 60.60 | -0.65 | 61.2 | 0.3 | -0.60 | -0.95 |
| 9 | 61.04 | -1.86 | 61.5 | -1.2 | -0.46 | -0.66 |
| 10 | 61.08 | -2.04 | 63.0 | -0.2 | -1.92 | -1.84 |
| Mean | 60.77 | -1.33 | 61.95 | -0.02 | -1.18 | -1.30 |
| Standard Deviation | | | | | | |