**IEOR 140 Final Project Final Report - 11/30/2012**

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, experimental work, and project writing.

**Hours Spent**

Approximately 50 hours were spent on this final project.

**Project Code**

https://github.com/ieor140-team4/FinalProject

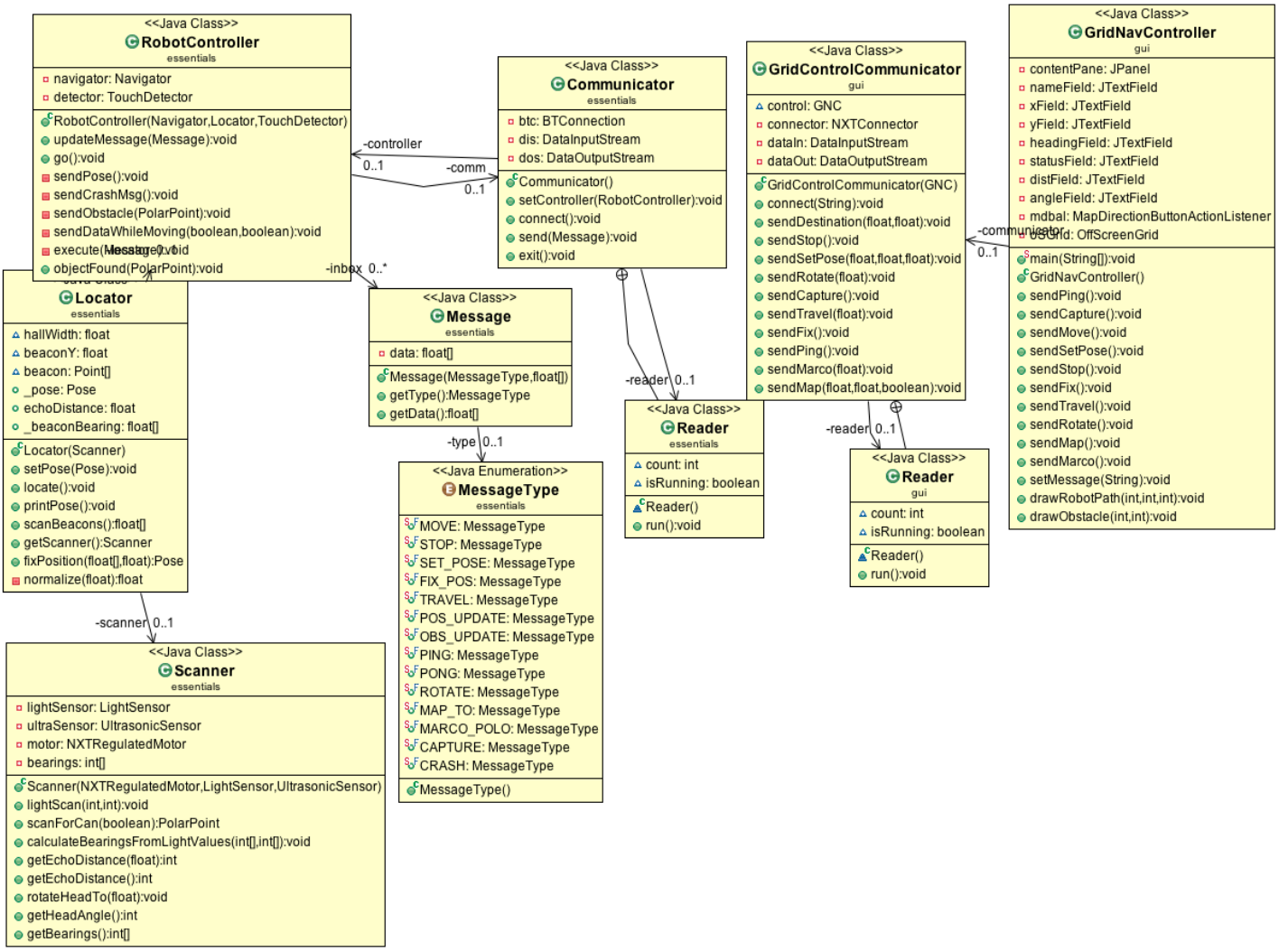
**Performance Specifications**

Our robot met all of the performance specifications (there were no bonus specifications to meet).

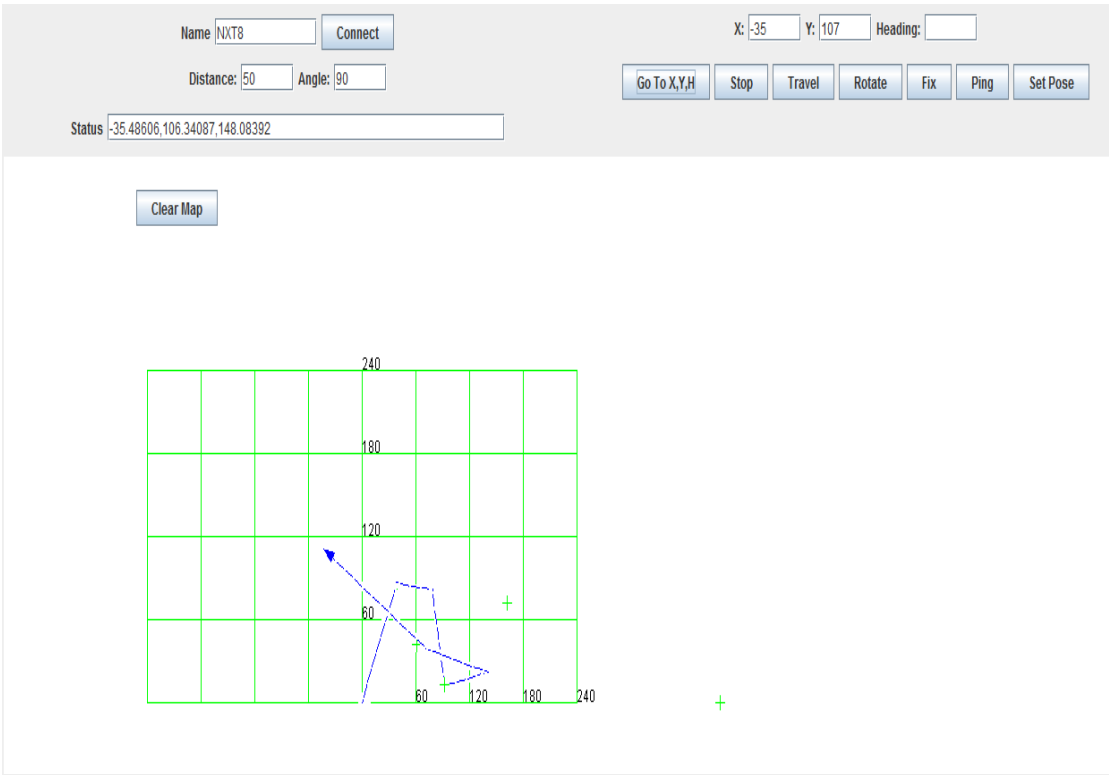
**Experimental Work**

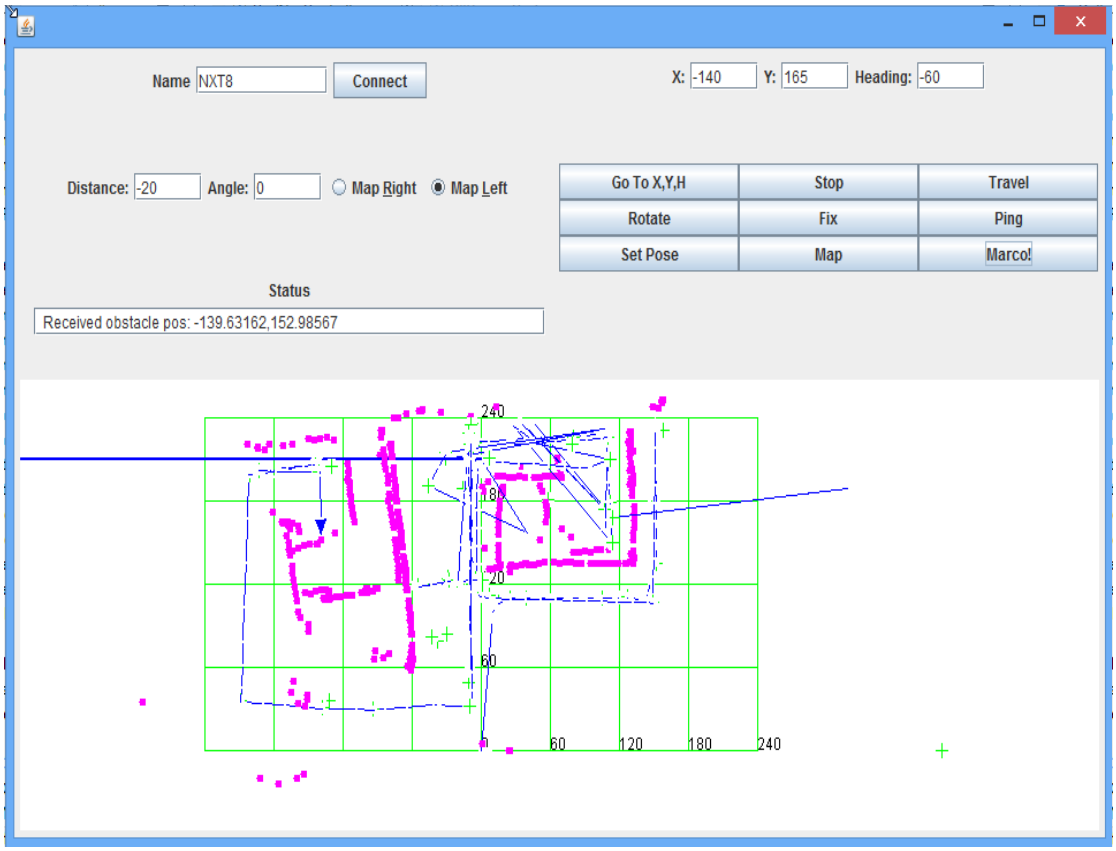
Light sensor and ultrasonic tests
Light sensor tests

## Class Diagram



## Screenshots



**Task/Problem Analysis**

- Use the motor to rotate the sensor so that it can take various light readings at different angles.
- Take light readings at each angle
- Use logic to determine what light readings actually correspond to beacons
    - Ignore all values underneath a threshold light value (42 in our case, selected from our experimental work.)
    - Once the light value passes underneath the threshold after already seeing a peak and storing a highest light value, we know that we've passed the first peak and that the next highest light value we see will be the second peak.
- Collect data from Scanner
- Calculate Pose
- Communicator receives a message from the computer over bluetooth
- Communicator then adds message to the RobotController's inbox
- RobotController will pull the oldest message from the inbox, run that command, and then deletes the message
- If a stop command is issued, the inbox is cleared and the stop command is run immediately with all waypoints cleared
- Map Left - turn the head to 90 degrees from the front of the robot and move to the given coordinates while mapping
- Map Right - turn the head to -90 degrees from the front of the robot and move to the given coordinates while mapping
- Marco - pings the given angle to determine how far away obstacles are in that direction.
- Detect distance to can
- Turning so the back of robot is facing can
- Backing up until can connects with robot

**Class Responsibilities/Software Design**

Milestone 2 was performed pretty much entirely by the Scanner class in NXT Files. The scanner was in charge of rotating the motor associated with the light sensor, taking in readings from the light sensor at each angle, and using logic to determine how the light readings we observed translated into beacon headings.

In milestone 3, the Scanner had the added task of being able to detect the Ultrasonic distance. This milestone was performed pretty much entirely by the Locator class in NXT Files. The Locator was in charge of using the Scanner class to scan for the distance to the wall and the angles to the lights and then uses this information to calculate its Pose.

In milestone 4, the Communicator class is in charge of receiving messages from the computer over bluetooth and putting them in the RobotController's inbox. The RobotController is tasked with reading the oldest inbox message and executing the message at which point it will clear the task from the inbox. Depending on the message type, the RobotController will send the command to a different class. In the case of a MOVE command, the RobotController will send

x and y to the Navigator to move to the given coordinates. A TRAVEL command will give the move distance to the Differential Pilot in Navigator which will move the robot forward the given distance. A ROTATE command will send the desired number of degrees to turn to the Differential Pilot in Navigator once again. A FIX_POS command will activate the Locator which will use the Scanner to determine its current position from the two lights. The SET_POSE manually sets the pose in both Locator and Navigator. The special case of a stop command will clear the inbox of all messages and run the stop command (which also clears all waypoints).

In milestone 5, Map Left, Map Right, and Marco buttons were added to GridNavController. On the robot, two cases of MAP_TO and MARCO_POLO were added. MARCO_POLO called the Scanner in Locator which then finds an obstacle in the given direction the scanner head was pointed. For MAP_TO, the robot will rotate its head to the angle given using rotateHead in Scanner in Locator and then goes to the given coordinates by sending them to Navigator. While moving, the robot will send its pose back to the GUI in order to map it on the Grid as well as scanning for obstacles every 50ms using getEchoDistance in Scanner in Locator. These are also sent back to the GUI.

In milestone 6, a new case was added in RobotController called CAPTURE. CAPTURE first scans for the can's location by using scanForCan in Scanner in Locator. The angle is passed the DifferentialPilot to rotate the robot so the back faces the can. Now, the robot scans again for the can to make sure it is aligned by using scanForCan in Scanner in Locator. Then, the distance is passed to the Navigator to travel the backwards distance to connect to the can.

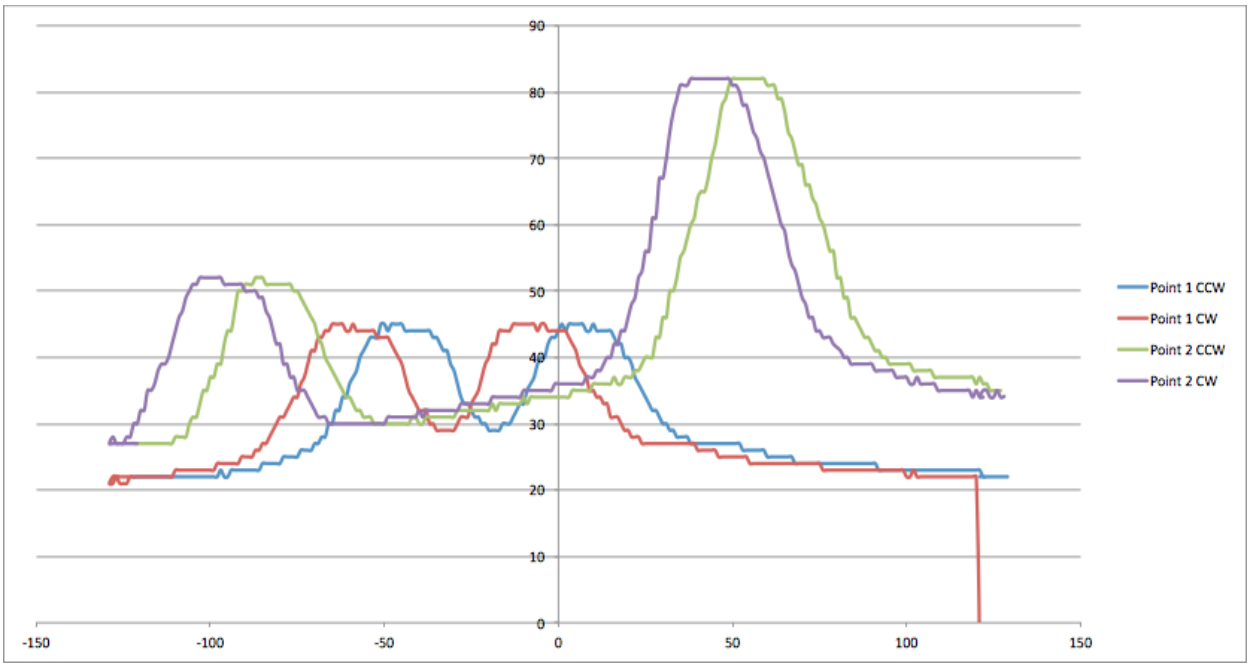
**Interesting/Challenging/Difficult**

The most interesting part of this project was getting the data that was received from the robot into an easy to read format of our GUI. The most challenging part of this project was while mapping, knowing which items were obstacles such as walls and barricades and which objects we wanted to interact with, such as the can. The most difficult part of this project was designing the logic to figure out the correct and most accurate way of calculating our Pose from the two light beacons.

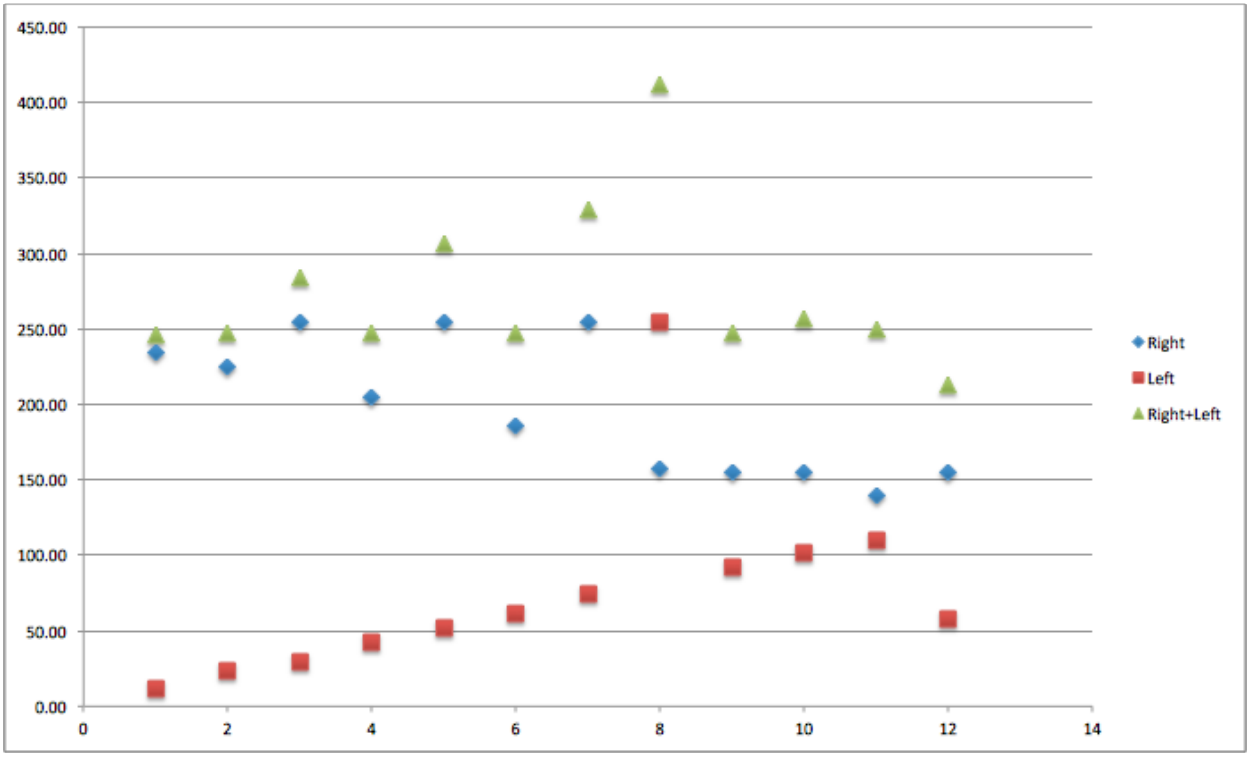
**Appendix**

Source Code | Java Docs

Light Sensor and Ultrasonic Tests




Light sensor data with Point 1 being the far away point (240, 30) and Point 2 being the close point at (20, 20).




Ultrasonic data gathered starting 10cm from the way and moving over into the middle of the hallway in 10 cm intervals. The Green dots are the added left and right points, giving us the total width of the hallway. Most readings give us an average of 248 (which is the the actual width), but we have some major outliers, which brings questions of consistency.

Light Sensor Tests

| (-30, 35) | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Heading 0 | | | Heading 90 | | | Heading 180 | | | Heading -90 | | | |
| X | Y | H | X | Y | H | X | Y | H | X | Y | H | |
| -24.098 | 36.934 | 0.687 | -24.188 | 36.5 | 89.931 | -28.75 | 34.844 | 178.377 | -28.138 | 35.5 | | |
| -28.186 | 36.662 | 3.526 | -26.484 | 36.5 | 90.259 | -28.748 | 34.94 | 179.377 | -28.903 | 35.499 | | |
| -25.424 | 37.258 | -2.691 | -24.975 | 36.5 | 89.71 | -28.748 | 34.94 | 179.377 | -27.292 | 35.498 | | |
| -24.755 | 37.24 | -2.501 | -26.484 | 36.5 | 90.259 | -29.656 | 34.916 | 179.127 | -28.138 | 35.5 | | |
| -31.853 | 36.474 | 5.486 | -27.302 | 36.5 | 90.029 | -27.859 | 34.868 | 178.623 | -28.138 | 35.5 | | |
| -26.795 | 37.007 | -0.078 | -25.772 | 36.5 | 89.485 | -27.859 | 34.868 | 178.623 | -27.292 | 35.498 | | |
| -31.853 | 36.474 | 5.486 | -18.047 | 20.504 | 92.058 | -28.748 | 34.94 | 179.377 | -28.138 | 35.5 | | |
| -24.755 | 37.24 | -2.501 | -26.58 | 36.5 | 89.259 | -28.748 | 34.94 | 179.377 | -28.138 | 35.5 | | |
| Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | A | |
| -27.2149 | 36.91113 | 0.92675 | -24.979 | 34.5005 | 90.12375 | -28.6395 | 34.907 | 179.0323 | -28.0221 | 35.49938 | | |
| Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | S | |
| 3.143426 | 0.336064 | 3.504766 | 2.971213 | 5.65544 | 0.857823 | 0.574572 | 0.04044 | 0.422359 | 0.522253 | 0.000916 | | |
| | | | | | | | | | | | | |
| Overall: | | | | | | | | | | | | |
| Average X: | | -27.2139 | | | | | | | | | | |
| Average Y: | | 35.4545 | | | | | | | | | | |
| Std Dev X: | | 2.518496 | | | | | | | | | | |
| Std Dev Y: | | 2.847518 | | | | | | | | | | |
| | | | | | | | | | | | | |
| (240, 185): | | | | | | | | | | | | |
| Heading 0 | | | Heading 90 | | | Heading 180 | | | Heading -90 | | | |
| X | Y | H | X | Y | H | X | Y | H | X | Y | H | |
| 239.392 | 177.711 | 3.012 | 244.043 | 176.565 | 98.809 | 219.424 | 175.154 | -167.892 | 236.271 | 174.483 | | |
| 233.48 | 177.743 | 2.681 | 243.948 | 176.551 | 97.809 | 219.424 | 175.154 | -167.892 | 212.222 | 217.341 | | |
| 233.48 | 177.743 | 2.681 | 243.948 | 176.551 | 97.809 | 224.076 | 175.353 | -176.319 | 236.271 | 174.483 | | |
| 239.392 | 177.711 | 3.012 | 250.139 | 176.555 | 98.124 | 229.195 | 175.387 | -175.965 | 235.903 | 175.481 | | |
| 233.48 | 177.743 | 2.681 | 211.715 | 176.638 | 102.848 | 214.325 | 175.38 | -176.038 | 236.271 | 174.483 | | |
| 233.48 | 177.743 | 2.681 | 250.139 | 176.555 | 98.124 | 251.562 | 175.04 | -179.579 | 236.271 | 174.483 | | |
| 239.392 | 177.711 | 3.012 | 250.139 | 176.555 | 98.124 | 229.201 | 175.291 | -176.965 | 236.271 | 174.483 | | |
| 239.392 | 177.711 | 3.012 | 226.942 | 176.581 | 99.845 | 224.076 | 175.353 | -176.319 | 236.271 | 174.483 | | |
| Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | Avg | A | |
| 236.436 | 177.727 | 2.8465 | 240.1266 | 176.5689 | 98.9365 | 226.4104 | 175.264 | -174.621 | 233.2189 | 179.965 | | |
| Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | Std Dev | S | |
| 3.160097 | 0.017105 | 0.176927 | 13.76508 | 0.029657 | 1.718157 | 11.35605 | 0.130675 | 4.31303 | 8.484996 | 15.10622 | | |
| | | | | | | | | | | | | |
| Overall: | | | | | | | | | | | | |
| Average X: | | 234.048 | | | | | | | | | | |
| Average Y: | | 177.3812 | | | | | | | | | | |
| Std Dev X: | | 10.80071 | | | | | | | | | | |
| Std Dev Y: | | 7.390092 | | | | | | | | | | |

From our experimental work, we discovered that the values deviated more when we were farther away from the lights as opposed to when we were closer to the lights. We have some large standard deviations when we are further away from the lights, but this is most likely caused by one bad reading out of multiple good readings.