Steven Cangul (260744412) and Noah Levine (260684940)

Lab 3: Navigation

Data

Table 1: Odometer Data

Odometer X (cm +/- 0.05cm)	**Odometer Y (cm +/- 0.05cm)**	**Actual X (cm +/- 0.05cm)**	**Actual Y (cm +/- 0.05cm)**	**Error X (cm +/- 0.05cm)**	**Error Y (cm +/- 0.05cm)**
60.4	0.2	60.1	-0.5	0.3	0.7
59.4	0.9	60.5	-0.8	1.1	1.7
60.8	0.1	59.7	-0.9	1.1	1.0
60.3	-0.9	60.7	0.7	0.4	1.6
58.5	-0.1	59.1	0.0	0.5	0.1
58.8	1.3	60.5	-0.8	1.7	2.0
60.7	0.6	59.0	1.1	1.7	0.5
61.0	1.0	59.7	-0.6	1.3	1.5
60.8	0.3	58.6	1.3	2.2	1.0
61.0	1.5	60.0	-0.9	0.9	2.4
			MEAN (cm)	**1.12**	**1.24**
			STANDARD DEVIATION (cm)	**0.58**	**0.67**

Data Analysis

Means were computed as follows:

MEAN(i) = $\frac{\Sigma Error(i)}{10}$ *[for i = x,y]*

Standard deviations were computed as follows:

$$STDEV(i) = \left(\frac{\Sigma\left(Error(i) - MEAN(i)\right)^2}{10}\right)^{\frac{1}{2}} \quad [for\ i = x,y]$$

Both the Odometer and Navigator are responsible for the errors seen above. The odometer introduces error as a result of wheel slippage resulting in the robot not accurately predicting the distance it has travelled. In this lab, we did not implement correction and thus the odometer was subject to some error. If the odometer class introduces error, then the navigator class will also introduce further error. This is because the navigator depends on the odometer. It obtains its values from the odometer in order to perform the proper navigation. Thus, if the odometer reports an incorrect position to the navigator, the navigator will navigate the robot incorrectly.

Observations and Conclusions

The robot's trajectory is constructed by a set of arguments and waypoints predefined in a class called Navigator. This class is essentially the driver for the robot, telling it where to go and how to calculate what it needs to do in order to get to point B starting from its actual position A set by its odometer. One of the methods composing this driver is the travelTo method; the travelTo method takes in the coordinates of the desired location and calculates both the distance the robot must travel to get there and the minimum angle it must turn to change its direction from its initial heading. This method continuously called a method names turnTo, which is responsible for taking in an angle theta and then setting the motors to turn through the proper input angle. Another thread running alongside Navigator detects the presence of obstacles using the ultrasonic sensor and then uses the methods discussed above to goa round it and get back on its designated path. All of this results in a robot that moves fairly accurately along its defined trajectory. Its reaches every waypoint as expected and lands within the 3cm range of where it is supposed to end up. Such deviations in the reported location may be caused by many factors such as slippage or a weak battery pack. Finally, the robot stops oscillating on its destination fairly quickly as we implemented the code in such a way that it would shut down the motors when arrived at its destination.

Increasing the speed of the robot would almost certainly negatively impact the accuracy of our robots odometry and its navigation abilities in general. Firstly, increasing the speed of the motors magnifies the chances of having slippage, especially when the robot turns, makes

abrupt changes in direction or accelerates from a standstill. This slippage would result in the robot's odometer thinking it has travelled a certain distance whereas the robot is in fact stationary for a brief period of time. Another area where speed would play a negative impact is in its ability to avoid obstacles along its path. Essentially, if the robot approaches the obstacle too quickly, it wont have enough time to detect it and apply the appropriate corrections to avoid it. Apart from slippage however, other factors such as the physical measurements of the robot's wheels and track width and its inherent error in the motor's angle rotation also play a huge role in the odometer's accuracy when considering large trajectories and several turns such as our tested path.

Further Improvements

As one of the main source of error arises from the fact that we have slippage when the motors accelerate abruptly, it is only logical to try to remedy this by increasing the static friction force between the wheel's contact patch and the smooth wooden test surface. One way to achieve this is by adding rubber balloons over the wheels of the robot.

Another way of going around this issue is to implement what is called the cross-coupled controller. Essentially, this controller continuously verifies the tachometer readings sent in by each motor and compares them to each other. If one of them is going too fast and has the potential to slip as it is held back by the other wheel, it is slowed down proportionally to its speed. The opposite happens if one of the wheels is seen as too slow.