ECSE 211 Lab 4
Localization Lab Report

Group 65
Kathy Yim 260741794
Hanwen Wang 260778557

Design Evaluation

The EV3 robot in this lab consists of five main components: a brick, two motors, an ultrasonic sensor, and a light sensor. The goal is to design a system that allows the robot to localize itself to its starting point using the ultrasonic sensor and light sensor. The ultrasonic sensor is placed in front center of the brick to detect walls and reorient itself accordingly. The light sensor is positioned in the back left corner of the robot and used to detect black lines. A hardware overview of our robot is shown in figure 1 below.

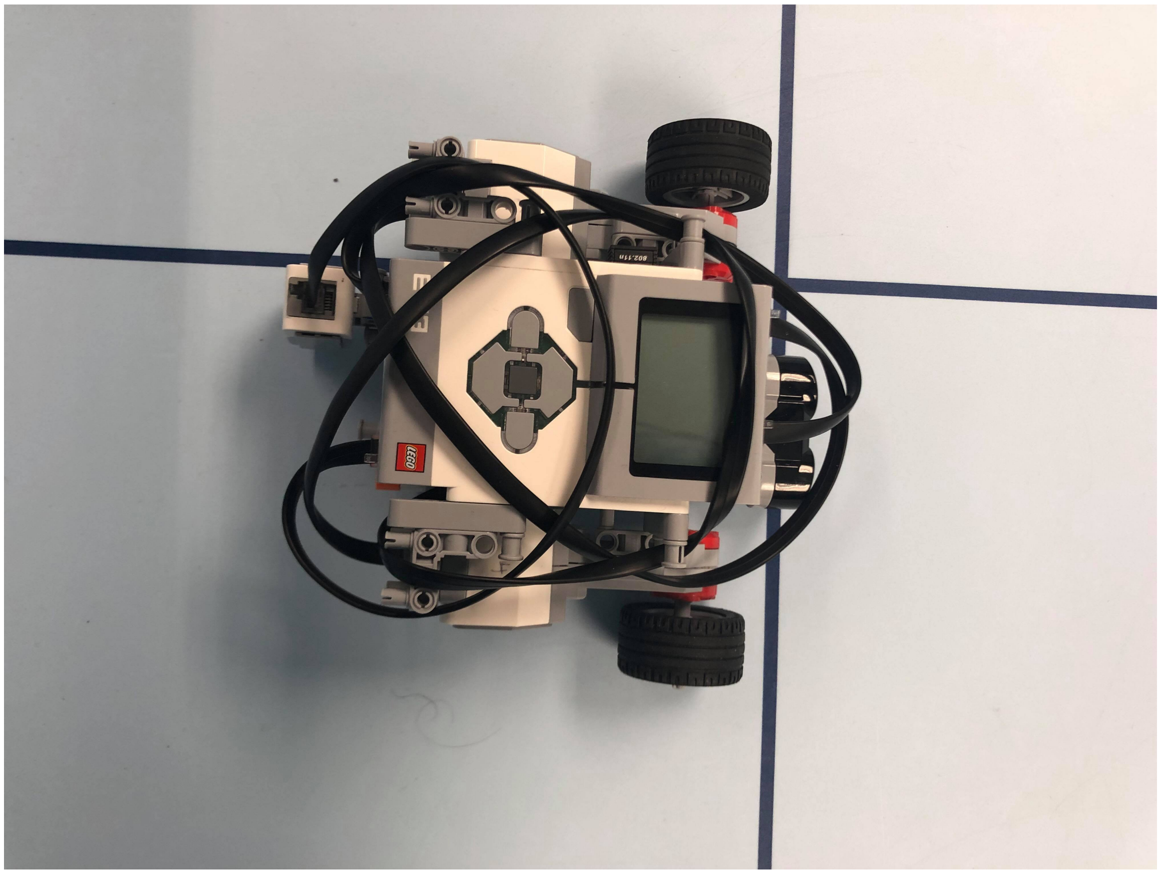

Figure 1: Hardware Overview

The software design of this lab consists of the USLocalizer and LightLocalizer. The USLocalizer determines the orientation of the robot by detecting its distance to the wall and then the robot turns to face the 0 degree direction. The light sensor detects four black grid lines and the robot localizes to the (0,0) point on the map facing the 0 degree direction.

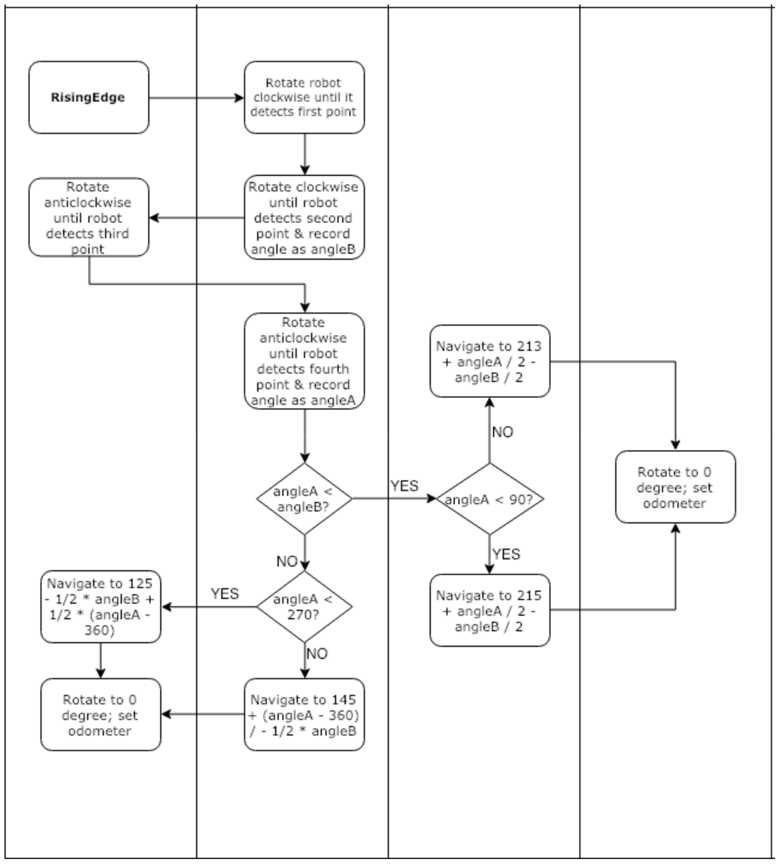
RisingEdge
Rotate robot clockwise until it detects first point
Rotate clockwise until robot detects second point & record angle as angleB
Rotate anticlockwise until robot detects third point
Rotate anticlockwise until robot detects fourth point & record angle as angleA
angleA < angleB?
YES
angleA < 90?
NO
Navigate to 213 + angleA / 2 - angleB / 2
YES
Navigate to 215 + angleA / 2 - angleB / 2
Rotate to 0 degree; set odometer
NO
angleA < 270?
YES
Navigate to 125 - 1/2 * angleB + 1/2 * (angleA - 360)
NO
Navigate to 145 + (angleA - 360) / - 1/2 * angleB
Rotate to 0 degree; set odometer

Figure 2: Rising Edge flowchart

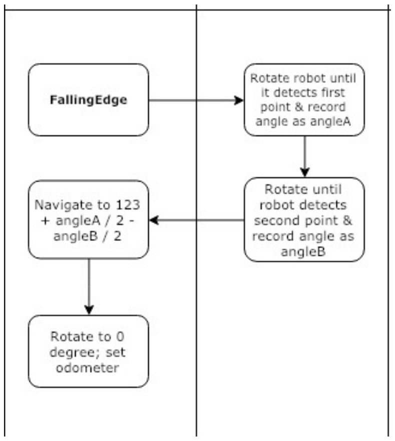
FallingEdge
Rotate robot until it detects first point & record angle as angleA
Rotate until robot detects second point & record angle as angleB
Navigate to 123 + angleA / 2 - angleB / 2
Rotate to 0 degree; set odometer

Figure 3: Falling Edge flowchart

Shown above in figure 2 and 3 are the rising edge and falling edge methods we implemented in the US Localizer. Both methods utilize the ultrasonic sensor to determine the robot's orientation and distance to the wall. These values are then used to reorient the robot towards the 0 degree direction.

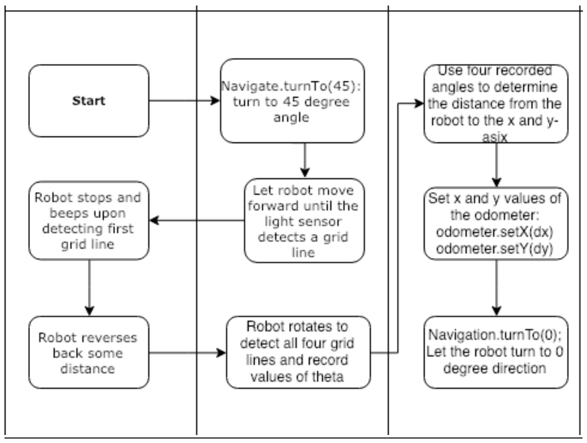
Start
Navigate.turnTo(45): turn to 45 degree angle
Let robot move forward until the light sensor detects a grid line
Robot stops and beeps upon detecting first grid line
Robot reverses back some distance
Robot rotates to detect all four grid lines and record values of theta
Use four recorded angles to determine the distance from the robot to the x and y-asix
Set x and y values of the odometer: odometer.setX(dx) odometer.setY(dy)
Navigation.turnTo(0); Let the robot turn to 0 degree direction

Figure 4: Light Localizer Flowchart

In figure 4 above, a brief workflow of light localization is shown.

Test Data

Trial	**Light Sensor Distance (x,y) (cm)**	**US Sensor Angle Error (degree)**	**Light Sensor Final Angle Error (degree)**	**Euclidean Distance Error**
1	(0.2, 0.4)	2.00	1.00	0.45
2	(0.1, 0.2)	1.00	4.00	0.22
3	(0.0, 0.0)	1.00	3.00	0.00
4	(0.4, 0.6)	0.00	1.00	0.72
5	(0.3, 0.6)	3.00	0.00	0.67
6	(0.1, 0.4)	2.00	1.00	0.41
7	(0.0, 0.0)	3.00	3.00	0.00
8	(0.5, 0.7)	1.00	5.00	0.86
9	(0.2, 0.5)	1.00	2.00	0.54
10	(0.7, 0.8)	0.00	2.00	1.06

Table 1: Rising Edge Trials

Trial	**Light Sensor Distance (x,y) (cm)**	**US Sensor Angle Error (degree)**	**Light Sensor Final Angle Error (degree)**	**Euclidean Distance Error**
1	(0.2, 0.4)	2.00	1.00	0.45
2	(0.1, 0.2)	1.00	2.00	0.22
3	(0.1, 0.3)	2.00	3.00	0.32
4	(0.3, 0.4)	0.00	0.00	0.50
5	(0.3, 0.6)	1.00	0.00	0.67
6	(0.1, 0.4)	1.00	3.00	0.41
7	(0.0, 0.0)	3.00	4.00	0.00
8	(0.3, 0.5)	0.00	0.00	0.58
9	(0.2, 0.6)	2.00	2.00	0.63
10	(0.7, 0.8)	1.00	2.00	1.06

Table 2: Falling Edge Trials

Euclidean Error Equation:

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

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)*

Test Analysis

	US Sensor Angle Error (degree)	**Light Sensor Final Angle Error (degree)**	**Euclidean Error**
Mean	1.40	2.20	0.49
Standard Deviation	1.07	1.55	0.35

Table 3: Error Mean & SD for Rising Edge Trials

	US Sensor Angle Error (degree)	**Light Sensor Final Angle Error (degree)**	**Euclidean Error**
Mean	1.30	1.70	0.48
Standard Deviation	0.95	1.42	0.29

Table 4: Error Mean & SD for Falling Edge Trials

Mean Calculation:

$$\bar{\boldsymbol{\epsilon}} = \frac{\sum_{i=1}^{i=\mathrm{N}} \epsilon_i}{N}$$

Where

- *ϵ_i is the i^{th} error*
- *N is the number of observations*

Standard Deviation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{i=N} (\epsilon - \bar{\boldsymbol{\epsilon}})^2}{N-1}}$$

Where

- *ϵ_i is the i^{th} error*
- *$\bar{\epsilon}$ is the mean error*
- *N is the number of observations*

Observations and Conclusions

Which of the two localization routines is performed the best?
Falling edge is the better localization routine in this lab as proven in table 3 and 4. The mean and standard deviation on error is smaller for falling edge than it is for rising edge. This is most likely because the total angle of rotation for rising edge is larger than that for falling edge; therefore contributing to a larger Euclidean error and lower performance.

Was the final angle impacted by the initial ultrasonic angle?
No, as long as the initial assumption that the robot is placed along the 45-degree line is satisfied, the final angle of the robot will not be impacted by the initial angle. The robot will always rotate some degree until the ultrasonic sensor detects a dramatic change in distance. It will then turn in the other direction until the sensor detects another dramatic change in distance. Therefore, regardless of the robot's initial angle, its final angle will be the same.

What factors do you think contributed to the performance of each method?
The accuracy of the odometer will contribute to the performance of the localization. Since the robot relies on the odometer to determine how much it needs to rotate and its exact orientation, a faulty odometer system can lead to significant error. Additionally, friction between the motors and the ground may influence the precision of when the robot is travelling in a straight path versus when it's turning by some angle. Ultrasonic wave interference can also lead to errors when detecting edge, which will cause errors in ultrasonic localization.

How do changing light conditions impact the light localization?
Since the method we used for the light localizer detects black lines by determining the intensity of the sample detected by the sensor, changes in light conditions will easily impact light localization. For example, if the room is very dark, the light sensor may fail to detect the presence of a grid line. On the other hand, if the room is very bright, the sensor will detect a brighter light reflected from the ground. In this case, it is also possible that the sensor will fail to detect a black line.

Further Improvements

One way to reduce errors using software is to implement a filter. It will help discard samples corresponding to a null signal. In essence, if the sensor detects an outlier value, the robot does not account for it unless this value is detected repeatedly.

One form of localization we could use is to let the robot rotate 360 degrees, and detect all the possible angles on a circular path. The most important angles are then recorded; the angle at which the distance decreases drastically, and the angle at which the distance increases drastically. These values can then be used to calculate the angle needed for the robot to rotate to the 0 degree angle.

If the color of each grid in different, light localization can be used when the robot is at the fringe of each grid. When the robot rotates, it detects a different color. Through odometry, the robot can determine how far it is from the fringe of the grid. However, this method can only take x or y-values. Alternatively, after navigating to the (0, 0) point and facing the 0 degree direction, the robot can continue to detect black lines. These black lines can be references to update the current x and y values stored in the odometer. In other words, each time the robot detects a black line, it knows it's 30.48cm from the previous point. To correct the odometry data, every time the robot detects a black line, the x and y values are multiplied by 30.48cm, and the odometer readings are updated based on the calculations. The algorithm implementation is very similar to that used in lab 2.