Yuhang Zhang [260787441] Group 7
Mohamed Youssef Bouaouina [260765511]

Lab 4 Report - Localization
ECSE 211: Design Principles and Methods

Design Evaluation

In Lab4, as it can be seen in figure 1, our workflow consists essentially of 3 stages: setting up the hardware, designing the software, and conducting tests. In this first section, we will mostly cover the challenges we faced and the decisions we made during the first two stages.

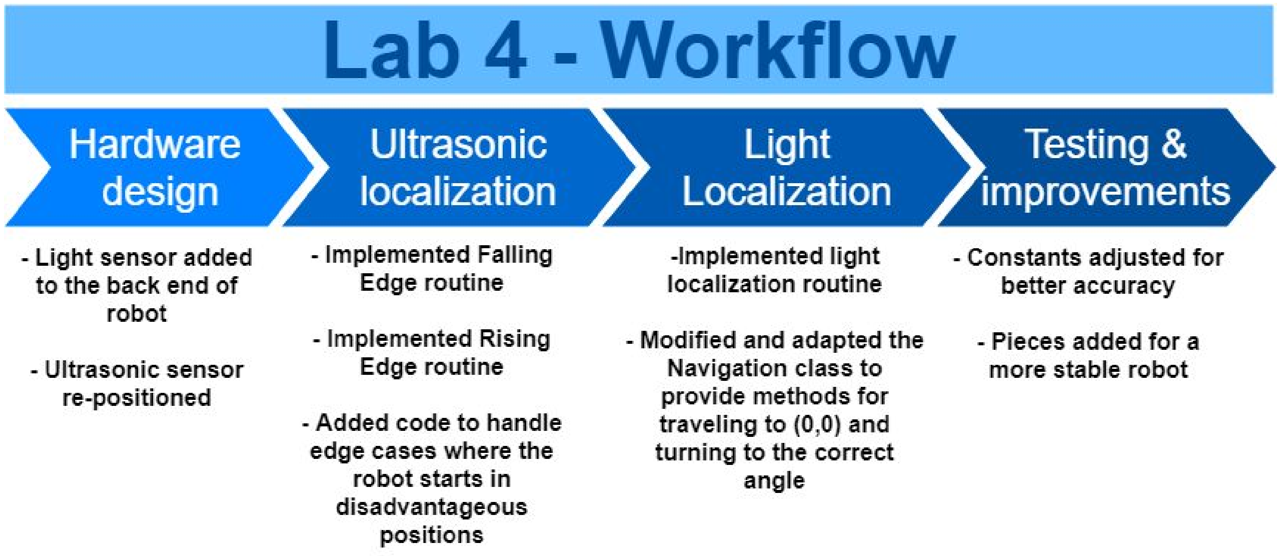
Lab 4 - Workflow
Hardware design
Ultrasonic localization
Light Localization
Testing & improvements
- Light sensor added to the back end of robot
- Ultrasonic sensor re-positioned
- Implemented Falling Edge routine
- Implemented Rising Edge routine
- Added code to handle edge cases where the robot starts in disadvantageous positions
-Implemented light localization routine
- Modified and adapted the Navigation class to provide methods for traveling to (0,0) and turning to the correct angle
- Constants adjusted for better accuracy
- Pieces added for a more stable robot

Figure 1: workflow during lab 4

a) Setting up the hardware:

Unlike the first three labs, the requirements in Lab 4 did not urge us to perform major changes to the overall structure of the robot: the chassis remained the same as the one used for Lab 3. However, as it can be seen in figure 2, we placed the color sensor at the rear end of the robot in an attempt to keep it as far as possible from the the center of rotation of the wheels. Indeed, this was done for an optimal detection of dark lines during the light localization phase.

Finally, the ultrasonic sensor of the robot used for lab 3 was connected to a small motor to make it rotate and detect obstacles ahead. Since the robot itself turns to detect the walls in this lab, we chose to dismantle the motor and simply position it at the front end of the robot as it can be seen in fig. 3.

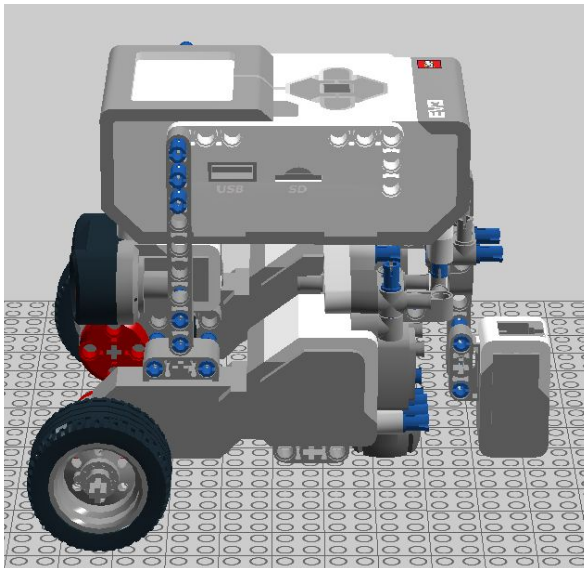

Figure 2: side view of the robot

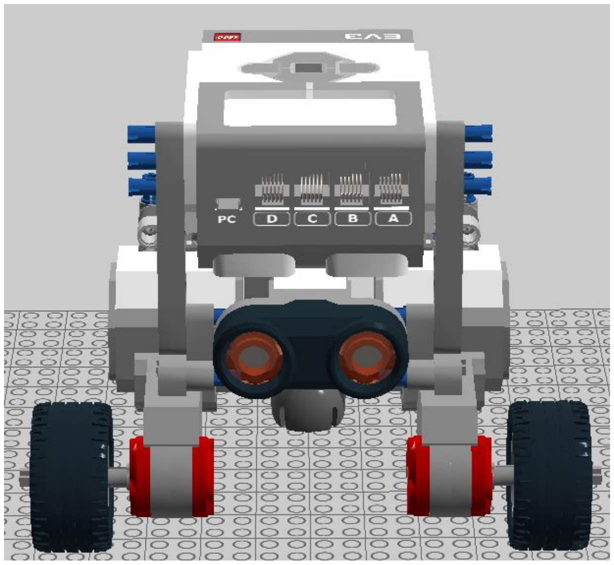

Figure 3: front view of the robot

b) *Software design:*

The software design process for this lab consisted of two majors phases. First of all, we created an UltrasonicLocalizer class that would be called from the main class in order to perform the ultrasonic localization. Secondly, we created the LightLocalizer class that is also called from the main class to perform the more accurate light localization.

Using the logic and the formulae from the Localization tutorial, we started by implementing the Falling Edge routine: the robot starts away from the wall and gradually turns towards it while polling the Ultrasonic Sensor for data. Once a distance below a set threshold is detected i.e. a falling edge, the localizer stores the angle at which the first falling edge was detected using the odometer methods. However, as the robot is supposed to start turning in the other direction to detect the other wall, we encountered a problem: the robot wrongly detects another falling edge right after it detects the first one. This produces erroneous calculations as only one wall is detected and, consequently, the robot does not turn to the 0° angle. To avoid this, we added code so that, once it detects the first falling edge, the robot immediately turns 90 degrees away from the wall before resuming the search for the second falling edge. The same procedure was applied in the code for the Rising Edge routine when we observed a similar problem.

Since the lab instructions indicated that the robot may be placed in any direction and position on the 45° line, we also incorporated additional code to make sure that the robot is in an optimal position before starting to look for falling/rising edges. Indeed, in the case of the Falling Edge routine, the robot is ideally placed facing away from the wall. Therefore, we instructed the robot to turn 180° if it detects a close wall before starting to search for falling edges. Similarly, a robot following the Rising Edge routine ideally starts operating while facing a wall. If this is not the case, we instructed the robot to keep turning in a clockwise fashion until it is facing a wall.

Once the Ultrasonic Localization tested and approved, we moved on to the implementation of the light localization. Once again, we used the tutorial formulae to write our code. However, as the tutorial material does not mention how to do it, our first challenge consisted of positioning the robot such that the light sensor's path crosses all four lines. We solved this problem by turning the robot 45° away from the 0° angle (obtained with US localization) and driving the robot forward until it detects a dark line. When a dark line is detected, the robot rolls backwards for a distance equal to the length of the robot (from light sensor to center of wheels). Therefore, the robot's center of rotation is now very close to the (0,0) point and it can start the light localization.

In order to be able to perform light localization on any board, we chose to implement a differential filter for the line detection: the robot starts first by sampling and storing the color reflection from the center of the board tile. Then, the robot continues to sample while traveling. If a sample indicates that the light reflection is much lower than the initial recorded value, the robot understands that it just crossed a line and performs the necessary and adequate action. For more details on this, please refer to the "Observations and conclusions" section.

Test Data

a) *Rising edge localization*

Trail	Distance Error (x,y) (cm)	Us sensor Angle Error (degree)	Light sensor Angle Error (degree)	Euclidean Error (cm)
1	(0.8, 1.2)	4	5	1.44
2	(0.2, 2.1)	9	0	2.11
3	(0.1, 1.5)	9	10	1.50
4	(0.1, 0.1)	6	8	0.14
5	(0.0, 0.0)	13	5	0.00
6	(0.5, 0.2)	10	2	0.54
7	(0.7, 0.4)	2	2	0.81
8	(1.1, 1.6)	7	0	1.94
9	(1.3, 1.1)	8	0	1.70
10	(0.3, 1.2)	3	5	1.24

Table . 1 rising edge localization

b) Falling edge localization

Trail	Distance Error (x,y) (cm)	Us sensor Angle Error (degree)	Light sensor Angle Error (degree)	Euclidean Error (cm)
1	(1.1, 2.1)	1	1	1.22
2	(0.2, 2.1)	5	2	2.37
3	(1.3, 1.5)	2	2	1.98
4	(0.1, 0.3)	2	0	1.14
5	(0.8, 0.4)	1	5	0.89
6	(1.2, 2.3)	0	5	2.59
7	(1.4, 2.1)	5	2	2.52
8	(0.4, 0.1)	2	12	0.41
9	(0.2, 0.0)	4	0	0.20
10	(0.1, 0.6)	3	1	0.61

Table . 2 falling edge localization

Test Analysis

$$\text{Mean} = \sum \frac{x_n}{n}$$

$$= \frac{1+5+2+2+1+0+5+2+4+3}{10}$$

$$= 2.50$$

$$\text{Standard Deviation} = \sqrt{\frac{1}{N-1}\sum_1^N (x_i - \bar{X})^2} =$$

$$\sqrt{\frac{(1.00-2.50)^2+(5.00-2.50)^2+(2.00-2.50)^2+(2.00-2.50)^2+(1.00-2.50)^2+(0.00-2.50)^2+(5.00-2.50)^2+(2.00-2.50)^2+(4.00-2.50)^2+(3.00-2.50)^2}{10}}$$

$$= 1.72$$

	Ultrasonic Angle Error (◦)	Euclidean Error (cm)	Light Sensor Error(◦)
Mean (cm)	2.50	1.40	3.00
Standard deviation (cm)	1.72	0.90	3.62

Observations and conclusions

Which of the two localization routines performed the best?

Based on our tests, the Falling Edge routine yielded better and more accurate results than the Rising Edge routine. This can be explained by the fact that an ultrasonic sensor makes more false negative errors than false positive ones. Indeed, we noticed in the Wall Following Lab that the sensor does not detect an obstacle ahead of it more often than it detects an obstacle that is not there. Therefore, when performing the Rising Edge routine, a false negative is more likely to happen which results in a enormous spike in the readings. The robot thinks that it reached a rising edge whereas the sensor simply reported a wrong value. The open space in the corner of the walls of the board may also result in a wrongly identified rising edge. All these elements combine to make the Rising Edge routine less accurate than the other one.

Was the final angle impacted by the initial ultrasonic angle?

Unless the ultrasonic angle is completely off, it does not significantly impact the final angle. Indeed, the final angle depends heavily on the light localization which is quite precise and covers for the inaccuracies of the ultrasonic angle. However, if the ultrasonic angle is really far from 0°, it might lead to inaccuracies in the final angle because the light localization did not start in the appropriate conditions.

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

There factors that we think contributed to the performance of each method is for one, the initial distance placed. If the initial distance from the wall is too small, the robot with a length greater than that value would crash to the wall when the robot rotates. This is the fundamental limitation of the localizer's performance.

Other factors can be the specific value of the radius of the wheel and the track as they would also influence the odometer reading and the actual angle turned, since these two variables will not completely match the robot's actual conditions; it creates an error. The friction of the two wheels and the accuracy of the motor rotation will also influence the performance.

In addition, the distance from the ultrasonic sensor to the wall and the tolerance of the distance that defines the falling and rising edge also influences the performance of the localization.

How do changing light conditions impact the light localization?

As mentioned in the software design section, we implemented a differential filter for detection of dark lines that does not depend on hard coded values. At the beginning of the light localization i.e. when the robot is located in the middle of the tile, a sample of the light reflection is taken and is stored as a reference value. Any subsequent sample taken by the light sensor during the localization is compared to reference and a significant difference between the sample and reference indicates that a darker line has just been crossed. Therefore, as no value is held constant as the reference throughout all the demos and since the reference value is measured at the beginning of each demo, changing light conditions should not impact the light localization.

Please note that we are assuming that most of the board bathes in a similar ambient light. However, if specific spots are being enlightened with different light sources, this might cause some disturbances. We can counter this problem by continuously updating the reference value with samples taken when we know (thanks to other indicators e.g. odometry, reference points on the board, etc.) that we are currently traveling over a reference point like the center of a tile.

Further improvements

Propose a software or hardware way to minimize errors in the ultrasonic sensor.

Ways to minimize errors in the ultrasonic sensor is to check whether the head angle is perfectly aligned when starting the light localization. To do so, the robot can be backed up against the wall with a flat tail, so that it is orthogonal to the wall and then move forward to perform the light localization. This approach can cancel out the error in terms of the angle.

Another way to minimize errors in the ultrasonic sensor would be to implement a filter similar to the one provided in the wall following lab. This filter ignores high values given by the ultrasonic sensor for a number of times. If these values are ignored more than a certain number of times (e.g. the constant *FILTER_OUT*), the filter will recognize that there is actually nothing ahead of it. This filter will address the problem of false negatives mentioned in the previous section.

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

A touch sensor can be used for localization. The touch sensor can be placed at the back of the robot. First, the robot can drive back and use the touch sensor to touch the wall, and then the robot can rotate 90 degrees and touch the other wall. After getting the accurate position of the wall, the robot can go forward an accurate distance and then rotate 90 degrees and go forward for a constant distance to go to the origin.

Another form of localization other than rising-edge or falling-edge could be rotating the robot for 360 degrees and recording the distance data. There will be specific distance values that will appear 4 times (some small tolerance would apply), i.e. twice for each wall. By analyzing the odometer angle reading, the mid points could be calculated and from that the robot would be able to calculate the 0-degree heading angle.

Finally, another form of localization could be to combine both rising-edge and falling-edge. If this method is implemented, the robot will turn until no wall is present. Once it sees a wall, falling edge would take that angle. Then, it would continue to rotate in the same direction until there is no wall and when it sees no wall, rising edge would take that angle. The two angles obtained can be used to calculate the angle the robot needs to rotate to face the 0-degree head angle.

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.

How and when the light localization could be used outside of the corner to correct Odometry errors is after the robot navigates to (0, 0), which is when every black line it sees becomes a reference to update the X or Y odometer data, with each line representing 30.48 cm from the previous black line. To perform the correction, two counters can be applied for X and Y, by multiplying 30.48 cm and updating the odometer value using the calculated result. That is implemented in the correction function in the lab 2.

In addition, the light sensor can be used to detect how long a time that the robot travels on a certain color, and by multiplying with the speed of the robot, the distance the robot traveling can be determined.