**Final Project - Milestone 3**

1. **Introduction -- Prototype of Bomb Disposal Robot.**

1.1. Name of the assignment: Milestone 3 Localization -- the goal of this milestone is to demonstrate and verify the accuracy of our Locator.

1.2. Team Number 5. Team members include Khoa Tran (Programmer), Phuoc Nguyen (Programmer), Trevor Davenport, Corey Short

1.3. Approximate number of person hours spent on the project: 20 Hours.

1. **Experimental work (if any):**

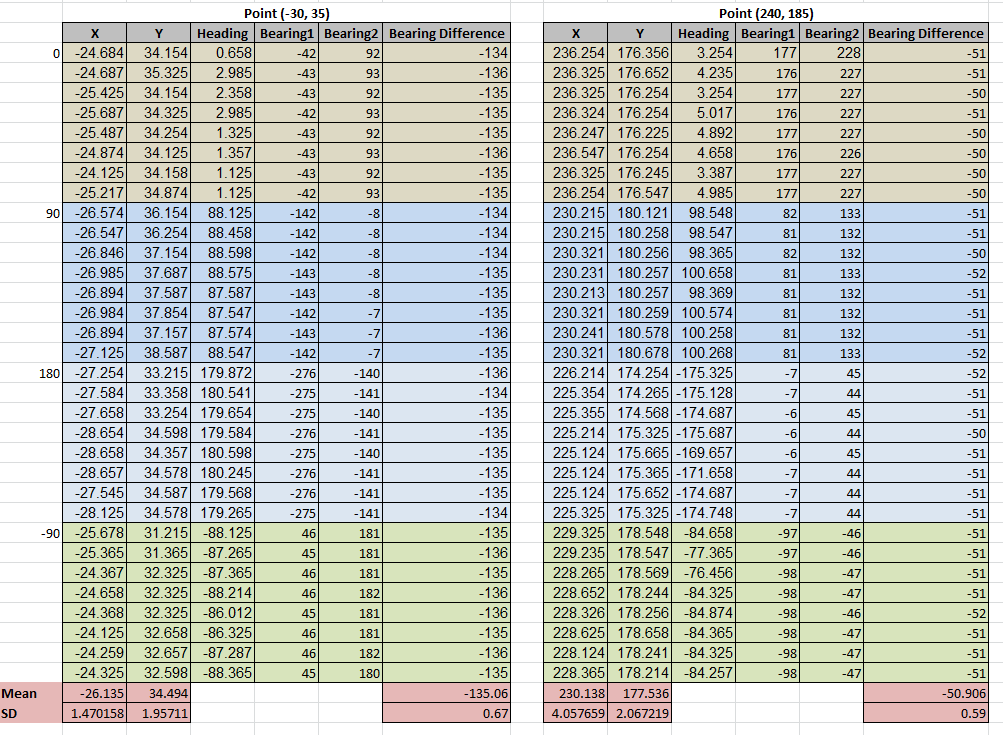
2.1. Experiment description and purpose: Coding and experimental work in preparation for final project. Design a scanning strategy and verify the accuracy of the Locator and its Scanner. In the final, our robot will start its mission at a known location and facing a known direction. That location and heading are saved in the Pose object. However, when the robot moves all the error will being built up and it will not be accurate. Our task is to code a Locator class in which we will update the location of the robot so whenever it goes, it will update the location. That should make the position more accurate.

2.2. Calculation and Analysis:

In this Milestone, we are trying to calculate the location of the robot based on the 2 bearings to the beacons and the distance to the wall. Given the width of the Hall (by our ultrasonic report, it is approximately 240-241 and we choose to use 241 for this Milestone). By using the basic trigonometry we could be able to calculate the position of the robot and set it to the pose, along with the heading.

2.3. How results were used in your code: The result will then be saved in pose along with the current heading. Based on that, we know the location of the robot and the heading it is. It is really useful for the next following milestones as we need to use that location to map and direct the robot to the bomb and get out.

2.4. Listing of data: (data can be found in MS3 - Data Chart.xml)

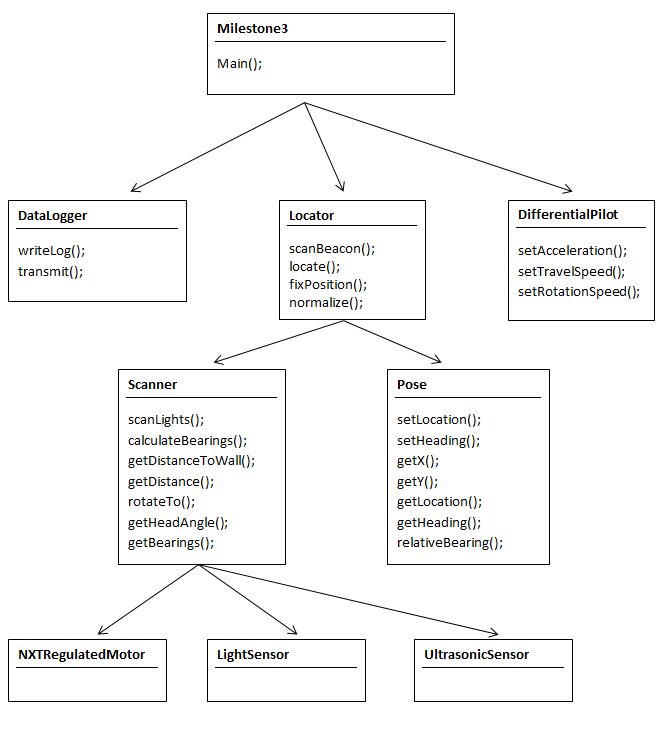


2.5. Classes and Responsibilities:

The milestone3 class is the main class that gathers up other classes that are used to finish this Milestone. Including 2 main classes:

* Locator: This class will use Pose object as an object to save the current location and heading of the robot. Then the method locate() is called, it will calculate the angle to both of the walls (North and South). Based on the current Y-coordinate, if it is near the North Wall then we are going to use the calculation based on the distance to the North wall. Otherwise, we will use the value based on the South Wall. The method will then save the bearings to both beacons, along with current X, Y coordinates. All the information will be passed into FixPosition() method, it is when the current location is calculated and saved back to the Pose object. The FixPosition() method plays a role as our calculator of the current location.
* Scanner: this class also plays an important role in this milestone. The role of it is to calculate the distance to the wall (distanceToWall() method). The scanLights() method will scan for the 2 lights and return their bearings to be used in Locator class. We perform 2 scans in the range of -90 to 90 degrees, then save the value of each beacon twice. By doing that, we can take the mean of the bearing to get the exact value for it.

**Class Dependency Diagram:**

[](https://raw.github.com/IEOR140-T5/FinalProject-NXT/master/reports/MS3_class_diagram.png)

2.6 Software design:

In order to scan for the beacons, we first decide which wall is nearer to the robot based on the current Y location, then that wall will be chosen to calculate other things. When the scanner is called, it will first scan and decide which angle to scan. No matter which heading it is, it will always scan from -90 to 90 in the heading that the 2 lights are. Then the bearings to the angle will be returned to the locator and those data will be calculated for the current location. The formula is given as:



B2

y2

y1

x

a22

a1

B1

R

We find the a1, a2 and we know that c = a1 + a2, with one of the B is 0 and the other is beaconY; Then by using Pythagorean we are able to find the quadratic formula for x. because there 2 solution for X so depends on where the robot is and the different angle, we can choose which solution is the correct one.

Based on the calculated value of X, Y coordinate, we can compare it to the right value to determine how accurate our robot is.

We used the data and algorithm from the second milestone for this project. It helps us to get the right values for both beacons and the distance to the wall that we want to have.

1. **The most interesting/challenging/difficult parts of the project:**

There are a lot of difficult parts in this milestone.

* To decide how to make the scanner always scan in the heading that have 2 lights, no matter what its current heading is.
* To calculate the exact location of the robot. We can never get the right value for the location, mostly because when the robot rotate to another heading, the location of the axis is changed that makes the angles to 2 beacons change. We first put the axis on the top of the point, that explains why the standard deviant for all of the values are really big. X and Y values will change according to the heading of the robot and the position of the axle.
* One difficult part of this milestone is to come up with the right formula to calculate the current location. Because the value is rounded, and we have to cast (float) to (int) or vice versa many times, it results in the loss of value.

The most challenging part of this milestone is the hardware. At first the threshold for light values keep changing for us, both of the lights don’t have the same values so it was really difficult to code the scanner, especially when it is at a far location. Moreover, when we call rotate for both the scanner and the brick, sometimes it doesn’t rotate the exact degree that we want. (let’s say if we want to turn 90 degree we now need to have a tuning factor in order to get to the right angle, same thing for the scanner). If we make the scanner go fast, then the value is not accurate. On the other hand, if we make it go slower, it will shake and doesn’t get the right value either. Our task was to find a right speed, acceleration for it in order to make it work accurately.

1. **Links to source code and JavaDocs:**

Steam/Team5/FinalPoject-NXT/Milestone 3/