**Abbreviated Lab Report**

Lab #2 Controlling the iRobot Create

Prepared by : Sean Lantto

Team Members: Sierra Portillo and Diana Quiroz

02/15/17

**1. Objective**

* Get familiar with the iRobot Create mobile robot platform;
* Learn how to collect data from the Create and display them in MATLAB;
* Learn how to control the motion of a Create.

**2. Observations and Discussions**

The iRobot create was initially placed atop a power supply to prevent any accidental driving. While on top, the robot was initialized, and it’s bump and cliff sensors were tested. Also during this time, its voltage and current draw were measured for 200 iterations, or about 10 seconds. After this the robot was programmed to drive a 1 meter square using t*ravelDist* and *turnAngle* commands.The robot appeared to drive in near perfect squares but was always slightly offset from its original starting position.

The robot was then programmed to perform the same task but using the *SetDriveWheelsCreate* function to command the drive motors. To ensure the robot traveled in the square, wheel encoder data had to be pulled to determine distance driven and angle turned. This was done using the *Read\_Create\_2* function as opposed to *AllSensorsReadRoomba,* which was too slow. Using the sensor data, and if statements embedded in a while loop, the robot drove in a near square.

A couple of safety precautions were added, such that if the cliff sensors were triggered all motion would stop, and the program would be ended. If the bump sensors were triggered, the robot would stop, reverse, and turn slightly to try a new direction, thus resetting the origin point of the square path.

Various wheel speeds were tried, and it was found that slower speed improved the accuracy to the prescribed square path.

**Answers to Deliverable Questions**

1) The create measures distance and angle using its wheel encoders

2) The robot comes close to its original position and heading, but is off by a few degrees each time.

3) The previously mentioned error could have to do with wheel speed, the faster the wheels are moving the more likely the wheels are to slip. Causing the robot to think it has traveled further or turned more than it has.

4) See figure in Appendix A. The plot appears to spiral outward. Visually, the robot appeared to move in a “square” shape but always was offset from its original position and heading a each “square” completion.

5) The average power consumption was 6 Watts at lower wheel speeds and 7 Watts at higher ones.

**3. Problems and Solutions**

A major problem encountered during the lab was the robot not turning when using *SetDriveWheelsCreate,* this was determined to be a coding error. In the while loop was a command to drive in a straight line, which overrode the turning command. This was fixed by moving the drive straight command to before the while loop. A lesser issue was the fact that the turn was never exactly 90 degrees. This was accounted for initially by giving a value lower than 90 degrees to stop the turn at, which worked to some degree. However a better fix to the issue was to slow the speed of the turn, giving the sensor read function a better chance of detecting 90 degrees in the turn.

**4. Learning**

I learned how to use the MATLAB toolbox for the iRobot Create, to read sensor data, and move the robot in a controlled manner.

**5. Comments and Suggestions**

To improve this lab, it would be interesting to attempt to use more than just the iRobot Create sensors, though I’m sure this is coming in the future labs. It would also be interesting to have some sort of competition between groups, like who can navigate a simple path fastest or with the least amount of bumps which could be identified by beeps.

**Appendix A – Figures and Tables**

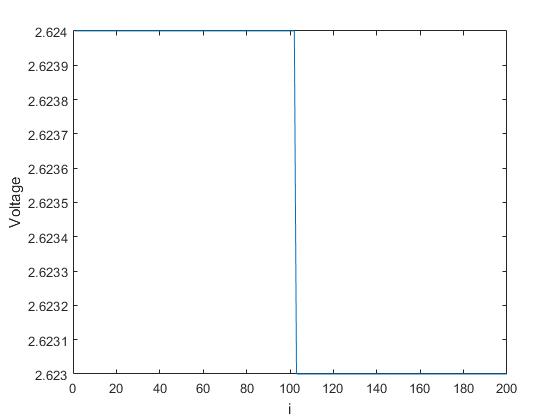
Figure 1: voltage use over 200 steps

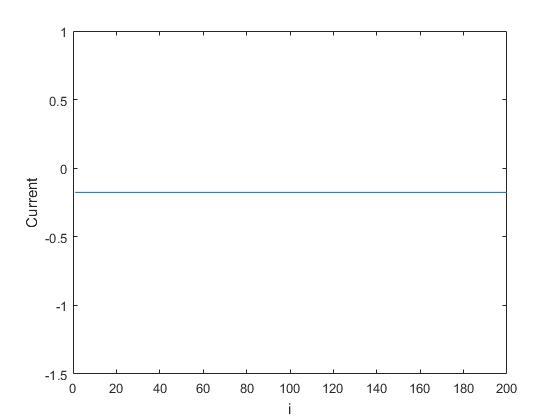
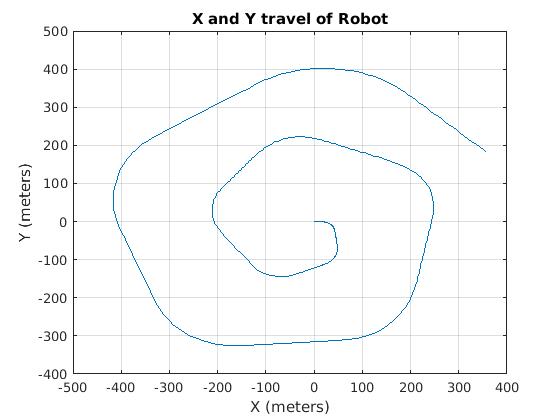
Figure 2: Batery current draw over 200 steps

Figure 3: Robot's X and Y position when commanded to run in a 1 meter square

**Appendix B – Source Code**

% Lab2

% Sean Lantto

% Sierra Portillo

% Diana L. Quiroz

clear

COM1=RoombaInit(1)

% [BumpRight, BumpLeft, BumpFront, Wall, virtWall, CliffLft, ...

% CliffRgt, CliffFrntLft, CliffFrntRgt, LeftCurrOver, RightCurrOver, ...

% DirtL, DirtR, ButtonPlay, ButtonAdv, Dist, Angle, ...

% Volts, Current, Temp, Charge, Capacity, pCharge] = AllSensorsReadRoomba(COM1)

%

%

% for i=1:200

% [Volts(i)]=BatteryVoltageRoomba(COM1);

% [Current(i)]=CurrentTesterRoomba(COM1);

% end

% figure;plot(1:200,Volts)

% xlabel('i');

% ylabel('Voltage');

%

% figure;plot(1:200,Current)

% xlabel('i');

% ylabel('Current');

SetLEDsRoomba(COM1, 3, 100,100)

BeepRoomba(COM1)

% for i=1:4

% travelDist(COM1, 0.25, 1)

% turnAngle(COM1,0.1,90)

% end

DistTot=0;

AngTot=0;

i=0;

SetDriveWheelsCreate(COM1, 0.5,0.5)

while 1

i=i+1;

fwrite(COM1,[142 0]);

pause(0.01);

[BumpRight, BumpLeft, BumpFront, Wall, virtWall, CliffLeft, CliffRight, CliffFrontLeft,CliffFrontRight, LeftCurrOver, RightCurrOver, ...

DirtL, DirtR,ButtonPlay,ButtonAv, Dist, Angle,CreateVolts,CreateCurrent,Temp,Charge,Capacity,pCharge] = Read\_Create\_2(COM1)

Dista(i)=Dist;

Anga(i)=Angle;

DistTot=DistTot+Dista(i);

AngTot=AngTot+Anga(i);

if DistTot>=1

SetDriveWheelsCreate(COM1,0,0)

SetDriveWheelsCreate(COM1,.1,-.1)

DistTot=0;

end

if AngTot>=90\*pi/180 || AngTot<=-90\*pi/180

SetDriveWheelsCreate(COM1,0,0)

SetDriveWheelsCreate(COM1,0.5,0.5)

AngTot=0;

end

if BumpLeft==1||BumpRight==1||BumpFront==1

SetDriveWheelsCreate(COM1, 0,0);

SetDriveWheelsCreate(COM1, -.5,-.5);

pause(0.5)

SetDriveWheelsCreate(COM1,.1,-.1)

pause(1)

SetDriveWheelsCreate(COM1,.5,.5)

DistTot=0;

AngTot=0;

end

if CliffFrontLeft==1||CliffFrontRight==1||CliffLeft==1||CliffRight==1

SetDriveWheelsCreate(COM1, 0,0);

break

end

Volta(i)=CreateVolts;

Amp(i)=CreateCurrent;

pause(0.05)

end

save('fastexptRun')