ECSE 211
Lab 3: Localization
Lab Report

Group 20

Linwei Yuan (260857954)

Le-Li Mao (260800098)

Section 1: Design Evaluation

Brief Description

For localization, the purpose of this lab required us to add an ultrasonic sensor and a light sensor to implement localization in two different ways. First, we had to make a choice of the position we want to place the ultrasonic sensor. We ended up changing lab1 ultrasonic sensor design and putting the sensor at the front of the robot. That was because our program involved algorithms to sweep through the wall with the ultrasonic sensor and change the robot's direction to face directly perpendicular to the wall. Second, we needed to decide to put the light sensor at the front or back. Finally, we figured that it was more beneficial to place it behind the robot. That was because the robot's tail is farther from the band center between two wheels, which created a larger radius leading to a larger sweeping area. The light sensor would detect black lines better in this manner. Another hardware adjustment is cleaning up the wire using a rubber band, which makes sure that the robot can move more freely without interruption of the wires.

The software worked in the following manner. First, the robot scanned the surrounding by performing a 360-degree turn. Upon finishing the turn, the robot would calculate at which angle the sensor detected the closest wall. Since the ultrasonic sensor only provides cm accuracy, there is a range of angles that measure the same distance instead of a single minimum angle. To improve accuracy, we took the range and performed a secondary sweep at a slower speed. Overall the robot has a great amount of accuracy and only falter when the sensor detects weird value. To localize the robot, the robot would face the closest wall using the previously described method and measure the distance and correct the distance by taking the difference between the tile distance and the measurement. The robot performs this for both walls and achieves a great deal of accuracy in reaching the (1,1) position.

(Bonus) The team also created localization for the color sensor. First, the robot would perform a 360-degree turn and record the angle which it saw a black line (Color ID: 13). Then the robot would move to the first seen black line and compute the angle difference between the two parallel black lines and calculate the half-angle between them. Turning to the half-angle would result in being perpendicular to the line which allows us to perform the movement in an ideal behavior. The distance that the robot should travel is based on the half-angle and the distance between the rotational center and the sensor. Performing the operation with the other two detected lines would result in the robot being in the center position.

Workflow

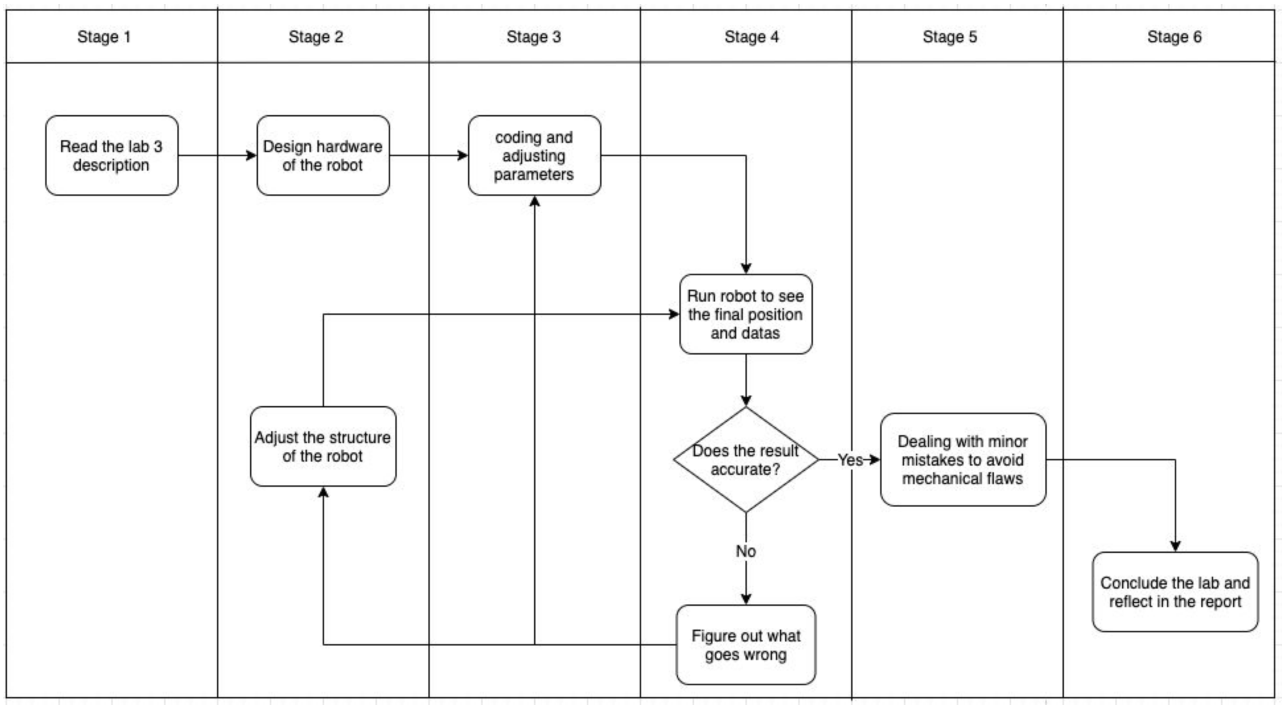
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6
Read the lab 3 description
Design hardware of the robot
coding and adjusting parameters
Run robot to see the final position and datas
Adjust the structure of the robot
Does the result accurate?
Yes
No
Dealing with minor mistakes to avoid mechanical flaws
Conclude the lab and reflect in the report
Figure out what goes wrong

Figure 1: Workflow diagram

Photos of Hardware Design

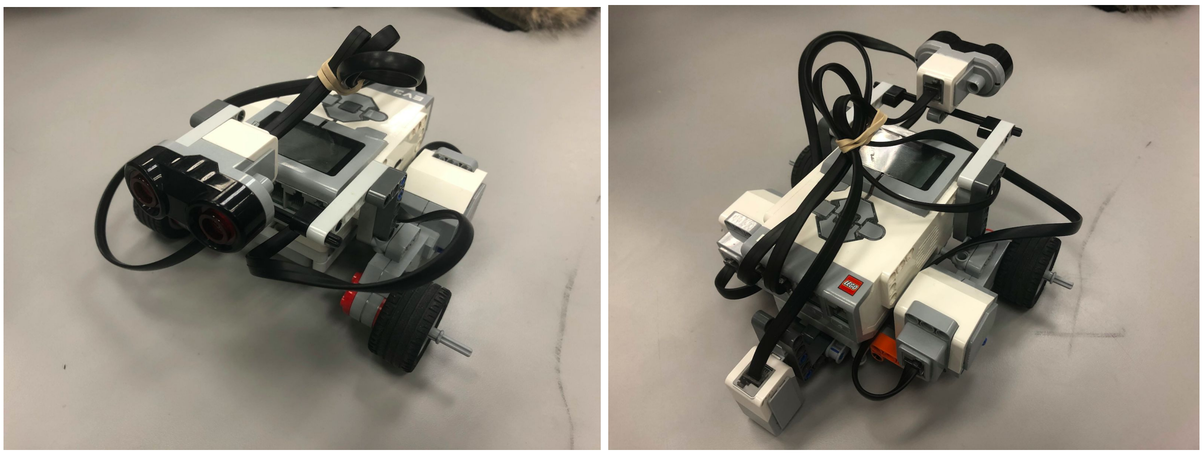

Figure 2: Front view (left) and back view (right) of the robot

Visuals for the Software Design

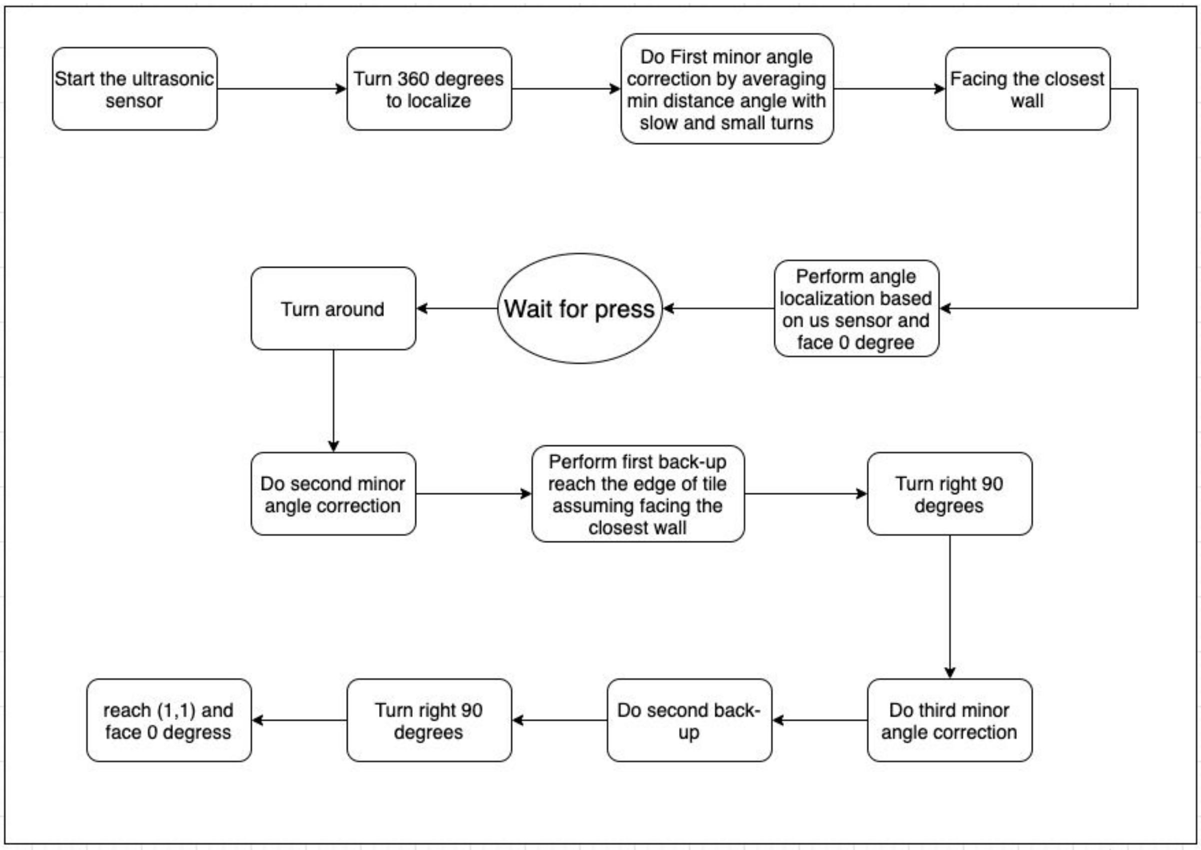
Start the ultrasonic sensor
Turn 360 degrees to localize
Do First minor angle correction by averaging min distance angle with slow and small turns
Facing the closest wall
Perform angle localization based on us sensor and face 0 degree
Wait for press
Turn around
Do second minor angle correction
Perform first back-up reach the edge of tile assuming facing the closest wall
Turn right 90 degrees
Do third minor angle correction
Do second back-up
Turn right 90 degrees
reach (1,1) and face 0 degress

Figure 3: Flow chart of ultrasonic sensor localization

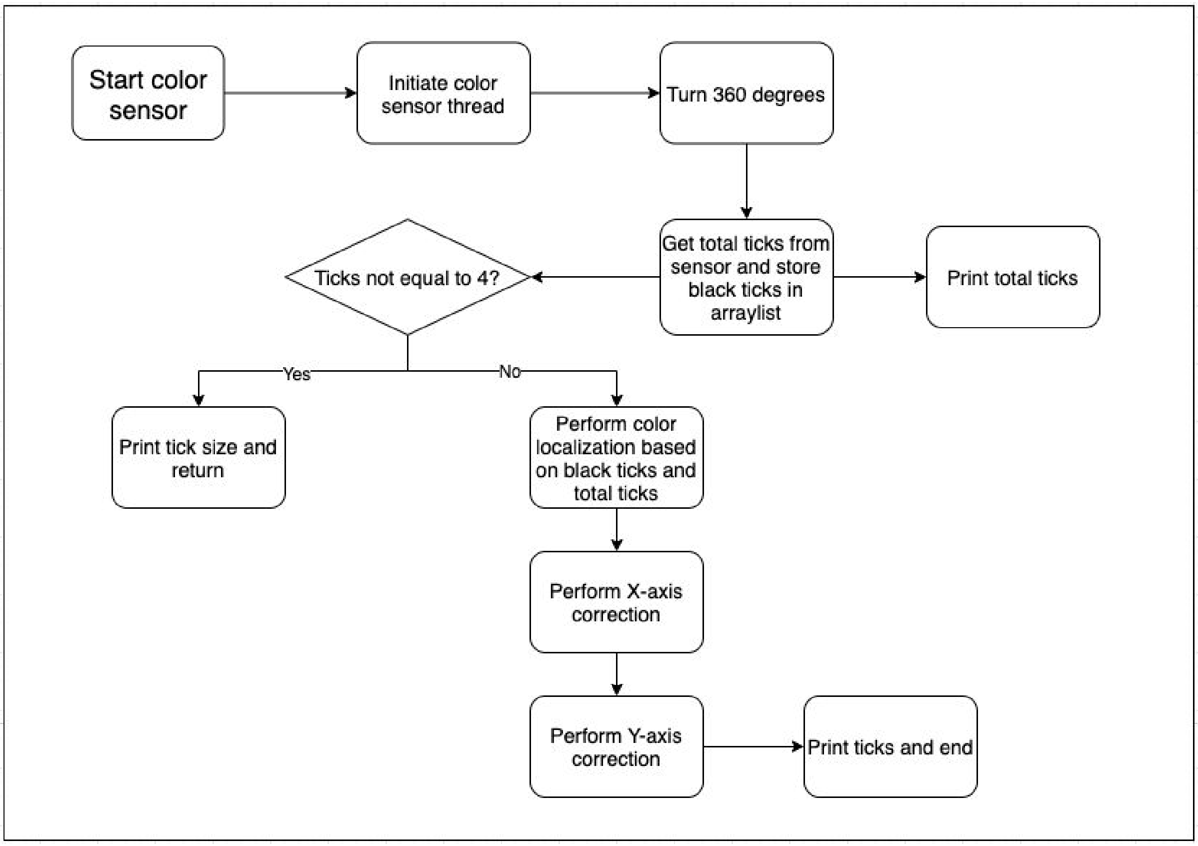
Start color sensor
Initiate color sensor thread
Turn 360 degrees
Get total ticks from sensor and store black ticks in arraylist
Print total ticks
Ticks not equal to 4?
Yes
No
Print tick size and return
Perform color localization based on black ticks and total ticks
Perform X-axis correction
Perform Y-axis correction
Print ticks and end

Figure 4: Flow chart of light sensor localization

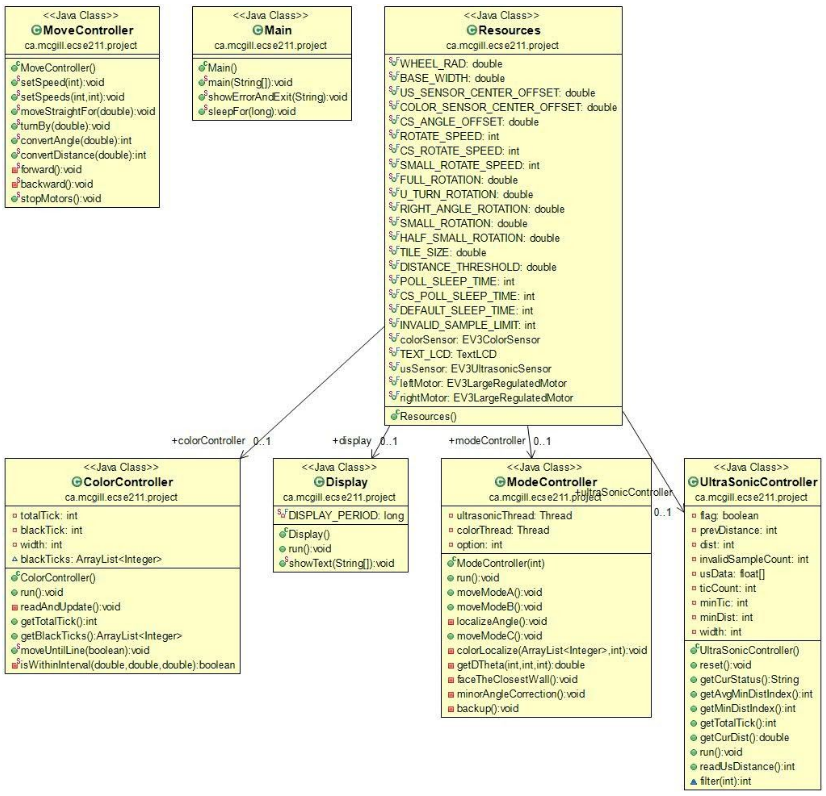

Figure 5: Class diagram of the project

Section 2: Test data

Trials	Ultrasonic Angle Error (degree)	Actual position (cm)	Euclidean Error (cm)	Final Angle Error (degree)
1	3.2	(30,29.43)	0.57	0.1
2	2.1	(30,31.56)	1.56	3.0
3	4.5	(29.58,31.8)	1.86	0.3
4	3.1	(33.3,33.36)	4.71	4.5
5	1.9	(30.3,24.39)	5.61	3.2
6	0.5	(28.62,31.5)	2.04	1.5
7	2.0	(29.58,31.2)	1.26	2.5
8	5.5	(29.73,30.6)	0.66	1.5
9	2.5	(30,30.6)	0.60	1.3
10	3.5	(29.85,31.5)	1.50	1.4

Table 2: Euclidean Distance Error and Final Angle Error

Euclidean Error Equation:

$$\epsilon = \sqrt{(X - X_F)^2 + (Y - Y_F)^2}$$

Where

- *X and Y are the expected (x, y) values (0, 0)*
- *X_F is the x-coordinate value measured from the origin (0, 0)*
- *Y_F is the y-coordinate value measured from the origin (0, 0)*

Section 3: Test Analysis

Mean Calculation:

$$\bar{\epsilon} = \frac{\sum_{i=1}^{i=N} \epsilon_i}{N}$$

Where

- *ϵ_i is the i^{th} error*
- *N is the number of observations*

Standard Deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{i=N} (\epsilon - \bar{\epsilon})^2}{N - 1}}$$

Where

- *ϵ_i is the i^{th} error*
- *$\bar{\epsilon}$ is the mean error*
- *N is the number of observations*

Sample Calculation:

For ultrasonic angle error:

Mean = $\frac{3.2+2.1+4.5+3.1+1.9+0.5+2.0+5.5+2.5+3.5}{10}$

= 2.88 degree

Standard deviation :

$$\sum_{i=1}^{10} (X_{mean} - X_i)^2$$

$= (2.88 - 3.2)^2 + (2.88 - 2.1)^2 + (2.88 - 4.5)^2 + (2.88 - 3.1)^2 + (2.88 - 1.9)^2 + (2.88 - 0.5)^2 + (2.88 - 2.0)^2 + (2.88 - 5.5)^2 + (2.88 - 2.5)^2 + (2.88 - 3.5)^2$

= 1.817 degree

$$\textit{standard deviation} = \sqrt{\sum_{i=1}^{10} (X_{mean} - X_i)^2} = \sqrt{1.817} = 1.35 \text{ degree}$$

Calculated data	Ultrasonic Angle Error (degree)	Euclidean Distance Error (cm)	Final Angle Error (degree)
Mean	2.88	2.04	1.93
Standard Deviation	1.35	1.65	1.29

Section 4: Observations and Conclusions

<u>Describe other localization techniques and explain why you chose yours.</u>

Other methods are rising edge and falling edge methods. As described in the tutorial, both methods utilize the ultrasonic sensor to determine the robot's orientation and distance to the wall. These values are then used to reorient the robot in the correct direction. The robot will detect rising or falling edge and react differently according to whether it is facing away from the wall or starting to face the wall. The mathematics behind falling and rising edges can increase the accuracy of localization.

We choose our method because, with the algorithm implemented, the robot can go to (1,1) when it is not in the 45-degree line, or even not in the 1-by-1 tile. Also, with adding the function of minor angle changes, the robot will correct the angle twice before navigate to (1,1). Therefore, functionality is expanded and accuracy is increased.

<u>Was the final angle impacted by the initial ultrasonic angle?</u>

The final angle will not be affected by the initial ultrasonic angle because we have a function of minor angle change implemented. What it does is basically sweep the wall back and force and find the angle that is perpendicular to the wall. So, when the robot turns 90 degrees, it would face the exact 90-degree angle. The robot will do that twice before reaching (1,1), so the error in the initial ultrasonic angle will be eliminated when the robot gets to the destination.

<u>What factors do you think contributed (positive and negative) to the performance of your method?</u>

Since our robot can do localization as long as it is within the 3-by-3 tile, we investigated its performance based on that scale. As a result, if the robot is placed near (1,1), the localization tends to work more accurately. Oppose to that, placing the robot far from (1,1) has a negative contribution to the performance. Also, using woodblocks as a wall may also impact negatively on the performance: the uneven surface of the wall will decrease the accuracy of the ultrasonic sensor.

<u>Do you think that the ultrasonic sensor or the light sensor method performs better? (If you did not do the bonus answer based on the reliability/quality of the readings of the light sensor vs. ultrasonic sensor). In future labs which one do you plan to use for navigating?</u>

We think the ultrasonic sensor performs better than the light sensor based on the test results. The reason for this is most likely due to the width of the line being small which would result in less accurate measurement. However, in scenarios where no wall exists, we would still need a color sensor to localize. Even so, given the environment, we currently have an ultrasonic sensor that is more likely to be used for navigation.

Section 5: Further Improvements

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

A hardware way to improve minimize the errors is to put the ultrasonic sensor vertically instead. In this way, the region that the sensor sees is more narrow and the data collected is more concentrated. So, it improves the accuracy of the ultrasonic sensor and thereby minimizing the errors.

In addition, we could use an infrared (IR) sensor instead of the US sensor. As we have seen in our test, the ultrasonic sensor can cause inaccurate results with more errors because the sound waves generated by it can bounce between blocks or walls. However, it will not be a problem for the IR sensor. The IR sensor is effective in detecting obstacles, especially in an indoor environment because there is less ambient light to interrupt the readings. The only drawback is the IR sensor costs more.

Propose another form of localization.

Except for all the methods of localization we have used and mentioned, there is another technique using the infrared (IR) sensor mentioned above. Since the IR sensor has much higher accuracy than the US sensor according to the EV3 website, the robot will have better performance. In terms of the accompanying algorithm, we could implement code that drives the robot to rotate a full circle and calculate all the angles along the circular path according to the precisely-measured distance. Special attention is paid to some important angles: the angle where distances drop dramatically, the angle where distances rise dramatically, and the angle rises and then falls. We could determine where the robot is facing by these angles, more accurate data would be collected to execute a better localization.

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.

When traveling through the course, the robot would use the detection of the line to localize the current location of the robot. For instance, if the robot starts from (1,1) and travels down the courses in the x-direction and passes the 3 lines, while it is on the third line, it is able to deduce it is 3 tiles down the x-axis and would able to compute and correct the distance by multiplying the tile and the amount of tile it sees. Performing similar actions would allow the robot to correct its location and stay true to its target.

How could the robot quickly localize given two light sensors?

Place each light sensor right beside each wheel. When the robot needs to localize, perform a turn in both wheels and stop the wheel only if the sensor detects a line. After one wheel rests on the line, perform a turn in another wheel until the other wheel is also on the line. At this stage, the robot should be perpendicular to the line. Using this fact and we could localize the angle of the robot relatively quickly in reference to the odometry data. After we acquire the angle we can quickly perform a 90-degree turn and find the other two points on the parallel line which would place us in the corner of the tile and we would be localized.