Laboratory 2: Odometry Report

ECSE 211: Design Principles and Methods

Prof. Ferrie, Prof. Lowther

Li Zhang (260743980)

Marwan Chehade (260675988)

Group 10

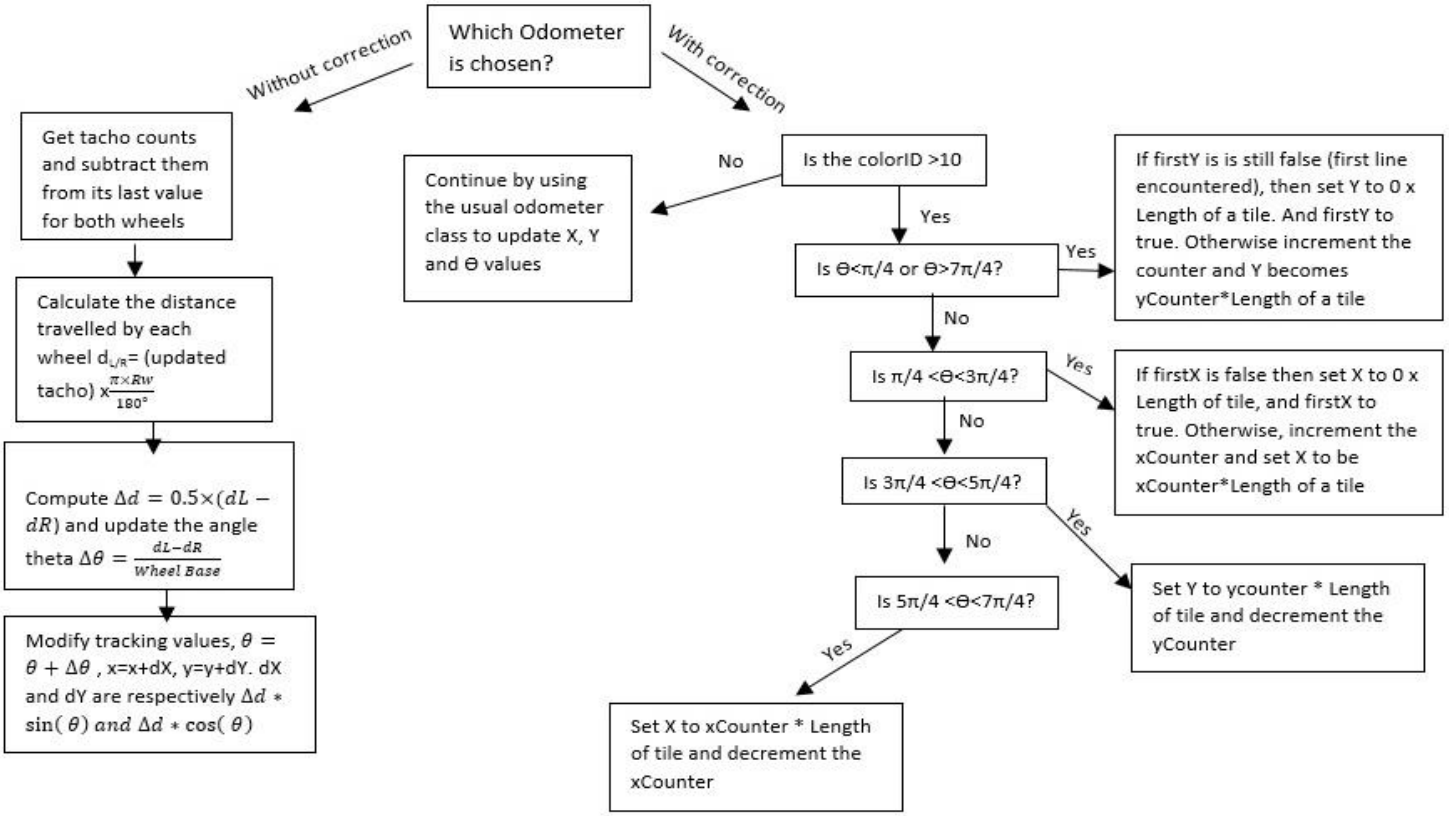
I. Design evaluation

Hardware design: The design consisted of the Mindstorms brick attached to two EV3 motors used for wheel rotation. Furthermore, a light sensor was attached to the front of the robot, between the two wheels, to detect black lines on the given grid. The sensor was placed as close as the ground as possible, to avoid error due to the ambient light.

Software design:

1- Odometer: The odometer software program relied on the tachometer to get the number of wheel rotations achieved during a movement or time frame. This number was then multiplied by $\frac{\pi \times Rw}{180°}$, R_w being the radius of the wheel, to calculate the distance travelled. Thus, the x-axis and y-axis coordinates were updated using the difference between the travelled distance in the left and right wheel as well as theta θ (the angle which allows to determine the direction of the robot). This angle is itself updated using $\frac{d}{Wheel\ Base}$ (d= d_L-d_R).

2- Correction for the odometer: The odometry correction software consists of creating an x and a y counter which take consider the theta at the moment a black line is detected (using the ColorID instance) to increase or decrease its value, by which is multiplied the length of a tile. The X and Y positions are then accordingly modified using these values. (*Refer to the flowchart below*)



II. Test Data

Without Correction					With Correction				
X	**Y**	**Real X**	**Real Y**	**ϵ**	**X**	**Y**	**Real X**	**Real Y**	**ϵ**
-0.12	-0.19	-1	-1	1.196035	-12.51	-13.36	-11	-13	1.552321
0.07	0.24	0	-0.5	0.743303	-14.32	-11.32	-13.2	-11.4	1.122854
-0.45	-0.02	-1.5	-0.5	1.154513	-14.2	-13.2	-13.9	-10.7	2.517936
-0.13	0.03	-0.5	0.4	0.523259	-12.04	-13.44	-11.3	-11	2.549745
-0.12	-0.18	-4	0.35	3.916031	-13.82	-10.06	-13.3	-9.2	1.004988
0.41	-0.06	-2	-0.8	2.521051	-12.47	-13.62	-11.3	-12.9	1.37379
0.16	-0.57	-3	-0.5	3.160775	-14.15	-12.78	-13.8	-14.2	1.462498
-0.34	-0.23	-0.8	-0.9	0.812712	-12.65	-11.34	-11.5	-10.2	1.61929
0.42	0.35	1.1	0.8	0.815414	-14.1	-13.27	-13	-11.5	2.083963
0.54	-0.12	1.2	0.8	1.132254	-14.3	-12.7	-12.7	-10.2	2.968164

Table 1 Results of 10 tests with and without correction

	Without correction			With Correction		
	X	**Y**	**ϵ**	**X**	**Y**	**ϵ**
Mean	-0.02571	-0.10714	1.887853	-13.456	-12.509	1.825555
Standard Deviation	0.271469	0.250181	1.311697	1.037073	1.517261	0.666366

Table 2 Mean and standard deviation for X, Y and ϵ for each design

III. Test Analysis

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 for the design with correction is less compared to the one with no correction. This variation is due to the fact that the correction program allows the robot to adjust the X and Y positions showed on the screen depending on where it is approximately on the grid. Thus, each time X or Y are corrected this allows to eliminate any accumulated error during the process of moving. However, this does not happen for the design with no correction. This means that the design with correction leads to more precision and more constant and predictable values.

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

A lower standard deviation means the values obtained vary less around the mean and thus for a low-enough mean, this leads to more precise values for X and Y with a relatively low margin of error. For example, for a mean of 1.8256 and a standard deviation of 0.666, the error obtained can be at most 2.492. While a mean of 1.89 and standard deviation of 1.31 for the design without correction, the error can go up to 3.2. The error in the X- or Y-position is thus expected to be smaller for the design with correction.

IV. Observations and Conclusions

1. *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 5-by-5 grid's distance?*

When the robot travels a 3-by-3 grid as opposed to travelling a 2-by-2 grid, the observed error grows but is still tolerable. However, if the robot travels a 5-by-5 grid the error is much larger and exceeds the tolerable limits for some of the trials.

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

As the distances become larger, there are more chances of slipping or of errors accumulating. For example, if the robot does not travel in an exact straight line, the larger the distance of its path the more this error will affect its final position and how far away from it does it end up being. Hence, as the travel distance increases, the error grows in a relatively linear fashion. Furthermore, if the robot is not placed parallel to the axis, the more it travels, the bigger the error will be. For example, if you place the robot on a slight angle, as the distance increases, the error will increase as well, as seen on a triangle: given an acute angle, the longer the hypotenuse is, the bigger the opposite side will be.

V. Further Improvements

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

The robot's wheels' slip can be reduced by a using the smoothAcceleration in the program which allows the wheels to reaccelerate gradually after each stop and thus gain adhesion to avoid slipping if full speed was applied directly.

Propose a means of correcting the angle reported by the odometer using software when:
The robot has two light sensors.

Using two light sensors, a sensor can be placed next to each wheel and thus detect when each wheel is reaching a black line. The difference in time between the two sensors to detect the black line can be translated into a distance using the known velocity at which the robot is travelling. Hence, the distance separating both wheels from the black line at the moment the second one (wheel which is further away) detects it can be offset by making the slower wheel travel faster (for the distance required only then comes back to its normal speed) such that it compensates and comes back to the same level as the other wheel, the robot will then go back to a straight line with an accurate theta.

The robot has only one light sensor.

Using one light sensor, one can find the distance travelled by the robot ($\sqrt{(x2-x1)^2+(y2-y1)^2}$) between two beeps, i.e. between two lines detected. If the robot is travelling in a straight line, the distance travelled should be equal to the length of a tile. Thus, if the distance is different than the actual width of a tile, using trigonometric formulas, we can determine the angle at which the robot is travelling relative to the straight line it should be at and thus the read value of theta can be adjusted or its path can be corrected to go back to a straight line.

For example:

$$\theta = \cos^{-1}\left(\frac{Length\ of\ a\ tile}{Distance\ travelled}\right)$$