ECSE 211: Odometry Lab Report

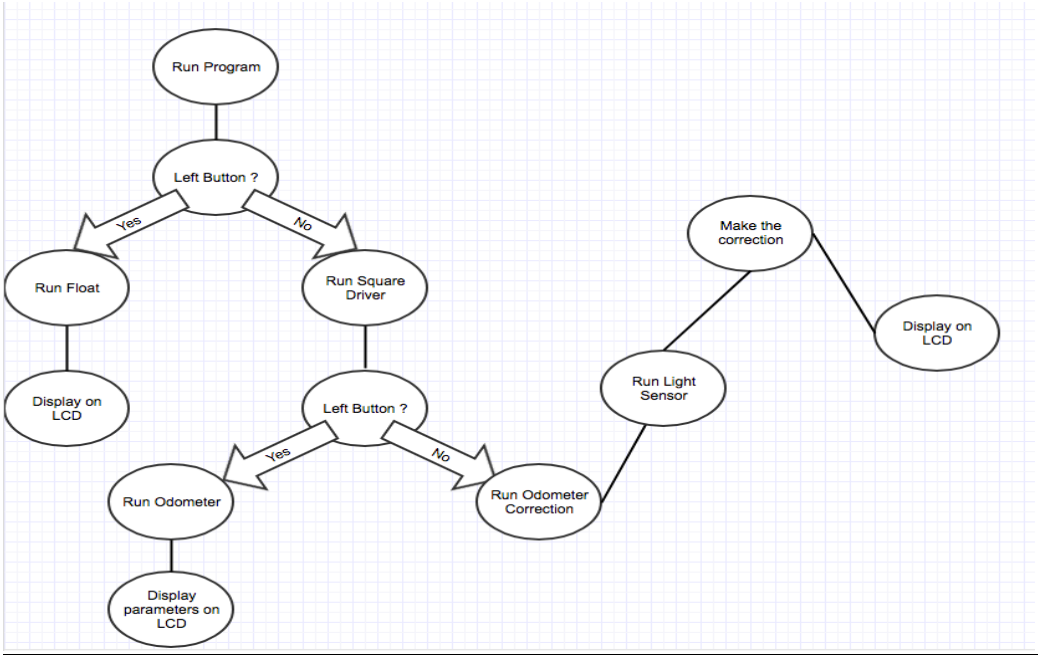
Ezz Aboulezz 260 677 463
Oscar Deceus 260 744 646

Group 57

Design Evaluation:

Hardware: Our robot design is very similar to the design we had for Lab 1. It consists of 2 motors with attached wheels at the front of the EV3 brick and is supported at the back by a metal sphere. The Ultrasonic sensor is placed on the left side of the brick at an angle of approximately 45 degrees. We placed the Light Sensor below the robot and to the front. The assembling of the robot was mostly driven by several videos and ideas found online. Refer to figure below.

Software implementation of the odometer:



Odometer

The odometer class gives the robot the ability to present parameters on the LCD display that define its position. Since the robot lives in a 2D world, it provides a number for the x-axis, a number for the y-axis and an angle for the orientation of the robot at all times. The robot does this through calculating the amount of rotations the motors made since starting to move. By calculating the number of rotations and then figuring out the amount of distance each wheel moved, the robot can calculate any distance moved in the x-y direction as well as calculate the orientation by comparing how much each wheel moved relative to the other. This is what our odometer basically did. It all came down to figuring out the math that converts motor rotations to distance and angle.

OdometyCorrection

This however does not give the robot any idea on where it is on the given board. This is where the odometry correction comes in. By giving the robot a light sensor that can detect black lines on the board, we gave it a method to find out where it is relative to a pre-determined origin (0,0). What we did is that we added a counter and initialized it at -1. The robot then checks what is its orientation and depending on what it gets it can only be in one of four states, either increasing x, decreasing x, increasing y or decreasing y (assuming it's driving in a perfect square). Then, every time it gets a black line reading the counter increases by 1 and we multiply the counter by 30.48 (tile) and set that as the x or y depending on the state. This means that the robot now marked the lines as 0,30,60 while increasing and then goes backwards in the second half of the trip.

Test Data and Test Analysis: (Refer to both tables)

The robot's X/Y positions without the correction of the odometer (in cm):

	Xs	Ys	Xf	Yf	X	Y	Error
	-14	-14.7	-7.5	-8.4	0	0	11.26099
	-14	-14.7	-9	-13	0	-0.23	15.62283
	-14	-14.7	-18.7	-17.4	0.04	-0.26	25.3962
	-14	-14.7	-18	-12.3	-0.23	-0.25	21.47034
	-14	-14.7	-19.8	-11.3	0.55	-0.24	23.16131
	-14	-14.7	-18.8	-14	0.3	-0.51	23.38354
	-14	-14.7	-21.7	-10	0.27	0	24.13878
	-14	-14.7	-19	-10.5	0.25	0.03	21.94182
	-14	-14.7	-18.2	-12.5	0.28	-0.26	22.16592
	-14	-14.7	-15.3	-18	0.5	0.04	23.98086
Mean					0.196	-0.168	21.25226
Stand. Dev.					0.2419	0.179059	4.397095

The robot's X/Y positions with the correction of the odometer (in cm):

	Xs	Ys	Xf	Yf	X	Y	Error
	-14	-14.7	-17	-11.5	-14.55	-13.16	2.959409
	-14	-14.7	-18.1	-18.5	-15.01	-18.34	3.09414
	-14	-14.7	-20.7	-12.5	-15.33	-19.01	8.439017
	-14	-14.7	-21.5	-13.1	-19.42	-12.63	2.13244
	-14	-14.7	-23.5	-11	-20.53	-19.49	8.994498
	-14	-14.7	-18.9	-13.5	-17.28	-18.48	5.236869
	-14	-14.7	-23.5	-9.9	-20.73	-18.71	9.235204
	-14	-14.7	-22	-13	-19.6	-19.92	7.32437
	-14	-14.7	-20.5	-11.5	-18.59	-18.82	7.565084
	-14	-14.7	-22.5	-4.3	-19.58	-22.19	18.12673
Mean					-18.062	-18.075	7.310777
Stand. Dev.					2.350559	2.947908	4.627115

1. How does the standard deviation change between the design with and without correction? What causes this variation and what does it mean for the designs?

The standard deviation of the x-y with correction is greater than the one without correction. However the error is lower. Many things could cause this variation. Without correction, no matter how the robot decides to go (perfect square or not), it will always keep the right X and Y measurements related to its starting point, which is supposed to give accurate results for the odometer in the end. With correction, we use the color sensor so that for each black line, the odometer should set X or Y position to a specific measure. This could be an issue in some situations because the odometer correction implies that the robot always performs a perfect square during its run, which does not always happen. This means that even if our main goal with the correction is to reduce the odometer errors caused by slips or drifts of the wheels, there are some improvements that could be done to give more accurate results in the odometer.

2.Given the design that uses correction, do you expect the error in the X position or in the Y position to be smaller?

We expect the X position to get the smallest error compared to the Y position simply because the X position correction is made in the last part of the 3-by-3 tile square which means that the Y position could change and because of the way that our design for the correction part is made we cannot control the odometer measurements for the Y position in the last part of the run. This is due to the fact that in the last quarter of the path we assume that we are moving in a straight line and change x only while this is not actually the case.

Observations and Conclusions:

Is the error you observed in the odometer, when there is no correction, tolerable for larger distances? What happens if the robot travels 5 times the 3-by-3 grid's distance?

The error does actually increase with greater distance travelled. However, the error does not increase linearly but rather irregularly due to several factors that accumulate the errors.

Do you expect the odometer's error to grow linearly with respect to travel distance? Why?

No, it is not expected that the error increase linearly with distance because there are several different factors affecting the error. Firstly, we have the dirt that accumulates non-linearly. The dirt accumulates irregularly hence it causes a non-linear increase in the error. Another factor could be the fact that every time we make a turn, or move forward, we don't make a perfect ninety degrees turn and we don't move in a perfect straight line. Hence these irregularities without any apparent pattern (due to slipping) create non-linear effects with regards to error increase with distance.

Further Improvements:

Propose a means of reducing the slip of the robot's wheels using software:

The robot slips mostly when it starts and/or stops moving so reducing the deceleration and/or acceleration would affect that slip. Also we could run a few tests to find out how much the robot slips on average and figure out a constant to multiply the correction with. This takes into account that we don't always move the distance that the odometer thinks it moves.

Propose a means of correcting the angle reported by the odometer using software:

1. When the robot has two light sensors:

   We can place them in parallel and then check which sensor reaches a black line first and then adjust the orientation accordingly. This will help us deal with the fact that we rarely ever move in a straight line.

2. When the robot has only one light sensor:

   First we calculate the time it takes from one black line to another, assuming straight motion. If the black line is reached at an angle then this will take more time. We can then find the total distance and then find the error in the angle using cosine (angle) = tile length/travelled distance. The error could be in any direction, meaning positive or negative. To figure out the sign of the angle we have 2 cases. If the robot takes less distance to reach next line this is due to a positive angle. If it takes longer distance this means the robot had a negative angle. Now that we have the angle error we can figure out the error in motion and not assume that we were moving in a straight line.