IEOR 140 Project 3 Final Report - 10/05/2012
Team 4: Nate Bailey and Raymond Ma

Responsibilities
In this project, Nate was in charge of program design and coding. Raymond was in charge of hardware design and project writing.

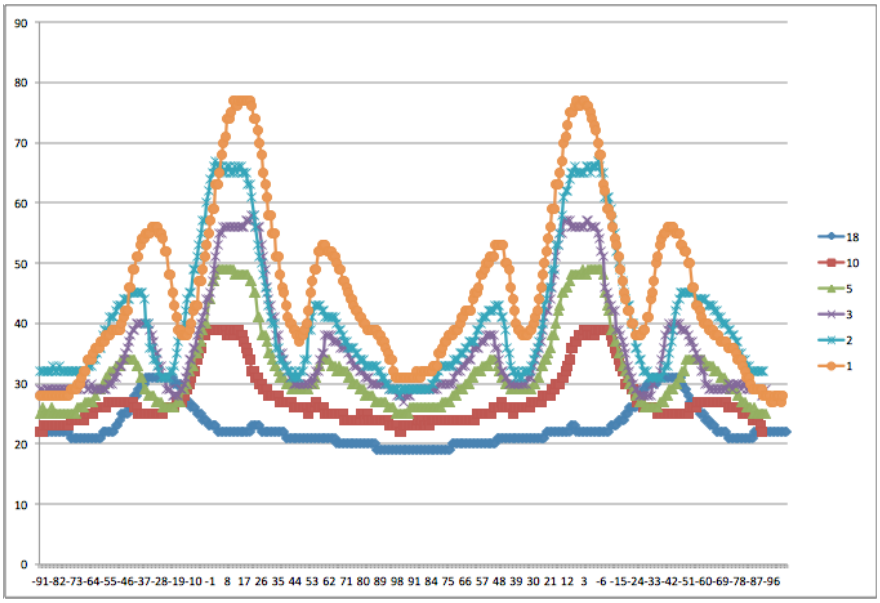
Hours Spent
Approximately 20 hours of work

Project Code
Our project code can be found on GitHub at https://github.com/ieor140-team4/Project3

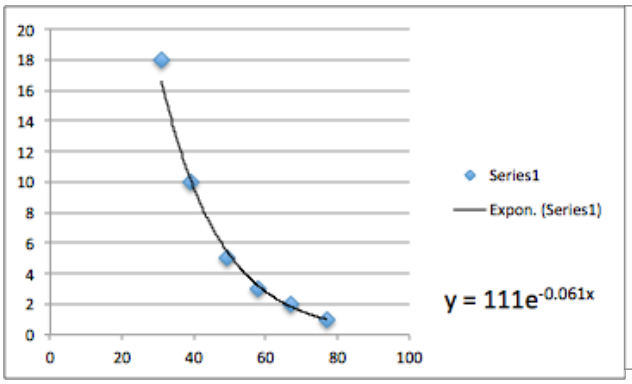
Program Specifications
We met all of the program specifications for this project (there were no bonus specifications). We also were the winners of the Great Race. Thanks for the cookies! :D

Hardware Design
Our robot was designed as a single drive robot with a castor wheel in the back. The sensors on our robot included two touch sensors, a light sensor, and an ultrasonic senor. The touch sensors were mounted on the front of the robot with attached bumpers to trigger them and the light sensor and ultrasonic sensor were attached to a motor and able to pan left and right, acting as the “head” of the robot. Touch sensors were used to detect obstacles that the ultrasonic was unable to pick up and the light sensor was used to detect and navigate toward a light on either end of the obstacle course.

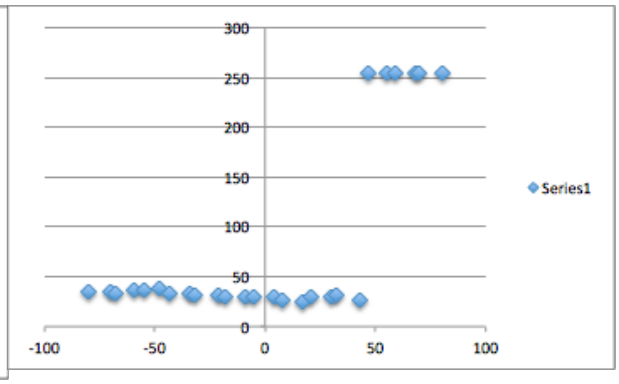
Experimental Work



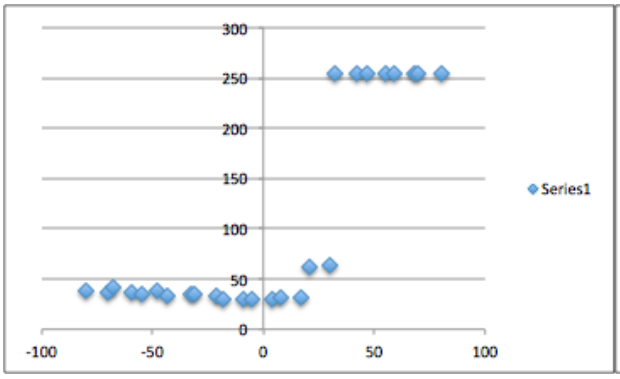
Light value as a function of angle test (at different distances from the light)



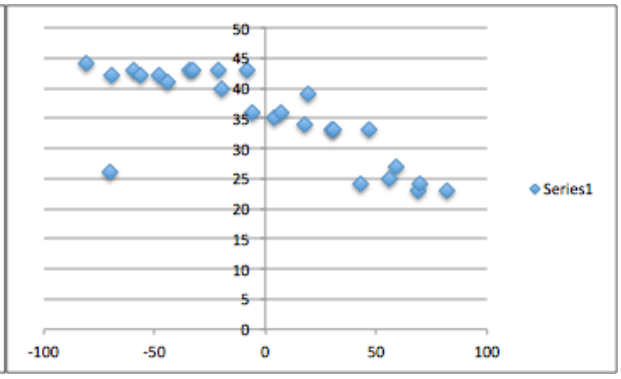
Peak graph of the highest point at each distance.



Obstacle tests using ultrasonic



Obstacle test using ultrasonic



Obstacle tests using ultrasonic

Problem Analysis

- **Control Scheme**
  To find the light, our robot does a scan of values for a cone area in front of it and find the highest light value in the range. Then, with the maximum light value, it find the angle of that reading and tells the Mover to move toward that angle, using a gain constant of ⅓. To make sure that the robot doesn't oscillate, the robot is programmed to turn no more than to the given angle so that it does not overturn in the time between successive scans. This process is repeated over and over until it reaches close to the light, which was determined by the experimental work we tested.
- **Obstacle Detection**
  The job of ObstacleDetector is to look to see if there is an obstacle in the path of the robot. Upon finding an object within the specified distance and angle thresholds (25 cm in front of the robot and within 15 degrees of the head), it calls any attached obstacle listeners' obstacle found methods. This means it sends the information back to the Navigator which then implements any obstacle avoiding algorithm.
- **Avoid**
  The Avoid algorithm calls a scan to the left and right to determine where the sides of the obstacle are (and the angle). It then picks the side with the smaller angle and turns 90 degrees in that direction, moves forward, and turns back again. If obstacle is detected all the way to 90 degrees on both sides of the robot, it back up and do another scan to find an angle.

Software Design

- Navigator

    - Takes input from the Sensor and passes it to the Mover.
    - Figures out when the robot is close enough to the light to stop and turn.
    - Controls the scanning / moving by having iterative scans between move commands
- Scanner
    - Finds both the angle and the distance to the light source
- Mover
    - Uses Pilot to steer towards light
    - Turns around when the robot gets close to the light
- ObstacleDetector
    - Uses an algorithm to detect objects
    - If one is detected, it passes that information to the Navigator to then initiate evasive action
- Avoid
    - Uses an algorithm to decide what to do when avoiding an obstacle
    - Evades obstacle and continues

Documentation
http://htmlpreview.github.com/?https://raw.github.com/ieor140-team4/Project3/master/doc/essentials/package-summary.html

Interesting/Challenging/Difficult
The most interesting part of this project was the Great Race. It was very interesting to see how the robots would act when faced with the moving obstacles that other robots became. Sometimes, there would be a blocking game occurring when other times they just got into each other's way. The most challenging and difficult part of this project was writing the logic to avoid when the avoid algorithm needed to avoid more than the simple case (such as hitting another obstacle while avoiding the first one or becoming trapped by one obstacle). The way we overcame this was to completely rewrite our avoidance algorithm to make is be able to work with more corner cases of obstacle avoidance.

Appendix
Source Code:
https://github.com/ieor140-team4/Project3

Documentation: http://htmlpreview.github.com/?https://raw.github.com/ieor140-team4/Project3/master/doc/essentials/package-summary.html