ECSE 211: Design Principles and Methods

Lab 2: Odometry

Arianit Vavla : 260868601

Cheng Chen: 260775674

Group 20

September 2019

Section 1: Design Evaluation

Hardware Design

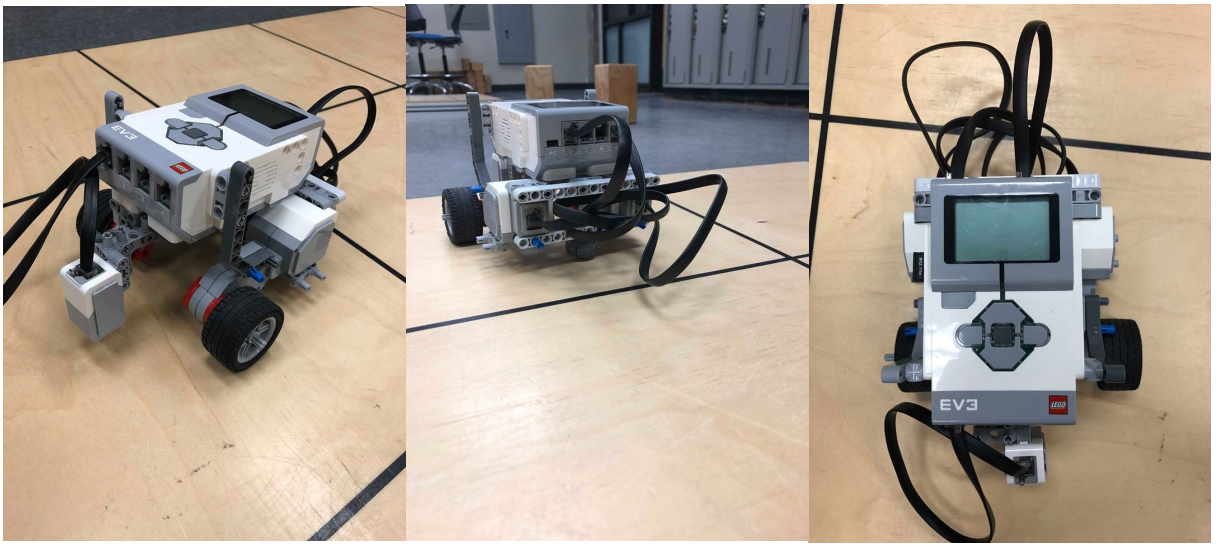

Figure 1. Robot

First of all, we wanted to make the robot as small and as compact as possible because it will make it easier for us to design on the software side. We fixed the motor to the sides of the EV3 controller and connect the motor to two tires in the front. On the back of the robot, we used the metal ball for the robot to gain more stability and the metal ball has less friction between the floor than other materials provided. Since there is a lot of empty space between two motors, we are able to put the wires inside the robot. For the first few tries, we put the light sensor at the top of the robot and pointed to the ground. We found that the data returned by the light sensor is not really accurate because there are many lights in the room. We decided to move the light sensor so that it is closer to the ground. The data returned was better than before, which made it easier for us to detect the black lines.

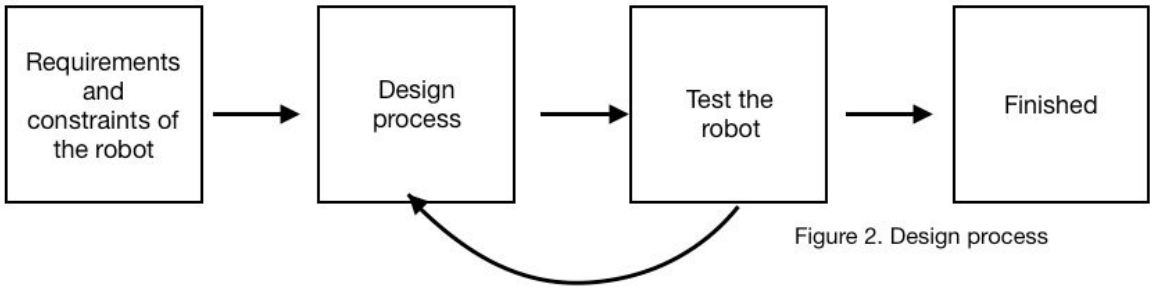

Figure 2. Design process

Software Design

There are six Java files which are Display, Main, Odometer, OdometerCorrection, Resources and SquareDriver.

For the Odometry, we started by translating the math formulas given in class into proper Java language. We also declared a few variables that are needed to implement this algorithm. When we ran our first try, we found that x is negative and theta is extremely large. After some modifications for theta, everything was working.

For the OdometryCorrection, we started by testing the light sensor and figure out how we can detect balck lines. We commented out all the code except the detection part. For the first few tries, we used red for the sensor mode, the robot sometimes skips a line or detects the line more than twice. After a few tries, we found that RGB is the best mode in our case. We collected a few data returned by the light sensor to find the threshold of detecting black lines. Then we implemented an algorithm that sets the coordinates of the robot every time it detects a line. For example, we set y equal to TILE_SIZE when it detects the first line.

The following flowchart will show OdometryCorrection.

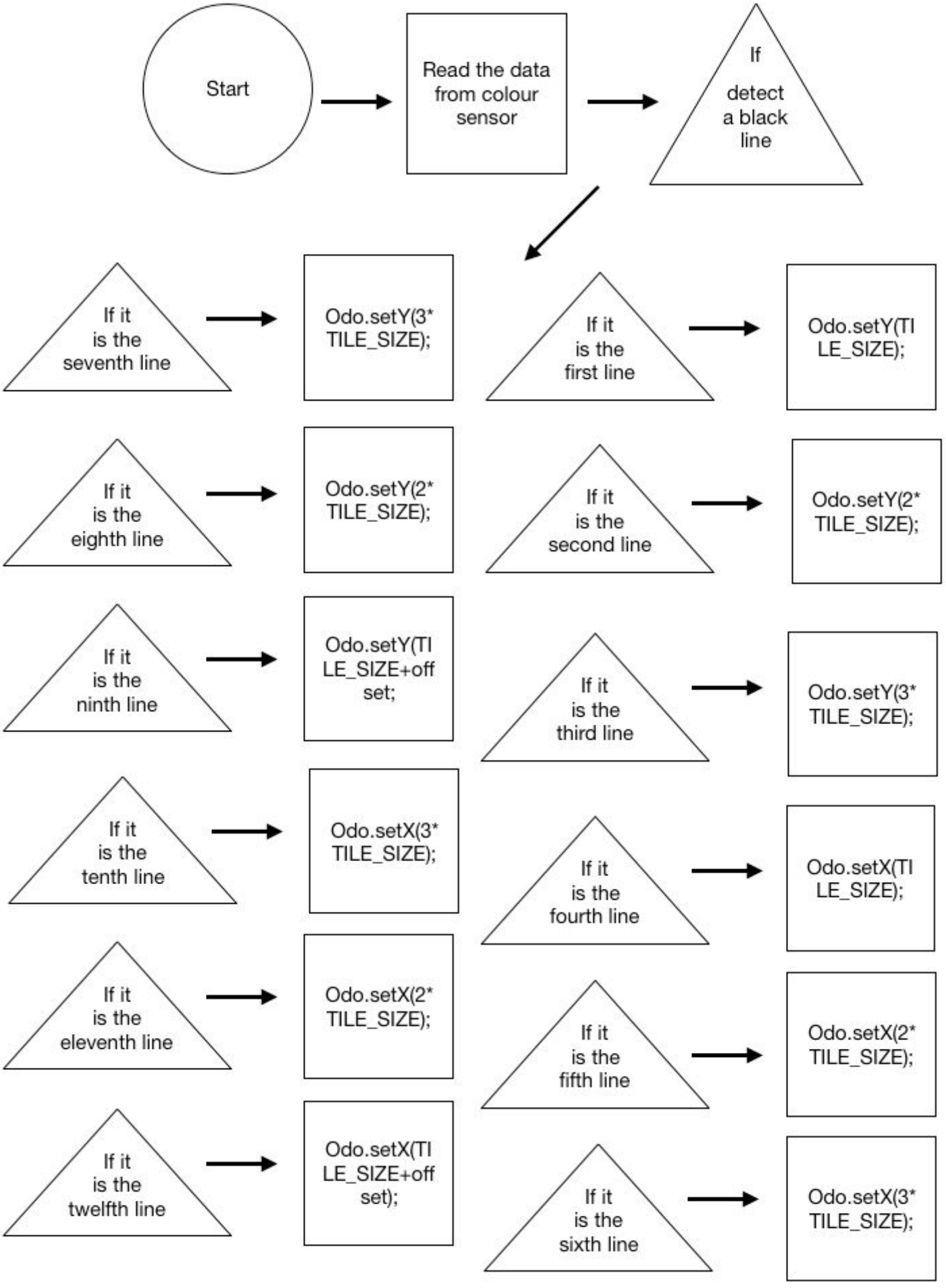

Figure 3. Flowchart for OdometerCorrection

Section 2: Data Test and Analysis

We used the following formula to find the error value:

$$\varepsilon = \sqrt{(X - X_F)^2 + (Y - Y_F)^2}$$

Data of Odometry Test Without Correction

X	Y	XF	YF	error value ε
1.158	0.18	-3.1	1.6	4.488536955
0.13	0.09	1.5	1	1.64468842
-0.871	-0.222	1.2	2.1	3.111386347
1.09	0.19	3	2.2	2.772760357
0.1	0.3	2.1	-1.7	2.828427125
0.334	0.11	3.1	4.1	4.854982595
0.53	0.47	-1.5	1.3	2.193125623
0.9	0.8	3.1	-0.9	2.780287755
0.8	0.6	2.4	4.7	4.401136217
0.2	0.2	-1.5	-1.1	2.140093456

Data of Odometry Test With Correction

X	Y	XF	YF	error value ε
12.34	25.41	12.5	22.9	2.515094432
21.5	17.88	22	20.1	2.275609808
17.8	23.63	16.6	22.8	1.459075049
12.22	19.9	12.9	19	1.128007092
18.43	23.81	20	20.5	3.663468302
15.33	12.25	15.9	13.5	1.373826772
19.46	16.17	21	18.2	2.548038461
21.09	13.66	21.5	15.8	2.178921752
18.6	22.4	19.2	20.5	1.992485885
10.55	14.45	11.8	17.5	3.296209945

Mean without correction

X	Y	error value ε
0.4371	0.2718	3.121542485

Mean with correction

X	Y	error value ε
16.732	18.956	2.24307375

Standard deviation without correction

X	Y	error distance ε
0.6042899323	0.2887004599	1.097536136

Standard deviation with correction

X	Y	Error distance ε
3.894608239	4.738209226	0.8148075116

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

The mean of X and Y are smaller without correction than with correction because the measure without correction was taken from the starting point of the robot, whereas for the odometry with correction, the point of reference was the bottom left corner of the tile where the robot starts its path. The mean value of the error value is smaller when there is a

correction because the robot modified its position each time it crossed a line, which means that there is less of a path made by the robot, which means less rotations to make an error. The standard deviation for X and Y is smaller in the test without correction because the odometer varies its values by not a lot and it is relative to the starting point, while for the test with correction, the starting point was at different positions in relative to the bottom left corner. The standard deviation of the error value is less for the test with correction than without for the same reason as the mean error value reason, which means more consistency. The design with correction is better because the mean is smaller for the error value, which means it has a smaller error, and the standard deviation is smaller too for the test with correction, which means more consistency of positioning.

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

I expect the X direction error to be smaller because the last line the machine detects and corrects the position is in the X direction, so the robot travels less distance in the X direction without correction which means less error for X than Y.

Section 4: Observations and Conclusions

The error observed in the odometer without correction would not be tolerable for larger distances and if the robot travels 5 times the 3-by-3 grid's distance, the error would be approximately quintupled because the error is expected to grow linearly as the distance is becoming larger. This is because the robot is modifying its position in a discrete manner and not a continuous one, which means that the robot is not taking measurements and modifying it the whole time, but only when the processor is able to take the next value, which means

there is time when the robot is not measuring its position but is still advancing, and this causes a small error on the distance, but more distance the robot is traveling, more traveling is done without taking the measurement, which means more error in a linear way is done because the errors are added by each sampling time.

Section 5: Further Improvements

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

As we know, as the speeds get faster and faster, the chance of slip of the wheels gets higher. To reduce the slip, we can reduce the speed of the wheel. Also, when the robot starts to accelerate, we should reduce the acceleration so that it reaches MOTOR_SPEED slower. Thus, less slips.

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

The robot has only one light sensor

We can put two light sensors on the left and right side of the robot respectively. When it reaches a black line, if one of the sensors detects the line before another, we know that the robot is not moving in a straight line. We can implement an algorithm that calculates the angle by the time difference between two sensors, the speed of the robot and geometry.

If we have only one sensor, we can record the distance it traveled from one line to another. Since we know the actual tile size, we can compare these two values. For the following figure, the red line is the actual distance traveled , and the black line is perpendicular to two balck lines which is equal to tile size. The length of red line can be calculated by using the time it takes to travel multiplied by the speed of the wheels. Then, We

can use software to calculate theta using the length of blue and length of red line by Pythagorean theorem.

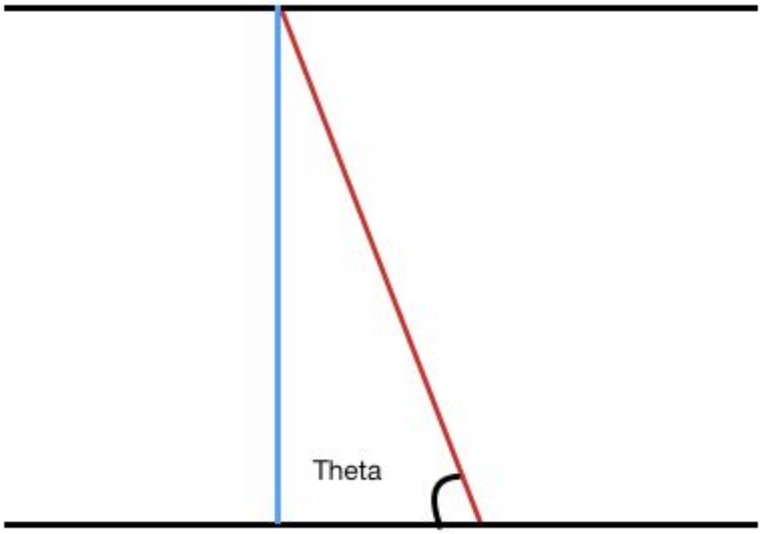

Figure 4. For question 2