Laboratory 3: Navigation and Avoidance Report

ECSE 211: Design Principles and Methods

Prof. Ferrie, Prof. Lowther

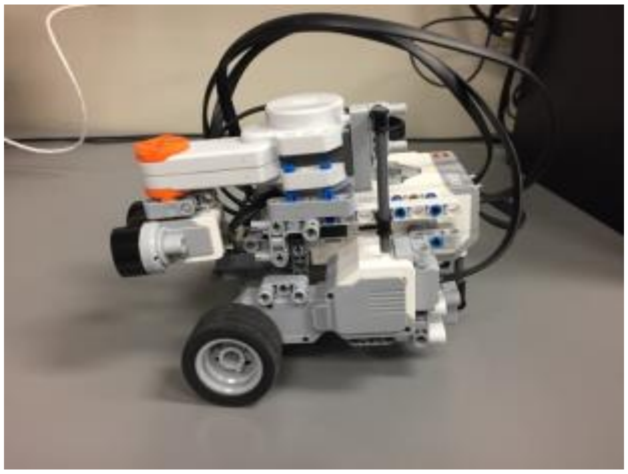
Li Zhang (260743980)

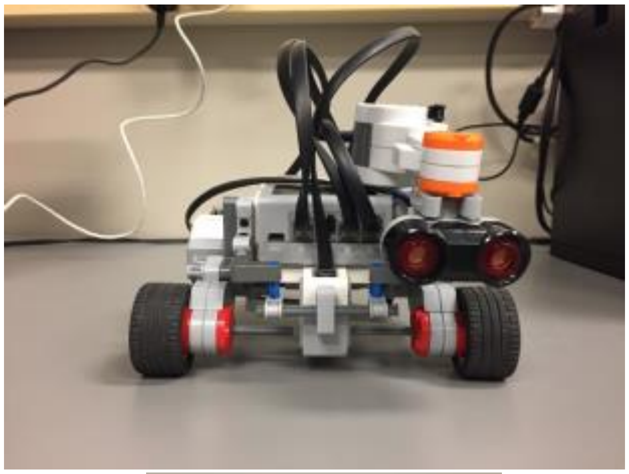
Marwan Chehade (260675988)

Group 10

I. Design evaluation

Hardware design: The Mindstorms brick was attached to two EV3 motors used for wheel rotation. Furthermore, an ultrasonic sensor was added to detect any blocks or obstacles on the robot's path. The sensor was placed facing the front and was mounted on a NXT motor in order to make it rotate when needed after detecting an obstacle.

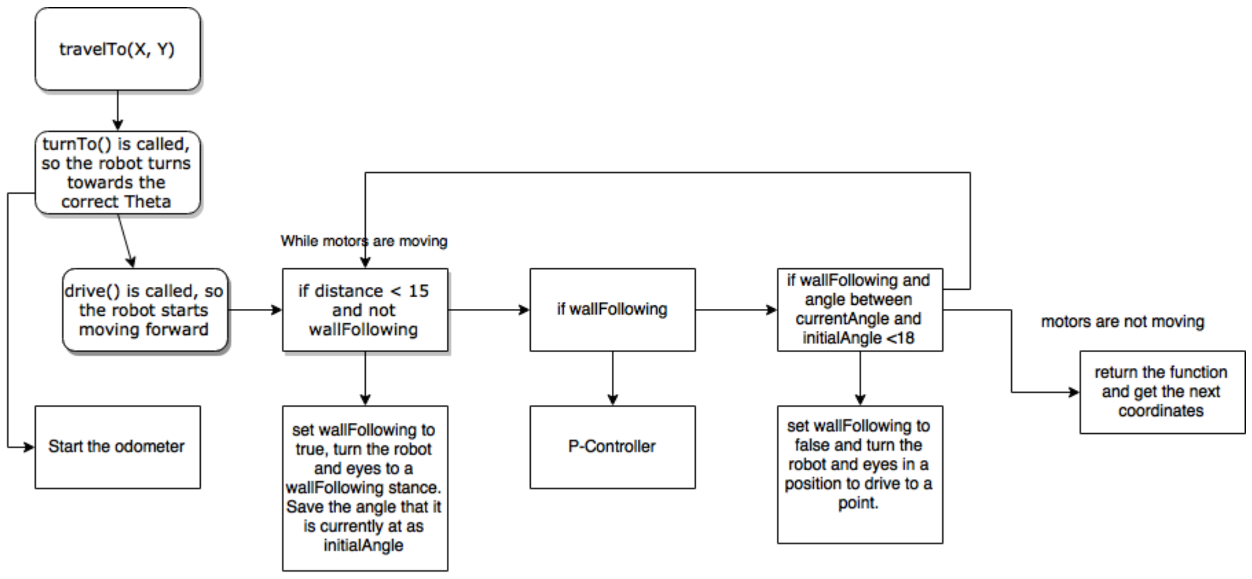




Software design:

1- Navigation: The navigation software directed the robot's movements. It contained a turnTo() method which computed the minimal angle the robot has to rotate to in order to face the next waypoint it has to go to. This was done using the arctangent function included in Java. Moreover, travelTo() was another method added which took 2 inputs, the coordinates of the waypoint, and by computing the distance between the actual position of the robot (Xi, Yi) and the specified waypoint (Xf, Yf) ($\sqrt{(Xf - Xi)^2 + (Yf - Yi)^2}$). It is important to point out that the odometry program is being used to obtain Xi and Yi. Finally, a boolean isNavigating() is set to true whenever the robot is travelling from a point to another and to false when the robot is rotating or finishes the track it was supposed to follow.

2- Wall following and Object Detection: When the robot is navigating, both wheels rotate command were set to true (non-blocking) in order for the program to read the following lines of codes which allows to acquire the ultrasonic sensor's values. When the robot is considered too close to an obstacle, the robot stops, rotates to its right such that the obstacle is positioned to its left and its sensor rotates towards the wall. Thus, a wall following (P-type Controller) code is then implemented (*Refer to the flowchart below*).


travelTo(X, Y)
turnTo() is called, so the robot turns towards the correct Theta
drive() is called, so the robot starts moving forward
Start the odometer
While motors are moving
if distance < 15 and not wallFollowing
set wallFollowing to true, turn the robot and eyes to a wallFollowing stance. Save the angle that it is currently at as initialAngle
if wallFollowing
P-Controller
if wallFollowing and angle between currentAngle and initialAngle <18
set wallFollowing to false and turn the robot and eyes in a position to drive to a point.
motors are not moving
return the function and get the next coordinates

II. Test Data

Table 1 Measurements made after 10 trials

Trials #	X (cm)	Y (cm)	X_f (cm)	Y_f (cm)	Error (cm)
1	56.76	-4.50	63.41	4.20	10.95046
2	54.66	-4.20	63.12	3.29	11.29919
3	57.96	-1.40	63.46	4.23	7.87064
4	57.66	-1.30	63.39	4.01	7.81211
5	58.26	-2.20	63.53	4.00	8.13713
6	57.26	-2.30	63.30	4.19	8.86576
7	57.66	-1.80	63.41	4.04	8.19561
8	57.36	-3.70	63.70	4.16	10.09828
9	58.96	0.00	63.50	3.98	6.03755
10	59.06	-1.70	63.29	4.05	7.13831

III. Test Analysis

1. *Compute the mean and standard deviation for the errors measured in the Navigation test. Be sure to show general formulas and sample calculations.*

$$\mu = \frac{\sum error}{trials}$$

$$\mu = \frac{10.95046 + 11.29919 + .. + 6.03755 + 7.13831}{10}$$

$$\boldsymbol{\mu = 8.64050}$$

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

where N = number of trials

so for each trial, do

$$error - \mu = 10.95046 - 8.64050$$

add all of them, divide by ten, and then find the root.

$$\boldsymbol{\sigma = 1.591935}$$

2. *Are the errors present because of the odometer or the navigator? Give reasons to back up your claim*

The errors are present because of the odometer. In each of the trials, the robot would stop at the coordinates given by the navigator. It means that the robot could indeed travel to specific coordinates on the board. Thus, the error found would be a result of the odometer. Furthermore, for lab 2, a light sensor was present to help correct the coordinates of the robot, which led to a decrease in error. However, in this lab, the light sensor was removed, meaning that the error was of larger magnitude.

IV. Observations and Conclusions

•How accurately does the navigation controller move the robot to its destination?

The average distance from the coordinates (2, 0) and the robot is 4.18948 cm with a smallest value of 2.00 cm and a largest of 6.16 cm. Therefore, we can conclude that the navigation controller moves the robot to its destination fairly accurately, but adding a light sensor would be much better in order to correct the robot's location.

•How quickly does it settle (i.e. stop oscillating) on its destination?

The robot does not oscillate in place because in the program, the motors stop once they have completed the give number of rotations. It does not do a double check to see if it has successfully arrived.

•How would increasing the speed of the robot affect the accuracy of your navigation?

Increasing the speed of the robot could result in the robot slipping and skewing the results of our odometer. However, if the acceleration of the motors is defined, as it is in our code, increasing the speed would not affect the navigation nor odometer because the motors will accelerate such that the wheel would not slip. The same can be said for coming to a stop: the motors will take into account the momentum and therefore slow the robot down before coming to a full stop.

•What are the main sources of error in navigation?

The main source of error in navigation is the fact that it does not do a final check to see if it has arrived to the correct location. The program calls drive(distance) which makes the wheels turn a specific number of times so that the robot will reach a location. However, after the wheels stop, there is no check to make sure that it has indeed arrived to the location. Instead, it assumes that it has. This could be improved by adding a check that double checks with the odometer if it has reached the coordinates.

V. Further Improvements

•What steps can be taken to reduce the errors in navigation and odometry?

In order to reduce the errors in navigation and odometry, we should try to get the robot to consistently reach a waypoint with a high precision. This can be done by making the robot correct its position depending on what it detects on the grid (lines, borders…). Another correction can be to make the robot retravel to the point it should be located at in case it is too far away (for example, if it is 0.5 cm away when looking at the odometer, make the robot travel to the waypoint again using its current location).

•In four to six sentences, identify at least one hardware and one software solution. Provide an explanation as to why they would work

A hardware solution would include using the light sensor so that it detects when lines are crossed such that the X and Y positions are adjusted in the odometer. Thus, the trajectory and angle that the robot has to take can also be recalculated and the precision obtained can increase. Alternatively, using two lines sensor would allow one to follow the X-axis line the robot should be travelling on (unless it is going diagonally) and the other can detect the Y-axis lines crossed to correct and the y-position.

A software solution is an implementation of the light sensor. When a black line is detected (for example, ColorID <11), depending on the path on which the robot is going and the next waypoint it is trying to reach, the X and Y values of the odometer can be corrected. Also, a line follower can be implemented to make sure the robot always follows a black line when it is not travelling diagonally and a correction would be made when it is too far from it such that the robot goes back to it.