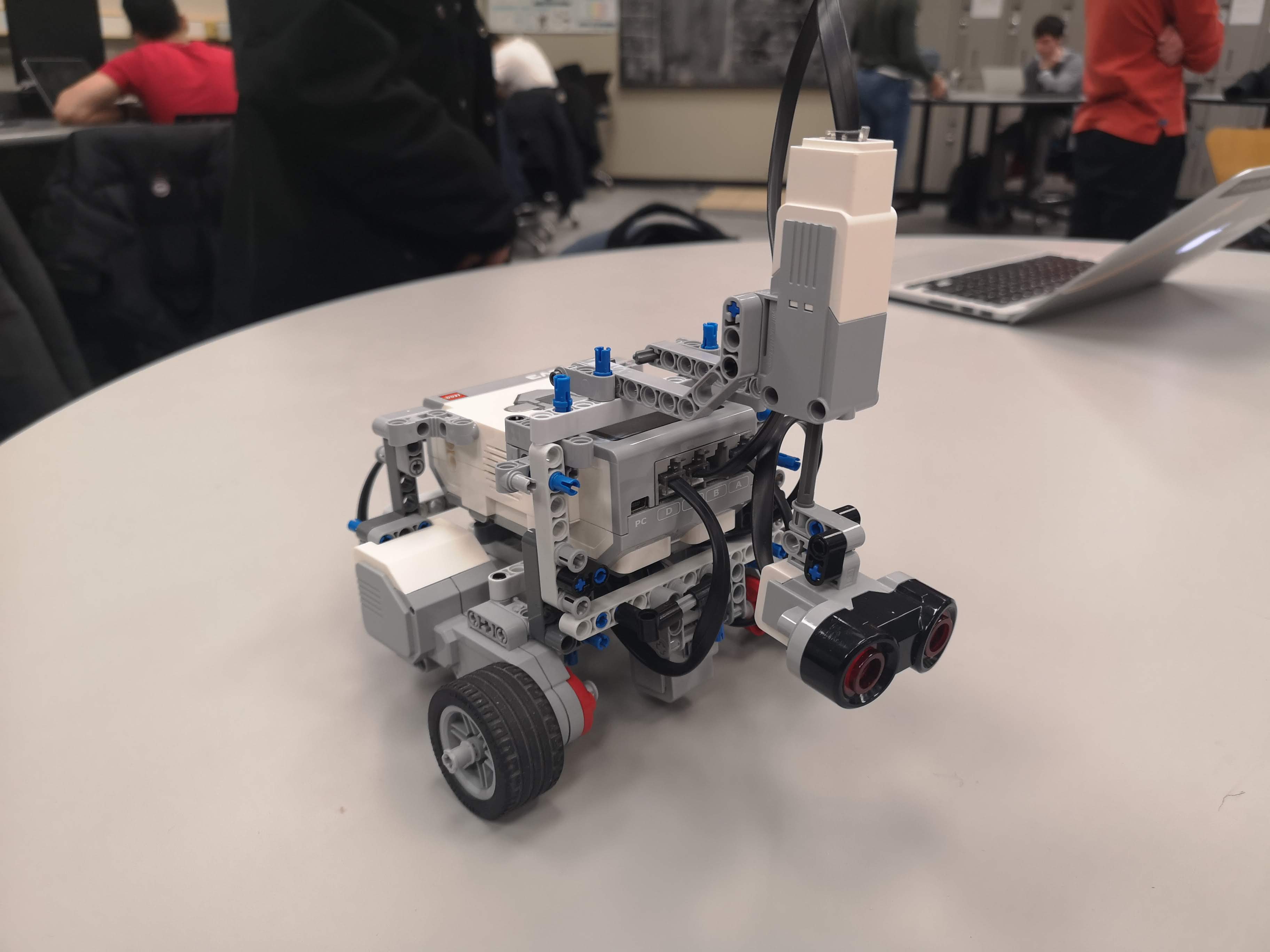
## Overview of Lab #3: Navigation and Obstacle Avoidance

*ECSE 211: Design Principles and Methods*

# SECTION 1: DESIGN EVALUATION

**Figure 1: Hardware design** Our robot is equipped with two motors attached to wheels as well as an ultrasonic sensor. The ultrasonic sensor also has a motor attached to it, which permits the sensor to perform a continuous sweeping motion to detect obstacles in front of the robot. As shown in figure (1), the color sensor is attached to the bottom of the robot due to it being use in earlier versions of the robot’s software design. After testing, it was determined that the color sensor did not contribute to the success of the lab, so although the sensor is still a part of the hardware, it is not used. Additional supportive structure was also added to the hardware of the robot following testing, in order to increase stability.

The software component of the design functions with the use of threads. Many classes run at the same time in order for the robot to simultaneously navigate on the tiles, display its location, detect the obstacles and avoid them.

The odometer class was not changed from the last lab as the results were sufficiently accurate. The obstacle detection is done using the ultrasonic sensor moving in a back-and-forth motion in order to detect the upcoming obstacles regardless of their angle in relation to the robot. Once the obstacle is detected, the robot avoids collision by a hard-coded series of movement. the robot simply turns to avoid the obstacle using pre-set values of the size of the block. The robot turns to either left or right to stay on the platform, depending on which quadrant it is situated on based on the odometer reading. Once the avoidance is completed, the robot checks its latest position using the odometer and continues its navigation to the next waypoint. Required distance and turning angle are calculated using basic trigonometry (Pythagorean theorem, and inverse tangent).

**Figure 2: Software design**

# SECTION 2: TEST DATA

**Table 1: Navigation test**

The first table presents the final distance of the robot from its destination waypoint, which is at the Cartesian point (2, 0). Since the width of a tile is of 30.49 cm, this means that from the origin the expected destination waypoint is of about (30.49 , 60.98) cm, which is represented by the odometer distance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Measured distance (cm) | | Odometer distance (cm) | |
| Trial | Xd | Yd | Xf | Yf |
| 1 | 61.1 | 30.6 | 61.27 | 30.48 |
| 2 | 61.0 | 31.1 | 61.31 | 30.51 |
| 3 | 60.8 | 30.8 | 61.19 | 30.49 |
| 4 | 60.2 | 30.2 | 61.17 | 30.50 |
| 5 | 60.7 | 30.1 | 61.21 | 30.53 |
| 6 | 61.2 | 30.3 | 61.13 | 30.44 |
| 7 | 60.8 | 29.9 | 61.16 | 30.41 |
| 8 | 61.3 | 30.4 | 61.24 | 30.59 |
| 9 | 60.6 | 30.2 | 61.22 | 30.45 |
| 10 | 61.2 | 30.6 | 61.21 | 30.46 |

# SECTION 3: TEST ANALYSIS

**Table 2 & 3: Euclidean Error ε, Mean and Standard Deviation**

The second table presents the Euclidean error of each trial.

(Table 2)

|  |  |
| --- | --- |
| Trial | Error |
| 1 | 0.208 |
| 2 | 0.666 |
| 3 | 0.498 |
| 4 | 1.015 |
| 5 | 0.667 |
| 6 | 0.157 |
| 7 | 0.624 |
| 8 | 0.199 |
| 9 | 0.669 |
| 10 | 0.140 |

|  |  |
| --- | --- |
| Mean | 0.484 |
| Standard Deviation | 0.280 |

(Table 3)

The Euclidean error distance ε of the position of the robot was calculated for every trial using the equation (1). The Euclidean error is a good method of determining the order of the error, as it takes into account the error in X as well as in Y. In the following equation, Xm is the measured value Xd and Xo is the value of Xf , displayed by the odometer.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | | (1) |
|  | |

Below is the sample calculation of the Euclidian error:

The mean μ of the Euclidean error was calculated using the equation (2). The mean is the sum of all the Euclidean errors Zi, divided by the number of trials N. The standard deviation was also calculated for these same error values using equation (3) in order to determine how constant the errors produced are. The standard deviation is calculated by the square root of the sum of the squared difference between each trial Zi and its mean μ, divided by the number of trials N.

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Below is the sample calculation of these two equations:

# SECTION 4: OBSERVATIONS AND CONCLUSIONS

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

The observation was that the odometer reading at the final position was relatively accurate, as it often gave a close reading to the current position of the robot. Therefore, it is more likely that the error was caused from navigation instead of odometer. It has to be noted that both odometer and navigation uses the same numerical value of track length and wheel radius to calculate the angle and distance. If the width of the track and the radius of the tires are not exactly the same as the ones we used, then the calculations for both odometer and navigation can be erroneous. For instance, if the set wheel radius is in fact smaller than that measured, then the robot will always move farther than it should. Thus, the error in navigation was more likely to be caused by the external causes.

One of the biggest causes of error is due to the starting angle and location of the robot on the (0,0) waypoint. Indeed, when the robot was placed at a slight angle, this error would accumulate throughout the course and the final error would be relatively larger. When the robot was exactly aligned at the 0° angle, this error would be significantly small. Since the robot does not check its initial angle, it never corrects a placement error, meaning that the placement of the robot is crucial to the success of the lab.

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

Without the obstacle avoidance, the navigation controller moves the robot very accurately to its destination. Indeed, the mean Euclidean error of the robot is of 0.484 cm. The standard deviation is of 0.280, meaning that the Euclidian errors of the trials do vary from about 0.2 cm to 0.75 cm. The navigation controller moved made the robot move slowly and deliberately, accurately landing on each waypoint.

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

The robot settles very quickly after it arrives at the destination. For the navigation without obstacles, the robot stops immediately upon arrival, as it simply calculates the distance traveled and stops once it has reached the required distance. As for the navigation with obstacles, the oscillation terminates slightly more slowly but still overall fast. This could be due to the fact that the robot is not necessarily reaching the waypoint from the planned direction, causing delay on the settling of the oscillation.

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

Increasing the speed of the robot would undoubtedly decrease the precision of the robot to navigate correctly to the waypoints. Firstly, as explained in the previous lab, an increase in speed leads to an increase in the slipping of the robot’s wheels. Higher speed requires higher acceleration. Higher acceleration in turn results in more force being applied to the floor, which increases slip and worsens the odometer reading. Secondly, an increase in speed would result in a lower accuracy of the robot’s final destination. Since the robot checks its position using the odometer, a higher speed would cause the robot to realise its arrival at each waypoint “late”. This inaccuracy would only worsen at each point.

# SECTION 5: FURTHER IMPROVEMENTS

A first hardware solution would be to add another ultrasonic sensor to the robot. Although the sweeping motion of the single sensor was sufficient it most cases, sometimes the robot was far and wide enough from an obstacle that the obstacle was not detected. The robot would then pass by the obstacle and bump into it since it was just in reach. Adding a second sensor would completely remove this problem. With two moving sensors, a wide enough angle would be covered so that no obstacles could be out of the field of view of the robot.

Another hardware solution to this same problem would be to simply reduce the width of the robot. For this lab, the robot was relatively wide, which greatly increases the risk of it colliding with obstacles since the track covers almost half of a tile.

As for the software, a good solution to the colliding of the robot into the obstacles would be to implement a wall following class into the code. Since the avoiding of the obstacles was simply hardcoded, if the obstacle is place at a small angle instead of at 90 in front of the robot, the hardcoding could fail (see figure 3). By implementing a class that uses the ultrasonic sensor in order to always stay at a certain distances from the obstacles, this problem would be completely avoided. Indeed, the hardcoding always makes the robot turn 90° and move forward the length of a cube, so any other angle of the robot causes issues.

**Figure 3: Obstacle angle affect the avoiding performance**

Robot

Obstacle

Obstacle at 30**°**

Robot

Obstacle

Obstacle at 90**°**