**Abbreviated Lab Report**

*Lab #5: Tilt Sensing and PID Control*

Prepared by: Sean Lantto

Team Members: I didn’t write their names down and I forgot

03/29/2017

# Objective

* Learn how to estimate tilt angles using accelerometers;
* Learn how to perform PID control on a mobile robot.

# Observation and Discussion

SMART\_RUN was modified to include equations 1 and 2, below, to estimate the pitch(θ) and roll(φ) angles. While running the modified code the SMART robot was rotated by hand to change the roll and pitch angles, the change in which can be seen in *figure 1* appendix A. To avoid damaging SMART, the robot was never rotated completely around, which can be seen in *figure 1*. The code was run again, but this time, SMART was placed on the ground to drive in a straight line. During the drive, there were no obstacle avoidance algorithms used, so when SMART approached an obstacle it was picked up and moved by hand, as can be seen by the sharp increase in pitch in *figure 2.*

SMART was modified to follow a 30-degree heading angle. Initially a proportional controller with respect to the error was used, to some success, as can be seen in *figure 3.*  A proportional derivative control was then implemented using the yaw rate gyro measurements as the numerical derivative. See *figure 4* for results from this run. With the PD control the robot started to oscillate its heading for no apparent reason part way through. A PI controller was then implemented and *figure 5* shows that despite the large overshoot, it settled very close to 30-degrees, relatively quickly. Finally, a PID controller was implemented. *Figure 6* demonstrates the results of the final tuned controller, unfortunately, it was unable to have zero overshoot, and perhaps more tuning would solve this issue.

**Deliverable answers**

1. Pitch is function of roll because the body frame rotates, so if roll was 90 degrees pitch will be in the same direction as yaw used to be.
2. Estimated pitch and roll when driving on a flat surface did not accurately reflect the observed pitch and roll, and the results seen in *figure 2,* could be assumed to be noise, or from vibrations on board.

|  |  |  |
| --- | --- | --- |
| **Rank** | **Controller** | **Approximate settling time(seconds)** |
| **1** | **Proportional** | **3** |
| **2** | **PI** | **7** |
| **3** | **PID** | **10** |
| **4** | **PD** | **12** |

# Problems and Solutions

The biggest issue encountered was running out of room when testing the controllers. While some sort of avoidance algorithm could have been used, the solution we chose was simply to pick up SMART and turn it around. Another issue was finding the proper tuning, which was fixed by trial and error until we deemed the performance acceptable

# Learning

I learned how to implement a PID controller in a physical system instead of in theoretical situations in other classes.

# Comments and Suggestions

# Appendix A: Figures and Table

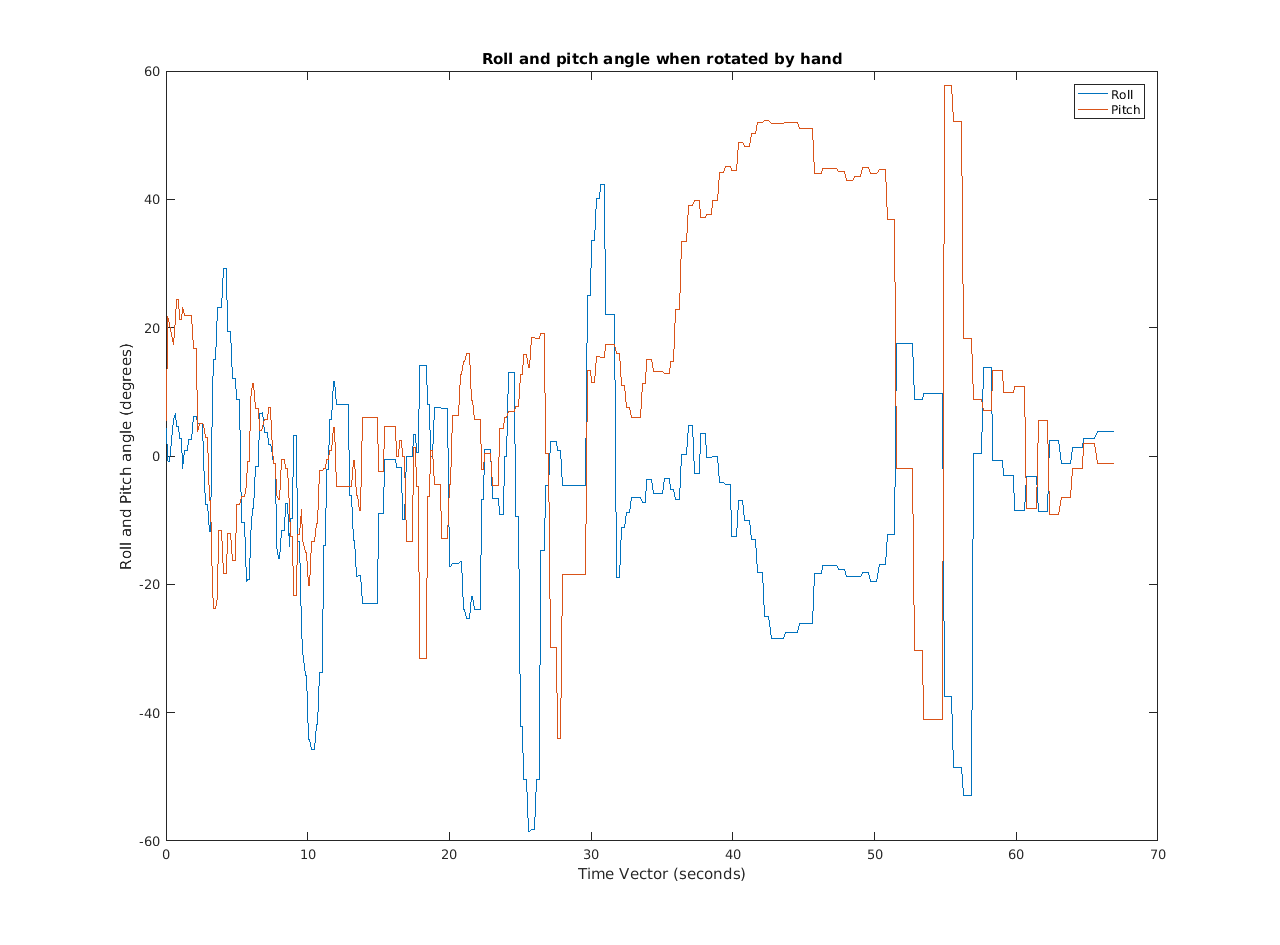


Figure : Roll and pitch of SMART when tilted by hand

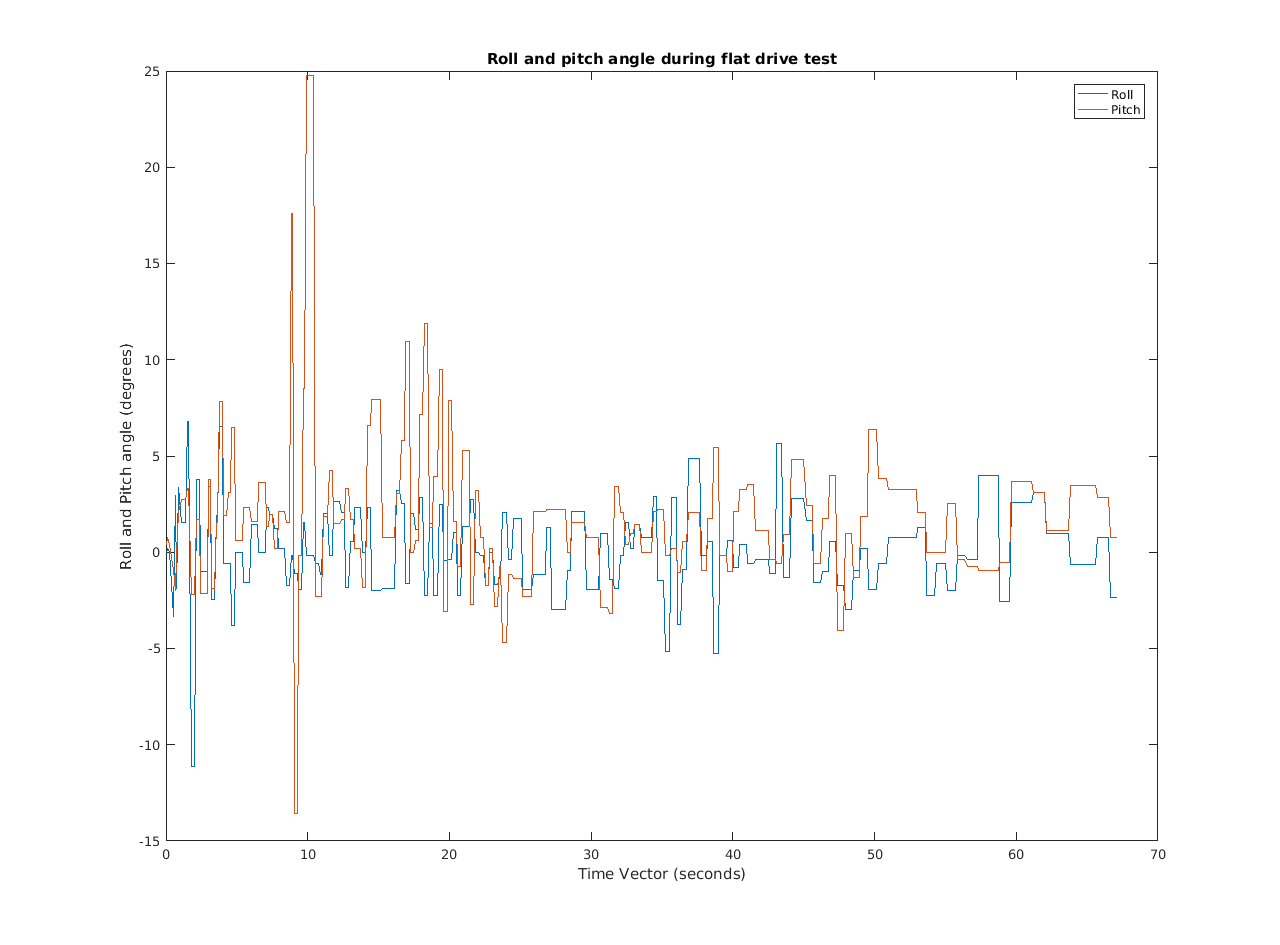


Figure : Roll and pitch while driving on the floor

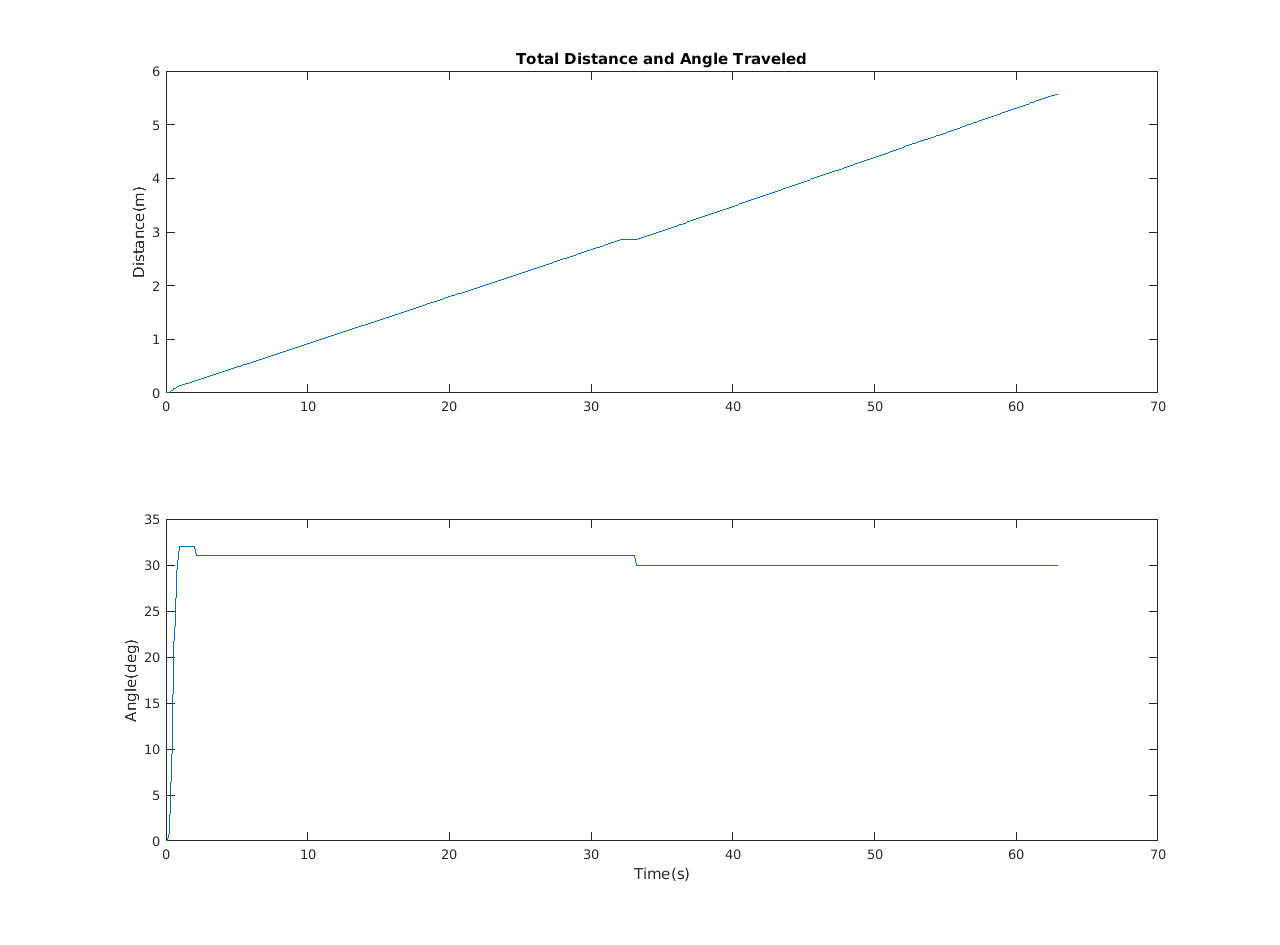


Figure : Distance travelled and total heading angle of SMART with a proportional controller

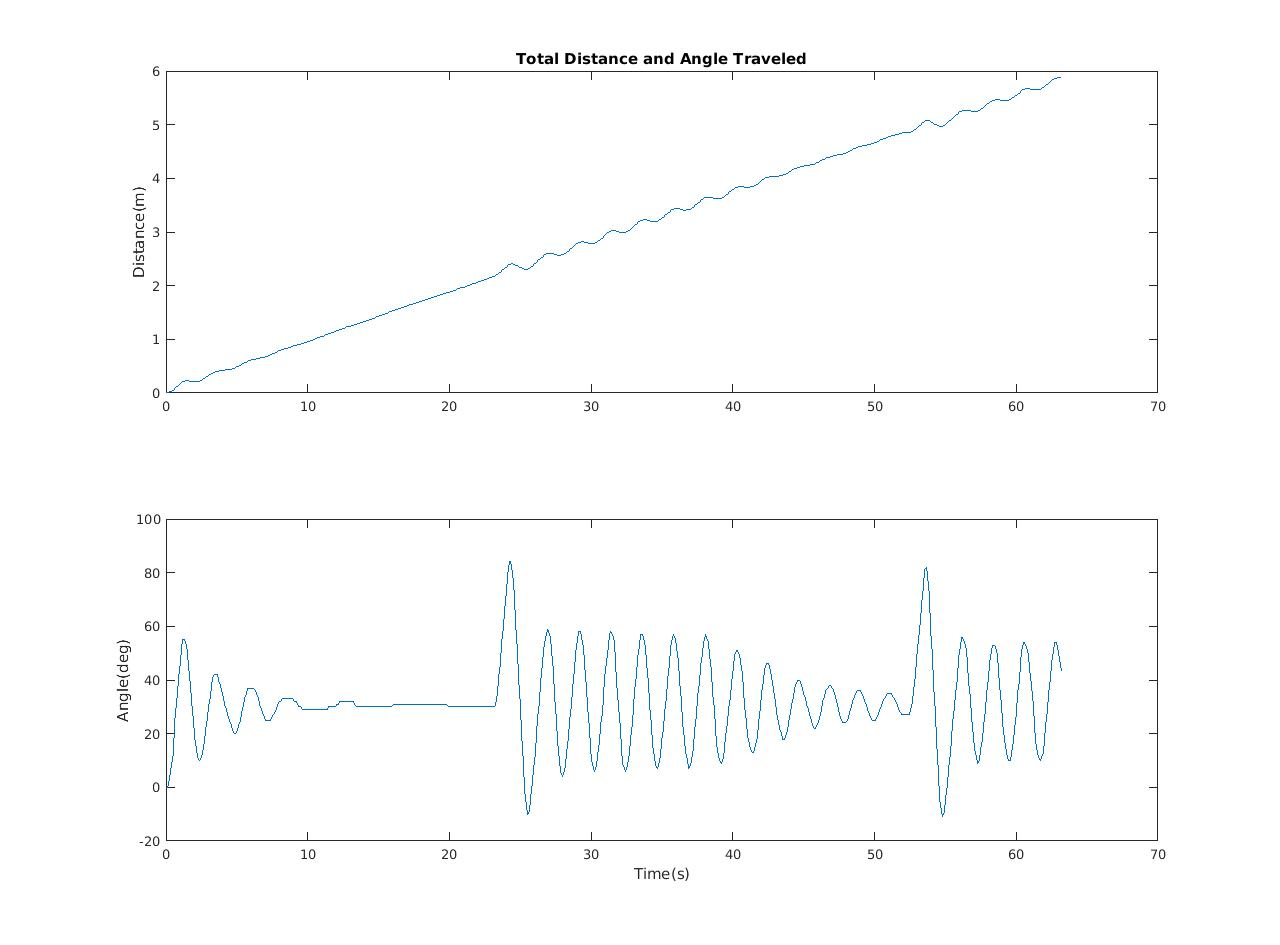


Figure :SMART distance travelled and total heading angle with a proportional derivative controller

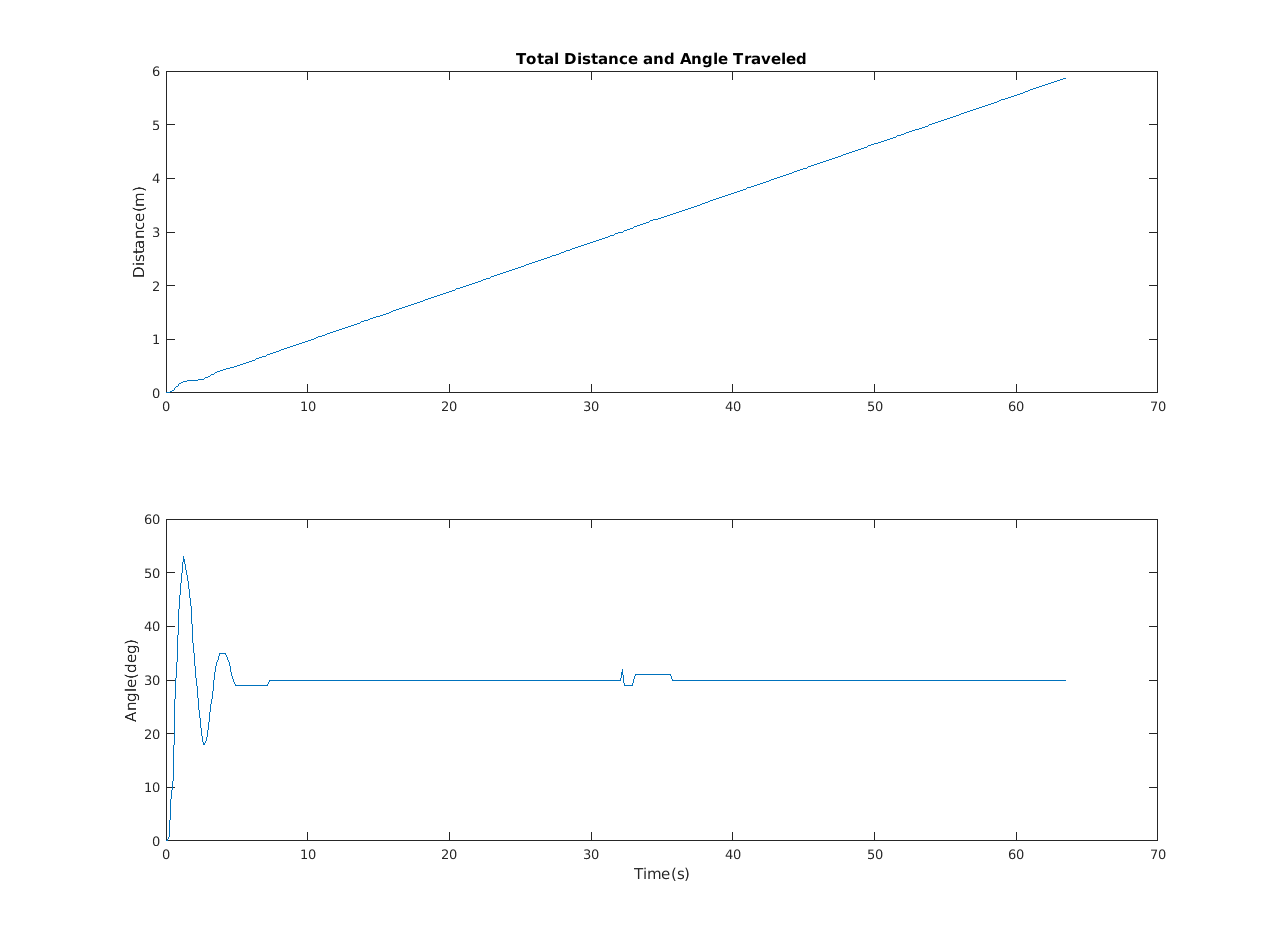


Figure :SMART distance travelled and total heading angle with a proportional integral controller

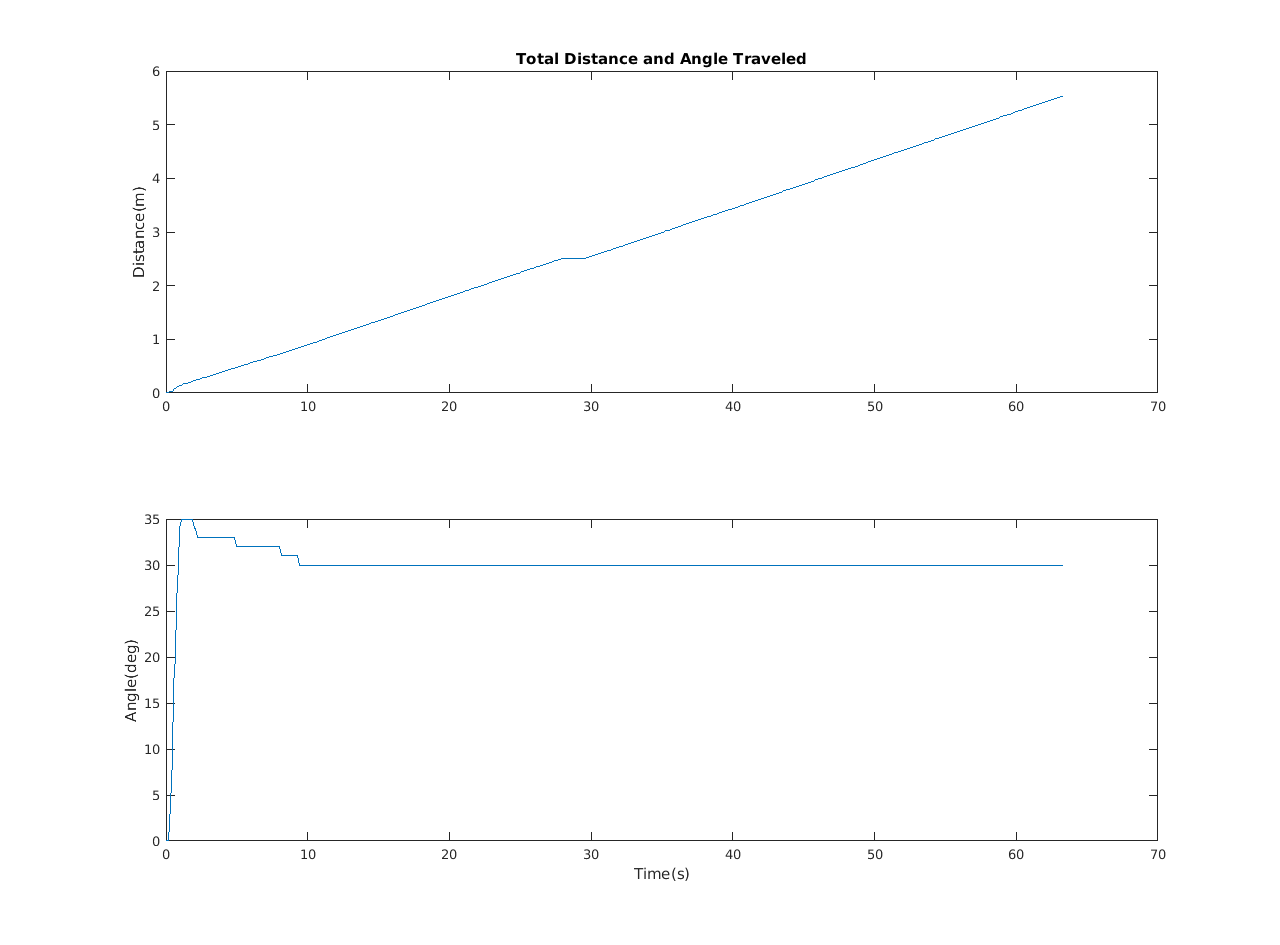


Figure :SMART distance travelled and total heading angle with a proportional integral derivative controller

# Appendix B: Source Code

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This is the main code for runing the SMART Robot %

% Author: Yu Gu %

% This is the version with Hukuyo lidar interface %

% The Kinect interface is removed %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all

close all

% Definitions

Ts\_Desired=0.1; % Desired sampling time

Ts=0.1; % sampling time is 0.1 second. It can be reduced

% slightly to offset other overhead in the loop

Tend=60; % Was 60 seconds;

Total\_Steps=Tend/Ts\_Desired; % The total number of time steps;

Create\_Full\_Speed=0.2; % The highest speed the robot can travel. (Max is 0.5m/s)

% Magnetometer Calibration Data

Mag\_A=[ 2.3750 0.2485 -0.2296

0 2.6714 -0.1862

0 0 2.5061]; % estimated shape of the soft iron effect

Mag\_c=[ 0.0067; 0.2667; 0.0473]; % estimated center of the hard iron effect

% Rate Gyro Biases

P\_Bias=0;

Q\_Bias=0;

R\_Bias=0;

% P\_Bias=-0.0019;

% Q\_Bias= 0.0053;

% R\_Bias=-0.0149;

% Define the main Data Structure

SD= struct( 'Index',zeros(1,Total\_Steps),... % SD stands for SMART Data, which stores all the robot data

'Time', zeros(1,Total\_Steps),... % The actual time at each time step (s)

'Time\_Diff', zeros(1,Total\_Steps),... % The time difference between two step (s)

'Delay',zeros(1,Total\_Steps),... % Delay needed between each time step. It indicate how much time avaliable for other compuations.

'Lidar\_Angle', zeros(682,Total\_Steps),... % Hukuyo Lidar Scan Angles

'Lidar\_Range', zeros(682,Total\_Steps),... % Hukuyo Lidar Scan Range

'Logger\_Counter', zeros(1,Total\_Steps),... % The Counter number for the SMART data logger

'Ax',zeros(1,Total\_Steps),... % Acceleration along the x-Axis (g)

'Ay',zeros(1,Total\_Steps),... % Acceleration along the y-Axis (g)

'Az',zeros(1,Total\_Steps),... % Acceleration along the z-Axis (g)

'P',zeros(1,Total\_Steps),... % Roll Rate (deg/s)

'Q',zeros(1,Total\_Steps),... % Pitch Rate (deg/s)

'R',zeros(1,Total\_Steps),... % Yaw Rate (deg/s)

'Mx',zeros(1,Total\_Steps),... % Magnetic strength along the x-axis (G)

'My',zeros(1,Total\_Steps),... % Magnetic strength along the y-axis (G)

'Mz',zeros(1,Total\_Steps),... % Magnetic strength along the z-axis (G)

'IMU\_T',zeros(1,Total\_Steps),... % IMU internal temperacture (C)

'Roll',zeros(1,Total\_Steps),... % Roll Angle (rad)

'Pitch',zeros(1,Total\_Steps),... % Pitch Angle (rad)

'Yaw',zeros(1,Total\_Steps),... % Yaw Angle (rad)

'Mag\_Heading',zeros(1,Total\_Steps),... % Magnetic Heading (rad)

'X',zeros(1,Total\_Steps),... % Robot X Position (m)

'Y',zeros(1,Total\_Steps),... % Robot Y Position (m)

'RF\_F',zeros(1,Total\_Steps),... % Front Range Finder (mm)

'RF\_FL',zeros(1,Total\_Steps),... % Front-Left Range Finder (mm)

'RF\_L',zeros(1,Total\_Steps),... % Left Range Finder (mm)

'RF\_B',zeros(1,Total\_Steps),... % Back Range Finder (mm)

'RF\_R',zeros(1,Total\_Steps),... % Right Range Finder (mm)

'RF\_FR',zeros(1,Total\_Steps),... % Front Right Range Finder (mm)

'Laser\_RF', zeros(1,Total\_Steps),... % Laser Range Finder (mm)

'Wall', zeros(1,Total\_Steps),... % Wall sensor of Create (0/1)

'VirtWall', zeros(1,Total\_Steps),... % Detect the Virtual Wall (0/1)

'Dist', zeros(1,Total\_Steps),... % Distance Traveled Since Last Call (m)

'TotalDist', zeros(1,Total\_Steps),... % Total Distance Traveled(m)

'Angle', zeros(1,Total\_Steps),... % Angle Traveled Since Last Call (rad)

'TotalAngle', zeros(1,Total\_Steps),... % Total Angle Traveled (rad)

'CreateVolts', zeros(1,Total\_Steps),... % Voltage of the Create Robot (rad)

'CreateCurrent', zeros(1,Total\_Steps)); % Current of the Create Robot (rad)

% Initialize the Serial Port for the Data Logger

S\_Logger=Init\_Logger('2'); % May be different on different computer

% Initialize the Serial Port for iRobot Create

S\_Create=RoombaInit(1);

% Initialize the Serial Port for LightWare Laser Rangefinder

S\_LightWare=Init\_LightWare('3');

% Initialize the Serial Port for Hukuyo Lidar

% S\_Hokuyo=Init\_Hokuyo(7); % initalize the Lidar

% pause(0.1);

% fscanf(S\_Hokuyo)

% rangescan=zeros(1, 682);

% for i=1:Total\_Steps

% SD.Lidar\_Angle(:,i)=(-120:240/682:120-240/682)\*pi/180; % an Array of all the scan angles

% end

% fprintf(S\_Hokuyo,'GD0044072500'); % request a scan

flushinput(S\_Logger); % Flush the data logger serial port

flushinput(S\_Create); % Flush the iRobot Create serial port

fwrite(S\_Create, [142 0]); % Request all sensor data from Create

BeepRoomba(S\_Create); % Make a Beeping Sound

pause(0.1);

phi = zeros(600,1);

theta = phi;

e=theta;

% scatter3(SD.Time,phi,theta)

% hold on

% Starting the main loop

for i=1:Total\_Steps

tic

SD.Index(i)=i;

% Acquire data from the sensor interface board

if i==1

[SD.Logger\_Counter(i), SD.Ax(i), SD.Ay(i), SD.Az(i), Raw\_P, Raw\_Q, Raw\_R, Raw\_Mx, Raw\_My, Raw\_Mz, SD.IMU\_T(i), A2D\_Ch1, A2D\_Ch2, SD.RF\_F(i), SD.RF\_FL(i), SD.RF\_L(i), SD.RF\_B(i), SD.RF\_R(i), SD.RF\_FR(i)] = Read\_Logger\_2(S\_Logger,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

else

[SD.Logger\_Counter(i), SD.Ax(i), SD.Ay(i), SD.Az(i), Raw\_P, Raw\_Q, Raw\_R, Raw\_Mx, Raw\_My, Raw\_Mz, SD.IMU\_T(i), A2D\_Ch1, A2D\_Ch2, SD.RF\_F(i), SD.RF\_FL(i), SD.RF\_L(i), SD.RF\_B(i), SD.RF\_R(i), SD.RF\_FR(i)] = Read\_Logger\_2(S\_Logger,SD.Logger\_Counter(i-1),SD.Ax(i-1),SD.Ay(i-1),SD.Az(i-1),Raw\_P, Raw\_Q, Raw\_R, Raw\_Mx, Raw\_My, Raw\_Mz, SD.IMU\_T(i-1), A2D\_Ch1, A2D\_Ch2, SD.RF\_F(i-1), SD.RF\_FL(i-1), SD.RF\_L(i-1), SD.RF\_B(i-1), SD.RF\_R(i-1), SD.RF\_FR(i-1));

end

SD.P(i)=Raw\_P-P\_Bias; SD.Q(i)=Raw\_Q-Q\_Bias; SD.R(i)=Raw\_R-R\_Bias; % Calibrate the gyro data

temp=Mag\_A\*[Raw\_Mx-Mag\_c(1); Raw\_My-Mag\_c(2); Raw\_Mz-Mag\_c(3)]; % Magnetometer Raw Data Correction

SD.Mx(i)=temp(1); SD.My(i)=temp(2); SD.Mz(i)=temp(3);

flushinput(S\_Logger); % Flush the data logger serial port

% Acquire data from the LightWare Laser Rangefinder

if i==1

SD.Laser\_RF(i) = Read\_LightWare(S\_LightWare,0);

else

SD.Laser\_RF(i) = Read\_LightWare(S\_LightWare,SD.Laser\_RF(i-1));

end

flushinput(S\_LightWare); % Flush the LightWare Laser Rangefinder serial port

% Acquire the data from the iRobot Create

[BumpRight,BumpLeft,BumpFront,SD.Wall(i),SD.VirtWall(i),CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,LeftCurrOver,RightCurrOver,DirtL,DirtR,ButtonPlay,ButtonAv,SD.Dist(i),SD.Angle(i),SD.CreateVolts(i),SD.CreateCurrent(i),Temp,Charge,Capacity,pCharge]=Read\_Create\_2(S\_Create);

flushinput(S\_Create); % Flush the iRobot Create serial port

fwrite(S\_Create, [142 0]); % Request all sensor data from Create

% Acquire data from Hukuyo Lidar

% rangescan = Read\_Hokuyo(S\_Hokuyo, rangescan);

% SD.Lidar\_Range(:,i)=rangescan;

% SD.Laser\_RF(i) = rangescan(341); % simulated laser rangefinder using the LIDAR center spot

% plot(rangescan);

% drawnow

% flushinput(S\_Hokuyo); % Flush the Lidar serial port

% fprintf(S\_Hokuyo,'GD0044072500'); % request a new Lidar scan

Time=clock; % Mark the current time;

SD.Time(i)=Time(6); % Store the seconds;

if i==1

SD.Roll(i)=atan2(SD.Ay(i), SD.Az(i)); % Calculate the Roll angle based on the gravity vector

SD.Pitch(i)=atan(-SD.Ax(i)/(SD.Ay(i)\*sin(SD.Roll(i))+SD.Az(i)\*cos(SD.Roll(i)))); % Calculate the Roll angle based on the pitch vector

SD.Mag\_Heading(i)=atan2(-SD.My(i), SD.Mx(i)); % The initial 2D magnetic heading of the robot

SD.Yaw(i)=0; % Set the current yaw angle as zero

Attitude\_P=zeros(3,3); % Initialization the error covariance matrix for attitude estimation

SD.X(i)=0; % Initial Robot X Position

SD.Y(i)=0; % Inital Robot Y Position

else % If i>1

SD.Time\_Diff(i)=SD.Time(i)-SD.Time(i-1); % Calculate the time difference between steps

if SD.Time\_Diff(i)<0

SD.Time\_Diff(i)=SD.Time\_Diff(i)+60; % Compensate for the minute change

end

SD.TotalDist(i)=SD.TotalDist(i-1)+SD.Dist(i); % Calculate the total traveled distance based on the encoder reading

SD.TotalAngle(i)=SD.TotalAngle(i-1)+SD.Angle(i); % Calculate the total traveled angle based on the encoder reading

SD.Mag\_Heading(i)=atan2(-SD.My(i), SD.Mx(i)); % The 2D magnetic heading of the robot

[SD.Roll(i), SD.Pitch(i), SD.Yaw(i), Attitude\_P] = Attitude\_Estimation(SD.Mag\_Heading(1), SD.Roll(i-1), SD.Pitch(i-1), SD.Yaw(i-1), SD.Time\_Diff(i), Attitude\_P, SD.Ax(i), SD.Ay(i), SD.Az(i), SD.P(i), SD.Q(i), SD.R(i), SD.Mx(i), SD.My(i), SD.Mz(i)); % Perform the attitude estimation

SD.X(i)=SD.X(i-1)+SD.Dist(i)\*sin(SD.Yaw(i)-SD.Yaw(1)); % Dead Reconing for X position

SD.Y(i)=SD.Y(i-1)+SD.Dist(i)\*cos(SD.Yaw(i)-SD.Yaw(1)); % Dead Reconing for X position

end

%% Put your custom control functions here

%----------------------------------------------------------

% Use the following function for the robot wheel control:

%SetDriveWheelsSMART(S\_Create, .4, .4, SD.CliffLeft(i),SD.CliffRight(i),SD.CliffFrontLeft(i),SD.CliffFrontRight(i));

phi(i) = atan2(SD.Ay(i),SD.Az(i));

theta(i) = atan(-SD.Ax(i)/(SD.Ay(i)\*sin(phi(i)) + SD.Az(i)\*cos(phi(i))));

% scatter3(phi(1:i),theta(1:i), SD.Time(1:i))

% if SD.TotalAngle(i) == pi/6

% SetDriveWheelsSMART(S\_Create, .1, .1, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

% elseif SD.TotalAngle(i) < pi/4

% SetDriveWheelsSMART(S\_Create, .1+.5\*abs((pi/6)-SD.TotalAngle(i)), .1, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

%

% elseif SD.TotalAngle(i) > pi/4

% SetDriveWheelsSMART(S\_Create, .1, .1+ .5\*abs((pi/6)-SD.TotalAngle(i)), CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

% end

if i>1

e(i)=e(i-1)+SD.Time\_Diff(i)\*((pi/6)-SD.TotalAngle(i));

end

if SD.TotalAngle(i) == pi/6

SetDriveWheelsSMART(S\_Create, .1, .1, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

else

SetDriveWheelsSMART(S\_Create, .1+(.35\*((pi/6)-SD.TotalAngle(i))+(.05\*e(i))-(0.1\*(SD.R(i)))), .1, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

end

% % Collision Avoidance Example(Set Tend=60s)

% if SD.RF\_F(i)<500

% SetDriveWheelsSMART(S\_Create, -Create\_Full\_Speed\*.4, -Create\_Full\_Speed\*.6, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

% elseif SD.RF\_B(i)<300

% SetDriveWheelsSMART(S\_Create, Create\_Full\_Speed\*.6, Create\_Full\_Speed\*.4, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

% elseif SD.RF\_L(i)<500

% SetDriveWheelsSMART(S\_Create, -Create\_Full\_Speed\*.5, Create\_Full\_Speed\*0.5, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

% elseif SD.RF\_R(i)<500

% SetDriveWheelsSMART(S\_Create, Create\_Full\_Speed\*0.5, -Create\_Full\_Speed\*.5, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

% else

% SetDriveWheelsSMART(S\_Create, Create\_Full\_Speed, Create\_Full\_Speed, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront);

% end

%----------------------------------------------------------

%% End of the Custom Control Code

% Calculate the time left in the constant interval

SD.Delay(i)=Ts-toc;

if SD.Delay(i)>0

pause(SD.Delay(i)); % Kill the remaining time

end

end

Total\_Elapse\_Time=SD.Time(Total\_Steps)-SD.Time(1) % Calcualte the total elapse time, not counting the minutes

SetDriveWheelsSMART(S\_Create, 0, 0, CliffLeft,CliffRight,CliffFrontLeft,CliffFrontRight,BumpRight,BumpLeft,BumpFront); % Stop the wheels

BeepRoomba(S\_Create); % Make a Beeping Sound

% Properly close the serial ports

delete(S\_Logger)

clear S\_Logger

delete(S\_Create)

clear S\_Create

delete(S\_LightWare)

clear S\_LightWare

% fprintf(S\_Hokuyo,'QT');

% fclose(S\_Hokuyo);

save('SMART\_DATA.mat'); % Save all the collected data to a .mat file

SMART\_PLOT; % Plot all the robot data