ECSE 211 Winter 2020

Lab 4: Navigation

Group 54
Joanna Koo (260865846)
Muhammad Mustafa Javed (260808710)
Due February 13th, 2020

Table of Contents

Section 1: Design Evaluation

Workflow:

The objective of the lab was to localize the robot with respect to the grid system, then navigate on a specified map. In order to correct its orientation to 0°, the robot used the ultrasonic sensor, and moved to (1, 1) on the grid using either the ultrasonic sensor or the light sensor. The only modification required was the driver class (renamed to navigation), which controlled the motion of the robot.

Hardware Design:

The hardware did not require any major modifications. A strengthening rod was added to the front to make the structure more rigid. The final robot design can be found in *Figures 1, 2,* and *3*.

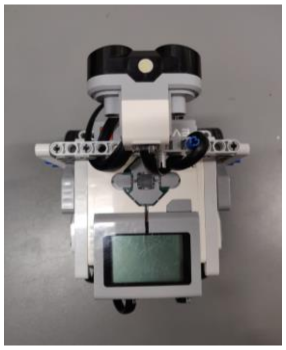
Figure 1. Bird's eye view of the robot.

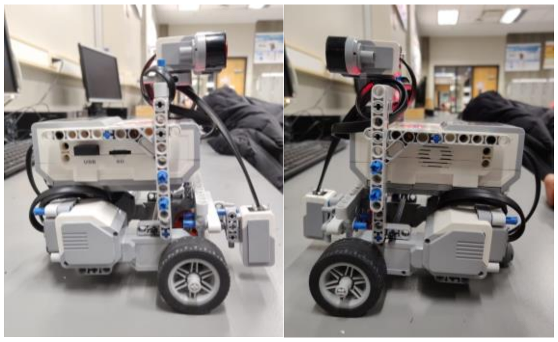
Figure 2. Left and right side profiles of the robot.

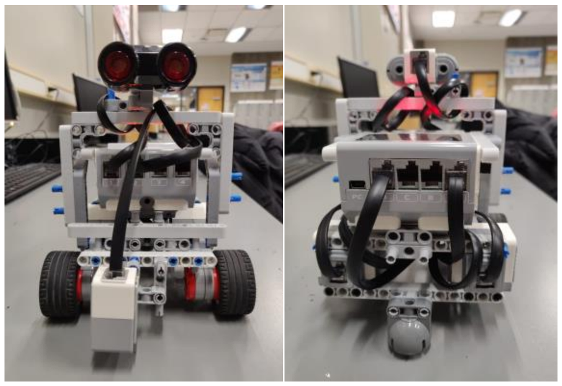
Figure 3. Front and back profiles of the robot

Software Design:

The Driver class was altered to include a travelTo(x, y) method and a turnTo(theta) method. The travelTo(x, y) takes the grid position as parameters. The algorithm would calculate the distance it has to travel and the bearing it has to turn based on the current position and the final position. The equations used are shown in *Equations 1* and *2*. The robot would then turn to that bearing using the minimal angle, then move that distance. The flowchart illustrating the software functionality is shown in *Figure 4*.

Equation 1.

$$distance = \sqrt{dx^2 + dy^2}$$

Equation 2.

$$\theta = arctan\frac{dx}{dy}$$

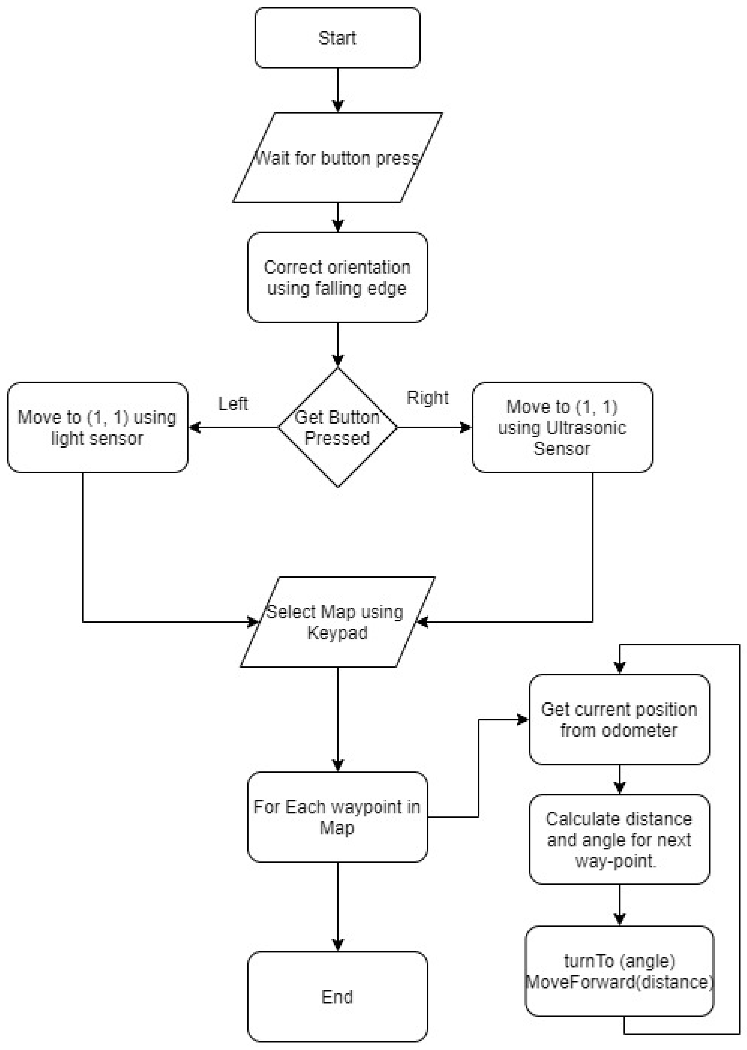
Start
Wait for button press
Correct orientation using falling edge
Get Button Pressed
Left
Right
Move to (1, 1) using light sensor
Move to (1, 1) using Ultrasonic Sensor
Select Map using Keypad
For Each waypoint in Map
Get current position from odometer
Calculate distance and angle for next way-point.
turnTo (angle) MoveForward(distance)
End

Figure 4. Flowchart for software functionality.

Section 2: Test Data

8 independent trials were performed for the navigation test and are recorded in *Table 1*. The robot was programmed to travel to waypoints (3,2), (2,2), (2,3) and (3,1). The destination waypoint, (3,1) is equivalent to X_D=91.44 cm and Y_D= 30.48 cm.

Trial	X_F (cm)	Y_F (cm)
1	90.50	31.00
2	91.44	32.50
3	89.00	31.50
4	90.50	32.75
5	89.50	30.48
6	92.25	31.45
7	90.00	31.75
8	89.90	32.90

Table 1. 8 independent trials for the navigation test.

Section 3: Test Analysis

The Euclidean error distance ϵ between (X_F, Y_F) and (X_D, Y_D) was computed and recorded in *Table 2*. *Equation 3* was used for calculations.

Equation 3.

$$\epsilon = \sqrt{(X_D - X_F)^2 + (Y_D - Y_F)^2}$$

Sample calculations are shown below.

$$\epsilon = \sqrt{(91.44 - 90.50)^2 + (30.48 - 31.00)^2} = 1.07$$

Trial	Euclidean error distance ϵ (cm)
1	1.07
2	2.02
3	2.64
4	2.46
5	1.94
6	1.26
7	1.92
8	2.87

Table 2. The Euclidean error distance for each trial.

The mean and standard deviation for the Euclidean distance error is computed in *Table 3*. *Equations 4* and *5* were used to calculate the mean and the standard deviation, respectively.

Equation 4.

$$\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$$

Equation 5.

$$\sigma_a = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})^2}$$

Sample calculations are shown below.

$$\bar{\epsilon} = \frac{1}{8}(1.07 + 2.02 + 2.64 + 2.46 + 1.94 + 1.26 + 1.92 + 2.87) = 2.02$$

$$\sigma_{\epsilon} = \sqrt{\frac{1}{10-1}(1.07-2.02)^2+(2.02-2.02)^2+(2.64-2.02)^2+\ \dots + (2.87-2.02)^2} = 0.63$$

	Euclidean error ϵ (cm)
Mean	2.02
Standard deviation	0.63

Table 3. The mean and Standard deviation for the error.

Section 4: Observations and Conclusions

Are the errors you observed due to the odometer or navigator? What are the main sources?

We observed that the errors, if any, were because of the odometer. The travelTo() method in the Navigation class depended on the coordinate values and bearing provided by the odometer. Therefore, any error in the odometer values induced an error in the navigation. The odometer values depended on the wheelbase and the wheel radius measurements, which are very sensitive; thus, even a small alteration caused a large variation in the travelled distance and the turned angle.

Another main source of error was the drifting of the tires. The odometer did not account for any slip of tires, which lead to incorrect values due to excess dust or scratches on the floor.

The battery level of the brick was also an important factor. The accuracy of sensors and motors depended on the battery level. If the battery level dropped below 7.2V, we noticed that the error increases as the ultrasonic sensor was reading an incorrect distance and the readings changed erratically. For testing and demonstration, it was ensured that the battery level was above 7.5V.

How accurately does the navigation controller move the robot to its destination?

We found the navigation controller to be very accurate if the initial localization was successful. When testing the Navigation (with the robot initially placed at (1, 1) position), the robot reached the final point with high accuracy and a mean error of 0.5 cm. If, however, the localization failed for any reason, it impacted the navigation and caused a larger error in the final position.

At which waypoints did you decide to localize and why? What are the advantages and disadvantages to localizing at every waypoint?

For the smaller maps, we did not implement localization at any waypoint other than the initial localization at (1, 1) as the final error was relatively small, making the light localization unnecessary. However, for future labs, we plan on implementing the light localization at every third waypoint as the error accumulates over the distance moved. This will ensure that the maximum error is removed and that the error does not become large enough that the recovery is impossible.

Localizing at every waypoint will minimize the error as much as possible by removing any error that may occur during any two waypoints. However, localizing at every waypoint will add unnecessary time and complication to the robot motion. This will also cause added processing time and load on the processor.

Section 5: Further Improvements

The following hardware and software solutions can be taken to reduce the errors.

Hardware solution

Using a color sensor in conjunction with an odometer correction system would reduce the errors. The color sensor would detect the black lines on the grid and would provide more accurate information on the surrounding of the robot. The robot would adjust accordingly and navigate through the points more smoothly and accurately.

Another hardware solution is to use rubber tracks or a four-wheel drive system, instead of a two-wheel drive. This will help to improve the grip over the board and increase the chances of consistent performance.

Software solution

Implementing the odometry correction would be an effective software solution to reduce the errors. Whenever the robot travels along the X or Y direction, or crosses the black line, the odometry correction that uses the color sensor could update the odometer readings accordingly. With the odometer readings being more accurate, the navigation will be more precise as the navigator's commands derive from the odometer readings.