GROUP 43
Ruptanu Chowdhury 260673074
Francis Duhamel 260746909

Lab 3: Navigation

Design Evaluation:

The physical design of the robot was very similar to the previous design. In fact, it was kept the same except for one small change: the sensor. We replaced the light sensor by the ultrasonic sensor. Keeping the same design allowed us to begin working on the software part much more quickly. Furthermore, the previous design was built with precision in mind. The robot had to be sturdy so that the uncertainty on the odometer readings could be reduced. For this lab, the robot had to move large distances using the odometer. Therefore, small uncertainties would accumulate over time. As will be explained in the software design part, the robot relies on these readings to know when a certain point on its path is reached. In other words, if the robot were wobbly, it could veer of course without the odometer knowing. Figure 1 describes visually the physical appearance of the robot.

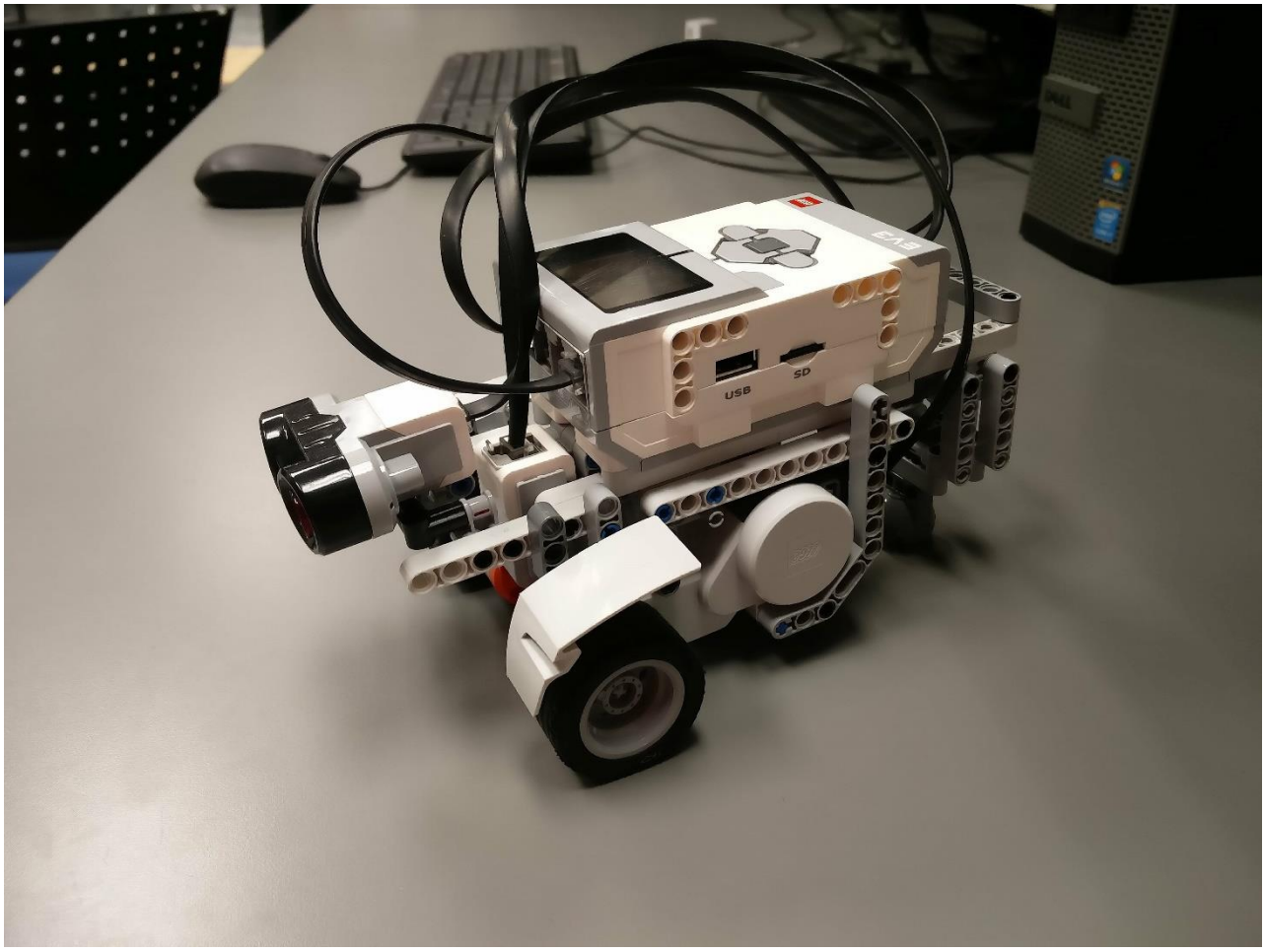

Figure 1. Hardware

Design

The software design of the robot was done with simplicity in mind. However, as noted by the teaching assistants, this simplistic approach may be too simple. The software aspect was divided into two parts. The first being the navigation, and the second being the obstacle avoidance. The navigation part was relatively straightforward to implement. To navigate through a given path, the software was given an array of integers representing points on the grid. It would break this array down into sets of two values, each set representing an x and y grid position. Taking into consideration only one set, the software would first start by determining in centimeters where it would need to go. Then, it would rotate towards that point and start moving forward. Only when the odometer's readings would be within a small margin of error would it consider that it is at the target point. It would then restart this procedure for the following point if there were any. Rotating to a certain angle was done by calculating the target angle based on the difference of the robot's position and its target position. The inverse tangent was used to calculate the target angle. The following describes this equation,

$$\Theta = \text{arc tan } dy/dx$$

Where

dy is the difference between the target y and the current y position,

dx is the difference between the target x and the current x position, and

Θ is the target angle.

The signs of dy and dx had to be considered as they were important in determining if the proper tan was being computed. The angle change was then calculated using the difference between the odometer's reading and the target angle. A small method was implemented to make sure the angle change was always the smallest possible. This angle change was then converted into wheel rotations

This conversion code was reused from the previous lab. The following figure 2 is a flowchart of the described code.

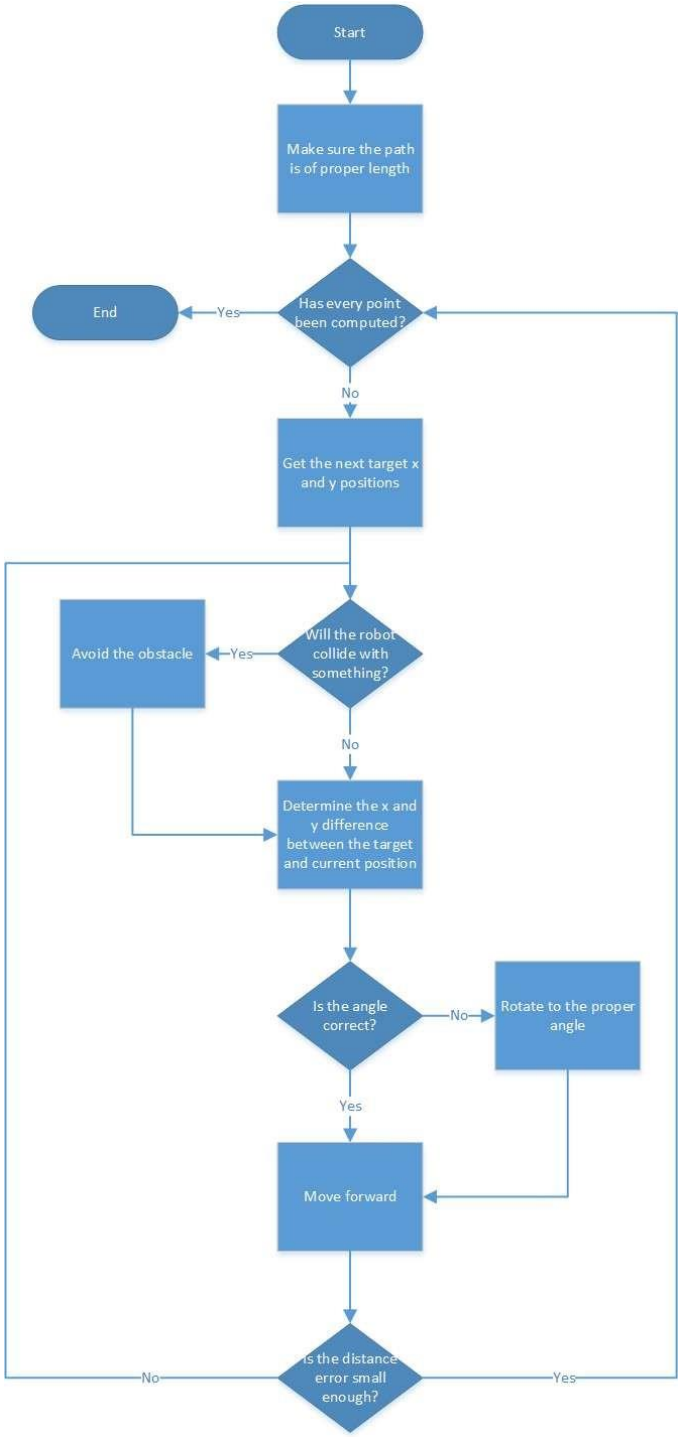

Figure 2: Navigation Flowchart

As explored during the classes, the simplest design is always the best. It is usually faster to implement and easier to understand. For this reason, the avoidance software was incredibly straightforward. Unfortunately, as mentioned by the teaching assistants, it did not fulfill every requirement. This simple implementation did not follow a wall, it rather travelled in a semi-predetermined path when it detected a wall in front of it. This means that if a second obstacle were on its avoidance trajectory, it would not be able to avoid it. Furthermore, the avoidance path would not account for the width of the obstacle, only its length. However, this algorithm was still capable of avoiding normal obstacles and was easy to understand and implement. Essentially, when an obstacle was in front of the robot, a trajectory that "should" clear the block was added to its trajectory. It would first stop moving and rotate 55 degrees. It would then move forward for 4 seconds and stop, rotate back 55 degrees and move forward. When moving forward, it would periodically turn 90 degrees to see if there was still an obstacle on its left. This was done until the obstacle was avoided. It would the rotate back towards its destination before the encounter and would resume its usual procedures. The following figure 3 is a flowchart of the described code.

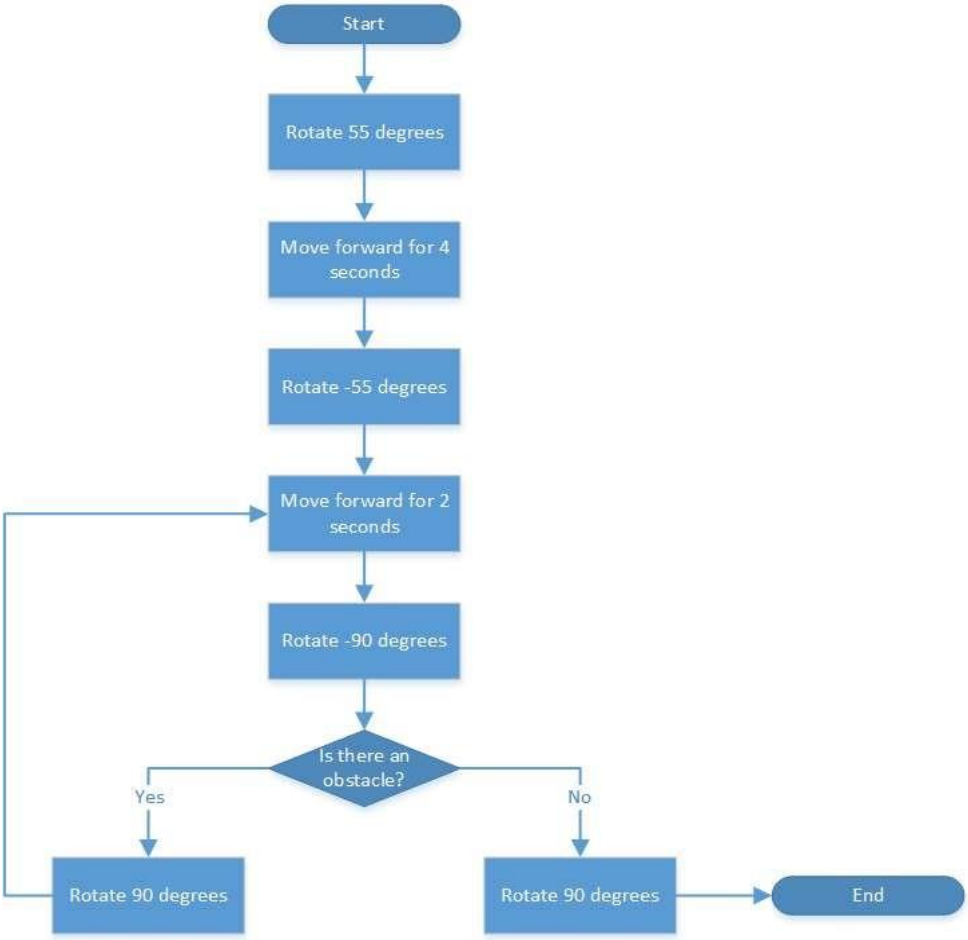

Figure 3: Avoidance Flowchart

Test Data:

Table 1: Navigation Test

Theoretical final X position (cm)	Theoretical Y position (cm)	Measured X (cm)	Measured Y (cm)	error ε
60.96	0	59.5	1	1.77
60.96	0	59.2	1.2	2.13
60.96	0	59.6	1.6	2.10
60.96	0	60.1	0.9	1.24
60.96	0	59	0.4	2.00
60.96	0	59.6	1.1	1.75
60.96	0	59.9	1.4	1.76
60.96	0	60.2	1.2	1.42
60.96	0	59.5	1.1	1.83
60.96	0	59.3	1.6	2.31

Test Analysis:

Mean Equation

$$\bar{x} = \frac{\sum x}{n}$$

Where,
$\bar{x}$ is the mean of the set of x values,
Σx is the sum of all x values, and
n is the number of x values.

Calculation

$\bar{x}$ = (1.77 + 2.13 + 2.10 + 1.2 + 2.00 +1.74 + 1.76 + 1.42 + 1.82 + 2.31)/10 = 1.83

Standard Deviation Equation

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

Where

sx is the standard deviation of the x values, and

xi is the ith value of the x values.

Calculation

$$s_x = \sqrt{\frac{\begin{pmatrix}(1.77-1.83)^2+(2.13-1.83)^2+\\(2.10-1.83)^2+(1.2-1.83)^2+\\(2.00-1.83)^2+(1.74-1.83)^2+\\(1.76-1.83)^2+(1.42-1.83)^2+\\(1.82-1.83)^2+(2.31-1.83)^2\end{pmatrix}}{9}} = 0.32$$

Are the errors present because of the odometer or the navigator? Give reasons to back up your claim.

The errors are present because of both the odometer and the navigator. The navigator contributed to errors as the angle it would turn was often off by a small margin. This is because perfect calibration was impossible to obtain. The robot would overshoot or undershoot slightly when turning. As the rotating was only dependent of the navigator and not the odometer, this error was caused by the navigator. The odometer would cause error because of its physical limitations. There are multiple physical factors influencing the robot that cannot be registered by the odometer. Therefore, the odometer's readings diverge slightly from the true position of the robot. The two wheels were not perfectly parallel to each

other and equidistant from the center of the robot. However, this was assumed by the odometer. Also, according to our measurements, both wheels had equal radii of 2.1 cm. However, the wheels can stretch and contract. This could have contributed to the odometer's errors. The axles can also move out of position, resulting in slight changes in the distance of each wheel from the center of the robot. Finally, the most significant cause of error was the slippage of wheels over the tile surface. All these factors resulted in noticeable error.

Observations and Conclusions:

How accurately does the navigation controller move the robot to its destination?
It does so very accurately. According the mean of the error, it would only be about 1.8 cm away from its actual target position. Making sure the robot would rotate the proper amount of degrees greatly contributed to highly precise displacement.

How quickly does it settle (i.e. stop oscillating) on its destination?
It does not oscillate at all. This is because the there was no correction if it missed its destination. The robot assumed it would reach its destination by doing one rotation. Over longer distances, this would result in a liability. The algorithms should therefore account the fact that the rotating method is not precise enough over longer distances. However, increasing the permissible "error" would fix this problem. Error accumulated by the imprecise direction would be caught by the larger destination error. This solution would not be ideal as the final position would be less exact. However, that error would be capped by a constant.

How would increasing the speed of the robot affect the accuracy of your navigation?
Increasing the wheel speed would obviously increase slippage, especially when turning. Therefore, the robot would think it is going in the right direction, yet its heading would be slightly off. This Less obvious, it would also affect the odometer's precision. This increased speed would increase the degree change between each sample fetch by the odometer. This would, in turn, increase the odometer's uncertainty which would contribute to an error in navigation as the navigation is highly dependent on the odometry to know when it has reached a certain position.

What are the main sources of error in navigation?
Two things would contribute to the error of the navigation. Firstly, there is the "allowed" error. If the odometry tells navigation it is within 1 cm of its destination, it may assume it is at the proper position. This is necessary because of the limitations of the system. Floating-point calculations are only approximations. Therefore, multiplying and dividing numbers always generates very small errors. These small errors are carried and amplified over time. This "allowed" error accounts for that. However, it also introduces small errors in the navigation. Secondly, the is the wrong angle change. As mentioned previously, it is impossible to absolutely calibrate the robot's constants (i.e. Wheel radius and distance from one another). Therefore, when rotating, navigation thinks it is at the proper angle (so does odometry) and will then navigate to the wrong destination.

Further Improvements:

What steps can be taken to reduce the errors in navigation and odometry?

Further reducing the speed and acceleration of the robot would reduce the slippage which in turn causes errors. Spending more time on calibration would also increase the precision of odometry and navigation when turning. Increasing the wheel width would also allow for more precise turns as the wheels would have to slip less to turn. Increasing the wheel width would reduce the ratio between the wheel angle change and its rotation. Finally, adding correction for the angle in navigation using odometry would help reduce error. By re verifying the angle is correct using odometry, it would prevent error from accumulating and would allow for reduced "allowed" error.

Identify one hardware and one software solution.

Hardware solution:

Installing a light sensor and implementing an odometry correction code (as used in Lab 2) would allow the odometer to update the coordinates of the robot using black lines on the tiles as reference. This would help remove errors accumulated over time.

Software solution:

We can modify the Odometer code by making it account for the tire slip over the tile surface. The wheels have difficulty catching the surface if the robot speeds up too fast, resulting in a tire slip. This can be solved by decreasing the speed and acceleration of the wheels, especially when the robot is turning or reaching a waypoint. This will allow for a smoother turn with less chances of the wheels rotating too fast.