# ECSE 211
# Design Principles and Methods
# Fall 2019

# Lab 3 Report:
# Navigation & Obstacle Avoidance

Group 21
Culha, Dafne 260785524
Udupa, Suhas 260867826

## Section 1: Design Evaluation

The objective of this laboratory was to design a robot capable of autonomously navigating a user-specified path composed of several vertices on a gridded environment. We were also required to implement obstacle avoidance and ensure that the robot would still arrive accurately at its final waypoint regardless of the presence of obstacles.

### Hardware Design

We began this laboratory with the hardware design. We had to modify our previous robot model to best fit the criteria of the new laboratory. Our robot is equipped with two motors attached to the wheels, a gyro ball in the back to provide support to the front wheels and an ultrasonic sensor, as seen in Figure 1, which was used to implement obstacle avoidance and replaced the color sensor from the previous lab. A question that rose up was whether we should place the ultrasonic sensor on a motor and make it rotate or fix it directly on the robot and keep it static. Since we worried it would be harder to keep track of the error and increase the cumulative error with a rotating sensor, we decided to go for a static positioning.

Based on our experience with the hardware in the previous laboratories, we wanted to keep the robot's design as compact as possible and to also distribute more weight onto the wheels for better traction to the ground. By slanting the EV3 processor brick on top of the wheels as seen in Figure 1 below, the increased pressure helped with the traction to the ground surface which better transferred the robot's rotational movement to lateral movement. In addition, the robot's compact design allowed for easier maneuvering around obstacles in relatively tight spaces. The ultrasonic sensor was placed in between both wheels and was directed to the front of the robot so that it could detect any obstacles directly in its path and have a broad view of what's ahead. Additional supportive structure was also added to the hardware of the robot following testing, in order to increase stability.

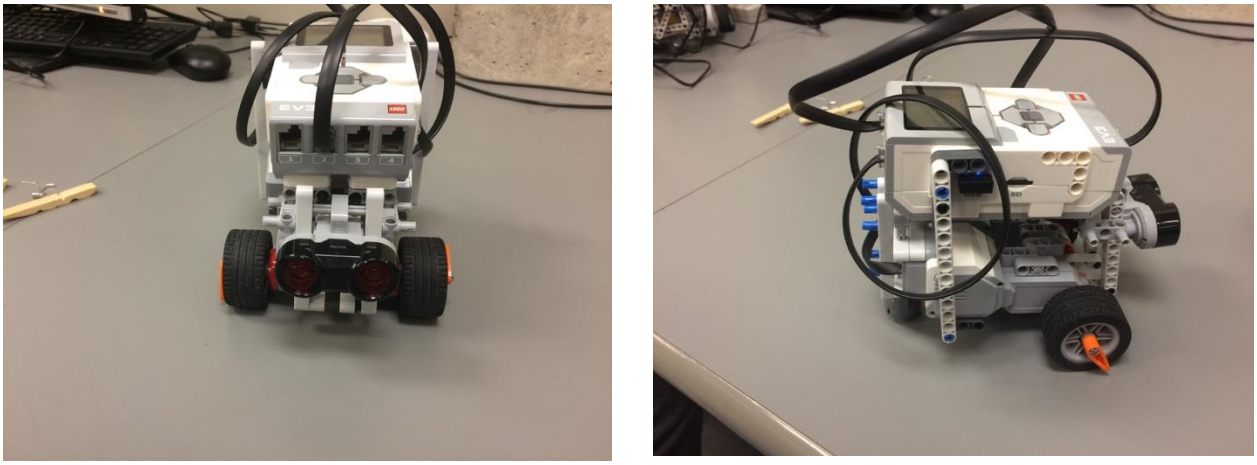

**Figure 1: Hardware design of the robot**

## Software Design

In terms of the software component of the lab, the design uses several threads running concurrently to allow the robot to simultaneously navigate to the specified waypoints, display its location, detect any obstacles and avoid them.

The odometer class was left mostly unchanged from the previous lab, as its functionality was sufficiently accurate.

The navigation class, which allows the robot to navigate towards specific waypoints, uses basic trigonometry to calculate the required displacements and trajectory angles for the waypoints. In fact, it computes the minimum angle at which the robot should turn (based on its current position) to get to the specified waypoint by calculating the tan of the angle between its x and y-coordinates and subtracting the angle read by the odometer (see Figure 2).

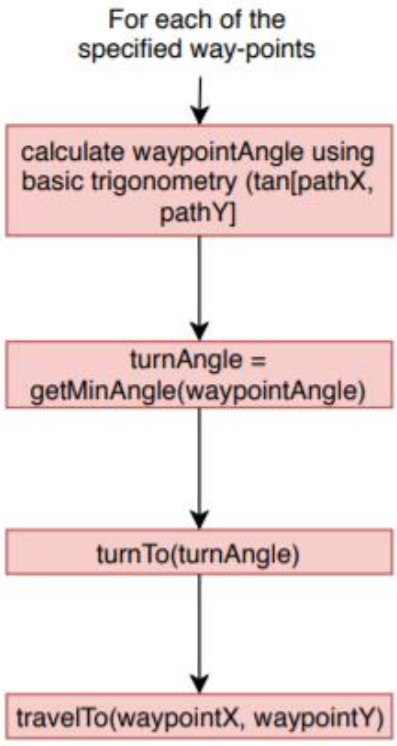


**Figure 2: Software design representation of the navigation class**

To implement the obstacle avoidance, we used the bang-bang logic from laboratory 1. Once an obstacle is detected, the robot avoids collision following a series of hard-coded movements. The robot simply skirts around the obstacle using the bang-bang approach and exits when the odometer reads the end angle (see Figure 3). Once the obstacle is avoided, the robot checks its latest position using the odometer and resumes its navigation to the next waypoint.

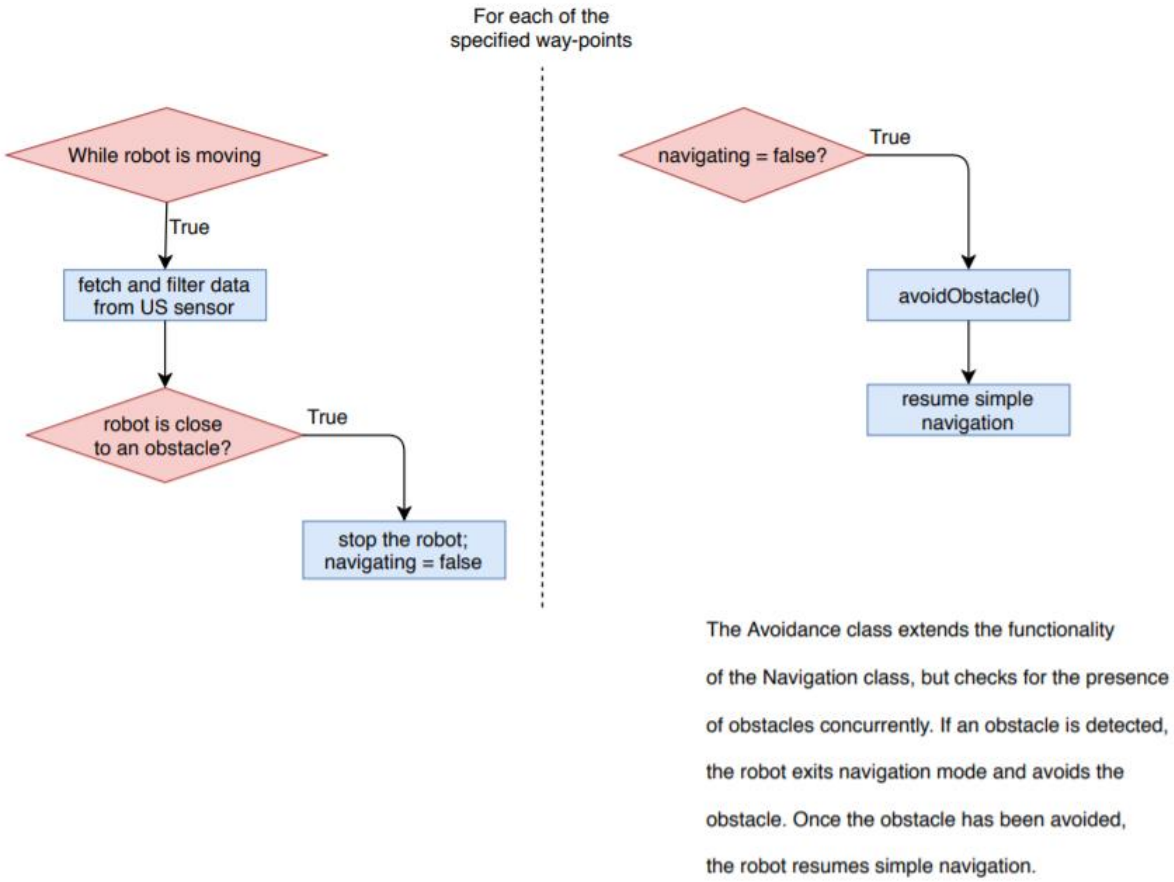


**Figure 3: Software design representation of the avoidance class**

To set the appropriate speeds for both wheels, we did several tests consisting of checking our robot's performance along a straight line to determine the ideal values. Furthermore, we conducted 360-degree spin tests to adjust the track of the robot to a value that allowed the robot to return to its initial angle.

## Section 2: Test Data

The expected final position of the robot and its actual final position at the end of its course are presented in Table 1. The robot's final destination point is the Cartesian point (3,1) and since the tile width is of 30.48 cm, we would expect the robot to finish its path around (91.44, 30.48) cm.

In the table 1, $X_d$ is the value expected and $X_f$ is the measured distance. The same notation is followed for the y-axis variables too.

**Table 1: Expected and Measured Distances in Navigation Test**

| | **Expected distance (cm)** | | **Measured distance (cm)** | |
|---|---|---|---|---|
| **Trial Number** | **Xd** | **Yd** | **Xf** | **Yf** |
| 1 | 91.44 | 30.48 | 91.59 | 30.69 |
| 2 | 91.44 | 30.48 | 91.56 | 30.72 |
| 3 | 91.44 | 30.48 | 91.40 | 30.28 |
| 4 | 91.44 | 30.48 | 91.46 | 30.62 |
| 5 | 91.44 | 30.48 | 91.82 | 30.45 |
| 6 | 91.44 | 30.48 | 91.48 | 30.53 |
| 7 | 91.44 | 30.48 | 91.50 | 30.72 |
| 8 | 91.44 | 30.48 | 91.66 | 30.38 |
| 9 | 91.44 | 30.48 | 91.57 | 30.47 |
| 10 | 91.44 | 30.48 | 91.71 | 30.48 |

## Section 3: Test Analysis

The Euclidean error ε of distance has been calculated for each trial using equation 1. In the equation, same notation has been followed as in table 1 which $X_d$ is the expected value and $X_f$ is the value measured in x-axis and $Y_d$ is the expected value and $Y_f$ is the measured value in y-axis.

$$\varepsilon = \sqrt{\left(X_d - X_f\right)^2 + \left(Y_d - Y_f\right)^2} \quad (1)$$

A sample calculation of the Euclidean error can be found below for trial 1.

$$\varepsilon = \sqrt{(91.44 - 91.59)^2 + (30.48 - 30.69)^2} = 0.130$$

We repeated the same calculation for our values in other trials. The Euclidean error of each trial is presented in Table 2.

**Table 2: Euclidean error distance ε between ($X_f$, $Y_f$) and ($X_d$, $Y_d$) for each trial**

| **Trial Number** | **ε** |
|---|---|
| 1 | 0.258 |
| 2 | 0.268 |
| 3 | 0.204 |
| 4 | 0.141 |
| 5 | 0.381 |
| 6 | 0.064 |
| 7 | 0.247 |
| 8 | 0.242 |
| 9 | 0.130 |
| 10 | 0.270 |

Table 3 presents the standard deviation and the mean value of Euclidean error distance ε between ($X_f$, $Y_f$) and ($X_d$, $Y_d$) for each trial.

**Table 3: Standard deviation and mean value of Euclidean error**

| **Standard Deviation** | **Mean** (μ) |
|---|---|
| 0.090 (σ) | 0.221 |

Mean (μ) is the sum of all the Euclidean errors divided by the number of trials. The mean of Euclidean error was calculated by using the equation 2. The mean of Euclidean errors is found to be 0.221 and our calculation can be found below.

$$\mu = \sum_{i=1}^{N} Z_i \qquad (2)$$

$$\mu = \frac{0.258 + 0.268 + 0.204 + 0.141 + \cdots + 0.130 + 0.270}{10} = 0.221$$

The standard deviation (σ) is calculated in order to determine how much our error values are spread out from the average (mean). It is calculated by using the equation 5. After using the formula in equation 5, we found the standard deviation of Euclidean errors to be 0.315 and our calculation to obtain the standard deviation can be found below too.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Z_i - \mu)^2} \quad (3)$$

$$\sigma = \sqrt{\frac{(0.258 - 0.221)^2 + (0.268 - 0.221)^2 + \cdots + (0.270 - 0.221)^2}{10}} = 0.090$$

## Section 4: Observations and Conclusions

Based on our observations while testing the robot, we concluded that the errors were due to the odometer rather than the navigator. This can be seen in the accumulation in error as the robot travels through the waypoints. At first, we noticed that the robot navigated accurately but near the end, it would miss the waypoints by a small offset. Since the navigator uses values from the odometer for its calculations, the error is carried forward.

However, we believe this to be a minor contributor to the error and that the more significant cause is the hardware. In fact, the navigator controller would send the proper signals to the motors to move the bot, but the hardware of the robot often led to systematic errors. These systematic errors are due to the low accuracy and precision of the hardware components and the lack of perfect symmetry. These errors are not accounted for by the encoders of the motors and the odometer, which caused the error to accumulate as the robot progressed in its trajectory. For example, the robot would often understeer or oversteer during turns and would deviate off course. As the robot deviated from its course, it was unaware of the error it was accumulating and therefore its calculations to reach the next waypoint were not accurate, leading the robot to finish away from the specified destination.

To continue, the navigator controller for its part was accurate. During our trial runs, it would bring the robot to its final destination with relatively decent precision (mean error of 0.221 cm).

Moreover, the navigator controller did not cause any oscillation as it arrived at its destination. Our robot would settle the moment the wheel's center was above the center of the destination waypoint. This is because the actual calculation for the displacement and angle needed in order to get to the waypoint was done before the robot's displacement itself. The robot's turns were accurate and were performed at a stable velocity, hence the robot could maintain a straight trajectory and didn't need to rectify its path before reaching the waypoint. Once the robot had completed its traversal, it would stop immediately and move on to the next waypoint.

Increasing the speed of the robot would first result in swifter and more abrupt movements on turns which could cause more deviation from the trajectory due to a higher angular error resulting from the lack of time to perform the turn accurately. Even if this error is negligible at one turn, it will be accumulated until it noticeably deviates the trajectory of our robot. Furthermore, an increase in speed would decrease its contact with the floor per unit time, which will provide less grip to the wheel. This decrease in friction would effectively increase wheel slip and cause the wheels to have less traction, causing the robot to slightly deviate from its path. As mentioned in section 1, this reduces the amount of rotational movement of the robot that is converted in real-time to forward movement and this delta is not taken into consideration by the encoders of the motors. Hence, a loss in accuracy in the robot's navigation is bound to occur since the wheel slip causes error in the actual effective displacement of the robot and since the robot relies on the encoders to make its calculations, its data is erroneous. Thus, increasing the speed would therefore accumulate more error during the robot's traversal and would undoubtedly decrease the precision of the robot to navigate correctly to the waypoints.

## Section 5: Further Improvements

In order to overcome the accumulated error in the displacement of the robot, we could implement a correction system using the color sensor which would optimize the trajectory of the robot. By adding the light sensor right below the ultrasonic sensor, facing the floor, we could use it to detect every black line the robot traverses. Every black line on the tiles could therefore be associated with a corresponding coordinate position, and every time the robot encounters one perpendicular to its path or when it crosses one diagonally one, the coordinates read by the odometer would be rectified to match the actual position of the robot, thus updating the robot's current position more accurately and also more frequently. This would effectively help reduce the error in displacement and the calculation of the robot's displacement required to travel to the next waypoint would be more precise. If the robot knew that it had surpassed or failed to reach its intended destination, it can then accommodate itself before moving to the next waypoint by adjusting its trajectory angle to account for this delta. In other words, the displacement calculation would be based on the robot's actual current position and not where it thinks it is.

Another software solution is implementing a continuous sweeping motion to the ultrasonic sensor as to increase its peripheral view so that it can better scan for obstacles.

A first hardware improvement would be to add a high-friction material around the robot's wheels to increase their traction to the ground. Among other, we could use rubber bands, small metal chains or even silicon, and doing this would significantly minimize the bot's deviation from its trajectory and would allow it to reach its destination waypoint more accurately.

Another possible hardware improvement would be to implement a gyroscope. Gyroscopes are small sensors that are capable of measuring the angular velocity of the body to which it is attached to. By recording the angular velocity of the robot, it is possible to calculate the current position of the robot in time. This would allow us to correct the robot's rotational angle during turns, as well as, the x and y- positions of the bot, which depend on the angle. This would allow our robot to maintain more accurately its trajectory course, during even longer paths.