Group 26
Rene Gagnon 260801777
LiLy Li 260664379

Lab 4: Localization

Section 1-Design evaluation:

For this lab, we basically kept the same hardware design as lab 3. The only modification made was that we moved the light sensor from the middle of the wheel base to the back of the robot. This was necessary for the light localization part of the lab. Since the robot needed to turn 360 degrees and cross four lines, we needed a good distance between the light sensor and the center of rotation (middle of the wheel base). In our case, this distance was 15 cm in our case. With testing, we determined that this distance was adequate (not too short and not too long). This modification took the previous location of the third wheel. We simply moved this wheel more toward the front of the robot. The hardware was built before we started working on the software and testing.

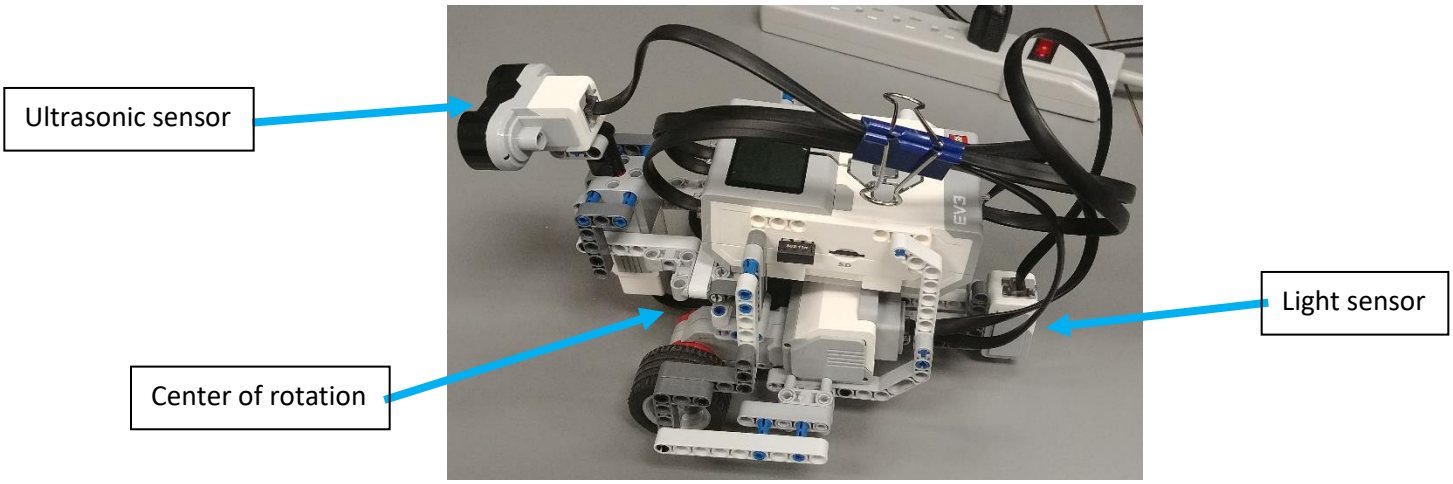

Figure 1: Left side view of the robot

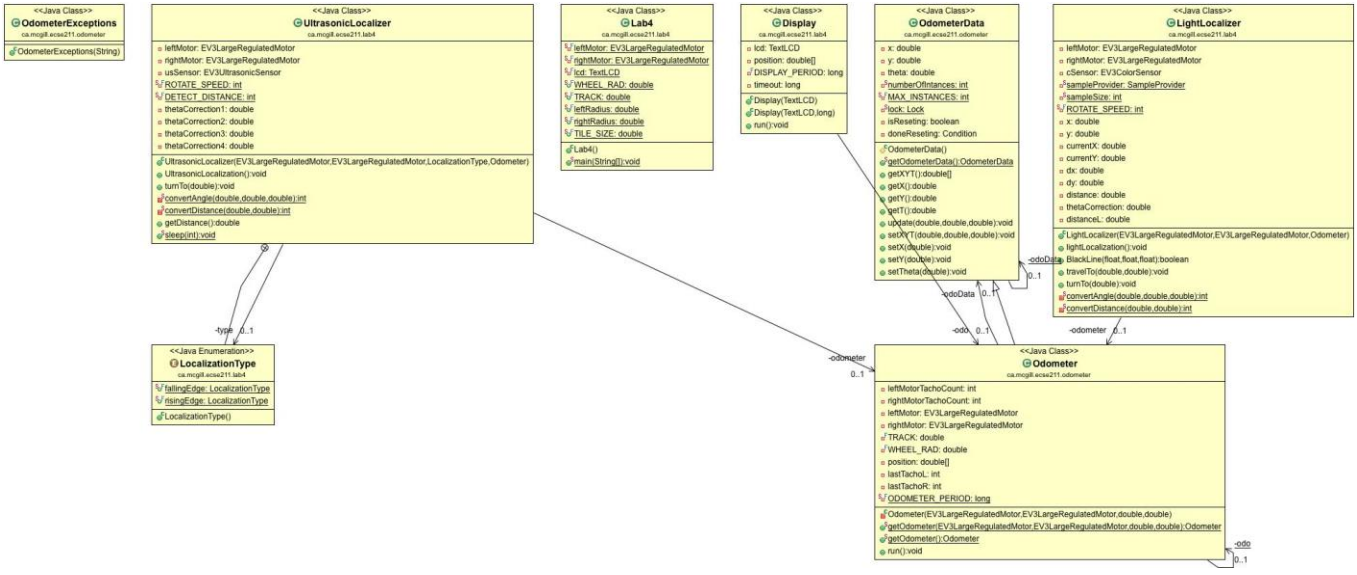

Figure 2: class diagram

For the software design, we kept the odometer from the previous labs, and added two additional classes: UltrasonicLocalizer and LightLocalizer.

From the main method, a new object of UltrasonicLocalizer is created given two motors, the odometer and the Localization type. The UltrasonicLocalization method checks which type has been chosen by the user and perform

localization accordingly. Using the example of rising edge type: first the robot starts turning clock wise, since we want the robot to perform localization regardless of the orientation, we used a while loop which breaks when the robot detects a wall, so that if the robot is heading towards nothing at first, the program stays in the while loop. Next, once the robot is facing to a wall, both motors stop when the distance has reached 45cm which is the threshold value found using trial and error method, and record the current theta from the odometer as alpha. Then the robot turns counter clock wise to find the next rising edge and record the theta as beta. Using the formula from the tutorial slides, the angle to correct the odometer with is calculated and set to the odometer. Then turn the robot to 0 degrees by using the method turnTo from the previous lab.

For the LightLocalizer, first turn 45 degrees and move straight for about 5 cm so that the light sensor can detect all four lines when making a 360 degrees turn. Then make the robot turn counter clock wise and record all four thetas in an array when it detects a blackline. Then use the formula provided from the tutorial slides, calculate the current position and set it to the odometer, then move the robot to coordinate 0,0 and heading 0 degree.

Section 2- Test data:

Localization using rising edge:

Test	Expected theta	Theta after ultrasonic localization	Expected final x(cm)	Expected final Y(cm)	Final X(cm)	Final Y(cm)	Final theta (degrees)	Euclidean error distance(cm)	Error of theta (degrees)	Final theta error (degrees)
1	0	-2.450	0	0	0.0	0.3	-5	0.30	-2.450	-5
2	0	1.636	0	0	0.0	-0.8	2	0.80	1.636	2
3	0	-0.818	0	0	0.0	-0.4	-2	0.40	-0.818	-2
4	0	0.409	0	0	0.0	-0.2	0	0.20	0.409	0
5	0	3.670	0	0	0.5	-0.4	10	0.64	3.670	10
6	0	-2.860	0	0	0.0	-0.2	-5	0.20	-2.860	-5
7	0	2.800	0	0	0.2	-0.4	5	0.45	2.800	5
8	0	2.050	0	0	0.0	-0.9	5	0.90	2.050	5
9	0	2.045	0	0	-0.1	-0.5	4	0.51	2.045	4
10	0	-1.636	0	0	-0.2	-0.3	-6	0.36	-1.636	-6

Table 1: Data from 10 runs of localization using rising edge (random starting orientation).

Localization using falling edge:

Test	Expected theta with US sensor (degrees)	Theta with US sensor (degrees)	Expected final X (cm)	Expected final Y (cm)	Final X (cm)	Final Y (cm)	Expected Final theta (degrees)	Final theta (degrees)	Euclidean error distance (cm)	Theta with US sensor error (degrees)	Final theta error (degrees)
1	0	-1	0	0	0.3	0.7	0	10	0.76157731	-1	10
2	0	0	0	0	0.0	0.5	0	5	0.50000000	0	5
3	0	4	0	0	-0.4	0.6	0	-5	0.72111026	4	-5
4	0	2	0	0	1.0	1.0	0	-5	1.41421356	2	-5
5	0	-2	0	0	0.1	-0.2	0	-10	0.22360680	-2	-10
6	0	2	0	0	-0.5	-0.4	0	-7	0.64031242	2	-7
7	0	8	0	0	-0.5	-0.5	0	6	0.70710678	8	6
8	0	5	0	0	-0.5	-0.5	0	-4	0.70710678	5	-4
9	0	-1	0	0	-0.2	-0.4	0	3	0.44721360	-1	3
10	0	4	0	0	-0.3	-0.3	0	3	0.42426407	4	3

Table 2: Data from 10 runs of localization using falling edge (random starting orientation).

For these tests, the rotating speed was 80 degrees/sec for the ultrasonic localizer and 100 degrees/sec for the light localizer. The forward speed was 100 degrees/sec. The track of the robot was 12.2 cm and the wheel radius 2.19 cm. The threshold for detecting a rising or a falling edge was 45 cm. We had a theta correction factor of -0.8 degrees for falling edge and 7.5 degrees to the light localizer.

Section 3-Test analysis:

Euclidean error distance mean (cm)	Euclidean error distance standard deviation (cm)	Final theta error mean (degrees)	Final theta error mean (degrees)
0.65465	0.31699	-0.40000	6.60303

Table 3: Mean and standard deviation of the 10 Euclidean errors shown in table 2.

Sample Calculations:

- Euclidean error distance for falling edge test 1:

$$\epsilon_1 = \sqrt{(X_D - X_F)^2 + (Y_D - Y_F)^2} = \sqrt{(0 - 0.3)^2 + (0 - 0.7)^2} = 0.76157731\ cm$$

- Euclidean error distance mean for falling edge:

$$mean = \bar{\varepsilon} = \frac{\sum_1^n \varepsilon_i}{n} = \frac{\sum_1^{10} \varepsilon_i}{10} = \frac{6.5465}{10} = 0.65465\ cm$$

- Euclidean error distance standard deviation for falling edge:

$$standard\ deviation\ of\ \epsilon = \sqrt{\frac{\sum_1^n |\epsilon_i - \bar{\epsilon}|^2}{n}} = \sqrt{\frac{\sum_1^{10} |\epsilon_i - 0.65465|^2}{10}} = 0.31699\ cm$$

Section 4-Observations and conclusions:

- Which of the two localization routines performed the best?

Falling edge and rising edge had very similar result when used to determine the theta using the ultrasonic sensor. Therefore, we cannot really say which one was better.

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

The final angle was impacted a bit by the initial ultrasonic sensor. Usually, if the initial angle was off by a lot, then the final angle would also be off. But as we improve our design, the initial angle got better and better up until a certain point where it would always be under 10 degrees of error. At this moment, we observed that the final angle was not really affected by the initial angle.

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

For the ultrasonic localization, the odometer contributed negatively to the performance. Since the odometer is never corrected, a wheel slip can cause the reading to be wrong and this error is further carried on to later localization. The initial position of the robot also contributed to the performance, if the center of rotation is not placed exactly on the 45-degree line, then the ultrasonic localization will be off.

For the light localization, the errors we had from testing are always within 2cm. The light sensor never missed a line because the range we set works with almost all light conditions in the lab.

- How do changing light conditions impact the light localization?

We used the RGB mode for the light sensor and have a separate method to determine if it is a blackline given the RGB values provided by the sensor. During testing, we have noticed that those values can vary depending on the lighting of the lab room, so we checked the values using the tools of the brick to make sure the range we set is reliable for all conditions.

Section 5-Further improvements:

To improve the precision of the ultrasonic sensor, we could calibrate it before using it. We observed that the distance reported by the sensor is always a bit smaller than the reality. We account for this in our code, but a better way to overcome this issue would be to do a proper calibrating of the sensor before using it.

Instead of using rising or falling edge for localization, we could make the robot do a full 360 degree turn and record a distance value at every degree for example. We then take the shortest distance and make the robot head in this direction. Since we are in a corner, on the 45 degrees line, we know for sure that the shortest distance to a wall is either going to be at 180 or 270 degrees. To determine which one of these it is, we can make the robot turn clockwise for a short amount and if it is sees no wall then we know the right orientation is 270 degrees and in the other case the right orientation is 180 degrees.

We could use the light correction in the middle of the grid, or pretty much everywhere on the grid. If the robot is on the 45 degrees line, we can apply the same technique as in this lab and get values for X and Y. These values are with respect to an origin that is on the upper left corner of the tile our robot is presently in. But if we know, in which tile our robot is in, we can correct this X and Y to be with respect to the original origin. We can then update the X and Y of the odometer of our robot.