**ECSE 211: Design Principles and Methods**

Lab 4: Localization

Arianit Vavla : 260868601

Cheng Chen: 260775674

Group 20

October 2019

## Section 1: Design Evaluation

### Hardware

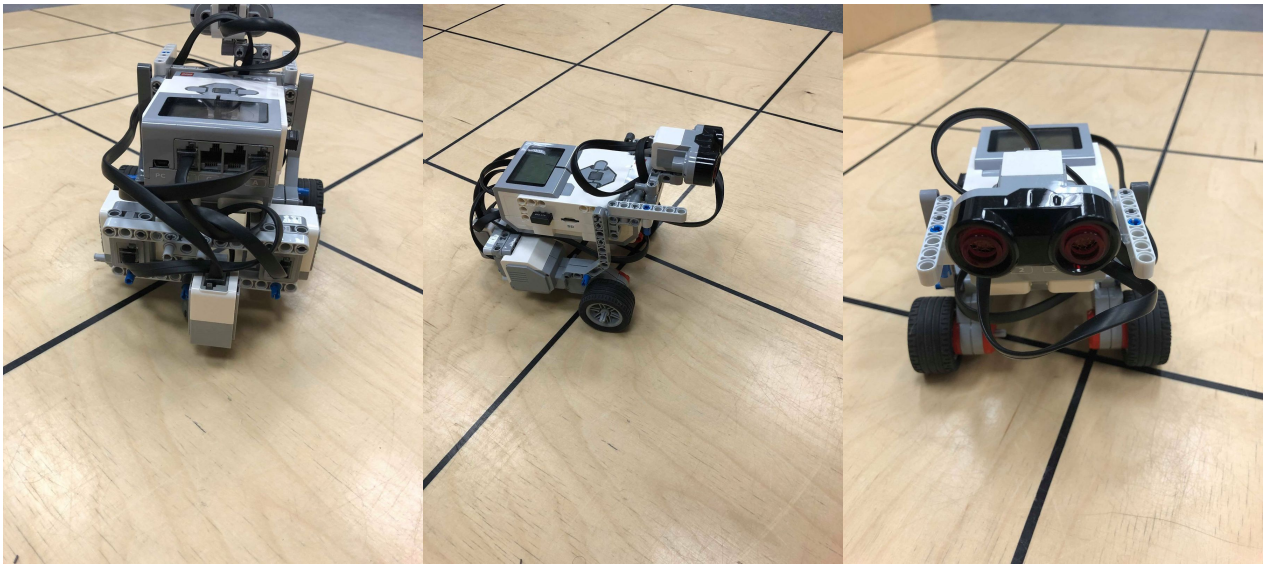

Figure 1. Robot

In this lab, we kept similar structures from previous labs. However, both light sensor and ultrasonic sensor are used this time. The light sensor is placed at the back of the robot because we want the robot to spin around to detect all the black lines around the origin. After the ultrasonic localization finishes, the light localizer will be activated.

At the first stage of the development of the robot, we placed the light sensor on the top of the robot and pointing downwards. After a few tests, we found that the light sensor missed a black line sometimes or detected a black line while in reality it is not on a line. Based on this observation, we decided to put the light sensor closer to the floor which lets the sensor collects better data since the ambient light effect on the sensor is minimized.

Also, during the tests, we found that the angle after ultrasonic localization is usually off by about 5 degrees. We thought that this was caused by the imprecise rotation of the wheels.

After some investigation, we found that the wheel touched the motor which results in the wheel not turning at a right angle because of the friction between the wheel and the motor. After we pull the wheel out by 4 cm, everything was working perfectly.

**Software**

In this lab, we used some of the classes from previous labs such as odometer, display and simple navigation. Also, we put all the constants in a separate class called resources just like what we did for the previous labs.

We implemented the ultrasonic localizer first. We used the formula explained in the tutorial. For the falling edge method, the robot will turn to the left until it detects that it is less than 33cm to the wall and record the angle it turns to the left. Then it turns to the right until it detects that it is less than 33cm to the wall and records the angle it turns to the right. The angle it needs to turn in order for it to go to angle 0 degree will be calculated by getting the average of the two angles and add the average to the angle shown by the odometer. For the first few tries, we found that the angle recorded is less than the angle turned because there is a delay between the time it detects that it is close to the wall and the time the robot stops. To solve this issue, we added an offset to reduce the turning angle. For the rising edge part, it did not work as well as the falling edge part. After running some tests, we managed to find a different offset that works for the rising edge. After these adjustments, the robot was able to turn to 0 degree within an error of 5 degrees.

After that, we implemented the light localizer. After the robot finishes the ultrasonic localization, the light localizer will be activated. The robot will turn to the origin. Once the sensor detects a line, it will go backwards for 12cm and spin around until 4 lines are detected. It

calculates the difference of the coordinates between the position where line 3 and line 1 detected and the difference of the coordinates between the position where line 4 and line 2 are detected. Then it uses the difference to calculate the average which gets us the current coordinates of the robot. Then it calls setXYT() method from the odometer class to set the correct coordinates of the robot. As we have the coordinates of the origin, we use the travelTo method in the simple navigation class to travel to the origin. For the first few tries, we found that the line detection is not very accurate because of the threshold we chose. We did some testing of the light sensor by printing the data collected by the sensor and compare the difference between the yellow wood and the black line. With some modifications of the threshold, we were able to detect all the lines accurately. The following flow charts describe the algorithm for the ultrasonic falling edge localization and light localization.

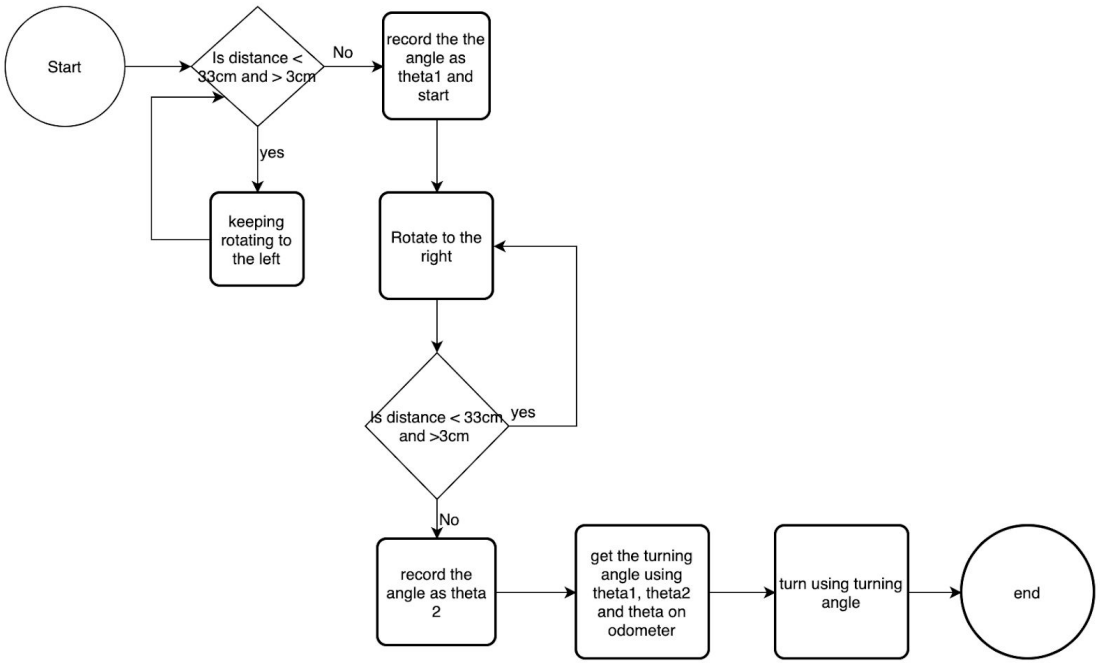


Figure 2. Flow chart for ultrasonic localization(falling edge)

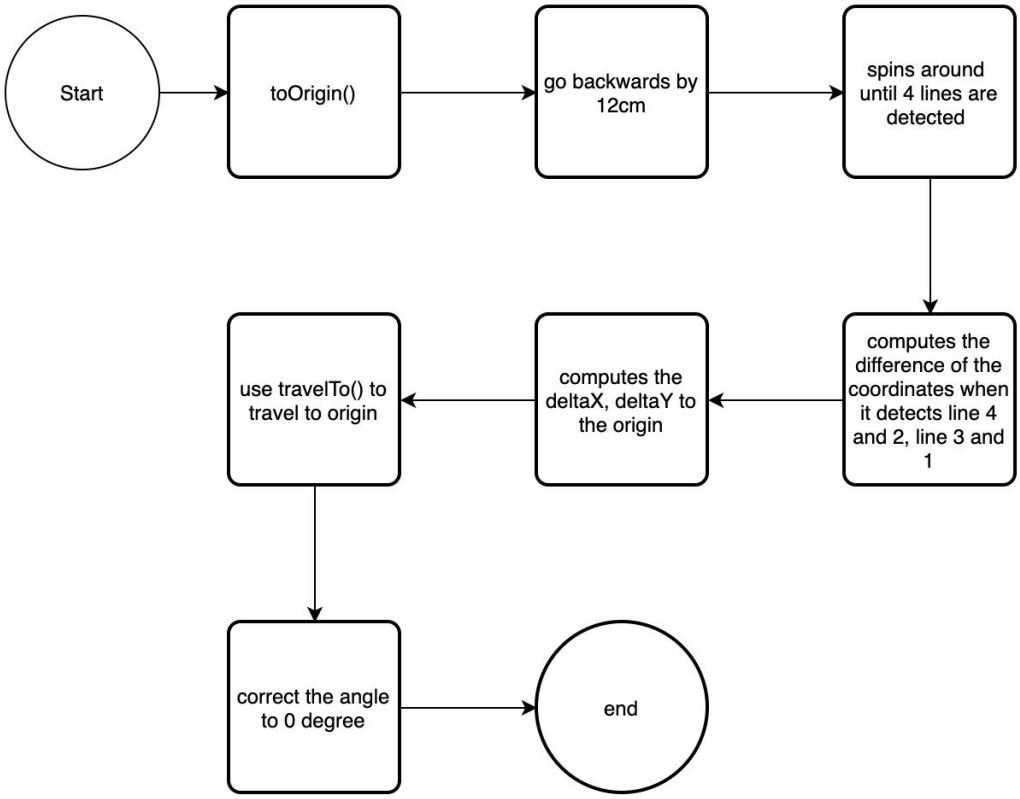


Figure 3. Flow chart for light localization

## Section 2: Test Data

Table 1: Rising Edge Test Data

| | Ultrasonic Angle Error (degrees) | Final Angle Error (degrees) | Euclidean Distance Error (cm) |
|---|---|---|---|
| 1 | 9 | 4 | 4.24 |
| 2 | 8 | 3 | 4.47 |
| 3 | 4 | 3 | 0.71 |
| 4 | 12 | 4 | 1 |
| 5 | 2 | 2 | 1.22 |
| 6 | 13 | 0 | 0 |
| 7 | 0 | 3 | 0.94 |
| 8 | 1 | 3 | 3 |
| 9 | 5 | 0 | 6 |
| 10 | 3 | 4 | 0.5 |

Table 2: Falling Edge Test Data

| | Ultrasonic Angle Error (degrees) | Final Angle Error (degrees) | Euclidean Distance Error (cm) |
|---|---|---|---|
| 1 | 2 | 0 | 5 |
| 2 | 1 | 3 | 0 |
| 3 | 3 | 0 | 1 |
| 4 | 8 | 3 | 4.4 |
| 5 | 6 | 0 | 0 |
| 6 | 5 | 0 | 2 |
| 7 | 6 | 0 | 0 |
| 8 | 0 | 3 | 0.5 |
| 9 | 3 | 1 | 0 |
| 10 | 8 | 5 | 5 |

**Section 3: Test Analysis**

To calculate the mean, we used the following formula:

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

Where n is the number of trials.

Example of calculation of the mean of the ultrasonic angle error for the falling edge:

$$\frac{\sum_{i=1}^{10} x_i}{10} = 4.2 \text{ degrees}$$

Table 3: The Mean Values of the Rising Edge Test

| Ultrasonic Angle Error Mean (degrees) | Final Angle Error Mean (degrees) | Euclidean Distance Error Mean (cm) |
|---|---|---|
| 5.7 | 2.6 | 2.20859371 |

Table 4: The Mean Values of the Falling Edge Test

| Ultrasonic Angle Error Mean (degrees) | Final Angle Error Mean (degrees) | Euclidean Distance Error Mean (cm) |
|---|---|---|
| 4.2 | 1.5 | 1.80 |

To calculate the standard deviation, we used the following formula:

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

Where n is the number of trials.

Example of calculation of the standard deviation of the ultrasonic angle error for the falling edge:

$$\sqrt{\frac{\sum_{i=1}^{10} (x_i - 4.2)^2}{10}} = 2.8 \text{ degrees}$$

Table 5: The Standard Deviation Values of the Rising Edge Test

| Ultrasonic Angle Error Standard Deviation (degrees) | Final Angle Error Standard Deviation (degrees) | Euclidean Distance Error Standard Deviation (cm) |
|---|---|---|
| 4.6 | 1.5 | 2.1 |

Table 6: The  Standard Deviation Values of the Falling Edge Test

| Ultrasonic Angle Error Standard Deviation (degrees) | Final Angle Error Standard Deviation (degrees) | Euclidean Distance Error Standard Deviation (cm) |
|---|---|---|
| 2.8 | 1.8 | 2.2 |

## Section 4: Observations and Conclusions

**Which of the two localization routines performed the best?**

In terms of the angle error, the best technique was the ultrasonic localizer with falling edge. However, this technique can still fail sometimes. When the gap between the robot and the wall is too large, the sensor can detect the wall twice when it starts to turn in another direction. For example, if we put the robot far away from the wall on the left. After the robot finishes turning left and starts to turn right, it detects that it is close to the wall which means the robots stop turning and records the angle of turning right. This issue can be easily eliminated by putting the sensor to sleep for 20ms.

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

The final angle was impacted slightly by the initial ultrasonic angle. As described earlier in the design evaluation section, the robot will travel to the origin using the odometer. Theoretically, the error of the final angle should not depend on the ultrasonic angle. However, based on our observations, if the robot is moving towards the origin at 45 degrees, the final angle was more accurate.

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

First of all, for the robot to move as described by the software, we have to make sure that the robot does not touch the wall while it's rotating or moving. Also, the floor should be clean enough for the light sensor to properly read the data so that it does not confuse other small details with the black lines. The friction of the floor can also contribute to the performance. If the floor does not have enough friction, there will be a lot of slips of the wheels which leads to inaccurate angles.

**How do changing light conditions impact the light localization?**

Changing light conditions will impact our light localization severely. We set threshold to detect the black lines. If the light conditions change, the light sensor will collect different data which does not comply with the current threshold. Thus, our code can be improved by using ratio between the previous collected data and the current data collected to detect the black lines. In other words, when the light sensor reads a very different color, it tells us that a balck line is detected.

## Section 5: Further Improvements

An improvement in the software we can make for the ultrasonic sensor localization is for the software to get the minimal angle to the origin because we made it so it would turn to the left to go back to zero and did not implement the minimal angle in there. For example, if it was at 300 degrees, it would still turn to the left, and with the error of the odometer, there would be an angle that can be reduced if we implement that method in the ultrasonic sensor localization.

We can use a combination of falling edge and rising edge. So we would know that the robot would start turning to the left at first, so we would click based on that. For example, if the robot is facing away from the wall, we can click first detection falling edge and keep turning left and then detect the rising edge, so there would be a swipe of the robot turning only once, not changing direction of turn. This would be saving a small amount of time because the robot is required to stop only once when it has to go back to 0 degrees.

Knowing that the middle has the same navigation from every corner of the field, we could use the odometer to navigate and then when the line leading to the middle square is crossed, it would follow the same instructions as localizing in the origin and detecting the four lines and adjust itself in the corner of the middle square at 0 degrees, correcting the odometer, and then proceed with the rest of the navigation or task.