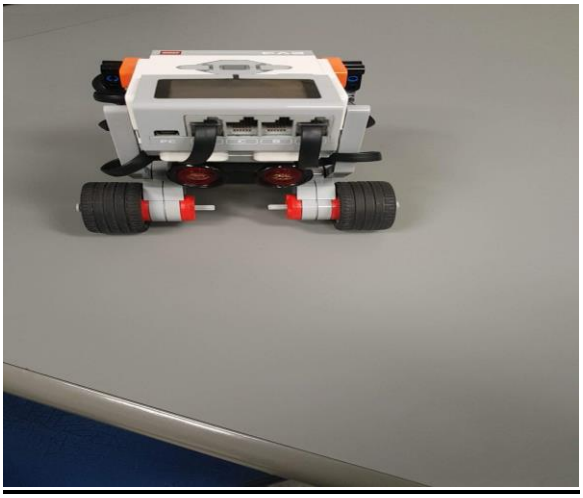
ECSE 211: Localization Lab Report

Ezz Aboulezz 260 677 463
Oscar Deceus 260 744 646

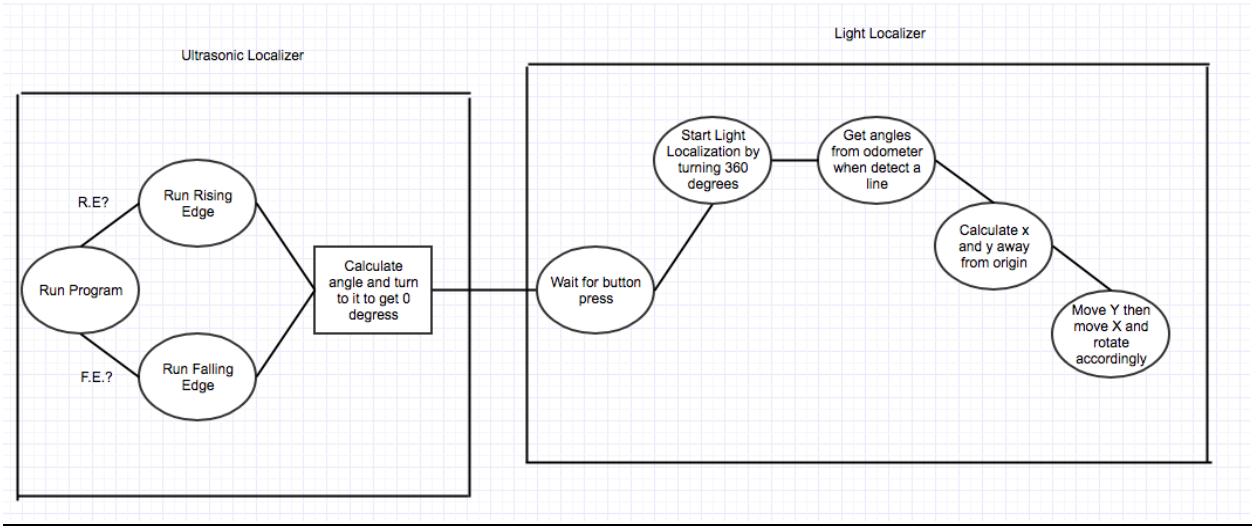
Group 57

Design Evaluation:

Hardware: **(Refer to pictures below to see sensor placement and robot design)**
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 light sensor is installed on the center back of the robot so that it is in line with the robot's center of rotation. The assembling of the robot was mostly driven by several videos and ideas found online.



Software: **Refer to workflow chart below**



Software Evaluation:

Ultrasonic Localization:

Ultrasonic localization uses the US sensor to detect walls and then orient the robot based on those angles. There are two types of US methods, which were developed during the lab.

Falling Edge*:* In the falling edge method, the robot rotated counter-clockwise until it saw a wall and then switched direction until it found the other wall. It then took note of the angles at which it found the wall. It then stopped and calculated the angle to turn to, that was calculated using trigonometry and the fact that the robot was placed on the 45° line.

Rising Edge: The rising edge method was the opposite of the falling edge method. The robot rotated counter-clockwise until it did not see a wall anymore and then switched direction until it stopped seeing a wall once more. It then took note of the angles at which it got too big signals. The robot then stopped and calculated the angle to turn to, that was calculated using trigonometry and the fact that the robot was placed on the 45° line.

Light Localization:

The light localization method used the light sensor and gridlines on the ground in order to determine the position of the robot. The method assumed that it started facing 0° since it came right after US Localization. First, the robot turned 45 degrees and moved a fixed distance in order to get closer to the origin and be able to detect all the gridline it needs. Then the robot turned 360 degrees and every time we detected a line the robot would get the current orientation from the odometer and save the angles in the order of finding them. Then after the robot stops it calculates the x and y distances away from the origin using the given trigonometry rules found in the tutorial. Then the robot drives in the Y direction a distance of y, which it calculated previously. Then it turns 90 degrees and does the same thing for the x distance found in he calculation. Finally it turns -90 degrees to go back to 0 degrees orientation at the origin.

Test Data: (Refer to tables below)

Table 1: Test Data for localization using Rising Edge

	Ultrasonic Angle Error (º)	Euclidean Distance Error (cm)	Final Angle Error (º)
1	0.25	0	4.83
2	0.25	0	4.91
3	0.05	1.015332	5.01
4	0.35	0.62434	4.81
5	0.06	0.665282	5.12
6	0.35	0.604401	4.81
7	0.75	0.710634	4.91
8	0.05	0.474236	4.75
9	0.25	0.51225	4.8
10	0.25	0.364005	4.83

Table 2: Test Data for localization using Falling Edge

	Ultrasonic Angle Error (º)	Euclidean Distance Error (cm)	Final Angle Error (º)
1	0.2	0.434166	4.95
2	0.25	0.403113	4.85
3	0.2	0.320156	4.91
4	0.33	0.518556	4.89
5	0.05	0.494975	4.87
6	0.15	0.429418	4.92
7	0.2	0.318277	4.93
8	0.25	0.367967	4.88
9	0.2	0.339706	4.87
10	0.15	0.390512	4.82

Test Analysis:

Rising Edge:

$$A = \frac{1}{n} * \sum_{i=1}^{n} x_i$$

Using the arithmetic mean formula , where n is the number of elements and Xi is an element, we can see that the mean is equal to 0.261º for the **ultrasonic angle error**, 0.497 for the **Euclidean distance error** and 4.878 for the **final angle error**.

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

Using the standard deviation formula , where N is the number of elements, Xi an element and **μ** is the mean, we can see that the standard deviation is 0.207º for the **ultrasonic angle error**, 0.313cm for the **Euclidean distance error** and 0.113º for the **final angle error**.

Falling Edge:

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 0.198º for the **ultrasonic angle error**, 0.0.402cm for the **Euclidean distance error** and 4.889º for the **final angle error**.

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 is 0.074º for the **ultrasonic angle error**, 0.069cm for the **Euclidean distance error** and 0.039º for the **final angle error**.

Observations and conclusions:

- Which of the two localization routines performed the best? Which preformed the worst?

If we are comparing the US-Localization to the Light-Localization, the Light-Localization preformed better with regards to reaching 0 degrees orientation at the end of the program run-time. Comparing rising-edge to falling-edge, the falling edge was better and more consistent. The main reason for this is the fact that falling edge relies on finding the first point on the wall, which is an easier task for the sensor as it can always make errors when the wall is too close.

- Was the final angle impacted by the initial ultrasonic angle?

Yes, the final angle was impacted by the initial angle. This is a product of our implementation. In the light localization method, the angle does not get corrected. We were using the odometer to get our angles and this assumed that the US performed within a very small degree of error. A further improvement would be to add a correction in the algorithm to be able to get a correct final orientation.

- What factors do you think contributed to the performance of each method?

Firstly, if the placement of the robot is not almost perfectly on the specified diagonal then the error can become huge due to the fact that the algorithm is based on this assumption. Also, the inaccuracy of the ultrasonic sensor, even after filtering, introduced a lot of error. Finally, the performance varied depending on the initial orientation of the robot. The best results were obtained when the robot was facing 45° for falling edge, and 225° for rising edge.

- How do changing light conditions impact the light localization?

During the weeks of the labs, the light was kept constant hence there was no effect on the light sensor. However if this were not the case then one would have to get the rate of change of light intensity instead of setting a threshold as a condition. This would ensure that any light change in the room would not affect the readings and the performance of the code would remain unchanged.

Further Improvements:

- Propose a way to minimize errors in the ultrasonic sensor.

The errors in the ultrasonic sensor are caused by the noise in the values. The noise tends to be large outliers hence instead of the current filter, which clips and averages the values of the sensor, a median filter should be added which ignores the outliers. Since the noise is much larger than the actual values, the median filter works better. The median filter works by looking for the median in a window and replacing values that are larger than the median value with the median.

- Propose another form of localization other than rising-edge or falling-edge.

Another form of localization using the ultrasonic sensor would be to rotate 360° while sampling the values. The sampled values and their respective angles should be recorded. Then, the angles where the robot is facing perpendicular to each wall can be found by finding the smallest distances. The angle to turn to can then be calculated using the angles found.

- 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.

If we assume that the odometer can give us our location to within a square, one step can be added to the light localization method to correct in a square that's not in the corner. First, the robot should turn to 0°, drive until it reaches a line and then reverse by a smaller distance. The robot should then turn 90°, drive until it reaches another line and then reverse by a smaller distance. After having detected both lines the robot can check and see which of the coordinates corresponds to the square it is in. After that it can perform light localization as demonstrated above.