LAB 3:

NAVIGATION REPORT

GROUP 62

MUHAMMAD ANZA KHAN 260618490

RYAN CHALMERS 260581055

DATA COLLECTED:

Table 1 shows the reported error as read by the robot, and real error as read by a ruler and the difference between them.

Reported x (in cm)	Actual x (in cm)	Delta x= Reported x – Actual x (in cm)	Reported y (in cm)	Actual y (in cm)	Delta y= Reported y – Actual y (in cm)
61.72	60.65	1.07	-0.79	-1.5	0.71
62.54	61.35	1.19	-0.55	-2.1	1.55
61.83	60.37	1.46	-0.3	-2.2	1.9
63.12	61.75	1.37	-0.9	-1.8	0.9
62.45	60.93	1.52	-1.1	-1.9	0.8
63.67	62.35	1.32	-0.97	-2.4	1.43
61.87	61.32	0.55	-0.84	-2.5	1.66
60.97	60.21	0.76	-0.91	-2.3	1.39
62.55	61.35	1.2	-0.75	-1.7	0.95
61.47	60.87	0.6	-0.67	-1.9	1.23

DATA ANALYSIS:

a) **Calculating the mean for the error values we collected**

-For x:

To calculate the mean, we sum all entries in the error column of x and divide by the number of entries.

(1.07+1.19+1.46+1.37+1.52+1.32+0.55+0.76+1.2+0.6)/10 = 1.104

-For y:

To calculate the mean, we sum all entries in the error column of y and divide by the number of entries.

(0.71+1.55+1.9+0.9+0.8+1.43+1.66+1.39+0.95+1.23)/10 = 1.252

Calculating the standard deviation for the errors in x and y:

-We subtract the mean from every entry in the delta column for both x and y

dx1 = 1.07 - 1.104 = -0.034, **dx2** = 1.19 – 1.104 = 0.086, **dx3** =1.46 – 1.104 = 0.356

dx4 = 1.37 – 1.104 =0.266, **dx5**= 1.52 – 1.104 = 0.416, **dx6**= 1.32 – 1.104 =0.216

dx7 = 0.55- 1.104 = -0.554, **dx8** = 0.76 – 1.104 = -0.344, **dx9** = 1.2 – 1.104 = 0.096

dx10 = 0.6 – 1.104 = -0.504

dy1 = 0.71 - 1.252 = -0.542, **dy2** = 1.55 – 1.252 = 0.298, **dy3** = 1.90 – 1.252 = 0.648

dy4 = 0.90 – 1.252 = -0.352, **dy5** =0.80 – 1.252 = -0.452, **dy6** = 1.43 – 1.252 = 0.178

dy7= 1.66 – 1.252 = 0.408, **dy8** = 1.39 – 1.252 = 0.138, **dy9** = 0.95 – 1.252 = -0.302

dy10 = 1.23 – 1.252 = -0.022

Next, we square the above values

(dx^2) =
(0.001156,0.007396,0.126736,0.070756,0.173056,0.046656,0.306916,0.118336,0.009126,0.254016)

(dy^2) =
(0.293764,0.08804,0.419904,0.123904,0.204304,0.031684,0.166464,0.019044,0.091204,0.000484)

We sum up the entries we collected for (dx^2) and (dy^2)

sum(dx^2) = 1.114174

sum(dy^2) = 1.43876

We divide the sums obtained above by (n-1), again n being the number of entries

X: sum(dx^2)/ (10-1) = 0.1237971

Y: sum (dy^2)/ (10-1) = 0.159862

Finally, taking the square roots of the two values obtained will give us the standard deviation

Standard deviation in x = 0.3518

Standard deviation in y = 0.3998

b) Are the errors present as a result of the odometer or the navigator? Give reasons to back up your claim.

Based on our analysis we have come to the conclusion that errors are produced by both the odometer and the navigator. The Navigator class, which in our case is the ***Navigator.java***

program, is responsible for simply driving the robot from one position to another. This class processes the data collected by the odometer to calculate the new heading, based on this it sets the motors accordingly in the forward direction and when the odometer values are within a predetermined distance from the desired destination, the robot comes to a stop. The navigator creates error which this group believes arises due to the inaccuracies in the ev3 motors. When the navigator class calls the ***turnTo ()*** method it makes both motors rotate by a specific amount but the motors do not stop rotating exactly at the tacho count specified, as a result error builds up every time the navigator calls this method.

Errors that arise due to the inaccurate readings of the odometer mostly occur because the odometer does not take into account the tire slips that happen when the robot is in motion, this happened to us multiple times, particularly during a trial demo in front of a TA where the tires slipped at launch and the robot went in the opposite direction. When the tires slip, there is a change in the odometer reading which results in these inaccuracies.

We also made several assumptions, as mentioned in the tutorial slides, that contributed to the errors as well. For example, we assumed that both wheels had the same radius and were parallel to each other. This may not have been the case, since the wheels can be subject to the physical laws of expansion and contraction, as a result the actual radii of the wheels might not be the same. In addition, the wheels tend to shift during operation, slightly altering the robot's width, this adds error to the odometer's angle calculation and since we did not implement an odometer correction in this lab the errors keep accumulating as the robot traveled further, which naturally led to errors in our measurements.

Observations and Conclusions

The purpose of the controller that we implemented was to get to the desired waypoints by using the odometers current values to calculate the distance and direction that the robot had to travel. The odometer values were constantly being updated during the navigation. The robot used the ***turnTo()*** method to figure out the precise degree by which it had to turn and the ***travelTo()*** method to figure out the distance it had to travel in order to reach its destination. Despite our attempts to make sure the robot was dead accurate in reaching its final destination, there was a slight error in the final destination of the robot due to several factors. One of them was the tolerance we had allowed in the calculation of the angle so that the robot did not oscillate frequently to correct the angle. Another limitation was the accuracy of the odometer which often gave incorrect values to the controller. Other hardware constraints such as the wheel slips while turning and taking a sudden start also contributed to the error. The accuracy of navigation would be reduced by increasing the speed. At higher speeds there will

be greater wheel slippage thus the error in the odometer will be larger. The sudden start of the robot at the turning points also caused the slippage hence at higher speeds the jerky motion would increase.

Further Improvements

As mentioned in the observations the main source of error was the wheel slippage that the wheels would encounter as the robot was moving along its path. A software solution to reducing this error is to reduce the acceleration at which the robot moves; this becomes an issue when time is important. Use of an additional sensor, such as the light sensor, to aid in determining the robots position is a very effective hardware solution. Reducing the acceleration of the robot will reduce the error in odometry because wheel slippage causes the odometer indicate erroneous values. Wheel slippage can cause linear and angular error. Linear error can be corrected however angular error, where there is slippage in one wheel that introduces error in the direction the robot thinks it is facing, can be more serious; as the robot move forward, the error is linear in the distance traveled. The use of the light sensor to correct angular would be a very effective solution. The robot is able to correct the angle with a single light sensor by remembering when it crosses a line, determining based on its knowledge of the distance between lines when it should detect the next line, and using trigonometry to determine the error in the angle.