Yuhang Zhang [260787441] Group 7

Mohamed Youssef Bouaouina [260765511]

Lab 3 Report - Navigation

ECSE 211: Design Principles and Methods

Design Evaluation

As it can be seen in figure 1, our workflow consisted 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 stage

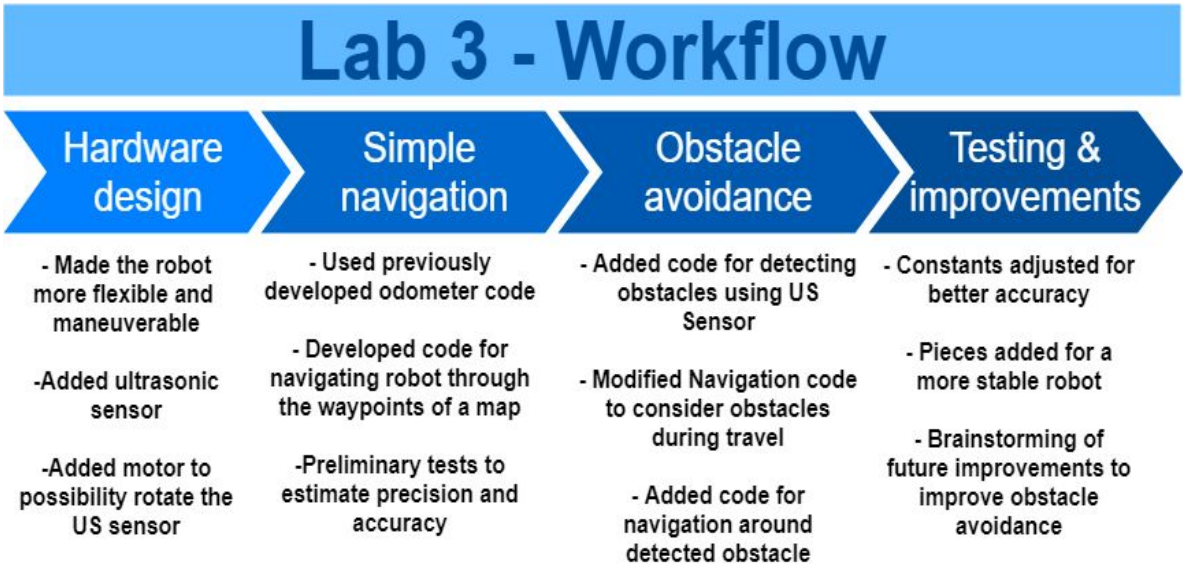

Figure 1: workflow during lab 3

a) Setting up the hardware:

Alongside navigating through the waypoints of a map, the robot in this lab was also required to avoid obstacles (i.e. blocks) placed along its pathway. As a result, an ultrasonic sensor was mounted at the front to be able to detect obstacles ahead of the robot. We also designed a structure that holds a motor at the front of the robot to be able to rotate the ultrasonic sensor. We initially built this structure with the intention of using it to sweep a larger area ahead of the robot, hence detecting the obstacles on its pathway.

Based on observations made during the previous labs, the following modifications were also made to improve the hardware design of the robot. First of all, we noticed that the robot's wheels were too close to the motors and often resulted in friction between the tires and the walls of the motors. Previously, we used to manually separate the wheels from the motors which resulted in inaccurate track values. Therefore, a piece was added to mechanically keep a distance between the wheels and the motors and prevent friction with the tires.

Secondly, in contrast with Lab 2, we wanted to make the robot as maneuverable and agile as possible since this lab involved obstacle avoidance. However, we also aimed for a robust robot that would not slip easily on the surface of the boards as this would incur errors to the odometry system. Therefore, we set the distance between the wheels to be 11.7 cm. We thus achieved a reasonable trade-off between weight and maneuverability.

After our first demo, we acted on a TA's advice by mounting an additional piece that connected the driving motors and prevented the wheels' axles from bending under the weight of the brick and the sensor.

Finally, we chose not to mount a light sensor as odometry correction was not required for this lab and timing constraints did not allow us to venture into this initiative (c.f. Improvements section).

b) Software design:

For this lab, the main task was to design a navigation program to guide the robot through the waypoints of the user chosen map.

First, we focused primarily on a simple navigation that did not consider the possibility of obstacles in the pathway of the robot. We implemented this by running a Navigation thread that is started in the main method of the Lab3 class. This thread successively goes through the waypoitns of the chosen map and calls the travelTo() method on each of them.

This travelTo() method reads the current position of the robot and determines the distance to cover to reach the destination waypoint. In order to access the current position of the robot, we chose to use the odometer code we developed in lab 2. As required by the lab instructions, the travelTo() method calls the turnTo() method after it determines the orientation the robot needs to be in to reach its destination point. When implementing the turnTo() method, we use helper methods we wrote in order to ensure that the minimal angle is taken to turn to the correct heading. For more information about the implementation of our helper methods, please refer to the Javadoc comments provided in the uploaded code.

After preliminary tests, we were satisfied with the accuracy of the simple navigation and decided to add some code to make the robot avoid obstacles in its pathway. This was done by modifying the travelTo() method so it polls the ultrasonic sensor for the distance detected. This distance is passed to a method from the Avoidance class that is in charge of handling the decision making regarding the avoidance of obstacles.

Unfortunately, due to time constraints, we "hard coded" to a certain extent the path to be taken when the robot comes within a certain distance of an obstacle. Please refer to the improvements section for more details about this and the initiatives we were planning to venture in. However, this hard coded path revealed itself quite successful in avoiding obstacles during the multiple tests we ran.

Test Data

Trial	*Destination Waypoint (cm)* (X_D , Y_D)	*Final position of the robot (cm)* (X_F , Y_F)	*Euclidean Error distance* $\in$ *(cm)*
1	(60.96 , 0.00)	(61.55 , -1.25)	1.38
2	(60.96 , 0.00)	(58.12 , 2.25)	3.62
3	(60.96 , 0.00)	(62.09 , 1.05)	1.54
4	(60.96 , 0.00)	(60.25 , -2.21)	2.32
5	(60.96 , 0.00)	(63.10 , 0.00)	2.14
6	(60.96 , 0.00)	(61.90 , 0.15)	0.95
7	(60.96 , 0.00)	(60.25 , 0.95)	0.99
8	(60.96 , 0.00)	(60.60 , 0.00)	0.36
9	(60.96 , 0.00)	(60.20 , 0.50)	0.91
10	(60.96 , 0.00)	(60.96 , 0.00)	0.21

Figure 2: test data table

Test Analysis

We use the following formulae to measure the required means and standard deviations:

Mean $\bar{x}$:

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

Where,
x_i is the i^{th} sample point
and n holds the number of sample points

Standard deviation σ:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

Where,
x_i is the i^{th} sample point,
$\bar{x}$ is the mean,
and n holds the number of sample points.

$$\text{Mean} = \sum \frac{x_n}{n}$$
$$= \frac{1.38+3.62+1.54+2.32+2.14+0.95+0.99+0.36+0.91+0.21}{10}$$
$$= 1.44$$

$$\text{Standard Deviation} = \sqrt{\frac{1}{N-1}\sum_1^N (x_i - \bar{X})^2} =$$
$$\sqrt{\frac{(1.38-1.44)^2+(3.62-1.44)^2+(1.54-1.44)^2+(2.32-1.44)^2+(2.14-1.44)^2+(0.95-1.44)^2+(0.99-1.44)^2+(0.36-1.44)^2+(0.91-1.44)^2+(0.21-1.44)^2}{10}}$$
$$= 1.02$$

Mean (cm)	*Standard deviation*
1.44	1.02

Observations and conclusions

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

We believe that the observed errors came largely from the odometer. The navigator calculates the required rotation angle by trigonometry and the robot's path is constrained by the value of ***TILE_SIZE***. Therefore, the values returned by the navigator are quite accurate and rarely caused the robot to deviate from its path.

On the other hand, we thought the odometer could be the major factor contributing to potential errors for the following reasons: odometer data uses the wheel radius and the value of the track, which are set manually, and such an issue could have potential errors on the calculated position by the odometer.

Besides the possible inaccuracy of the odometer, other sources of error could be the wheel slippage, low battery, difference of friction on the two wheels, and the unbalanced weight distribution of the robot.

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

Since the navigation class is responsible for travelling along the waypoints, and based on the test data, we can find that after slightly adjusting the navigation parameters, the Euclidean error of navigation can be is usually kept under 2 cm, or even less than 1 cm occasionally.

The navigation implements the travelTo (double x, double y) and turnTo(double theta) methods which calculate the distance and angle required to reach the next waypoint. These calculations are based on the current position and the angle of the robot and is updated each time it reaches a waypoint. As a result, the robot moves accurately to the final waypoint if the

odometer is correct. With no deceleration method called when stopping, the robot stops instantly when it reaches the destination. However, because the driving speed is low, the acceleration is not large, and a sudden stop will not cause any issues.

How quickly does it settle (i.e. stop oscillating) on its destination?

Since the travelling and rotating speed of the robot are relatively slow, the acceleration during the stop and start movement is not large. Therefore, only slight oscillations can be observed at each waypoint.

How would increasing the speed of the robot affect the accuracy of your navigation?

Increasing the speed of the robot means increasing the acceleration of the wheels. However, as mentioned in the previous lab, a large acceleration is associated with an increase in slipping on the board. This incurs errors to the odometer system. Therefore, the slower the robot moves, the less likely are the wheels to slip on the board, making the final reading of the odometer much more truthful to the robot's actual location.

Further improvements

As in any project in industry, this lab came with time, budget, and skills constraints. Therefore, we dedicate this section to improvements and features that we considered but, due to the aforementioned constraints, did not implement.

First of all, as mentioned in the software design section, we "hard coded" to a certain extent the obstacle avoidance section of the program. Alternatively, we suggest to use the knowledge acquired during Lab 1 to implement a Bang Bang or a proportional controller that uses data polled by the ultrasonic sensor to maintain a certain distance with the obstacle while going around it. This not only ensures a safer avoidance of the obstacle but also ensures that multiple obstacles can be detected and avoided accordingly.

For an even better performance of the previous improvement, we also considered (as mentioned in the hardware section) making the ultrasonic sensor rotate and sweep the area ahead of the robot to enhance obstacle detection.

Finally, we contemplated the possibility of using a light sensor as in Lab 2 in order to implement the odometry correction logic and make our odometry system more accurate.