Lab Group: 43

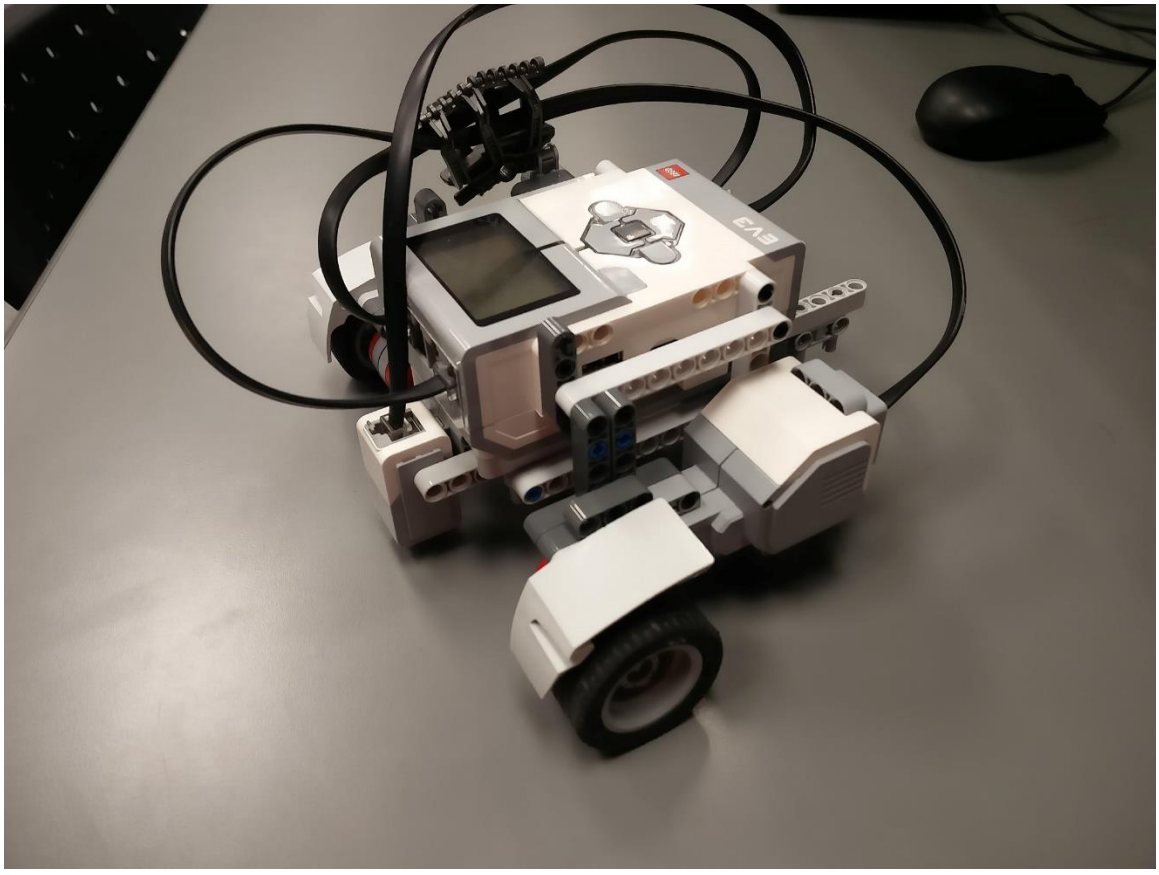
Name: Ruptanu Chowdhury - 260673074

Name: Francis Duhamel - 260746909

ECSE 211: Lab 2 Report

Section 1:

Our robot consists of four main parts, two motors to which we connected wheels, one color sensor, and one EV3 LEGO brick. To add more stability, we added a steel ball in the back of the robot. The ball is located at the back of the brick and the motors are attached to each side of the brick. The sensor is placed at the front of the robot, close to the brick to minimize movement during rotations. It is attached in such a way that it is as near as possible to the ground in order to get more accurate readings. The wires connecting the motors and the sensor are passing through a holder to prevent them from disturbing the robot.



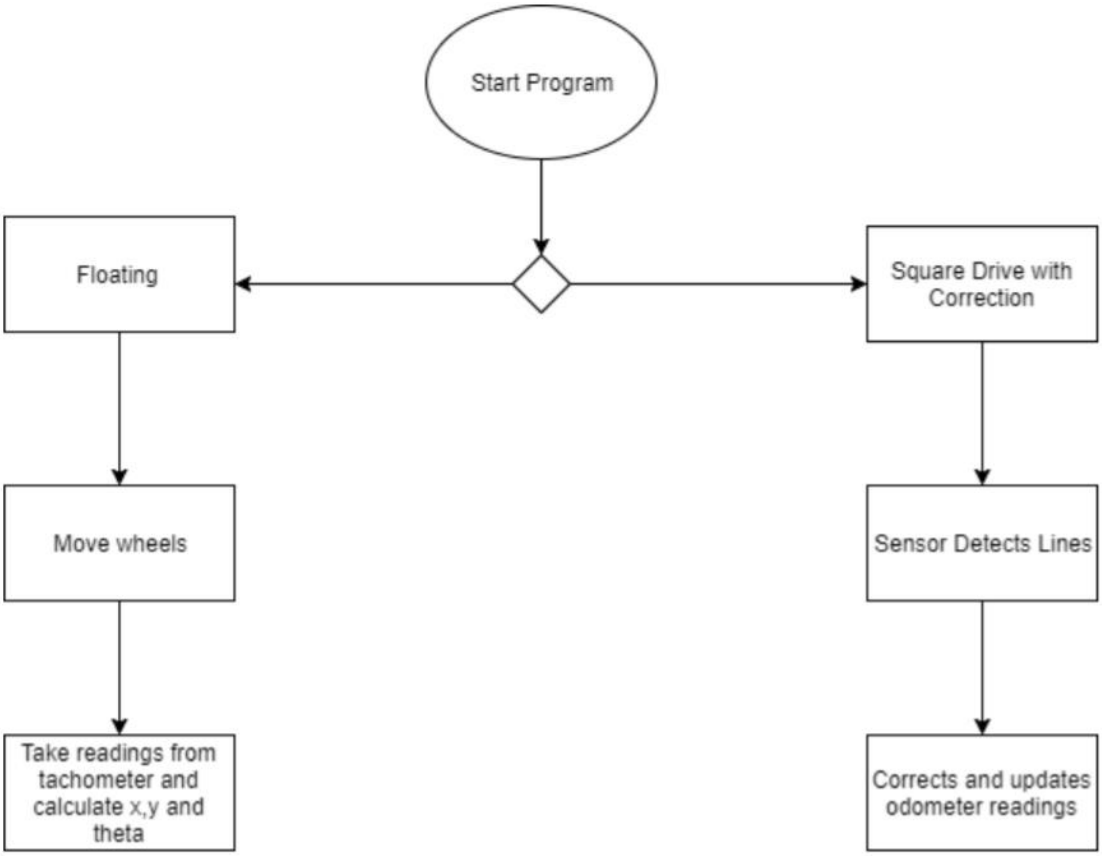
The Square Driver is implemented in such a way that, when executed, the robot should advance 91.44 cm, which is the distance of three tiles, then turn to the right at an angle of 90 degrees and repeat the operation three times. This means that it runs in a square with a side of 91.44 cm. This is implemented with a formula that uses the wheel radius and the width of the robot to

determine the amount of turns the wheels have to make in order to first move forward 91.44 cm and then do the 90-degree angle turn.

The Odometer uses the amount of turns both wheels make with the “TachoCount” to determine the position of the robot. It also uses the direction of the robot, hence the angle it is at, compared to the starting position, to determine whether to increase or decrease X and Y. In our case, the y-direction is the initial direction that the robot faces. In the calculations, the angle is in radians and not degrees as shown on the display, this is because the theta we use in the formula is not the same as the one that is shown. We also made it so that when the angle hits 360 degrees, it would drop to 0, and if it is lower than 0, 360 degrees would be added.

The Odometer Correction uses light sensors and known values of the distance of each tile to reduce the error in the odometer readings. As soon as the light sensor detects a line the robot checks the value of theta. If the theta is between 358 degrees and 75 degrees, it checks the number of lines previously detected in the y-axis. If it is 0 then the robot sets the point of the first line as 0 and increments the number of lines detected. If the number of lines detected is greater than 0, then the robot sets the distance for y as distance of each grid number of lines detected and then increments the number of lines detected. There is also a variable which saves the highest number of lines detected in the y-axis. If the theta is between 85 and 160 degrees the same thing is done along the X axis. The robot also saves the highest point of of Y as soon as the robot detects the first line along the X-axis. When the theta is between 161 and 245 degrees it compares to see if the number of lines previously detected along the y-axis is same as the highest number of lines detected. If it is then the robot sets the distance travelled as delta y by subtracting the value in the odometer with the highest value of y and sets the value of the highest point of X. After that the robot decrements the number of lines in the Y-axis. If the number of lines is less than the highest number, then the robot sets the value of Y by subtracting the highest value of Y with delta Y and the number of tiles detected while the robot is going down the y axis multiplied by the distance of a tile. The number of lines detected going down the Y-axis is calculated by subtracting highest number of lines detected with number of lines detected. If the theta is between 246 and 358 the same thing happens as described above on the X-axis.

Flowchart:



Section 2 & 3:

For both the correction and non-correction tests, we calculated the initial and final distance in respect to the origin as this made the measurements more accurate.

Table 1: Measurements without correction

Starting X	Starting Y	Final X	Final Y	Final X-Starting X	Final Y- Starting Y	X-Odometer	Y-Odometer	Error
-14.6	-10.4	-10.5	-12.5	4.1	-2.1	0.3	-0.68	4.056649
-14.6	-10.4	-12.3	-12	2.3	-1.6	0.62	-0.4	2.064558
-14.6	-10.4	-9.8	-12.5	4.8	-2.1	0.65	-0.32	4.515628
-14.6	-10.4	-10.8	-12.9	3.8	-2.5	0.61	-0.47	3.781137
-14.6	-10.4	-12.1	-13.4	2.5	-3	0.14	-0.41	3.503955
-14.6	-10.4	-9.9	-12.4	4.7	-2	0.38	-0.57	4.550527
-14.6	-10.4	-12.7	-12.56	1.9	-2.16	0.43	-0.2	2.45
-14.6	-10.4	-12.5	-11.9	2.1	-1.5	0.21	-0.13	2.334309
-14.6	-10.4	-10.6	-12.5	4	-2.1	0.58	-0.36	3.837186
-14.6	-10.4	-11.3	-13.1	3.3	-2.7	0.39	-0.68	3.542386

Table 2: Measurements with correction

Starting X	Starting Y	Final X	Final Y	Final X- Starting X	Final Y- Starting Y	X-Odometer	Y-Odometer	Error
-14.6	-10.4	-14.1	-11.9	0.5	-1.5	-11.75	-12	2.352126697
-14.6	-10.4	-14	-15.9	0.6	-5.5	-15.02	-14.63	1.628895331
-14.6	-10.4	-13.2	-13.4	1.4	-3	-11.8	-11.96	2.008382434
-14.6	-10.4	-10.4	-13.2	4.2	-2.8	-12.18	-14.09	1.9901005
-14.6	-10.4	-11.7	-11.5	2.9	-1.1	-13.73	-10.98	2.095542889
-14.6	-10.4	-13.5	-12.9	1.1	-2.5	-13.04	-11.82	1.173882447
-14.6	-10.4	-11.5	-12.4	3.1	-2	-15.37	-11.34	4.012542835
-14.6	-10.4	-13	-13.4	1.6	-3	-11.8	-11.96	1.874459922
-14.6	-10.4	-11.7	-11.1	2.9	-0.7	-11.55	-10.48	0.637887137
-14.6	-10.4	-12.7	-13.8	1.9	-3.4	-14.3	-14.5	1.74642492

Table 3: Mean and standard deviation

	Mean	Standard Deviation
X without correction	0.431	0.178738
Y without correction	-0.422	0.182323389
Error without correction	3.527359402	1.053299858
X with correction	-13.054	1.457167725
Y with correction	-12.376	1.488595162
Error with correction	1.952024511	0.8766225

The standard deviation in the odometer readings for both X and Y when the robot is run without correction is very small but the error has a high mean and standard deviation. On the contrary, we have high readings for the X and Y coordinates and low error when we apply correction. These high values are due to the fact that the origin isn't at the same place in both runs. When we run without correction, the origin is where we started and since we are supposed to do a perfect square, it makes sense that, if the code works, the odometer should show little distance from the starting point. However, since no correction is applied, it is normal to see a large margin of error. On the other hand, when we apply correction, we change the origin. Since the robot still makes the same square, it is normal to see a big difference in X and Y coordinates. Since we do correct the odometer here, we can see that the error has a smaller mean and standard deviation because it is more precise. This is why the standard deviations are different. For the design, it means that, the more complicated the trajectory gets, the higher the standard deviations will be if we do not use correction. Ultimately, this will lead to the failure of a task at a higher and higher percentile.

Given the way the robot is designed and how the code for the correction is written, we expected the error in the X position to be smaller. We expected this because the last correction made before the robot finishes a full square is a correction in X. Because of this, we thought that the final value in X would be closer to reality than the value in Y.

Section 4:

The error in the robot's odometer is too large for large distances therefore cannot be tolerable if we ran 5 times the distance of the 3x3 square. If we are talking of a 15x15 square, a small error in theta will be much more noticeable as well as the slip of the wheel. The small errors will accumulate and result in much bigger errors. If we are talking about doing 5 laps of the 3x3 square, the errors in theta will have a bigger impact due to the number of the turns that have to be done. The theta will keep on getting inaccurate which will make the X and Y too inaccurate to be reliable. In both cases, as the robot travels, the error grows proportional to the distance. There are a lot of other factors that cause error in the readings. Some of these factors are: where the weight is distributed in the robot, how fast the wheels turn, where the sensors are placed, the lighting in the area etc... Given these examples, if they are enough to make the robot move, then the error is not linear. On the other hand, if we can say that these factors are not important enough, we can say that, because the travel error should grow linearly, then the error is linear.

Section 5:

A software solution to reduce the slip of the robot's wheels would be to reduce the acceleration and speed. If the acceleration is too high, the robot might skid across the surface every time it starts either moving forward or turning. By reducing the speed, we can prevent the robot to slip if the surface is uneven at a certain area. If the robot goes too fast, then there would be less traction to prevent the slip whereas if it is slow, then it will stay more on route even if there is a slippery patch of track.

If we put two light sensors on the robot, one way of correcting the angle would be by putting the sensors right next to each other at the exact same distance from the wheels in the Y coordinate. Then, for example, we could take both inputs of the sensors and if one sensor detects the line before the other, then that sensor is in front of the other. The robot should then reduce the speed of the wheel on the side of the sensor that reads the line first. The time and speed reductions would be determined automatically by comparing the time difference between the two readings.

To correct the angle when the robot has only one sensor, we would have to measure the time it takes for the sensor to detect a certain number of lines in a row. This should be compared to the time the robot should take if it was going straight at the speed it is set. Then, with a few calculations, we would know by how much the angle is off, but we wouldn't know in which direction to correct it if the odometer does not report the error. To figure out in which direction the odometer should correct itself (whether to add or subtract degrees), we would have to wait until the robot has made a 90 degree turn and then measure the time it takes to detect the next line. We would then compare that time value to the expected time value for that line crossing. If the measured time is smaller than the expected, that means the robot is too close to the line and so the correction should add degrees to the odometer (assuming the robot turns to the

right to make the square). If the measured time is bigger, then subtract the correction to the odometer.