GROUP 43s
Ruptanu Chowdhury 260673074
Francis Duhamel 260746909

Lab 4: Localization

Section 1:

Our robot consists of five main parts, two motors to which we connected wheels, one ultrasonic sensor, one light sensor and one EV3 LEGO brick. The two motors are connected closely to the robot and a steel ball is used to offer stability with low friction. The ultrasonic sensor is attached at the front of the robot and aims directly at the 0-degree direction of the robot. The light sensor is attached at the back of the robot, pointing down, and is mounted in such a way that it is close to the surface. The motors are connected to the output of the brick while the two sensors are linked to the inputs of the EV3 brick.

For the Ultrasonic localization, there are 2 distinct methods that can use; the Falling Edge method and the Rising Edge method. Both of these are used to make the robot set itself to an angle of 0 degrees in the direction of the plan. This is implemented using the distance between the walls of the floor tiles.

For the Falling Edge method, the robot faces away from the walls. We ask it to turn to the right and stop when it detects a wall that is closer than 25 cm. It then saves the angle at which the odometer thinks it is as the angle "alpha". Then, we ask it to rotate the other way and stop when, again, it detects a wall that is closer than 25 cm and store the angle of the odometer as "beta".

$$\Delta\theta = 45 - \frac{\alpha + \beta}{2} \quad \alpha < \beta$$
$$\Delta\theta = 225 - \frac{\alpha + \beta}{2} \quad \alpha > \beta$$

Using the equation given, we determine a new angle, "theta": We then ask the robot to rotate (theta + beta) to the right which will leave the robot facing the intersection of the walls. From there, we know that, to get to 0, the robot has to turn 135 degrees to the right.

For the Rising Edge method, the robot starts by facing the walls. Again, we tell it to turn right 360 degrees, but this time stop when it detects that the wall is more than 35 cm away and store the value of the angle of the odometer as "alpha". Then turn to the left and stop when the distance is again more than 35 cm. Store the value of the angle as "beta". Using the same formula, we determine "theta". We then tell the robot to turn (theta + beta) to the right so that it will have its back turned to the intersection of the walls, therefore on the 45-degree axis facing away from the walls. Then all we have to do is turn 45 degrees to the left to realign with 0.

For Light localization method we assume that the US localization works to a certain extent as mentioned in the tutorial slides. After US localization is done, the robot rotates 45 degrees and then moves forward until it detects a line. It then goes backward and starts rotating while the light sensor picks up the lines. The robot records the value of the odometer at each intersection

and records them in an array and stops rotating after intersecting 4 lines. Using the angles detected the software then computes the X and Y distance from the origin as well as the angle it needs to turn to have a 0-degree orientation. After calculating the angle, the robot travels to point (0,0) and rotates so that it ends up facing in the 0-degree direction.

Figure 1: Hardware Design

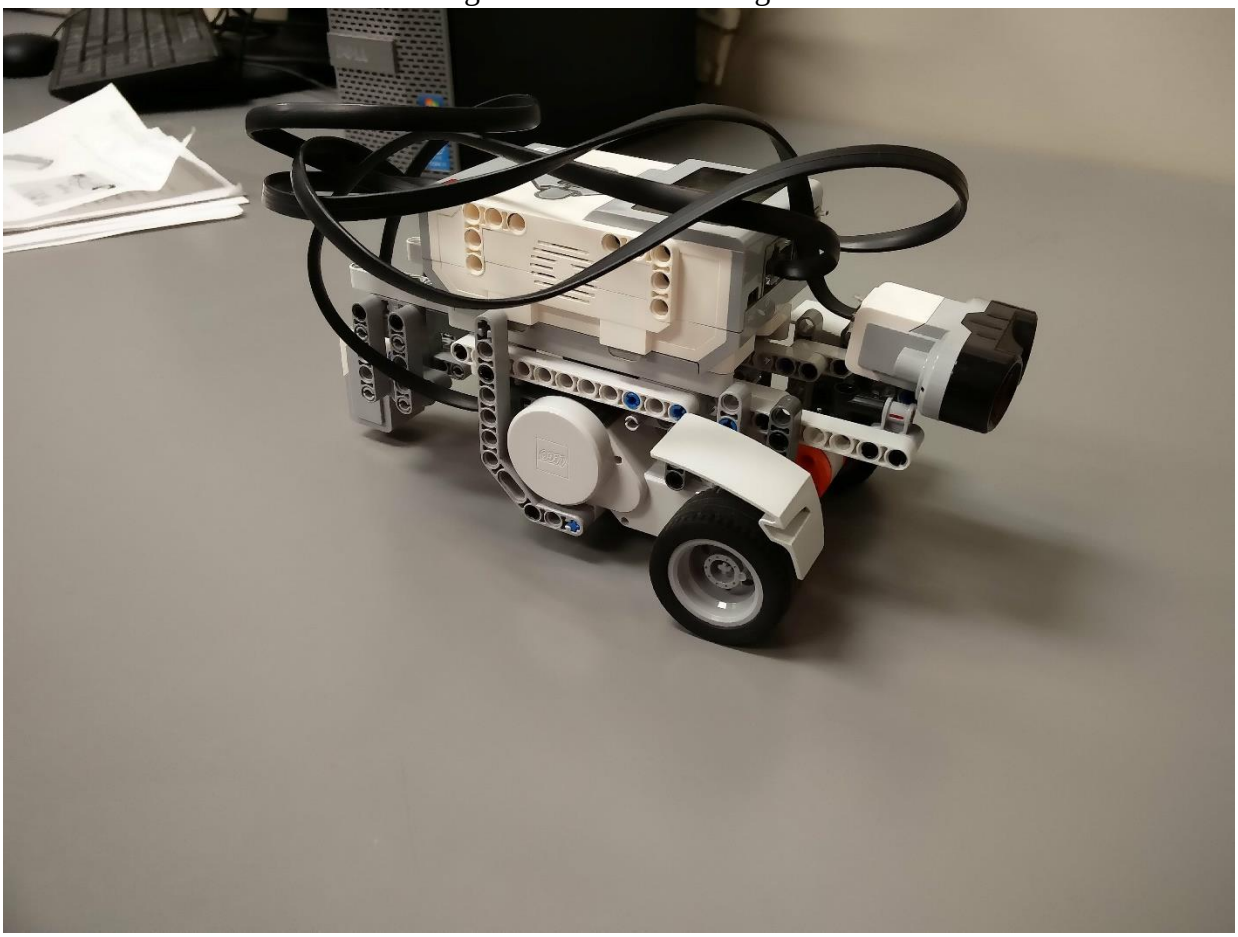


Figure 2: US Localization Flowchart

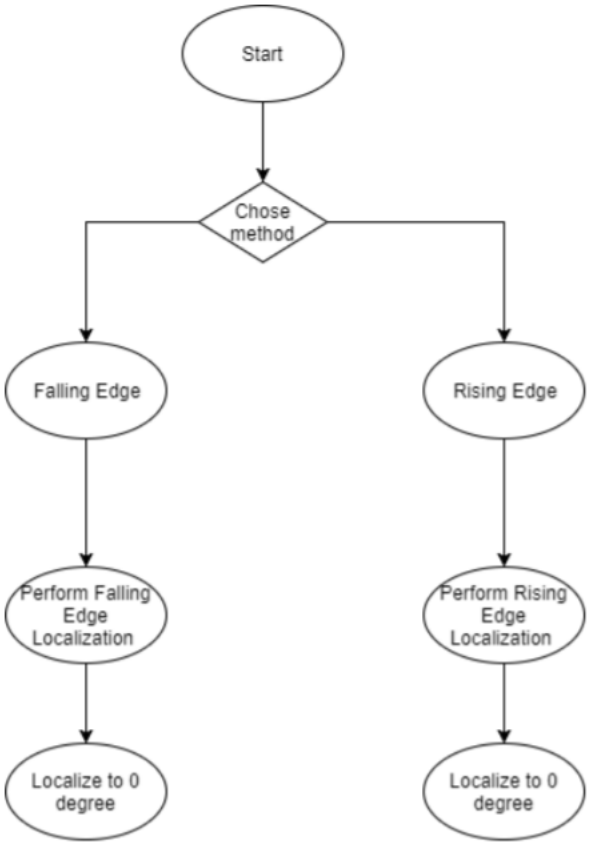
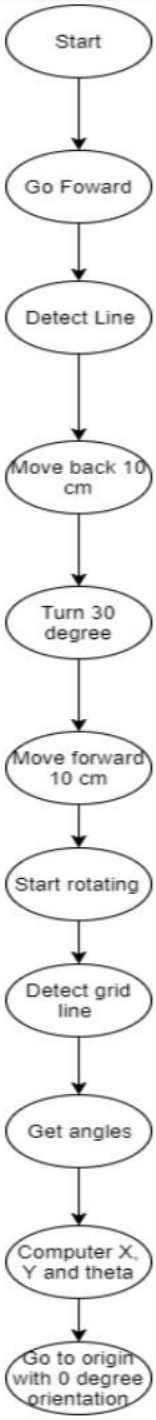


Figure 3: Light Localization Flowchart



Section 2:

Test data for Falling Edge

Trial #	Us Theta error (°)	Final X (cm)	Final Y (cm)	Final Theta error(°)	Euclidean distance error (cm)
1.0	13.0	1.5	0.4	15.0	1.6
2.0	-11.0	1.2	0.3	-9.0	1.2
3.0	16.0	1.7	1.6	13.0	2.3
4.0	11.0	1.1	0.6	14.0	1.3
5.0	10.0	1.3	1.0	9.0	1.6
6.0	8.0	0.8	0.4	6.0	0.9
7.0	14.0	0.9	0.6	9.5	1.1
8.0	18.0	1.9	1.3	15.0	2.3
9.0	16.0	1.6	0.6	16.0	1.7
10.0	9.0	1.1	0.6	8.0	1.3

Test data for Rising Edge

Trial #	Us Theta (°)	Final X (cm)	Final Y (cm)	Final Theta error(°)	Euclidean distance error (cm)
1.0	3.0	0.4	0.2	1.0	0.4
2.0	5.0	0.8	0.4	2.0	0.9
3.0	4.0	0.4	0.2	0.5	0.4
4.0	2.5	0.1	1.3	1.0	1.3
5.0	0.5	0.2	0.1	0.0	0.2
6.0	3.0	1.3	0.6	1.5	1.4
7.0	15.0	2.6	1.3	10.5	2.9
8.0	4.0	0.7	0.2	1.0	0.7
9.0	0.0	0.1	0.2	0.0	0.2
10.0	1.0	0.3	0.2	0.0	0.4

Section 3:
Formula for mean:

$$\bar{x} = \frac{\Sigma x}{N}$$

where,
Σ x: Sum of each value in the set
N: Number of elements

Formula for standard deviation:

$$Sample\ Standard\ Deviation = \sqrt{\frac{\Sigma(x - \bar{x})^2}{(n-1)}}$$

where,
x: Values of the set
$\bar{x}$: Average of the set of values
n: Number of values

The mean:

	Us Theta error	Final Theta error(°)	Euclidean distance error (cm)
Falling Edge	10.4	9.7	1.5
Rising Edge	3.8	17.5	0.9

Standard Deviation:

	error (°)	Final Theta error(°)	Euclidean distance error (cm)
Falling Edge	8.2	7.4	0.5
Rising Edge	4.3	3.1	0.8

Both the mean and the standard deviation were calculated in Microsoft Excel using the built-in formulas for mean and standard deviation which is the same was the formula stated above.

Section 4:
In our case, the localization routine that worked the best was the light localization, and therefore the ultrasonic localization had the worst performance. The final angle was a bit impacted by the original angle of the robot. In each of the methods, the further the original angle was from the 45-degree line, the bigger the impact would be on the final angle. For the ultrasonic methods, factors that contributed to the performance of the method were, amongst others; getting the robot

exactly on the 45-degree axis, the length of the robot and where the sensor was placed, the lighting in the room, the accuracy of the sensor, and the precision of the odometer.
The light sensor compares the value obtained from the board to a fixed value set by us, changes in light condition can result in the sensor missing lines if the brightness of the light increases. If the brightness of the light decreases, then the sensor might detect lines when there aren't any. The chances of these errors occurring can be reduced by ensuring the sensor is placed as close to the ground as possible without actually touching the ground.

Section 5:

In order to minimize errors in the ultrasonic sensor, we could test the accuracy in different ambient lightings, see which one is the best, and use that lighting every time we use the sensor.

Instead of using the rising and falling edge methods, we could use the light sensor to get the direction of the robot by scanning around itself and reading black lines that are near it. It would then use these readings to localize itself based on the position of the lines.

If we are to assume the odometer readings are accurate to a certain extent the robot can use those to navigate near the border of the tile, get the readings off the 4 lines from that border and localize to the coordinates it just calculated.