# QUADRUPEDAL ROBOT DESIGN AND CONTROL

NOVEMBER 18, 2021

**Submitted to: Prof. Dr. Kemalettin Erbatur**

**Name of Student: Moses Chuka Ebere**
**Student Number: 31491**
**Course: Autonomous Mobile Robotics (ME 525) Lab**

## TABLE OF CONTENTS

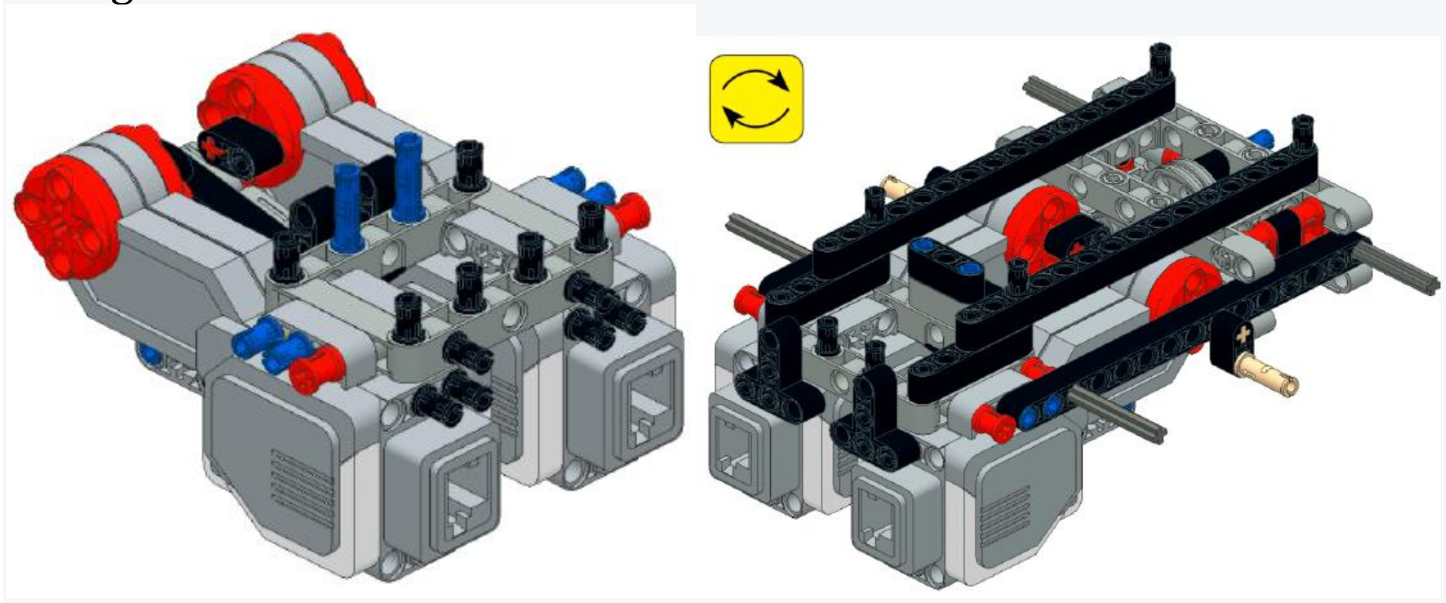
## INTRODUCTION

This lab involves the construction of a quadrupedal robot followed by the implementation of controls mechanisms using MATLAB to move the robot in different direction.

The major aim of this lab is to explore the locomotion of a four-legged robot. In this case, the robot is built using parts from the EV3 Mindstorm toolkit.

## PROCEDURE

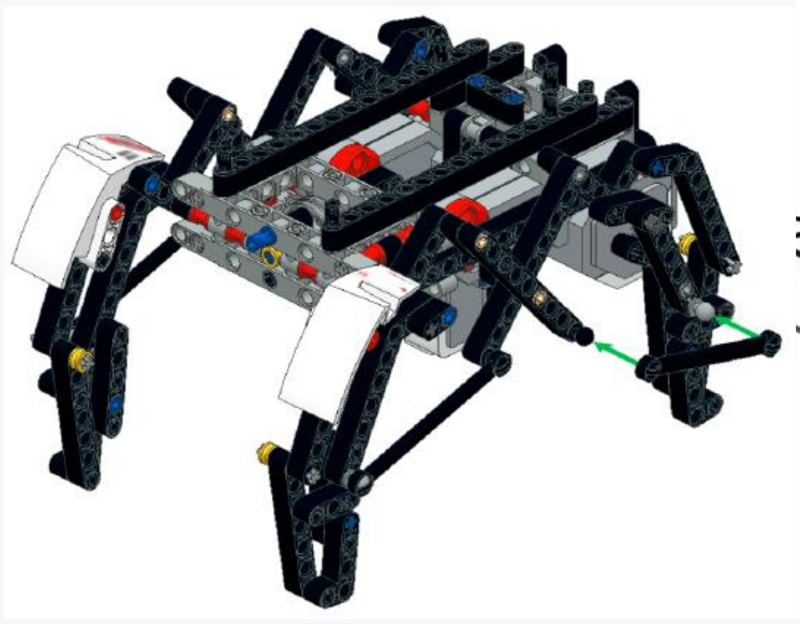
Firstly, the robot is assembled using the necessary parts as shown in the figures below:

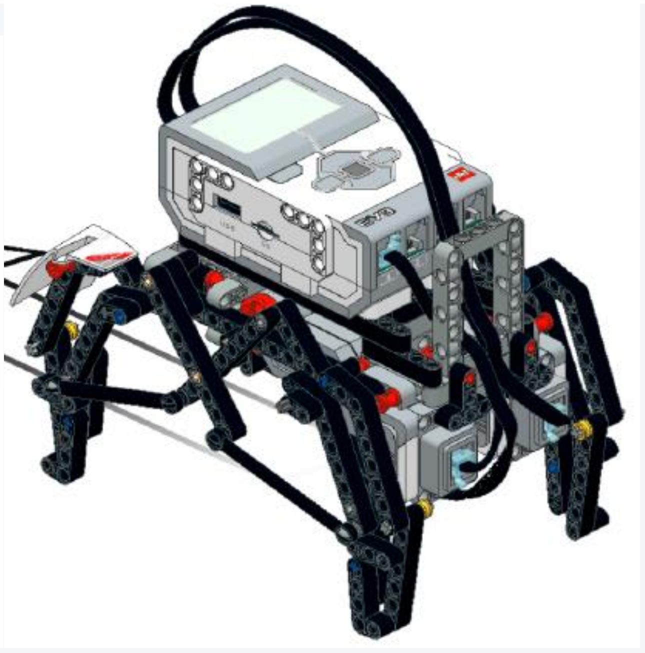


The major components used are:

- Two large motors
- One infrared sensor
- The EV3 Brick
- One touch sensor
- A remote infrared beacon

MATLAB is the underpinning software for this application.





To examine the locomotion of the robot some programs are written in MATLAB. Prior to this, a USB connection is established between the robot and the PC on which the MATLAB code is written.

## MATLAB Code

To ensure that we obtain a fluid walking pattern, the robot's motors are calibrated. As seen below, the touch sensor is instrumental in the calibration process.

```
Motor B = Left motor;
Motor C = Right motor;
Encoder B = Left motor's encoder;
Encoder C = Right motor's encoder;

if Touch Sensor is pressed then
    Run Motor B with 10 percent speed;
    Run Motor C with 20 percent speed;
else
    Stop both motors;
end

while Touch Sensor is NOT pressed do
    Run Motor B with 40 percent speed;
end
Stop Motor B;
Reset Encoder B;
while Encoder B > -72 degree do
    Run Motor B with -50 percent speed;
end
Stop Motor B;

while Touch Sensor is NOT pressed do
    Run Motor C with 40 percent speed;
end
Stop Motor C;
Reset Encoder C;
while Encoder C > -72 degree do
    Run Motor C with -50 percent speed;
end
Stop Motor C;
```

In the main script called remoteControlledMotion, the motors and IR sensors are properly defined.

- Arrays are created for the left and right motors to store the necessary data that will be obtain from the walking trajectory of the robot.
- A counter is created to facilitate storage of data in the aforementioned arrays.
- The calibration function is called to ensure that the robot is ready for the different motions.
- A while loop is used to make the robot walk for some time for proper data collection.
- Within the while loop, forward, backward, leftward, and rightward motions are performed using some lines of code.
- To avoid running all motion types at once, the beacon channel is defined and, with the remote control, if statements are used to properly separate each motion type.

Forward motion: After a simple test, it was determined that the robot proceeds forward with negative encoder speeds. Therefore, for direct forward motion, the speeds of the left and right motors are set at the same negative value. Results are obtained for different speeds.

Backward motion: For this, the speeds of the left and right motors are set at the same position value. Results are obtained for different speeds.

Leftward motion: The speed of the right motor is set significantly higher than that of the left motor (with the same signs (negative)).

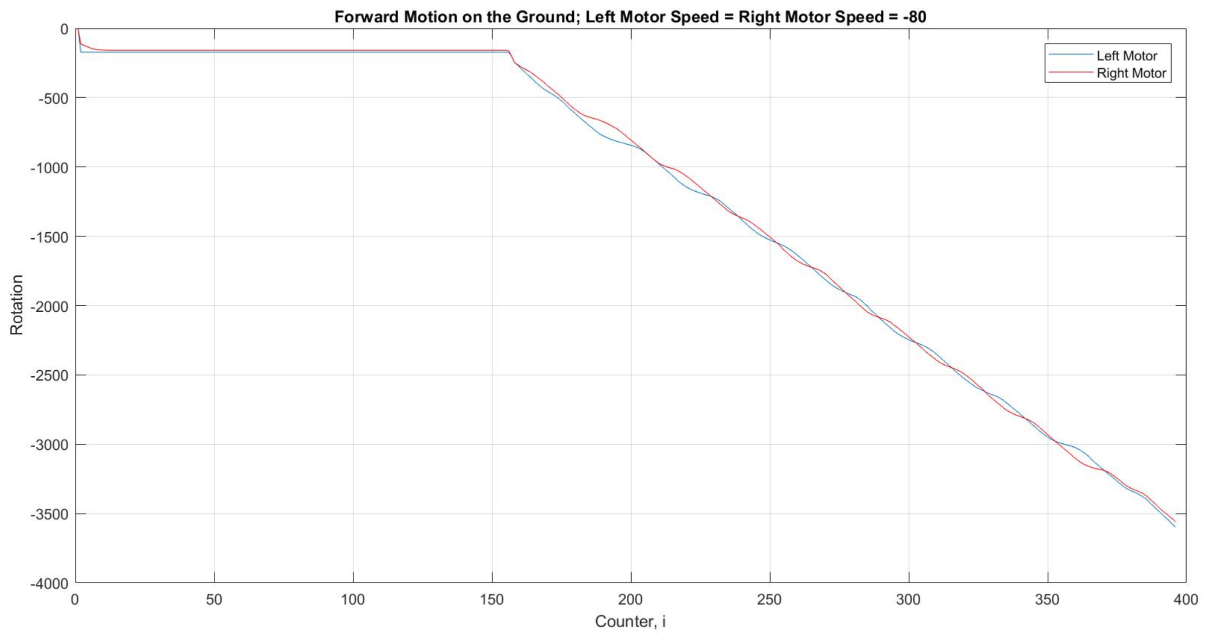
Rightward motion: The speed of the left motor is set significantly higher than that of the right motor (with the same sign (negative)).

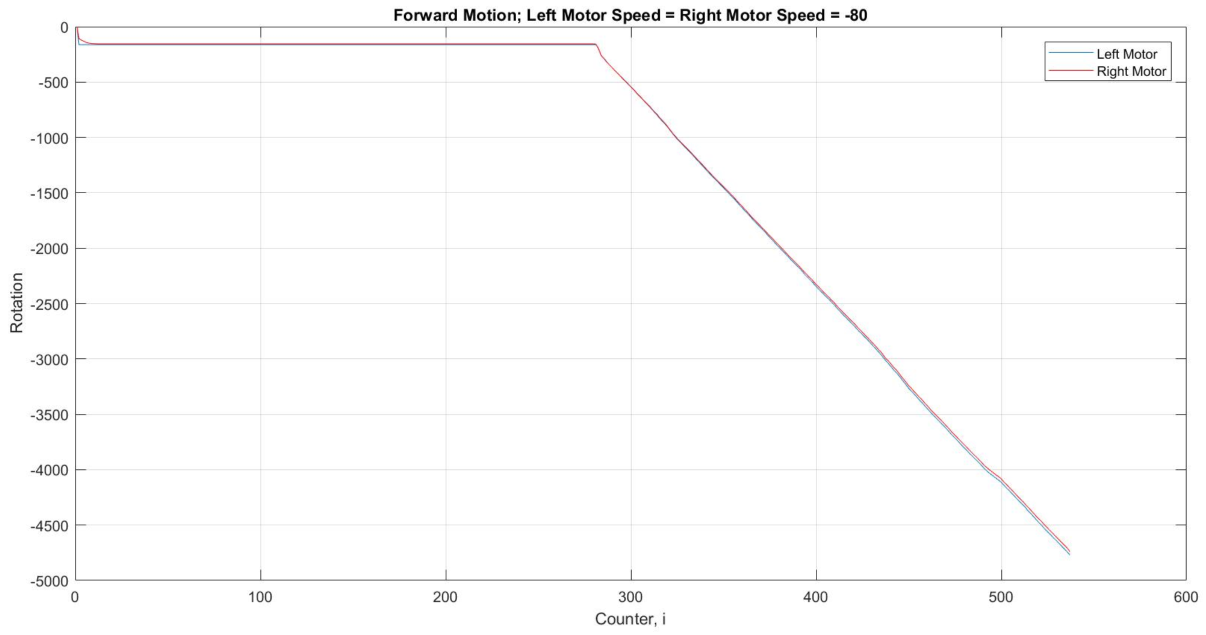
- After every run within the while loop, the encoder speeds for the left and right motors are stored in the earlier defined array before the counter is incremented for the next run.
- Finally, the results are plotted.

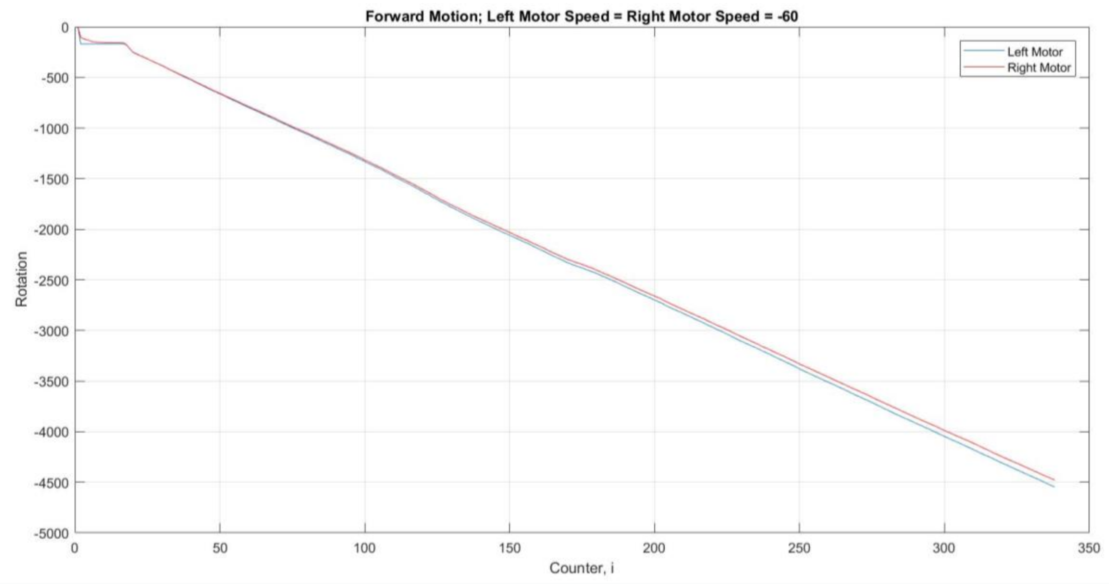
## RESULTS

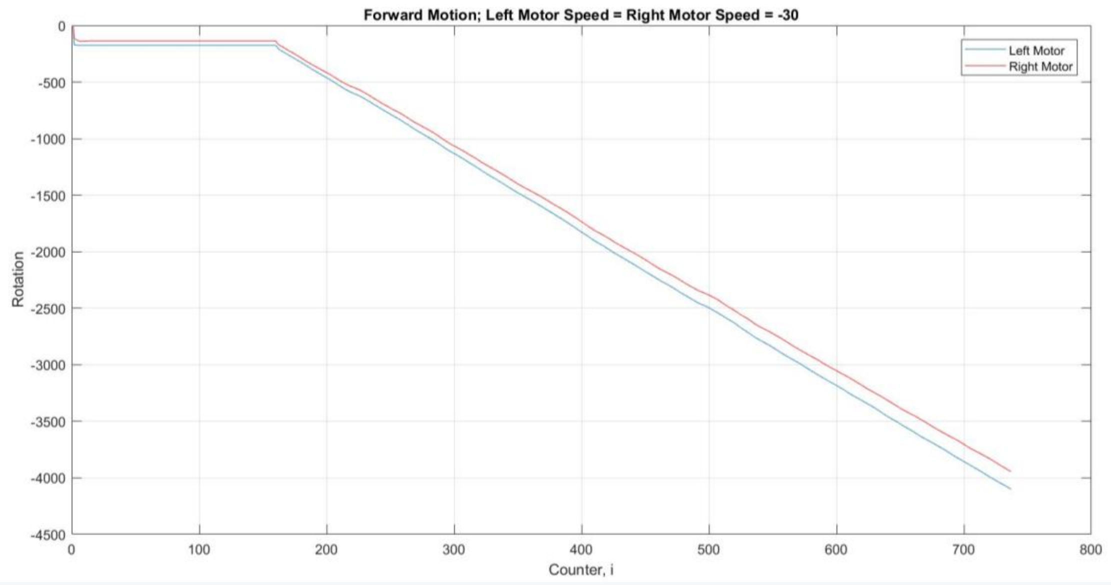
The following results were obtained from the implementation of the MATLAB code on the quadrupedal robot.

Forward Motion:

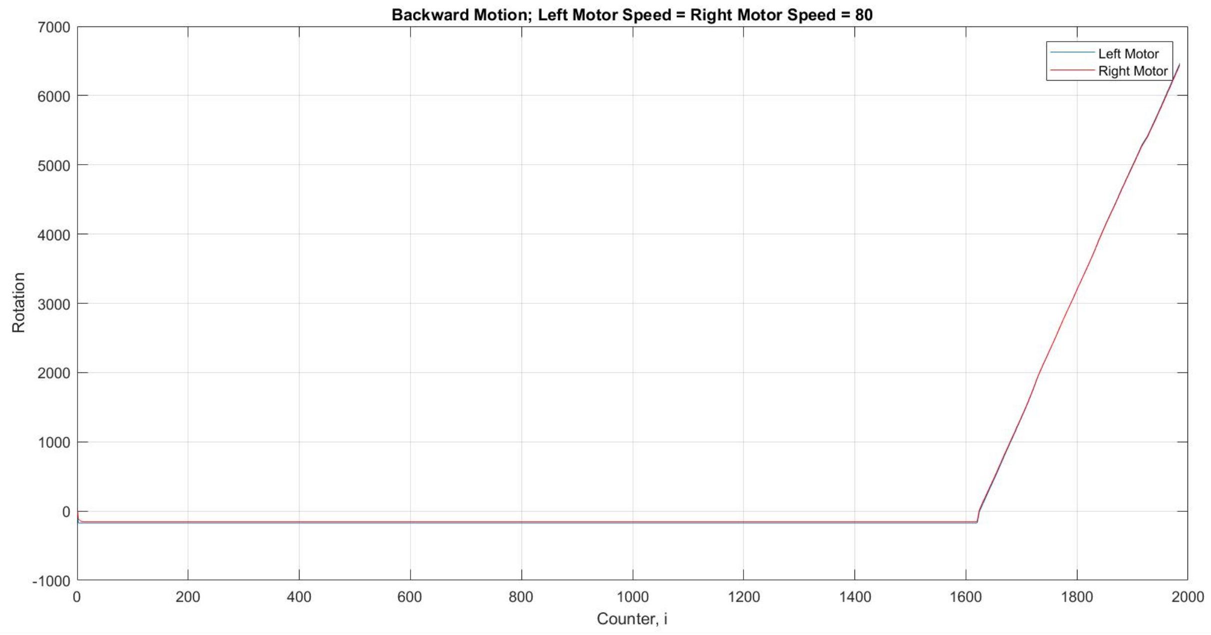




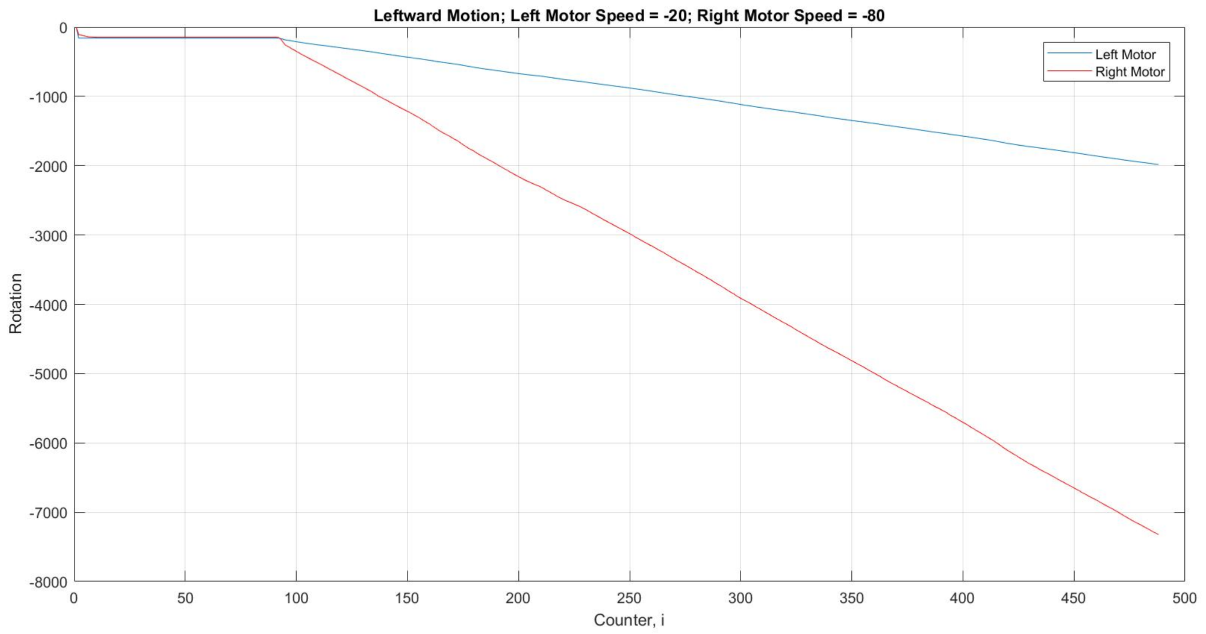




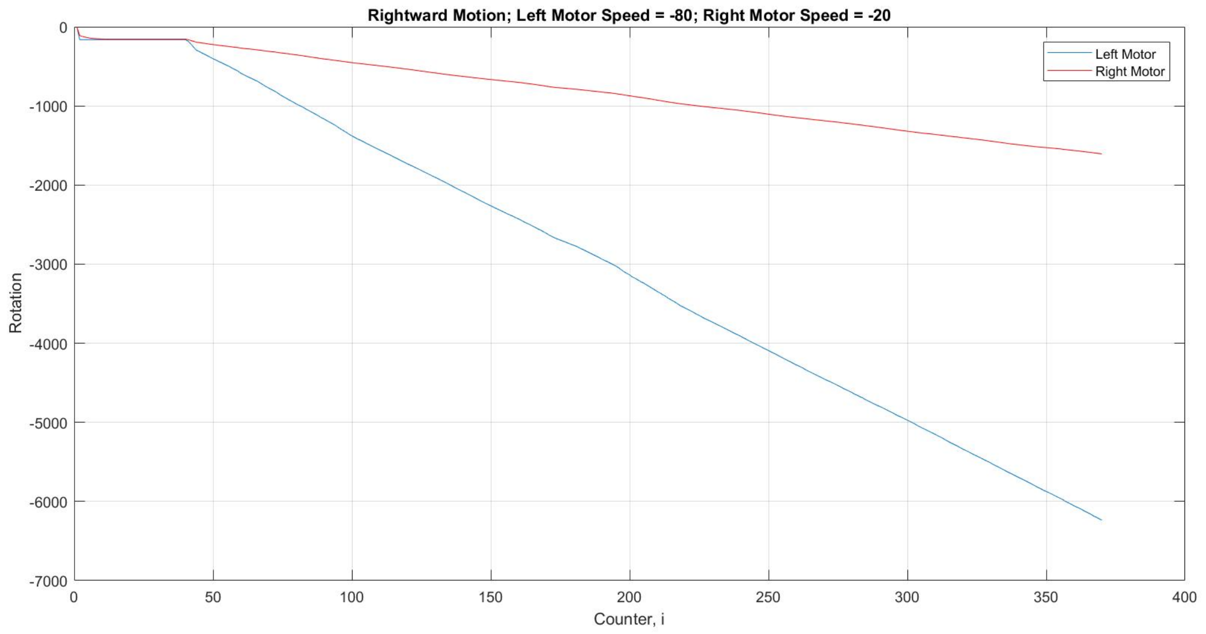
## Backward Motion:



## Leftward Motion:



## Rightward Motion:



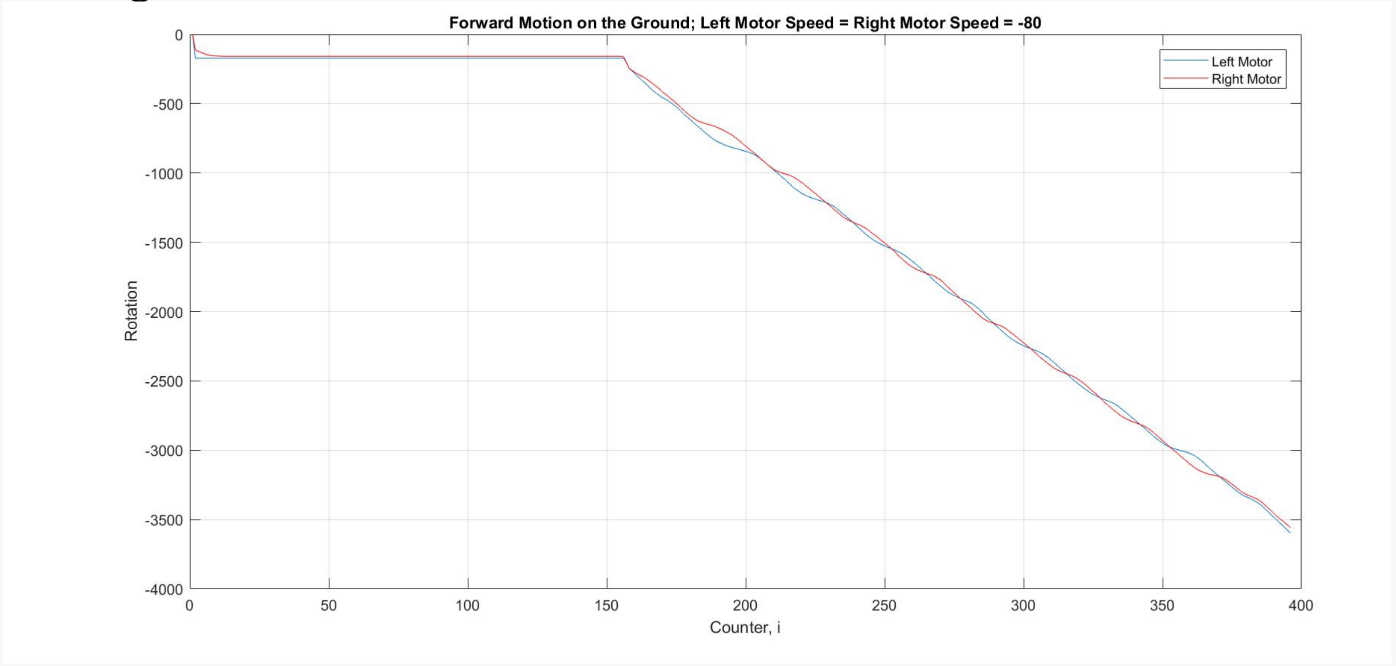
# DISCUSSION

## Static and Dynamic Stability

The robot ensures static stability because while at rest, all four legs are in full contact with the ground, and we know from general knowledge of stability that for a legged robot to be statically stable, the body weight must be supported by at least three legs.

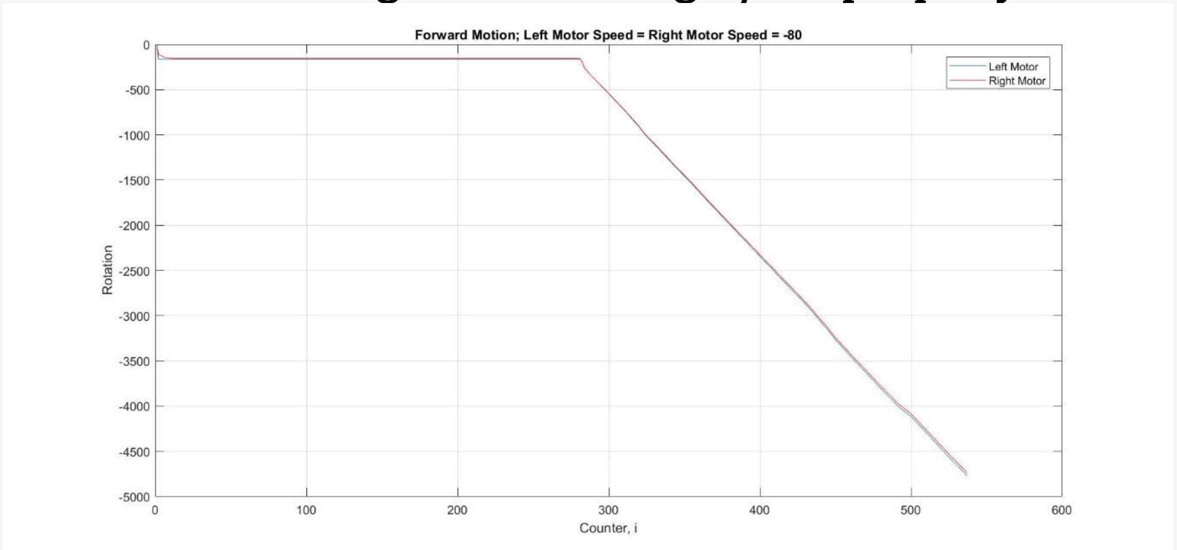
- Also, we can observe the center of mass projection of the robot. We'll inevitably notice while the robot is at rest or moving slowly, the center of mass projection is always within the support polygon. However, though this is safe, it is inefficient.

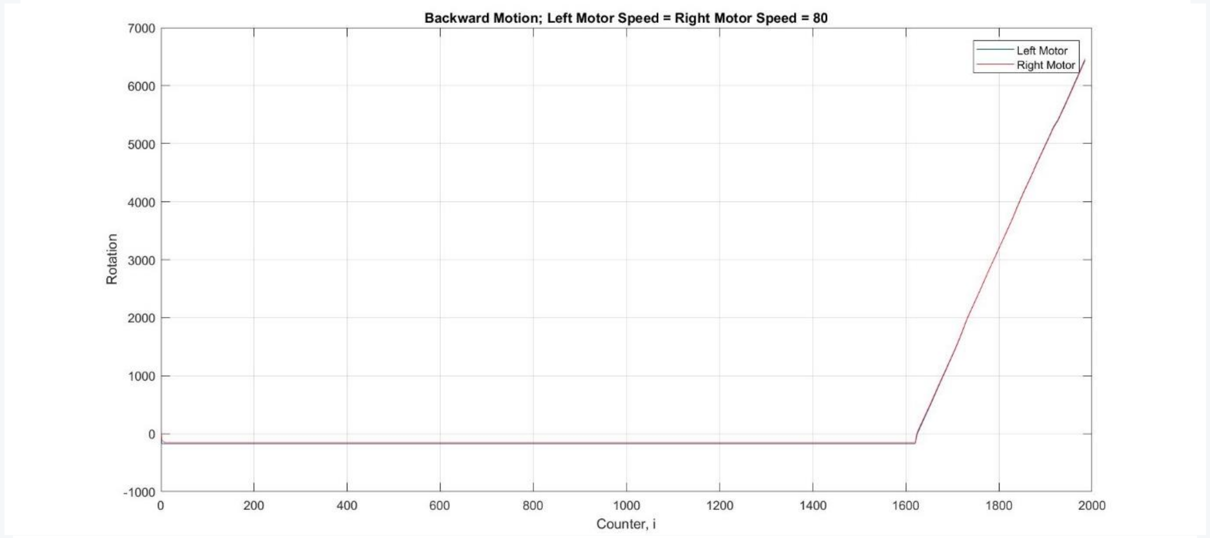
- The robot ensures dynamic stability by continuously moving.
- Due to the structure of the robot, its center of mass projection momentarily falls outside its support polygon during the swing of a pair of legs on one side; however, since the robot keeps moving, dynamic stability is maintained.
- We can observe this in the forward motion plot for the robot on the ground below:

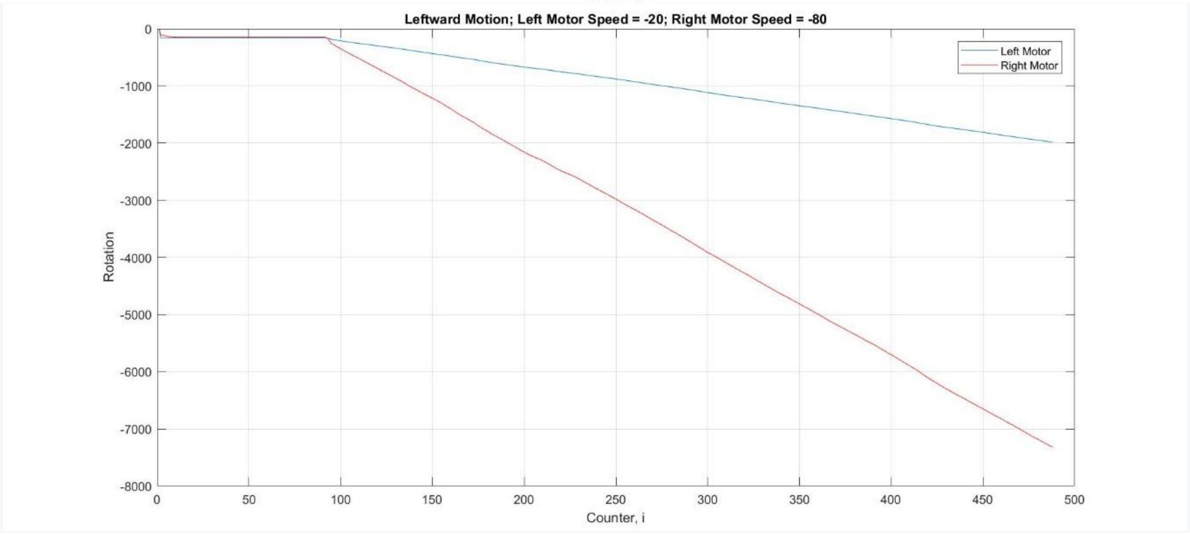


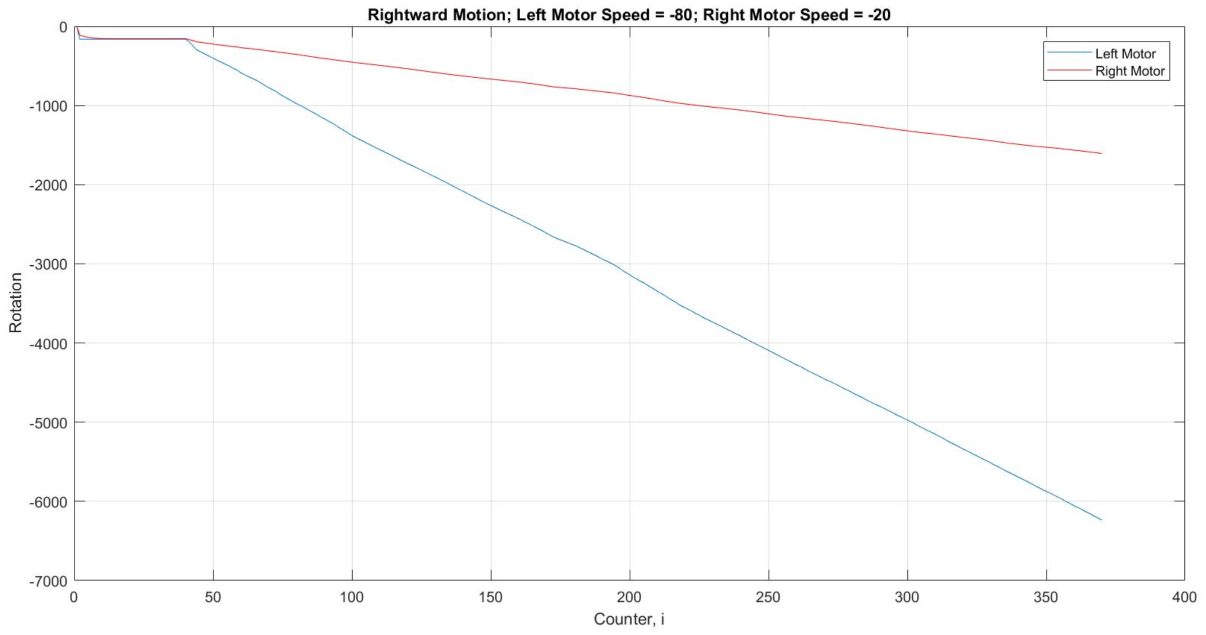
Side note: I also notice that there seems to be a slight oscillatory motion as observed in the above plot. My thoughts on this is that the robot also maintains its dynamic stability by the slight changes in the rotational speed of the motors during ground contact and during swings.

## Can the robot move straight and turn right/left properly?







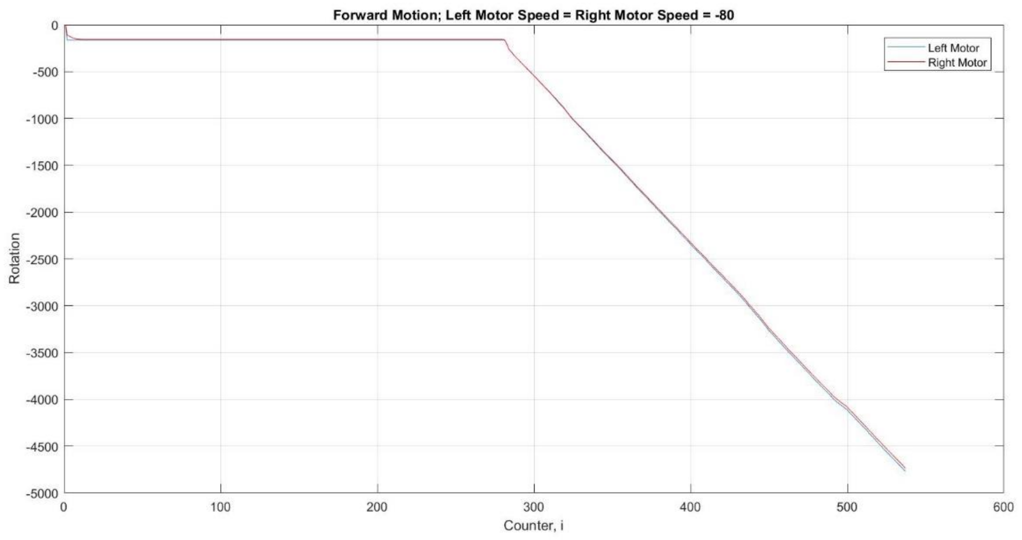


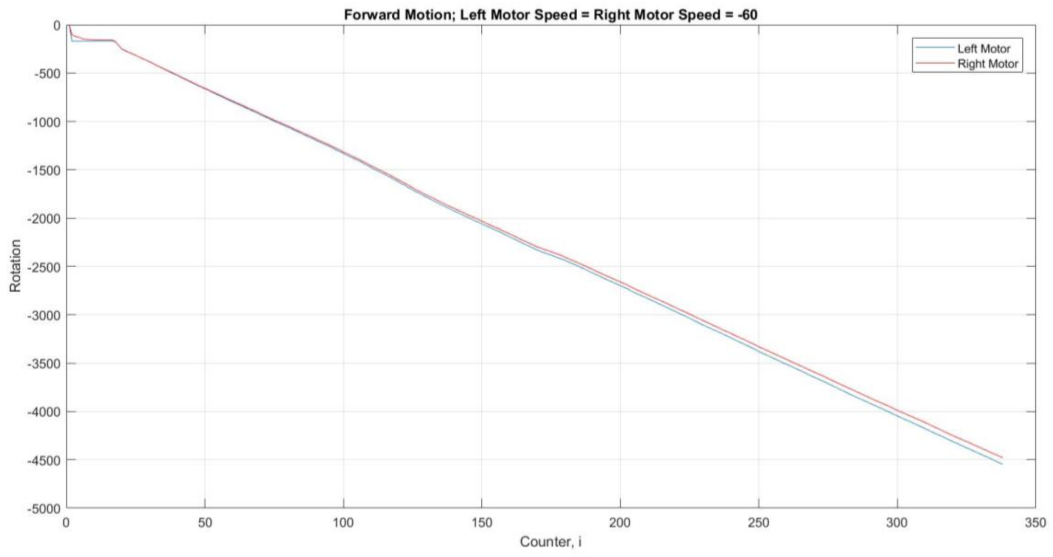
From the above plots of the encoder readings of the motors, I can say that the robot does move:

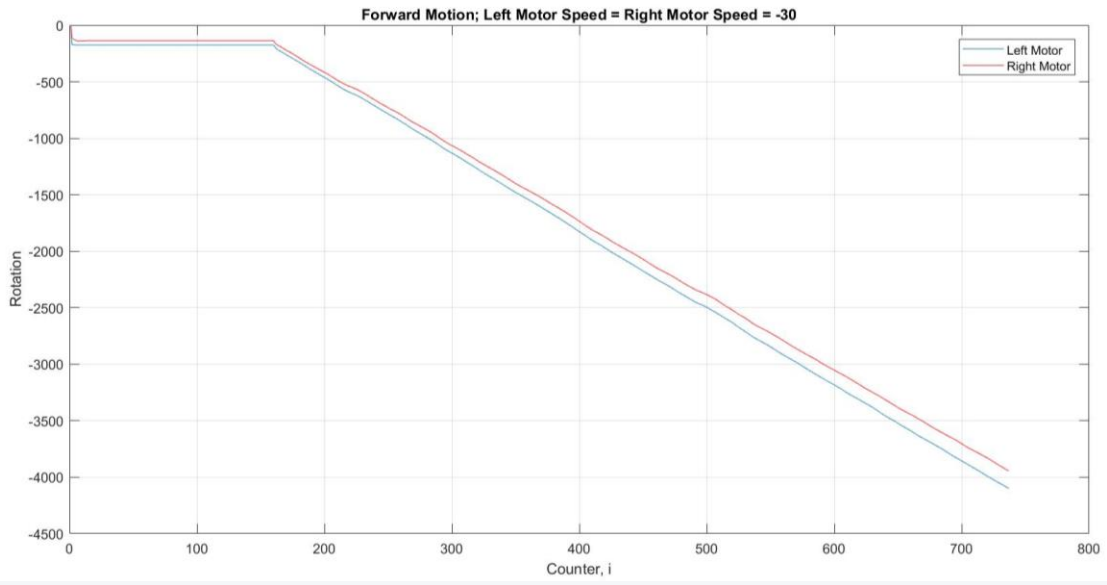
a. Straight properly: This is because the trajectory of the left and right motors (for forward and backward motion) advances in an approximate linear fashion almost with the same slope throughout. However, there are minor deviations here and there.
b. Right properly: As expected, the slope of the right encoder values is much smaller than that of the left encoder values. Also, the lines are straight and there are not gaps. Therefore, the robot moves right properly.
c. Left properly: As expected, the slope of the right encoder values is much larger than that of the left encoder values. Also, the lines are straight are there are not gaps. Therefore, the robot moves left properly.

## <u>Does the robot speed have an impact on accurate motion?</u>

The forward locomotion of the robot was examined at different speeds: -30, -60, and -80 to determine the effect of speed on motion. The following plots were obtained:







Looking at the above results, we immediately observe that speed has a direct impact on motion. As the speed reduces, the line plots for the left and right encoders start deviating. This will definitely hamper accurate motion at low speeds.

- So, if we want more accurate motion using the same motors, it'll be best to run them at high speeds.

## CONCLUSION

This lab includes a condensed study on the locomotion of legged robots. In this case, a four-legged (or quadrupedal) robot was used. A key principle utilized during this work was the differential drive principle. This was particularly implemented while making the robot turn. In general, the outcome of this experiment is a desirable one barring some inevitable discrepancies due to hardware constraints. For example, the deviation in left and right encoder plots as the speed is lowered could be a result of the motors used. Wear and tear could also be a culprit in this situation.

## APPENDIX

**Calibration Code**

```
function calibration(mylego)

% Create connections for the left (B) and right (C) motors.
MotorB = motor(mylego, 'B');
MotorC = motor(mylego, 'C');

% Read the rotation of each motor.
EncoderB = readRotation(MotorB);
EncoderC = readRotation(MotorC);

% Set up the Touch Sensor on input port 1
mytouchsensor = touchSensor(mylego, 1);
% Read the state of the touch sensor.
% Such that 1 - pressed; 0 - not pressed
pressed = readTouch(mytouchsensor);

if readTouch(mytouchsensor) == 1
    % Set the speed of left motor to 10%
    MotorB.Speed = 10;
    % Run the left motor
    start(MotorB);
    % Set the speed of right motor to 20%
    MotorC.Speed = 20;
    % Run the right motor
    start(MotorC);
else
    stop(MotorB);
    stop(MotorC);
end

while readTouch(mytouchsensor) == 0
    % Set the speed of left motor to 40%
    MotorB.Speed = 40;
    % Run the left motor
    start(MotorB);
end

% Stop the left motor
stop(MotorB);
% Reset the left Encoder
resetRotation(MotorB);

while readRotation(MotorB) > - 72
    % Set the speed of left motor to -50%
    MotorB.Speed = -50;
    % Run the left motor
    start(MotorB);
end
% Stop the left motor
stop(MotorB);

while readTouch(mytouchsensor) == 0
    % Set the speed of left motor to 40%
    MotorC.Speed = 40;
    % Run the left motor
    start(MotorC);
end
% Stop the right motor
stop(MotorC);
% Reset the right Encoder
resetRotation(MotorC);

while readRotation(MotorC) > - 72
    % Set the speed of right motor to -50%
    MotorC.Speed = -50;
    % Run the right motor
    start(MotorC);
end
% Stop the right motor
stop(MotorC);

end
```

### remoteControlledMotion Code

```
clear all
close all
clc

mylego = legoev3('usb');

% Set up infrared sensor on port 2
myirsensor = irSensor(mylego, 2);

Rightmotor = motor(mylego,'C');
Leftmotor = motor(mylego,'B');

resetRotation(Rightmotor);
resetRotation(Leftmotor);

Rightmotor.Speed = 0;
Leftmotor.Speed = 0;

Leftarray = [];
Rightarray = [];

Leftarray(1)= 0;
Rightarray(1) = 0;
```

```
calibration(mylego);
stop(Rightmotor);
stop(Leftmotor);

i=1;

while ~readButton(mylego, 'up')
    % Read the button from the beacon
    button = readBeaconButton(myirsensor,2);

    % Forward Motion
    if button == 1
        Rightmotor.Speed = -80;
        Leftmotor.Speed = -80;
        start(Rightmotor);
        start(Leftmotor);

    end

    % Backward Motion
    if button == 2
        Rightmotor.Speed = 80;
        Leftmotor.Speed = 80;
        start(Rightmotor);
        start(Leftmotor);

    end

    % Rightward Motion
    if button == 3
        Rightmotor.Speed = -20;
        Leftmotor.Speed = -80;
        start(Rightmotor);
        start(Leftmotor);

    end

    % Leftward Motion
    if button == 4
        Rightmotor.Speed = -80;
        Leftmotor.Speed = -20;
        start(Rightmotor);
        start(Leftmotor);

    end

    % Read the Encoder values
    Leftarray(i+1)= readRotation(Leftmotor);
    Rightarray(i+1) = readRotation(Rightmotor);

    i=i+1;
end

 stop(Rightmotor);
 stop(Leftmotor);
```

**Code for Plotting the Required Graphs**

```
% Syntax for Plots
% -------------------------

figure
plot(Leftarray)
hold on
plot(Rightarray, 'r')
title('Forward Motion on the Ground; Left Motor Speed =
Right Motor Speed = -80')
ylabel('Rotation')
xlabel('Counter, i')
legend('Left Motor','Right Motor')
grid
```