# ME 425 Post Lab Report 2

Demir Demirel, Mert Gunduc, Selcuk Ertug

***Abstract*— This report include implementation of Two Wheeled Robot made with Lego Mindstorms™ EV3 and discussion on the test results gathered with MATLAB.**

## I. INTRODUCTION

In this lab we were required to construct a two wheeled robot. The aim of this group project was to observe and gather information on the motion of the robot which we provided after implementing our MATLAB code to gain access to the robots motor control. We were required to drive, observe and plot the received information about how robots linear and circular motion changed due to different speeds of each motor, with the help of a sensor mounted under the constructed vehicle. Rest of the report we will be explaining the construction, electrical connections and the programming of the robot and how did the change in speed affected our desired trajectory.

## II. PROCEDURE

In this lab, we have experienced implementation of differential-drive robot model. To achieve this, we again used MATLAB environement to give necessary commands to our mobile robot that we constructed from LEGO pieces. We managed to give two types of motion to our robot. During the motion of the robot, we collected the position and orientation data to plot to see our results.

First we introduced our robot with a circular motion. In order to give the circular motion, one of the two motors must be stopped and the other one should keep its constant motion. So we stopped the left motor and activated the right one and calculated robots odometry equations in order to this, we used the encoder values of the right motor and we modeled the motion of the robot in discrete manner

$$x(k+1) = x(k) + D_c cos(\Theta(k) + \frac{\Delta\Theta}{2}) \tag{1}$$

$$y(k+1) = y(k) + D_c sin(\Theta(k) + \frac{\Delta\Theta}{2}) \tag{2}$$

$$\Theta(k+1) = \Theta(k) + \Delta\Theta \tag{3}$$

Equations in (1)(2)(3) shows the odometry equations that we used in our robots motion. Dc is the distance traveled by the center of the wheels and $\Delta\Theta$ is the change in the angle for specified time period. We calculated our next step position and orientation values from these inputs. Dc and $\Delta\Theta$ can be calculated in terms of the distance traveled by the right and left wheels.

$$D_r = 2\pi R\frac{\Delta E_r}{360} \tag{4}$$

$$D_l = 2\pi R\frac{\Delta E_l}{360} \tag{5}$$

$$D_c = \frac{D_r + D_l}{2} \tag{6}$$

$$\Delta\Theta = \frac{D_r - D_l}{L} \tag{7}$$

Our odometry estimation equations appeared on MATLAB as shown in the second figure. We implemented this code and kept the outcomes of the functions in an array. Then we plotted those arrays in order to visualize our data coming from the encoders and checked if it is properly managing its circular motion.

We initiated position and the angle of the robot to zero and utilized the IR sensor to control the robot from remote beacon controller. We gave the distance between the motors (L), 9.7 cm and radius of the circular motion (R) is 2.1 cm.

Second, we introduced the motor to its linear motion. Again we used the same calculated odometry equations but this time we actuated the both motors. We kept our initilizations and we declared the same L and R values. Further more, we planned to use the linear motion for better idea. We obtained a A3 paper and a pen and we attached the pen to the robot and let the robot draw our desired trajectory sketch on an A3 paper. After we plotted the data that we obtained from the encoder of the motors, we were able to see the same image both at the plot figure and in our paper.

## III. RESULT

In this lab we observed the trajectory of the two wheeled robot with MATLAB plots. We try to plot trajectory with circular movement and forward movement. As seen in the figures there is slipping in the movements. For example we can observe the different lines on each circle in Figure: III. In the figure above we decide the *L* variable as 9.1cm . Since all calculations are done with the mm we couldn't obtain the graph that we need. After that step we couldn't recognize what is the problem and try to change sampling time as 1 second. But it also has polygonal shapes as seen in Figure:III. Since we did not convince with the result we change the sampling time as 0.001 second and observe lots of circle instead of one circle that we expect.Figure:III After that point we realize our *L* value is wrong and we change the value with the corresponding mm. In the figure III we see the trajectory as we expect.

The trajectory lines are not met with each other and we can conclude that, the slipping is one of the result of there is no such a control algorithm.

Fig. 1. Circular Movement t=0.1s

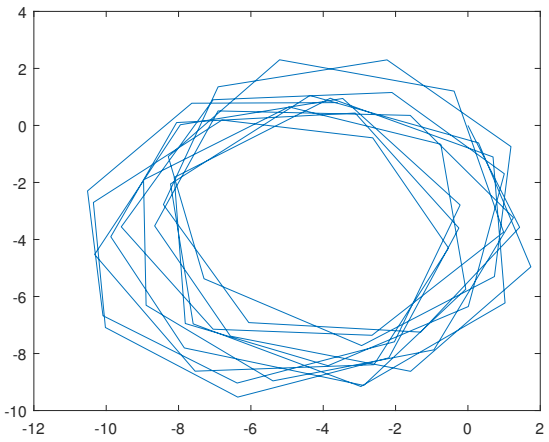



Fig. 2. Circular Movement t=1s

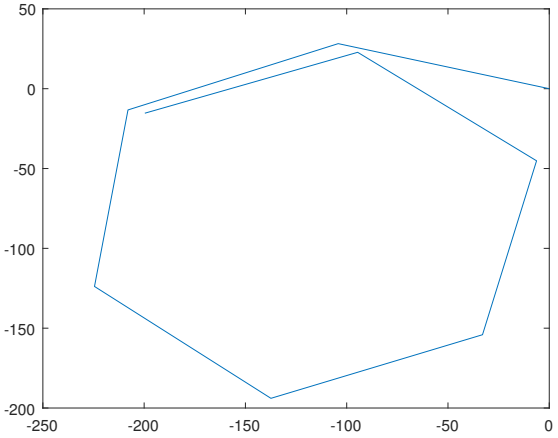
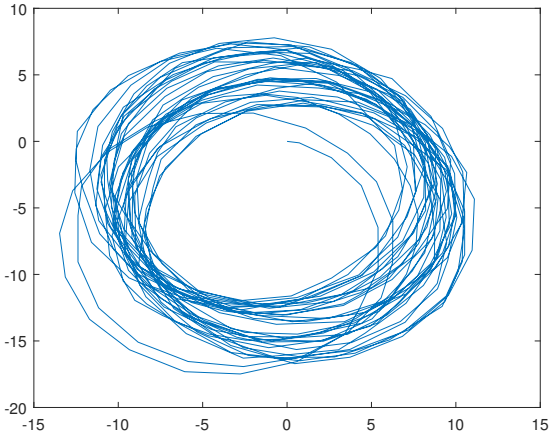


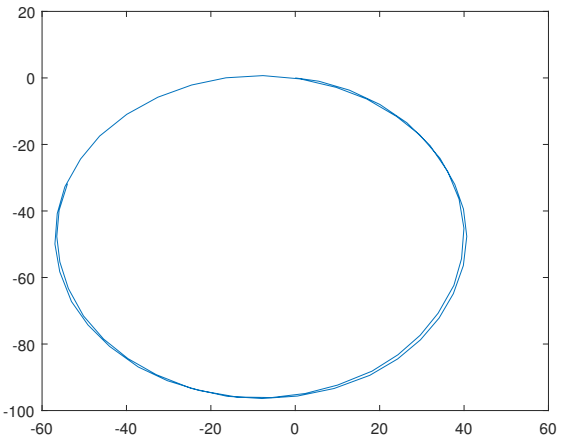

Fig. 3. Circular Movement t=0.001




## A. CONCLUSION

The two wheeled differential drive robot experimental platform LEGO MINDSTORMS EV3 is described in this lab report. The data gathered from actuators, sensors and control hardware are reviewed in terms of graphs that represent location of the robot. The theoretical expectations was not satisfied well. There are environmental constraints which prevent the accurate motion like friction. Beside environmental constraints, missing of a control algorithm also affect the real motion. Also measurements of the wheel base and wheel diameter has a small effect on the estimation observed as a result of the experiment.

Fig. 4. Circular Movement t=0.1




## IV. INDIVIDUALS

### A. *Selcuk Ertug*

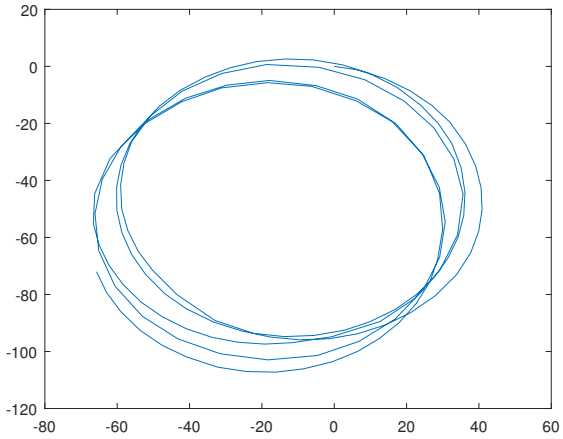
We could have gotten more precise values by using more sensitive tools for calculating the distance between wheels. Also; during the circular motion, due to the friction between the surface of the table and the robots stopped tyre, the distance between 2 wheels have changed in time. We could have prevented it by using a locking mechanism to fix the wheels to the axle. Major deflection in our linear motion was caused by angled surface forcing the car to pull in the direction of the gravity. To compare our MATLAB data for reliability, we have altered the design of the robot and managed to fix a pen to get the drawing of its motion on an A3 paper. However, we werent able to do so in the first try because, the center of the mass of the car shifted towards the back that lead to loss of traction and spinning of the wheels. We could save some time by anticipating such action and take precaution beforehand.

### B. *Demir Demirel*

In this lab we were tried to plot the trajectory of the robot with the data that we gather from encoders of the motors. From the motor encoders we can only obtain the rotation data. This rotation data helps us to find out distance that traveled of each tire. This distance is obtain from equations of (4) and (5). And we write the code with the equations of (1),(2),(3). At the lab we are asked to made circular movement at first. The graphs **??**, III and III are representing circular movements with different sampling times that we

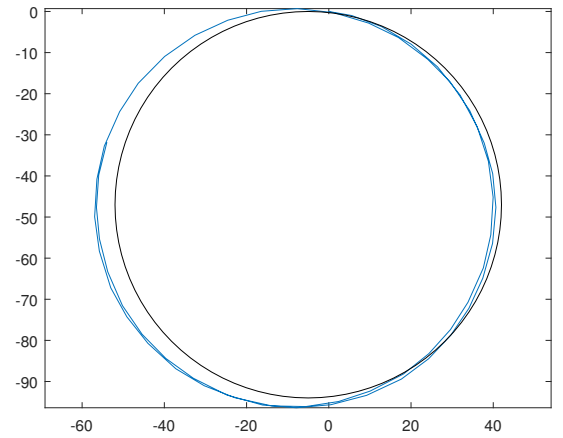
obtain. We write the sampling times in the label above the graphs. While sampling time gets smaller, the trajectory lines become more frequent and the error become more. When

Fig. 5. Circular Movement t=0.1



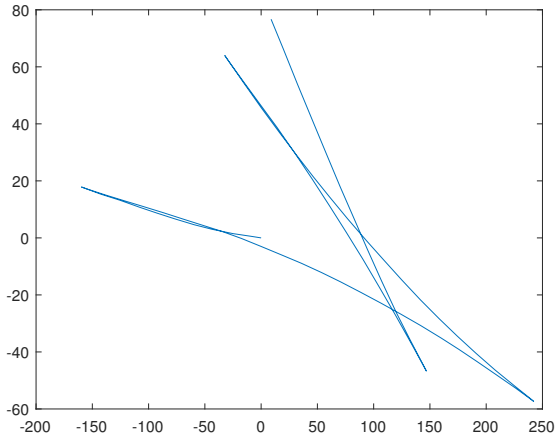

we compare the figure III and IV-B we can observe the difference between constant speed and with different speeds at same graph. As it can be observed when we change the speed the error gets larger for an instance and after that the trajectory lines draw similar circles with same radius. And it show us, the change in the rotation just affect the movement as moving a bit more to the front. Beside the plotting just a trajectory, we combine it with the desired trajectory in the figure IV-B. And it could be observed that there is a difference between them. This difference might be related with measure the wheel base wrong. Because the wheel base affect the radius of the plot directly we might obtain wrong radius. After circular movement we were expected

Fig. 6. Circular Movement




to made linear movements. We have only two motion for linear motion; forward and backward but as seen in the figure IV-B motion is not linear, we observe some zig-zags. The reason of that movement could be there is no controller in this implementation and we couldn't guarantee ideal linear movements. For extra work we change the two wheeled robot

Fig. 7. Linear Movement t=0.1




a bit as can be seen in the figure below. And put a pen between two wheel since there is a centre for the trajectory. In other words the trajectory lines are obtain from that point as well. When we put the pen we can see the real trajectory in paper and could compare with the odometry estimation plot. In following Figures we can see that, robot gives us

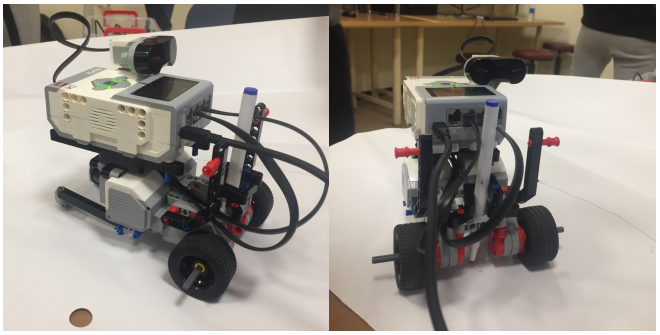


Fig. 8. Modified Two Wheeled RobotFig. 9. Modified Two Wheeled Robot

very similar trajectory in both plot and paper.

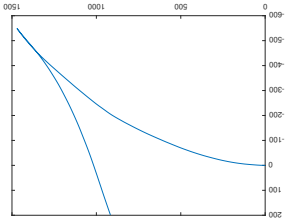


Fig. 10. Modified Two Wheeled Robot MATLAB Plot

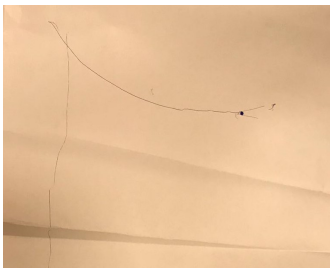


Fig. 11. Modified Two Wheeled Robot Real Life

To sum up this lab shows us, hard coded movement commands sometimes wouldn't work. We need to apply control algorithms instead of hard coded implementations to

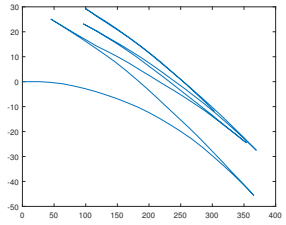


Fig. 12. Modified Two Wheeled Robot MATLAB Plot

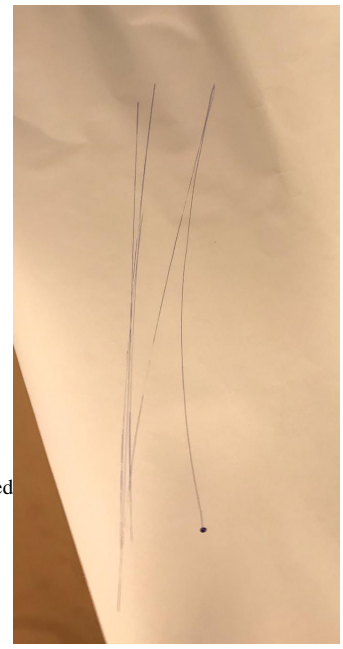


Fig. 13. Modified Two Wheeled Robot Real Life

make our trajectory more consistent that we aim for. With the exercises that we did also we have chance to see effects of speed and the sampling time to make correct estimations.

### C. Mert Gunduc

For this lab assignment we implemented differeantial-drive robot model and we managed to mobilize the robot in two type of motion, linear and circular motion. We calculated the odometry equations to understand the second step of the robots trajectory. We were able to plot and see the results in a Matlab figure. For linear motion we tried to have a staright line and for the circuar motion we wanted the trajectory to be proper circle. But, we observed some errors in both motion type like, our straight line and proper circle lines were deviated from each other. After changing the motor speeds and initating the sampling time as 0.001 second, we started to observe the perfect circle and a straight line. Despite of not having a control over motors and friction due to environmental causes, we managed to get our encoder data properly and observed the robots trajectory. Furthermore, we tried to observe our linear motion on a A3 paper in order to check whater the plot is correct. To do this we attached a pen to the robot which changed robots center of mass, we arranged our mobile robot again to mobilze it stabilly.

## APPENDIX

*Initialization*

```
clear all;
mylego = legoev3;
myirsensor = irSensor(mylego);
beep(mylego);
button = readBeaconButton(myirsensor,1);
motorleft = motor(mylego,'B');
motorright = motor(mylego,'C');
motorleft.Speed = 50;
% Start the motor
l=97;
r=21;
dt=0.1;
q=[  ];
x=[  ];
y=[  ];
q=[q 0];
x=[x 0];
y=[y 0];
start(motorleft);
start(motorright);
while ¬readButton(mylego, 'up')
    button = readBeaconButton(myirsensor,1);
    if button == 1
        motorleft.Speed=50;
        motorright.Speed=50;
    else
        motorright.Speed=-50;
        motorleft.Speed=-50;
    end
    resetRotation(motorleft);
    resetRotation(motorright);
    pause(dt);
    el = double(readRotation(motorleft));
    er = double(readRotation(motorright));

    dr= (2*pi*r*er)/360;
    dl= (2*pi*r*el)/360;
    dc= (dr+dl)/2;
    dq= (dr-dl)/l;
    q= q(end)+dq;
    x= [x (x(end)+dc*cos(q+dq/2))];
    y= [y (y(end)+dc*sin(q+dq/2))];

end
stop(motorleft);
stop(motorright);

plot(x,y);
```

```
clear all;
mylego = legoev3;
myirsensor = irSensor(mylego);
beep(mylego);

motorleft = motor(mylego,'B');
motorright = motor(mylego,'C');

% Start the motor
l=97;
r=21;
dt=0.1;
q=[  ];
x=[  ];
y=[  ];
q=[q 0];
x=[x 0];
y=[y 0];
start(motorleft);
start(motorright);
while ¬readButton(mylego, 'up')
    button = readBeaconButton(myirsensor,1);
    if button == 1
        motorleft.Speed = 50;
        motorright.Speed = 50;
    elseif button == 2
        motorleft.Speed = -50;
        motorright.Speed = -50;
```

```
    elseif button == 3
        motorleft.Speed = 50;
        motorright.Speed = -50;
    elseif button == 4
        motorleft.Speed = -50;
        motorright.Speed = 50;
    end
    resetRotation(motorleft);
    resetRotation(motorright);
    pause(dt);
    el = double(readRotation(motorleft));
    er = double(readRotation(motorright));

    dr= (2*pi*r*er)/360;
    dl= (2*pi*r*el)/360;
    dc= (dr+dl)/2;
    dq= (dr-dl)/l;
    q= q(end)+dq;
    x= [x (x(end)+dc*cos(q+dq/2))];
    y= [y (y(end)+dc*sin(q+dq/2))];

end
stop(motorleft);
stop(motorright);
plot(x,y);
```