ECSE 211

Lab 4: Localization

Group 21
Theodore Janson (260868223)
Marwan Khan (260762251)

February 14th, 2019

1. Design Evaluation

a. Hardware

For this lab, we kept the same update design as from lab 3. The wide design allows for more stability, which would reduce the error due to imbalances in the robot's weight distribution. Large EV3 motors, placed on either side of the brick, are used as main anchors for stability. The robot uses front wheel drive, so a marble support was placed at the back of the robot to support the rear half. The ultrasonic sensor was placed at the front of the robot, and the light sensor was placed at the back. We placed the light sensor at the back in order to maintain the overall for-aft balance of the robot. This offset was corrected by a constant in the code. To keep all the wires from taking up to much room, which could cause issue when avoiding obstacles, we used a clamp to reduce wire volume and attached this to the side of the robot. After some testing, we rotated our ultrasonic sensor vertically to get a more accurate 0° heading. Because the US sensor sends a sound wave out in a triangle of 30° and returns the first echo signal, there could be some error if the sensor detects wall at an angle greater or less than when the sensor is at the supposed minimum or maximum distance. A sensor placed vertically sends out a wave at vertical triangle of 30°, which reduces the potential error. Moreover, thanks to experience from lab 2, we knew that it was important to place the light sensor as low to the ground as possible.

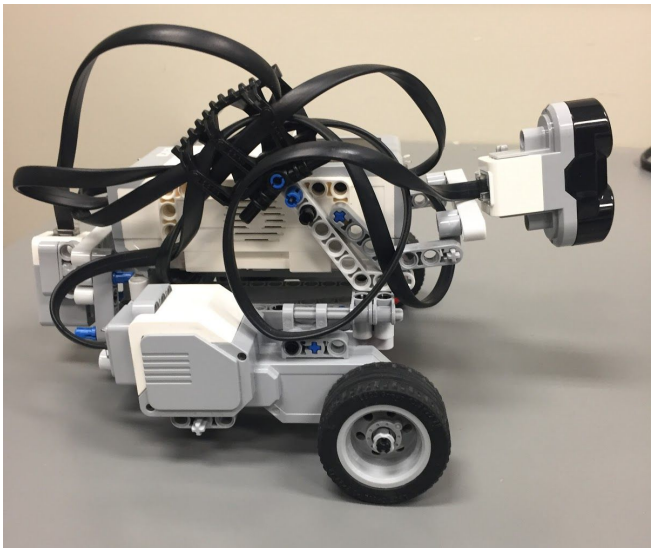


Figure 1: Hardware design

b. Software

The initial entry point to the code is the LocalizationLab class which initializes the instances for the motors, the light sensor, the ports, as well as the main constants (wheel radius and track) and variables used throughout the project. The main method initializes and starts the odometer, display, ultrasonic localizer and after a button has been pressed, the light localizer threads. The two classes written for this lab are the LightLocalizer and UltrasonicLocalizer classes. The Ultrasonic localizer class is used to find the robots heading during a falling edge or rising edge routine, identify the 0° heading and turn towards it. The run method of the US localizer class calls either the falling or rising edge method based on the user's choice. In the falling edge method, the robot is called to rotate until the US sensor no longer detects any wall. It keeps rotating until it detects the wall below a minimum distance threshold. It then turns the other way, and repeats the process on the other side. After some testing, we decided to record two angles on each side and use the average of the two to calculate 0° heading. After testing, we noticed that the robot would sometimes detect the wall again after it rotates to the the opposing edge. To correct this, we called a delay for 3 seconds. After the average of the two falling edge angles are calculated for each side, we call on a method which makes the robot turn to the accessed 0° heading. The method calls the turnTo method, as the robot turns to the angle computed from its current heading and the necessary change in heading.

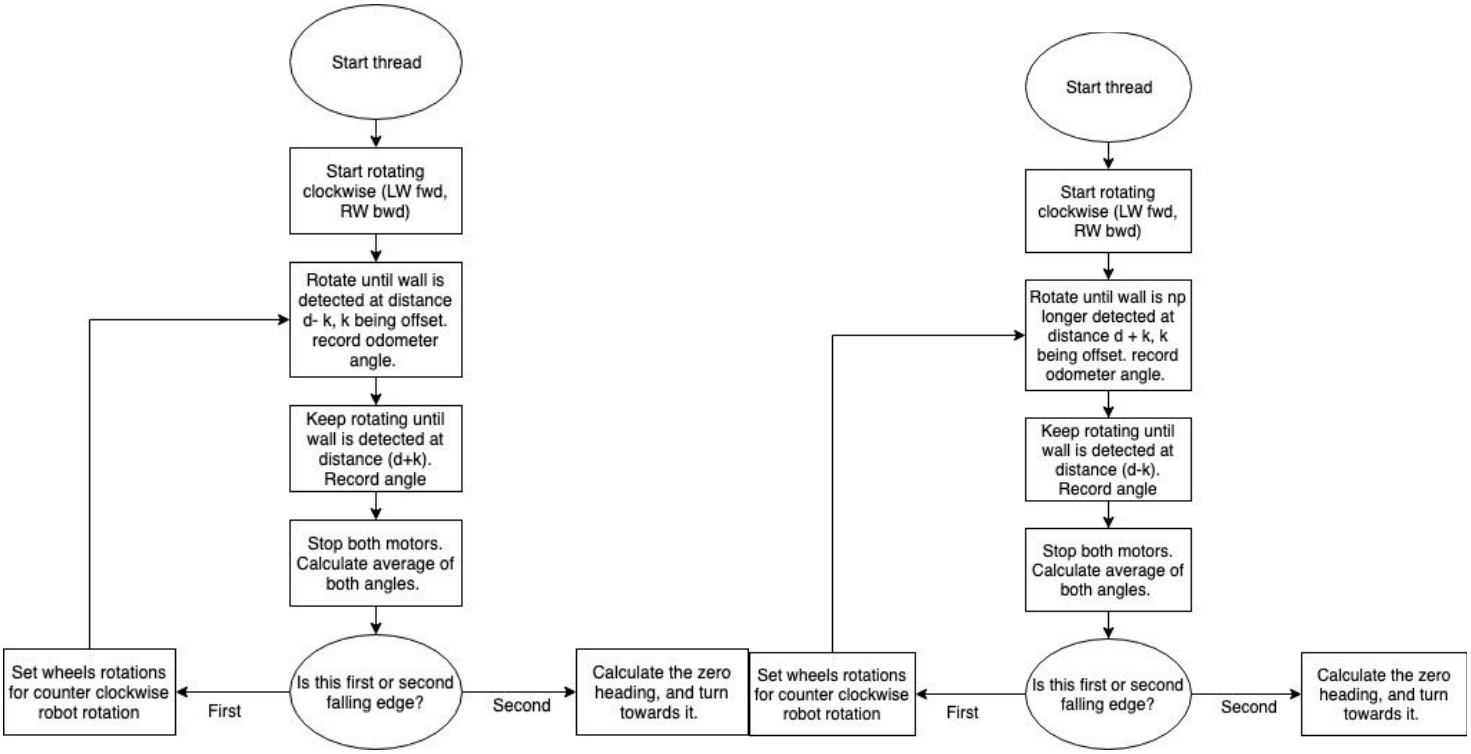
Angle to turn to (0° heading) = (WallAngle2 - current heading) - 45 + (360 - (WallAngle1 - WallAngle2)/2).

This change is calculated by taking the average of the two falling edge angles and adding it to the difference between the current heading and second falling edge angle. This angle is then corrected by subtracting 45°, so the robot heads at 0° and not towards the origin. The rising edge method follows the same flow, only it initially turns away from the wall until a maximum distance is detected, at which the angle is recorded. This routine was less reliable than the falling edge one because of the detection of false negatives when the robot was facing the wall.

The LightLocalizer class is used after this heading has been determined, and is used to identify the x-axis, y-axis and by extent, the origin. The light localizer contains the turnto method and some methods for calculating motor rotations as a function of distance or angle needed to travel to or by. These methods are from previous labs. In this class, the lightlocalize method is

used to detect the x and y axes independently. The robot moves forward at 0° heading until it detects a black line (negative x axis). The odometer's Y value is recorded. It keep moving forward (delay of 0.5 seconds) so that the black line is not detected twice. It then moves in reverse, detects the black line again, and keep moving back (delay for 4 seconds) so that the entire robot has cleared the axis. The robot rotates by 90° to repeat the process with the negative y axis. The recorded difference between the robot's current position and the X and Y odometer values are then used to find the distance to the origin using pythagoras. The robot turns by 45°, drives this distance, and turns by another 45°.

Most of the workflow involved in the software portion concerned adjusting certain constants such as track and the distance between the center of the robot and the light sensor. It also involved adding corrector constants to reduce recurring errors such as the over rotation of the robot when turning to its 0° heading. We also had to change the threshold for light detection a few times, as it often wasn't constant based on where the robot was placed on the testing board. An attempt to correct this involved filling an array with light samples. A black line would then be detected whenever the difference between two elements was greater than a threshold. This solution was implemented too soon before the demo, and wasn't used as there were issues. *Figure 2 & 3: Falling edge and rising edge routines.*



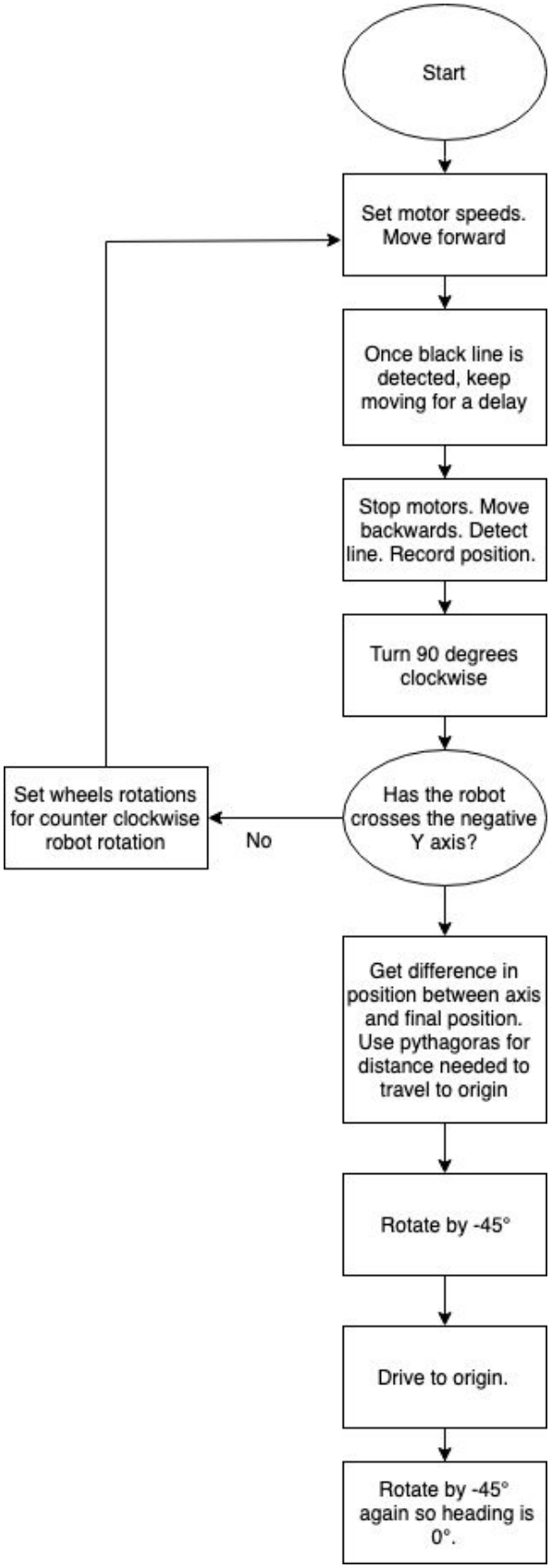


Figure 4: Light Localization

2. Observations and Conclusions

Rising Edge:

Trial Number (N)	Ultrasonic Angle Error - ∈ (°)	Euclidean Distance Error - ∈ (cm)	Final Angle Error - ∈ (°)
1	5	2.110	7
2	4	2.000	5
3	2	1.860	3
4	4	1.921	4
5	3	1.803	3
6	5	1.702	5
7	4	1.825	4
8	4	1.921	4
9	2	1.655	3
10	4	2.025	7

Falling Edge:

Trial Number (N)	Ultrasonic Angle Error - ∈ (°)	Euclidean Distance Error - ∈ (cm)	Final Angle Error - ∈ (°)
1	2	1.920	1
2	1	1.700	1
3	2	1.860	3
4	2	1.844	2
5	2	2.000	3
6	3	1.942	4
7	2	1.921	3
8	2	1.970	2
9	2	2.000	3
10	1	1.838	1

3. Test analysis

Rising edge:

	Ultrasonic Angle Error (°)	Euclidean Distance Error (cm)	Final Angle Error (°)
Mean - $\bar{x}$	3.7	1.8823	4.5
Standard Deviation - σ	1.059349905	0.142572438	1.509230856

Falling Edge:

	Ultrasonic Angle Error (°)	Euclidean Distance Error (cm)	Final Angle Error (°)
Mean - $\bar{x}$	1.9	1.8996	2.5
Standard Deviation - σ	0.567646212	0.091968835	0.971825316

Sample Calculations:

Euclidean Error:

$$\epsilon = \sqrt{(0 - X_F)^2 + (0 - Y_F)^2}$$
$$\epsilon_1 = \sqrt{(0 - 1.7)^2 + (0 - 1.1)^2}$$

Euclidean Mean:

$$\bar{x} = \frac{\sum_1^N \epsilon_N}{N}$$
$$\bar{x} = \frac{2.025 + 1.655 + 1.921 + 1.825 + 1.703 + 1.803 + 1.921 + 1.860 + 2.000 + 2.110}{10}$$

Euclidean Standard Deviation:

$$\sigma = \sqrt{\frac{\sum_1^N (\epsilon_N - \bar{x})^2}{N - 1}}$$

$$\sigma = \sqrt{\frac{(2.025 - 1.8823)^2 + (1.655 - 1.8823)^2 + (1.921 - 1.8823)^2 + (1.825 - 1.8823)^2 + (1.703 - 1.8823)^2 + (1.803 - 1.8823)^2 + (1.921 - 1.8823)^2 + (1.860 - 1.8823)^2 + (2.000 - 1.8823)^2 + (2.110 - 1.8823)^2}{9}}$$

4. Observations

1 - Which of the two localization routines performed the best?

Falling edge performed better than the rising edge as evidenced by its slightly lower Euclidean error. This is because the rising edge routine can be unreliable due to detection of false negatives, whereas the detection of false positives during the falling edge is a lot more uncommon because the sensor doesn't detect distances above 255 cm. In the falling edge routine, the final angle and ultrasonic angle errors are considerably less than in the rising edge routine (mean of 2°).

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

The final angle was impacted by the placement of the robot and initial ultrasonic angle. This is because the ultrasonic angle accumulates and yields a greater error in the final angle. This is because the robot assumes it is on a perfect 0° heading after the ultrasonic localization, so when the robot moves forwards and backwards during light localization, it is sometimes not actually on a perfect heading.

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

The light sensor's performance was improved by placing the sensor really close to the ground, without actually touching it. It was also dependent on the final angle after the ultrasonic localization method. If the robot is on a perfect 0° after this, then the performance of light localization is very reliable, unless the light localizer fails to identify black lines correctly, which generally isn't the case. Other errors are only due to wheel slippage and errors in motor timing, which are minimal.

4 - How do changing light conditions impact the light localization?

If there are different light sources in the room, dim conditions, random ambient lighting such as shadows, the light sensor might not detect a black line or might detect black lines where there are none, depending the error in the colour sensor threshold. If the robot detects a black line where there is none, it will reverse into the wall. If it fails to detect a black line, it will never reverse back into place. This can be corrected with a differential threshold.

5. Further improvements

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

If the speed of the wheels were reduced as the robot approached an angle corresponding to the minimum/maximum distance of the falling/rising edge routine, then the data samples would be more continuous, which would allow for both more values to be read and a more precise 0° to be calculated. Moving slower would also reduce errors due to tire slip.

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

One could use the ultrasonic sensor to turn the robot away from the wall so that it is no longer facing the wall. Once the sensor detects a distance greater than a threshold, a delay can be called so that the robot keeps rotating for a short amount of time away from the wall. Then, using two light sensors, one placed on each of its sides, the robot would move forward until one of the sensor detected a black line. The robot would then stop. The motor on the side corresponding to the sensor that didn't detect a line would be called on to move forward alone until that sensor detects a black line. The robot would then be either on the x or y axis, at a 90° or 0° heading. One could then rotate the robot by 90°. If the ultrasonic sensor detected wall, the robot would be on the y axis.

3. *Discuss how and when light localization could be used outside of the corner to correct*

The above method of using two light sensors and correcting the headings until the second sensor detects a black line too could be implemented anywhere on the board in

order to keep the robot on certain specific headings such as 0°, 90°, 180°, 270°. One could then add or subtract angles from its corrected heading to go anywhere on the board.