ROBOT WITH LOCALIZATION– OVERVIEW AND ANALYSIS

**Design Evaluation**

Our robot is equipped with 2 large EV3 motors. In deciding between the EV3 and NXT motors, we evaluated that the precision and accuracy provided by the EV3 motor would be much more appreciated for a lab that required a high precision for measurements. The major hardware decision made during this lab regarded the positioning of the ultrasonic and light sensors. The light sensor was positioned at the back of our robot as this would make it easier to detect black lines when rotating in place during localization. The ultrasonic sensor was positioned at the front of the robot at 0 degrees, as this would give us a very good line of sight when detecting the walls. Also, it was placed very low, as this would make sure no false positives were detected. Furthermore, a beam runs across the back of the robot to deal with the wiring of the motors. Lastly, a gyro ball was added at the middle of the back of the brick to support the front wheels. Our robot was equipped with ability to navigate to certain waypoints and avoid objects as well. Figure 1 visually depicts the outlined hardware design.

*Figure 2: Hardware Design of Robot*



*Ultrasonic Sensor*

*Light Sensor*

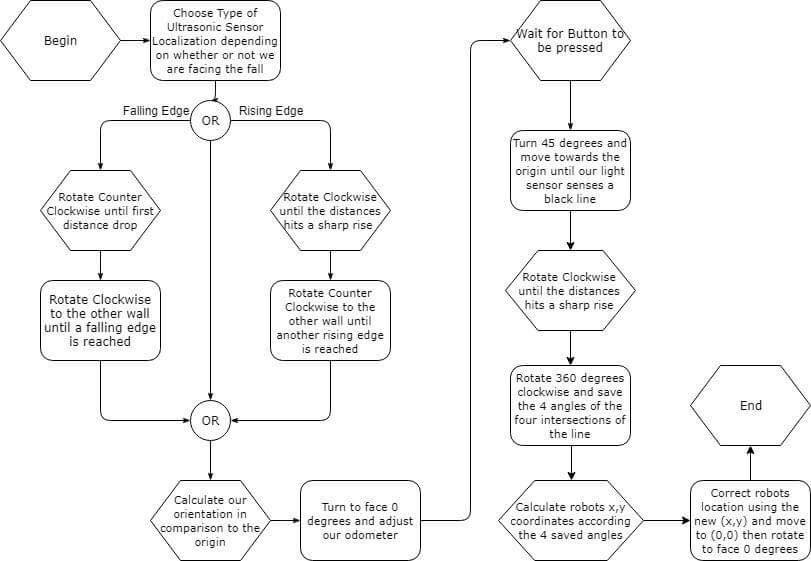
**Workflow**

The initial step for this particular lab was to determine the positioning of the two sensors. Mr Jay is the hardware expert within our group and he handled this process. Following this, both Mr Jurisic and Mr Jay proceeded to outline the pseudocode for the Ultrasonic sensor localization. Both members then performed rigorous testing of both the rising and falling edge methods for this localization. Once both members were satisfied with the robot’s performance and had collected sufficient data for the lab report, the same sequence of steps was repeated for the light localization method. Following the completion of the software and hardware components, both members assisted in the writing of the lab report, with Mr Jurisic being the lead during this final task.

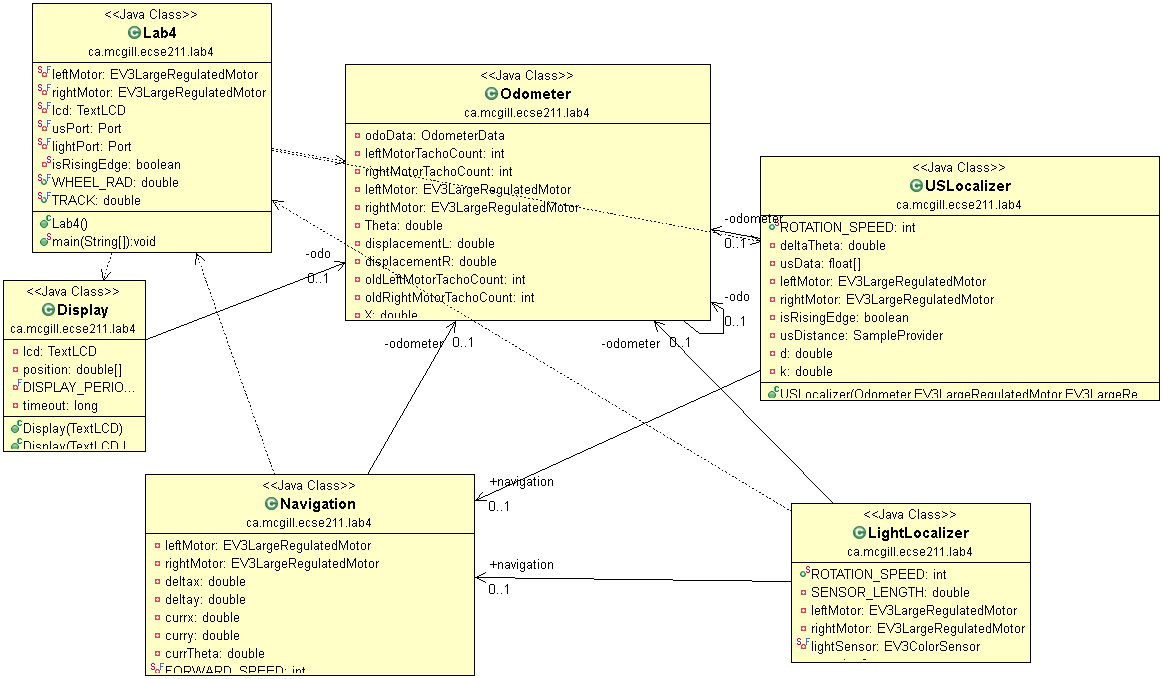
**Software**

The software design of our robot involved 2 sub problems. Firstly, we needed to utilize our ultrasonic sensor to allow the robot to figure out its orientation when placed along a diagonal in the first quadrant of the 4x4 grid. Secondly, once it had successfully aligned itself with a 0-degree heading, we needed to use the light sensor to make the robot traverse itself from its current position to the origin of the quadrant, (0.0). Figures 2 and 3 below summarize the functionality of our software design.

*Figure 2: Outline of Software Functionality*



*Figure 3: Class Diagram*



**Test Data and Analysis**

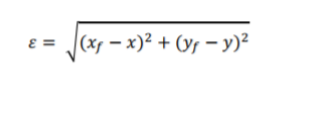
To test the accuracy of the ultrasonic sensor localization, we measured the final heading of the robot against the expected value of theta, which is 0 degrees. Table 1 details the actual angle recording during 10 trial runs of both methods, rising and falling edge, and the error in angle is calculated simply as

*Table 1: Ultrasonic Localization Results*

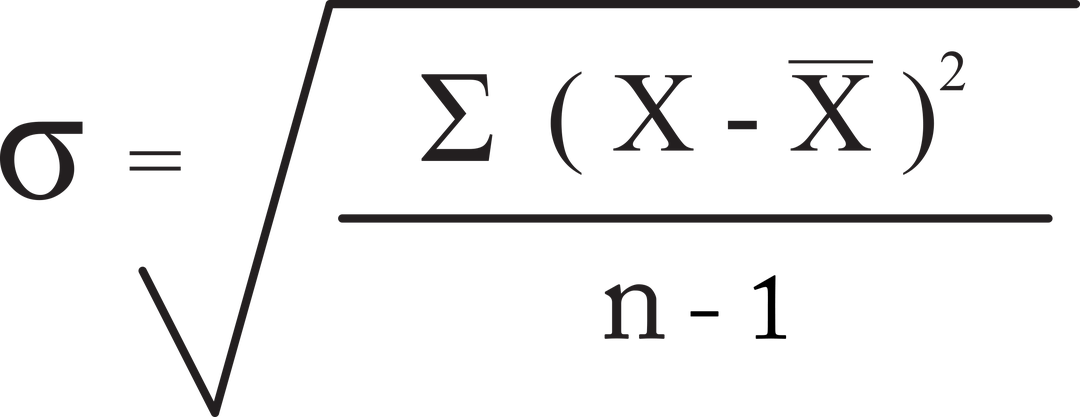
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Falling Edge Ultrasonic Localization** | | | | **Rising Edge Ultrasonic Localization** | | |
| **Run #** | **Expected angle (**°**) ± 0.05**° | **Actual Angle (**°**) ± 0.05** ° | **Error in Angle (**°**) ± 0.05** ° | **Expected angle (**°**) ± 0.05** ° | **Actual Angle (**°**) ± 0.05** ° | **Error in Angle (**°**) ± 0.05**° |
| 1 | 0.0 | 2.34 | 2.34 | 0.0 | 2.04 | 2.04 |
| 2 | 0.0 | 1.14 | 1.14 | 0.0 | 2.56 | 2.56 |
| 3 | 0.0 | 1.55 | 1.55 | 0.0 | 3.04 | 3.04 |
| 4 | 0.0 | 1.54 | 1.54 | 0.0 | 3.08 | 3.08 |
| 5 | 0.0 | 0.53 | 0.53 | 0.0 | 2.52 | 2.52 |
| 6 | 0.0 | 1.12 | 1.12 | 0.0 | 2.59 | 2.59 |
| 7 | 0.0 | 2.03 | 2.03 | 0.0 | 3.03 | 3.03 |
| 8 | 0.0 | 2.57 | 2.57 | 0.0 | 2.05 | 2.05 |
| 9 | 0.0 | 358.54 | -1.46 | 0.0 | 2.57 | 2.57 |
| 10 | 0.0 | 1.43 | 1.43 | 0.0 | 357.10 | -2.90 |

To test the accuracy of the light sensor localization method, we measured the final x, y and theta co-ordinates and compared them against the expected values.

Notice that that the Euclidean Error and mean have already been calculated and included in table 2. The error was calculated by using the formula below, where xf and yf are the co-ordinates of the final position of the robot and x and y are the actual co-ordinates displayed by the odometer.



By calculating the Euclidean error in this manner, it is easy to analyse the accuracy of the light localization method. Similarly, calculating the standard deviation of the error in x, the error in y and the error e for the trials serves as an effective marker as to the consistency in the readings. The expression for standard deviation used is as follows:



Where σ is the standard deviation, xi an element of the data set, u is the average value of the data set and n the number of elements in the data set.

*Table 2: Falling Edge Light Localization: X and Y co-ordinate results*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Falling Edge Light Localization** | | | | | | | |
| **Run #** | **Theoretical X XA ± 0.05 cm** | **Theoretical Y YA ± 0.05 cm** | **Actual X XA ± 0.05 cm** | **Actual Y YA ± 0.05 cm** | **Error X EX ± 0.05 cm** | **Error Y EY ± 0.05 cm** | **Euclidean Error ± 0.05 cm** |
| 1 | 0.0 | 0.0 | 1.00 | 0.57 | 1.00 | 0.57 | 1.151043 |
| 2 | 0.0 | 0.0 | 1.57 | 1.23 | 1.57 | 1.23 | 1.994442 |
| 3 | 0.0 | 0.0 | 1.45 | 0.45 | 1.45 | 0.45 | 1.518223 |
| 4 | 0.0 | 0.0 | 1.75 | 0.34 | 1.75 | 0.34 | 1.782723 |
| 5 | 0.0 | 0.0 | 0.55 | 0.78 | 0.55 | 0.78 | 0.954411 |
| 6 | 0.0 | 0.0 | 1.24 | 0.80 | 1.24 | 0.80 | 1.475669 |
| 7 | 0.0 | 0.0 | 0.54 | 1.12 | 0.54 | 1.12 | 1.243382 |
| 8 | 0.0 | 0.0 | 0.21 | 1.70 | 0.21 | 1.70 | 1.712921 |
| 9 | 0.0 | 0.0 | 1.47 | 1.32 | 1.47 | 1.32 | 1.975677 |
| 10 | 0.0 | 0.0 | 1.34 | 1.10 | 1.34 | 1.10 | 1.733667 |
|  |  |  | **Mean (cm)** | | 1.112 | 0.941 | 1.554216 |
|  |  |  | **Standard Deviation (cm)** | | 0.516049 | 0.427693 | 0.350782 |

*Table 3: Light Localization: X and Y co-ordinate results*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Rising Edge Light Localization** | | | | | | | |
| **Run #** | **Theoretical X XA ± 0.05 cm** | **Theoretical Y YA ± 0.05 cm** | **Actual X XA ± 0.05 cm** | **Actual Y YA ± 0.05 cm** | **Error X EX ± 0.05 cm** | **Error Y EY ± 0.05 cm** | **Euclidean Error ± 0.05 cm** |
| 1 | 0.0 | 0.0 | 1.05 | 1.54 | 1.05 | 1.54 | 1.863894 |
| 2 | 0.0 | 0.0 | 1.14 | 1.25 | 1.14 | 1.25 | 1.691774 |
| 3 | 0.0 | 0.0 | 0.87 | 0.75 | 0.87 | 0.75 | 1.148651 |
| 4 | 0.0 | 0.0 | 0.91 | 1.20 | 0.91 | 1.20 | 1.506021 |
| 5 | 0.0 | 0.0 | 1.43 | 1.75 | 1.43 | 1.75 | 2.259956 |
| 6 | 0.0 | 0.0 | 1.63 | 0.64 | 1.63 | 0.64 | 1.751142 |
| 7 | 0.0 | 0.0 | 0.98 | 1.36 | 0.98 | 1.36 | 1.676305 |
| 8 | 0.0 | 0.0 | 2.16 | 1.43 | 2.16 | 1.43 | 2.590463 |
| 9 | 0.0 | 0.0 | 1.19 | 1.27 | 1.19 | 1.27 | 1.740402 |
| 10 | 0.0 | 0.0 | 1.24 | 0.93 | 1.24 | 0.93 | 1.55 |
|  |  |  | **Mean (cm)** | | 1.26 | 1.212 | 1.777861 |
|  |  |  | **Standard Deviation (cm)** | | 0.393362 | 0.348259 | 0.400236 |

*Table 4: Light Localization: Final angle results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Falling Edge Light Localization** | | | | **Rising Edge Light Localization** | | |
| **Run #** | **Expected angle (**°**) ± 0.05** ° | **Final Angle (**°**) ± 0.05** ° | **Final Error in Angle (**°**) ± 0.05** ° | **Expected angle (**°**) ± 0.05** ° | **Final Angle (**°**)   ± 0.05** ° | **Final Error in Angle (**°**) ± 0.05** ° |
| 1 | 0.0 | 1.00 | 1.00 | 0.0 | 2.00 | 2.00 |
| 2 | 0.0 | 0.50 | 0.50 | 0.0 | 2.50 | 2.50 |
| 3 | 0.0 | 0.45 | 0.45 | 0.0 | 2.60 | 2.60 |
| 4 | 0.0 | 0.35 | 0.35 | 0.0 | 2.45 | 2.45 |
| 5 | 0.0 | 1.50 | 1.50 | 0.0 | 3.60 | 3.60 |
| 6 | 0.0 | 2.0 | 2.0 | 0.0 | 4.00 | 4.00 |
| 7 | 0.0 | 1.45 | 1.45 | 0.0 | 2.00 | 2.00 |
| 8 | 0.0 | 1.20 | 1.20 | 0.0 | 1.80 | 1.80 |
| 9 | 0.0 | 1.50 | 1.50 | 0.0 | 3.50 | 3.50 |
| 10 | 0.0 | 3.00 | 3.00 | 0.0 | 2.00 | 2.00 |

*Table 5: Data Analysis of Final Angle Results*

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Falling Edge Final Angle (**°**)** | **Rising Edge Final Angle (**°**)** |
| **Mean Error** | 1.30 | 2.64 |
| **Standard Deviation** | 0.81 | 0.78 |

**Observations and Conclusions**

*Which of the two localization routines performed the best?*

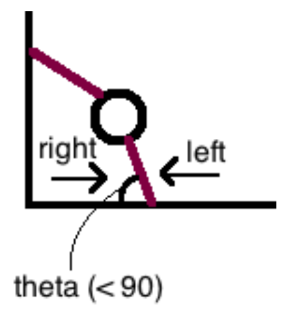
The falling edge localization performed better than it’s rising edge counterpart. Comparing tables 2 and 3, the mean errors in the x and y co-ordinates when using falling edge are 1.12 and 0.94 respectively. This is lower than the mean errors of 1.26 and 1.21 in the x and y measurements when implementing rising edge. Similarly, the Euclidean error of 1.55cm when using falling edge is slightly superior to the 1.78cm Euclidean error produced when using rising edge. Similarly, when looking at the final angle error between the two, the falling edge proves to be a more accurate method. The mean error in the final angle is 1.30 ° degrees. using falling edge, whereas the rising edge mean error in the final angle is slightly higher at 2.64° degrees. Using these metrics of comparison, it is logical to conclude that the falling edge localization method proved slightly more accurate.

*Was the final angle impacted by the initial ultrasonic sensor?*

The error produced during the initial ultrasonic localization did in fact not influence the final angle error. This is because as long as the light sensor could detect all black lines on the tile and calculate the correct rotating angle for each turn, the robot would be able to localize itself to the origin very accurately, regardless of any previous angle error.

*What factors do you think contributed to the performance each method?*

1. *Comparison of Ultrasonic methods*: The accuracy of both methods is again dependant on common recurring sources of errors such as wheel slippage and speed of rotation. Despite these sources of error being common to both, the falling edge method performed more accurately than the rising edge. There are a few reasons for this. Firstly, since using rising edge entails having the initial heading of the robot directly facing a wall, it is more likely to read the gap in the wall, the corner, as a maximum distance, and thus counting it as an angle. Furthermore, the sensor counted an angle at a distance around 30cm, the size of the square, and so the sensor was always past the 90-degree mark when recognizing an angle. This means that an overshoot in the falling edge method would be much less than the overshoot associated with the rising edge. This is due to the fact that the rate at which the distance is changing is increasing when it is moving is right, whereas it is decreasing when moving right.
2. *Comparison of Localization methods*: Again, accuracy errors due to slippage and acceleration are common to both methods, however the light localization method proves to be more accurate. The light sensor corrects the x,y and theta values given that it starts in a general position of known heading. From here, it can traverse itself to the origin very accurately. However, the ultrasonic localization only works if the robot is placed exactly along the diagonal of the first quadrant. Every amount that it is off by relative to the diagonal will correspond to an error in the final theta reading.



*How do changing light conditions affect the light localization.*

In previous labs involving the light sensor, the RBG mode of detecting black lines was being used. However, during this lab, the RED mode was implemented. This mode is more accurate than the RGB mode, because it is more sensitive to any changes in the ambient light detected. Therefore, any changing light conditions, such as a dark shadow falling over the quadrant during a run, would have more of an impact during previous labs. Any scratches or impurities on the grid also must monitored much more closely than in previous labs. In fact, during many runs, we had continuous issues with the sensor missing 2 of the 4 lines during light localization because those two lines had been scratched. Thus, during testing it is imperative to:

1. Ensure a clean and clear grid
2. Make sure that during the run, ambient light acting on the grid is constant.

**Further Improvements:**

*Propose a software or hardware way to minimize errors in the ultrasonic sensor*

One hardware improvement would be to replace the ultrasonic sensor with an Infra-Red one. The most obvious benefit with this would be the improved sampling rate. The waves of the electromagnetic radiation sensors, move at the speed of light. Hence, there would be no delay in between sending and receiving a signal. This would allow not only allow for more accurate results, but for a higher polling rate as well. Together, these would result in a more precise localization.

*Propose another form of localization other than rising-edge or falling-edge.*

Another form of localization would to utilize a more accurate sensor, such as the infra-red one outlined, and use it to precisely find the minima around it. Once the minima are established, it and determined which side of the minima was the corner and which was the open wall, it could determine its position. This is because the robot now knows its x-position based on the sensor distance at the x-minimum, and similarly its y-position at the y-minimum. Then it would be able to rotate 180 degrees from the x-minimum, which positions the robot at 0-degree heading. Then, it can precisely drive to any position within the grid.

*Discuss how and when light localization could be used outside of the corner to correct Odometry errors, e.g having navigated to the middle of a larger floor?*

In this case, the key difference would be that there would be no wall to assist the robot in orientating itself correctly towards the origin. However, the benefit of light localization is that its implementation can be independent of the robot’s closeness to a wall. The light localization can be used at any intersection of 4 lines. If the robot can detect its approximate location, then it is possible to improve the software, and adjust certain constants to correct the odometer’s precision. For example, if the robot is near the co-ordinate (2,2) then you simply need to add 2\*30.48cm to each of the x and y co-ordinate to correct the odometer’s positioning, of course assuming the origin is still at the bottom left corner.