ECSE 211
Lab 4: Navigation
Lab Report

Group 20

Linwei Yuan (260857954)

Le-Li Mao (260800098)

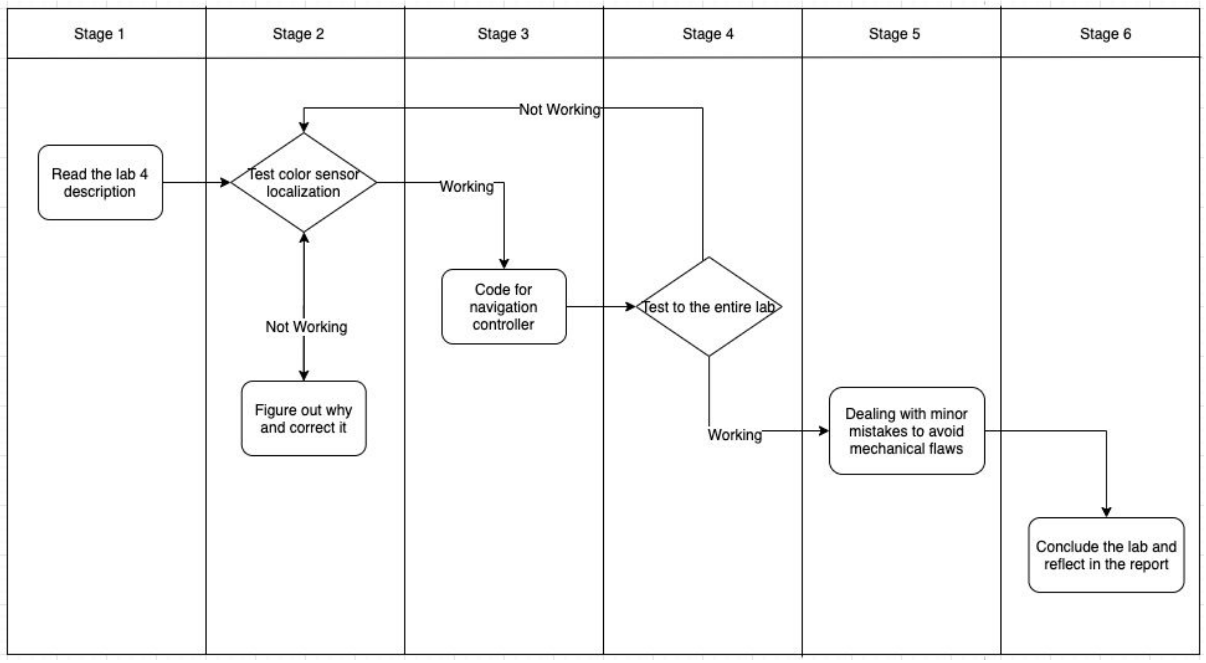
Section 1: Design Evaluation

Brief Description

The hardware for this lab was inherited from the last lab because the main function that the hardware needed to have did not change: it was required to collect data from the color sensor and ultrasonic sensor and then localize. The only difference was implementing one more function, navigation, which was similar to Odometer and required no extra sensor to achieve the goal.

The software consisted of many components of the previous lab. First, the odometer class was brought back to the project and was used when computing the angle to turn and position to move inside the navigation methods. Also, the US controller class and color controller class from our previous lab was also ported over to use for localization initially and when the robot reached a checkpoint. After we attached all the classes together, we started implementing two navigation methods. The methods work as follows. First, we implemented the turn to a method that takes an angle as input indicating the angle we would like to turn to. To do this, we read the current angle from the odometer and compute the difference between the angle to get the angle needed to reach the desired angle. When that angle is bigger than 180 or less than -180 degrees, we would compute the opposite direction angle which would be the minimum angle to reach the desired angle. Then, we implemented the “travel to” method. First, we read the current x and y coordinate from the odometer and compute the delta between the desired x and y. After computing the delta x and y, the magnitude which we would travel is given by the Pythagoras theorem and use the arctangent to compute the angle. After computing the angle, we would use the “turn to” method to turn to the indicated direction and use the move straight method to travel the calculated distance.

Workflow



Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
Stage 6
Not Working
Read the lab 4 description
Test color sensor localization
Working
Code for navigation controller
Test to the entire lab
Not Working
Figure out why and correct it
Working
Dealing with minor mistakes to avoid mechanical flaws
Conclude the lab and reflect in the report

Figure 1: Workflow diagram

Photos of Hardware Design

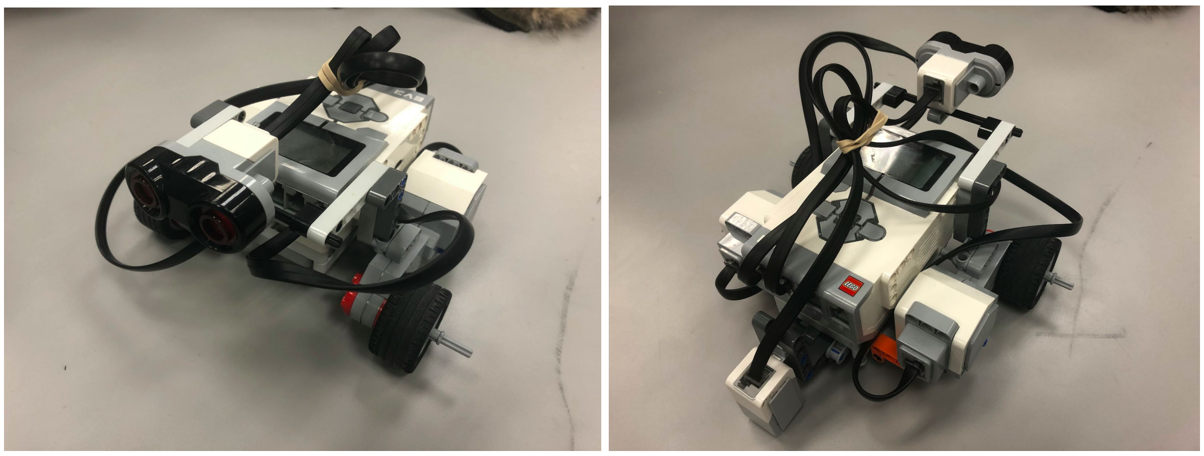


Figure 2: Front view (left) and back view (right) of the robot

Visuals for the Software Design

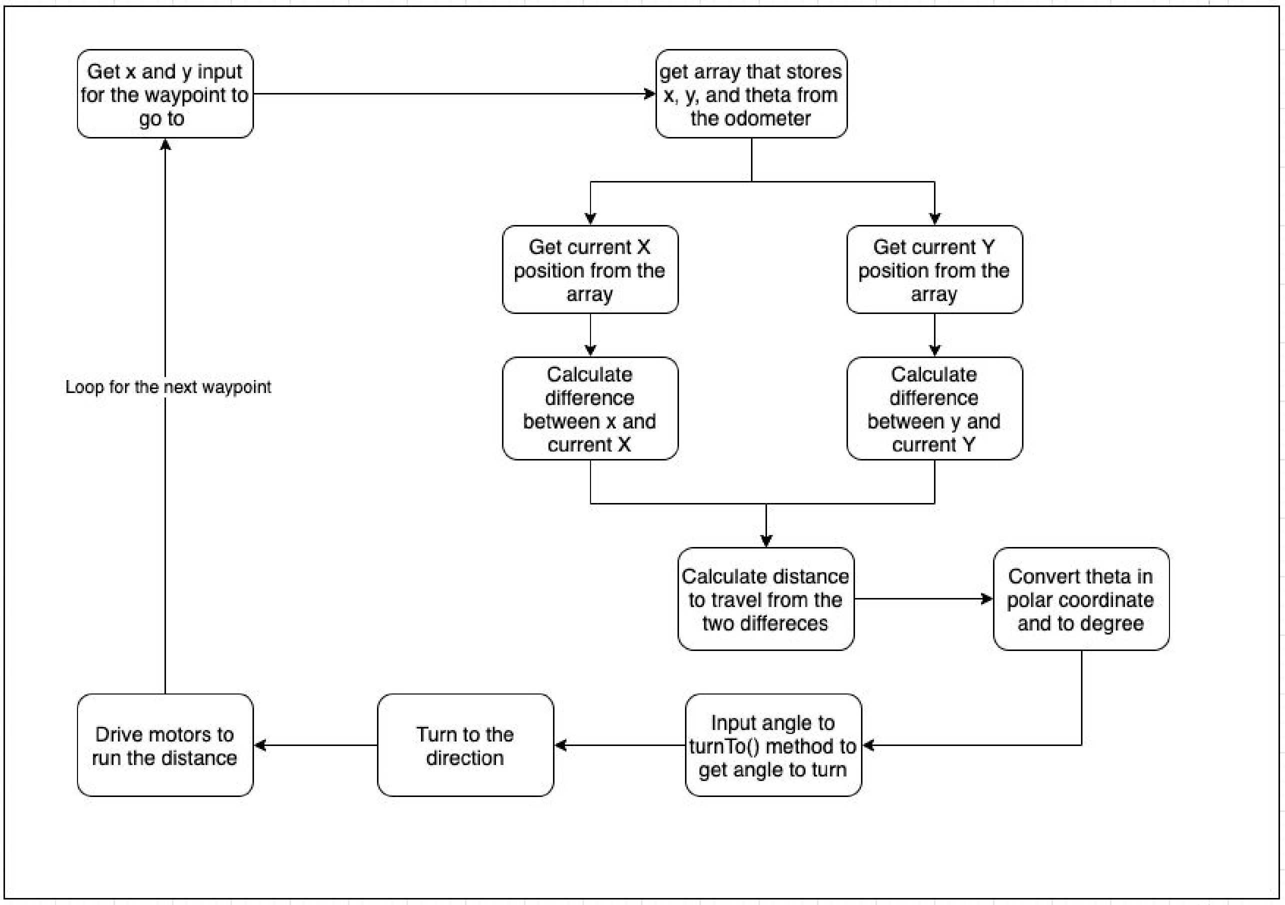
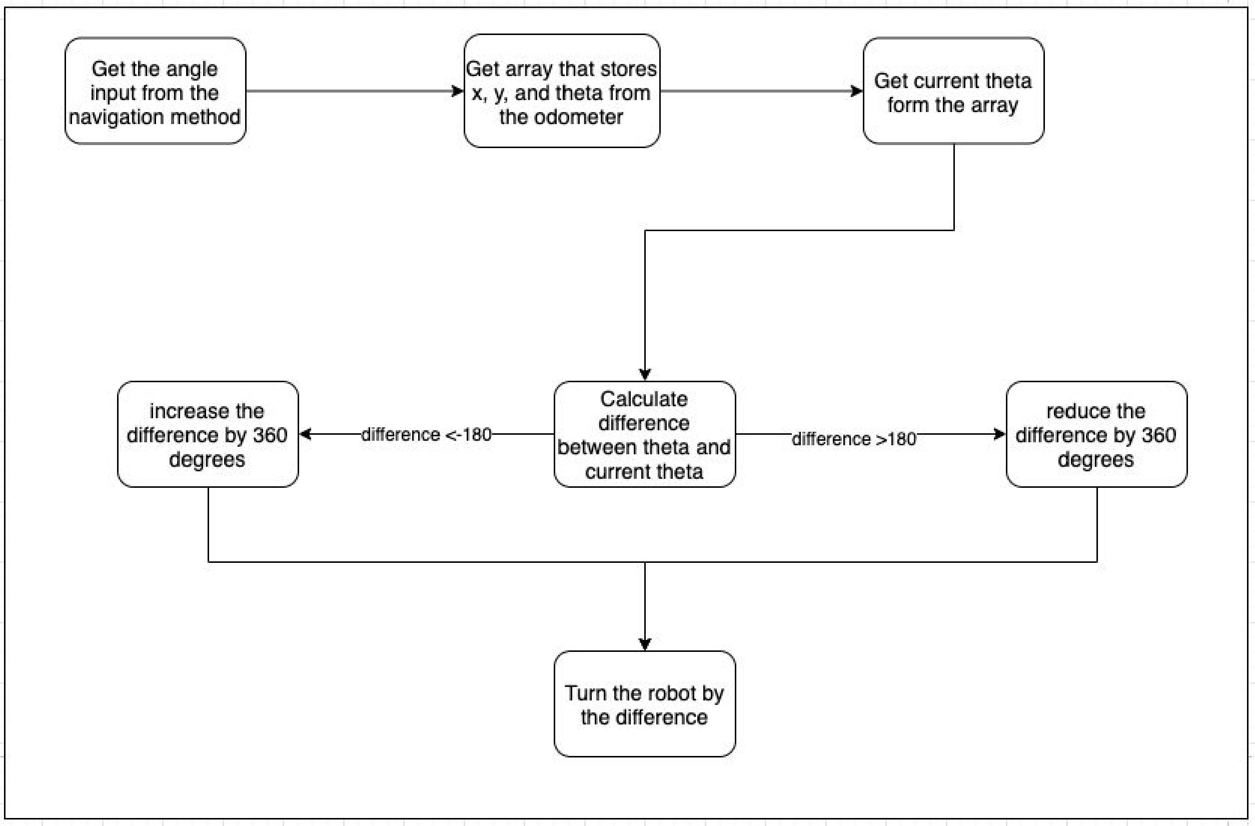

Get x and y input for the waypoint to go to
get array that stores x, y, and theta from the odometer
Get current X position from the array
Get current Y position from the array
Calculate difference between x and current X
Calculate difference between y and current Y
Loop for the next waypoint
Calculate distance to travel from the two differeces
Convert theta in polar coordinate and to degree
Drive motors to run the distance
Turn to the direction
Input angle to turnTo() method to get angle to turn

Figure 3: Flow chart of the method that navigates the robot to a waypoint


Get the angle input from the navigation method
Get array that stores x, y, and theta from the odometer
Get current theta form the array
Calculate difference between theta and current theta
difference <-180
increase the difference by 360 degrees
difference >180
reduce the difference by 360 degrees
Turn the robot by the difference
Figure 4: Flow chart of the method to calculate the angle to turn during navigation

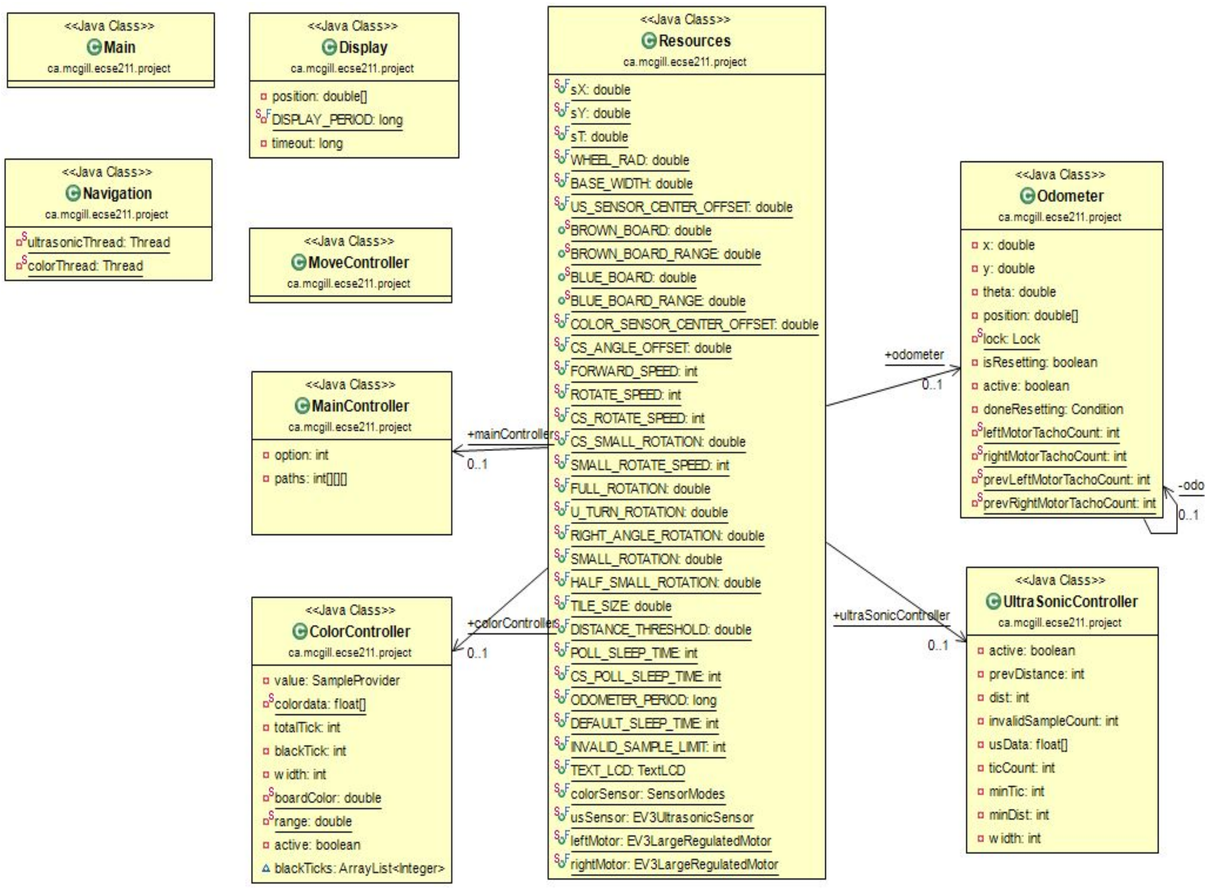


Figure 5: Class diagram of the project (attributes only)

Section 2: Test data

Trial	Destination Xd (cm)	Destination Yd (cm)	Final position Xf (cm)	Final position Yf (cm)	Error (cm)
1	90	30	90.85	30.95	1.27
2	90	30	91.40	29.10	1.66
3	90	30	90.55	30.55	0.78
4	90	30	92.11	33.00	3.67
5	90	30	93.40	31.50	3.72
6	90	30	89.35	29.50	0.82
7	90	30	90.15	30.50	0.52
8	90	30	89.10	32.45	2.61

Table 1: Destination, Final Position, and Euclidean Distance Error

Section 3: Test Analysis

The Euclidean error distance is shown in the table above, and here is the equation and sample calculation for the first trial:

Euclidean Error Equation:

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

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)*

$$\varepsilon = \sqrt{(Xd - Xf)^2 + (Yd - Yf)^2}$$

$$= \sqrt{(90.85 - 90)^2 + (30.95 - 30)^2}$$

$$= \sqrt{1.625}$$

$$= 1.27 \text{ cm}$$

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 Euclidean error:

Mean = $\frac{1.27+1.66+0.78+3.67+3.72+0.82+0.52+2.61}{8}$

= 1.88cm

Sample Standard deviation :

$$\sum_{i=1}^{8} (\varepsilon_{mean} - \varepsilon_i)^2$$
$= (1.88 - 1.27)^2 + (1.88 - 1.66)^2 + (1.88 - 0.78)^2 + (1.88 - 3.67)^2 + (1.88 - 3.72)^2 + (1.88 - 0.82)^2 + (1.88 - 0.52)^2 + (1.88 - 2.61)^2$

$= 11.73$

$So,\ \frac{11.73}{N-1} = \frac{11.73}{8-1} = 1.67$

$sample\ standard\ deviation = \sqrt{1.67} = \sqrt{1.67} = 1.29$ cm

Calculated data	Euclidean Distance Error (cm)
Mean	1.88
Sample Standard Deviation	1.29

Table 2: Mean and Sample Standard Deviation of Euclidean Distance Error

Section 4: Observations and Conclusions

Are the errors you observed due to the odometer or navigator? What are the main sources?

Errors that we observed are mainly due to the odometer because of the different friction of the two wheels. For example, if one wheel is covered with more dust than the other one, the wheels will not move at the same speed so the angle to turn will also not be exactly the same as we set. Moreover, the unbalance of two wheels due to frictional force will drive the robot away from the path. This deviation results in the inaccurate values detected by the odometer, then causing the robot to reach a deviated destination. Since the robot is going to turn at and navigate through several waypoints in this lab, the errors will accumulate and cause a deviation that can not be ignored. That has to be corrected by another localization with the color sensor.

How accurately does the navigator controller move the robot to its destination?

Since our robot will do the first color sensor localization when it reaches the second waypoint, it turns out to be very accurate most of the time in the testing. Even before the second color sensor localization at the final destination, the error is usually within 4 cm. Based on the measured result, the error is quite small and the sample standard deviation is also acceptable. So, the navigator controller is accurate in the lab most of the time. Even if the navigation is affected by an unexpected factor, the robot will be able to correct itself with a final color sensor localization.

At which waypoints did you decide to localize and why? What are the advantages and disadvantages of localizing at every waypoint?

We decided to localize at every two waypoints because localizing at every waypoint would be very time-consuming. On the other hand, localizing less would cause the error to propagate which could potentially lead to an unfixable state where the robot is too far from the cross-section of two lines to localize.

The benefit of localizing would be that you never deviate too far from your true value and it is very stable in terms of navigation and would be less prone to error. However, each localization takes time which is going to cause issues in time-sensitive scenarios such as competition and the next lab.

Section 5: Further Improvements

What step can be taken to reduce the errors you discussed above? Identify at least one hardware and one software solution. Provide explanations as to why they would work.

Software:

First, we would consider slowing down the speed of the motors as mentioned above in order to reduce the slip of the wheels. Second, the odometer can be corrected by implementing the odometer correction method from the odometer lab. Therefore, a second light sensor needs to be added to the robot between two tires. It can correct the x-y-theta values whenever the sensor detects a black grid line. Third, if time allowed, we could do light sensor localization on each waypoint, which would definitely make the robot start in the right position but will be time-consuming.

Hardware:

We can increase the weight of the robot to lower the center of mass of the robot increasing the speed limit of slipping occurrence. Also, as mentioned in software improvement, we can add a light sensor to the robot and implement the corresponding algorithm to adjust each time it meets the black line. Another improvement in the structural design is that a relatively long bar is suggested to connect two wheels to avoid the axles of wheels flexing. That is because this measurement will decrease the chance to have an unbalanced moving speed of wheels. Therefore, errors are minimized.