ECSE 211: Navigation/obstacle avoidance Lab

Ezz Aboulezz 260 677 463
Oscar Deceus 260 744 646

Group 57

Design Evaluation:

Hardware: The hardware design of our EV3 brick is very simple for this lab. It contains 2 motors on both sides of its front part with wheels attached to them. There is a metal sphere in the middle of the back to balance the brick and to facilitate its driving. The Ultrasonic sensor is placed between the wheels and points forward. The assembling of the robot was mostly driven by several videos and ideas found online.

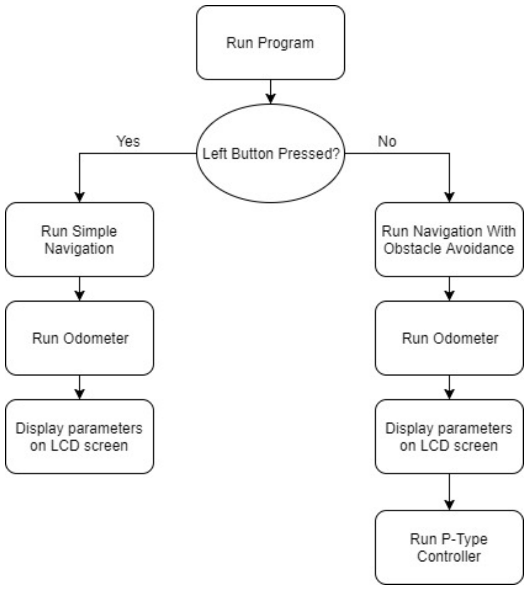
Software: **Refer to workflow chart below**

Navigation

The Navigation class contains all the instructions that allow the robot to move across the platform. For each waypoint, we calculate the angle for which the robot should position itself to get to the next point from its current position according to the x and y axis. We also make sure that the robot turns using a minimal angle. Furthermore, we get the distance that the robot has to travel using the Euclidean distance measure. With these data, we are able to let the robot drive effectively to the next waypoint. Finally, we update the current position of the robot and start the process again for the next waypoint until the robot reaches its final destination.

P-Controller

We've decided to use our P-Type Controller class from Lab-1 as the obstacle avoidance for the robot. While the robot is driving, the Ultrasonic sensor receives data and can recognize if there is an obstacle in front of it or not depending on the distance threshold. If an obstacle is detected, the robot stops its movements and goes to P-Type controller mode by following the wall from a specific distance until it gets to the other side of the wall. After that, we get the current position of the robot (x, y, angle) and calculate the new path that the robot has to follow to get to its waypoint.


Run Program
Left Button Pressed?
Yes
No
Run Simple Navigation
Run Odometer
Display parameters on LCD screen
Run Navigation With Obstacle Avoidance
Run Odometer
Display parameters on LCD screen
Run P-Type Controller

Test Data: (Refer to table below)

	X(cm)	Y(cm)	Xf(cm)	Yf(cm)	Error(cm)
1	60.96	0	57.25	2.5	4.473712
2	60.96	0	58	3	4.214451
3	60.96	0	58	2.5	3.874481
4	60.96	0	57.25	2.5	4.473712
5	60.96	0	58.5	3	3.879639
6	60.96	0	62	-1.25	1.626069
7	60.96	0	60.96	0	0
8	60.96	0	59	2.25	2.983974
9	60.96	0	60.96	1.75	1.75
10	60.96	0	60.96	0	0

Test Analysis:

Using the arithmetic mean formula, $A = \frac{1}{n} * \sum_{i=1}^{n} x_i$ where n is the number of elements and Xi is an element, we can see that the mean is equal to 2.727604.

Using the standard deviation formula $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$, where N is the number of elements, Xi an element and μ is the mean; we can see that the standard deviation for the errors measured in the table is 1.765895.

The errors are caused mainly caused by the Odometer. No odometer correction is implemented in our code, which means that the slips of the wheels during its accelerations or decelerations can cause the Odometer to have wrong values. Thus, getting the current position of the robot from the Odometer brings values that won't send the robot to the right destination. The Navigation class cannot bring errors because every method is based with formulas and precise values for the waypoints. The only issue is that it doesn't take in consideration the way that the wheels are placed on the robot. They could be unbalanced and can cause the Navigation formulas to be useless.

Observations and conclusions:

<u>How accurately does the navigation controller move the robot to its destination?</u>
It moves the robot to its expected destination quite accurately. The robot travelled the calculated distance precisely as long as there were no obstacles in its way to alter its movement.

How quickly does it settle (i.e. stop oscillating) on its destination?
The robot stops oscillating immediately once it reaches its destination after travelling the calculated distance that it is supposed to move to reach it.

How would increasing the speed of the robot affect the accuracy of your navigation?
Increasing the speed will result in more slipping by the wheels which will affect the odometer's reading and result in less accurate positions. As mentioned before, Navigation uses the readings retrieved from Odometer therefore an error in there will result in an error in Navigation.

What are the main sources of error in navigation?
The navigation class uses the wheel radius and base to calculate the heading angle it should turn to and the shortest distance it should travel to. Tweaking these two variables could make the navigation more accurate as sometimes, an imbalance of weight on both sides of the robot could cause a minor difference in the wheelbase. Also the radius and base are human calculations so they are prone to have error.

Further Improvements:

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

The robot slips mostly when it starts or stops moving so reducing the deceleration and/or acceleration would affect that slip. Also we could run a few tests to find out how much the robot slips on average and figure out a constant to multiply the correction with. We could also do the same with the angle by turning the robot a certain amount to determine the error when the robot is turning.

Identify at least one hardware and one software solution. Provide an explanation as to why they would work.

A hardware solution could be either mounting two sensors each facing forty-five degrees or mounting the single sensor on a servomotor hence allowing the sensor to access more angles. This will deal with blind spots and and give the sensor much wider vision. A software solution would be to add an odometer correction class similar to the one in lab 2 hence give us better calculations for our navigation when we get the x, y and theta from the odometer.