Stability Control System for the Bioloid Humanoid Robot

NSF Summer Undergraduate Fellowship in Sensor Technologies
Willie W Gonzalez (EE) – University of Puerto Rico Mayagüez campus

Advisor: Dr. Daniel D. Lee

ABSTRACT

A key characteristic in any autonomous system is stability. Without stability robots could not work properly. To achieve stability first the robot needs the means to know its position with respect with its original so that a correction in position can be obtained if necessary. This is true for the type of INS (inertial navigation system) known as dead reckoning. In this INS the current position of the robot or system is calculated from measurements of the acceleration and knowledge of its original position. Some IMUs (inertial navigational units) consist of a couple of accelerometers and gyroscopes. With this IMU acceleration can be obtained. On Strap-down navigational system, this IMU are directly attached to the system so that their measurements correlates with those of the system, so that they can be used to calculate the position of the system. Throughout this document a method on how to obtain the necessary measurements from the IMU and how to apply them, will be discuses.

Table of content

1. Introduction

The creation of autonomous robots is a task that many have undertaken for military, medical or other reasons. But no matter the reason, measurements of the position of the system relative to its environments are needed for it to act accordingly without an external intervention. A technique that humans have used to navigate through their environment is dead reckoning [1]. This approach consists of the calculation of current position with the use of the following: knowledge of an initial position, and measurements of speed and direction. The inertial navigation system (INS) uses an equivalent approach with the help of Newton's laws. These laws, which relate velocity, acceleration, and position, enable us to calculate the change in position. With knowledge of the position of a part of an autonomous system and some inverse kinematics, the position of every part of the system can be calculated. With this information an accurate displacement of each part can be achieved.

How do we obtain these measurements? To measure the acceleration of a body, inertial sensors are used: accelerometers and gyroscopes. The accelerometer is used to measure acceleration in a single direction. By placing three accelerometers orthogonally to each other we can sense the acceleration in any direction [2]. From the gyroscope we can obtain the angular velocity on a single axis. These two inertial sensors together can help to identify the change of position of a system. Inertial sensors are often attached to a fixed part of the vehicle or system so that the measurements that it senses are due to the system movement and not the movement of the sensor itself. This type of configuration is called Strap-down INS[1] Many types of this inertial sensor are well documented [3] but they come with some limitations: noise and drift on the measurement readings can disturb the outcome of the calculations.

This paper explores ways of eliminating or attenuating those limitations by using complementary filters. We created a control loop that used the readings of the IMU to allow the platform to return to its original position. Some application for this type of system could be a platform that holds a camera so that in case of undesired movements the camera continues filming in the same direction.

2. Background

2.1 Inertial navigation system (INS)

The inertial navigation system was first developed in the early 1950 by the Sperry Gyroscope Corp. Their objective was to be able to create a navigation system that did not require ground beacons. In time of wars these beacons can be jammed or destroyed [4].Their main applications are in missile, aircraft, marine and land vehicles.

Inertial navigation systems used the force laws postulated by Isaac Newton to estimate the current position and orientation of an object [1]. Using force equations that relate angles, forces, velocity, position and acceleration, the current position and orientation of an object can be determined. But before we are able to apply these equations we need to have devices that can measure the acceleration of the system and for that inertial sensors are needed.

Inertial sensors are external sensors that can measure the acceleration of a system and make almost no references to the external world. Inertial sensors are often used to refer to accelerometers and gyroscopes that can measure acceleration in a single axis and angular acceleration respectively. These types of sensors are commonly used in Strap-down INS.

2.2 Strap-down INS

Strap-down INS are like any other INS; the difference is that instead of having a big platform holding the INS, the inertial measurement unit (IMU) is attached or strapped down to the system. This results in a lighter, smaller and cheaper Inertial Navigation System. But for this type of system to work more accurate sensors and higher computational equipment are needed [1]. Thanks to the advance in technology these needs have been satisfied.

2.3 Inertial measurement unit

An IMU consists of the set of sensors that will be used in a certain application to retrieve the information needed to calculate the position and orientation of the body of a robot or any system. This project used the IDG300 gyroscope and the ADXL330 accelerometers IC's which are both located on the Inertia Measurement Unit developed by SPARK FUN

2.3.1 Gyroscope

Gyroscope can sense the angular rate of turn or angular acceleration about a single axis. There are many types of gyroscope with different applications depending on the environment and the accuracy needed.

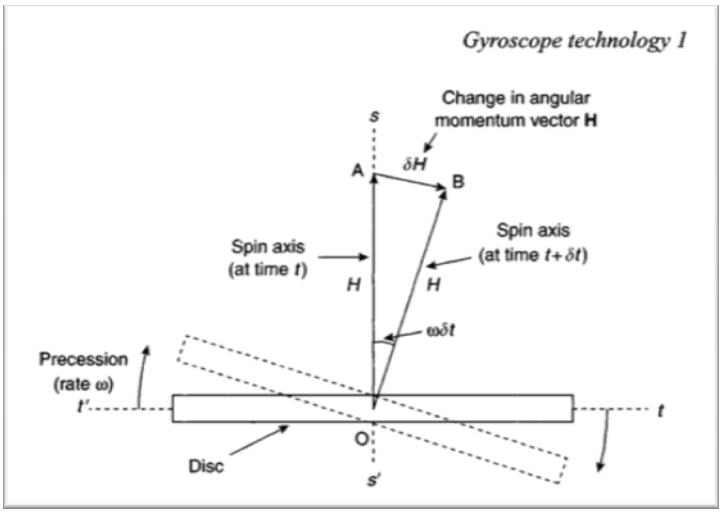


Figure 1 Gyroscope technology [7]

A component normal to the rotation axis appears, the precession w, to try to align the spin axis with the torque axis. T = ω H where T (torque) and H (angular momentum) this is [5], also known as the gyroscope law. If a torque is applied to keep it's spin axis aligned with a direction then the measurement of this torque will provide the angular movement of the object to which the gyroscope is attach.

2.3.2 Accelerometer

Accelerometers allow the measurement of the acceleration on a single axis to be ascertainable thanks to Newtown's second law F=ma where a could be represented as the sum of g (gravitational forces) and f, the acceleration produced by external forces of the object, thus a=g+f. Typically in NS there are three accelerometers orthogonal to each other to provide the acceleration reading in three different directions. It is not practical to use the entire mass of a system to determine its acceleration. Instead, a proof mass connected to a set of springs is used to measure the acceleration is found to be more efficient in discerning the acceleration. With the help of the Hook law F=x*k where k is a constant, a property of the spring and x it's the stretch displacement of the spring the acceleration of the object on a single axis can be measured. This is the simplest type of accelerometers there far more accurate and expensive accelerometers that involve technology such as Solid-state ferroelectric accelerometer and Solution electrolytic accelerometer [6].

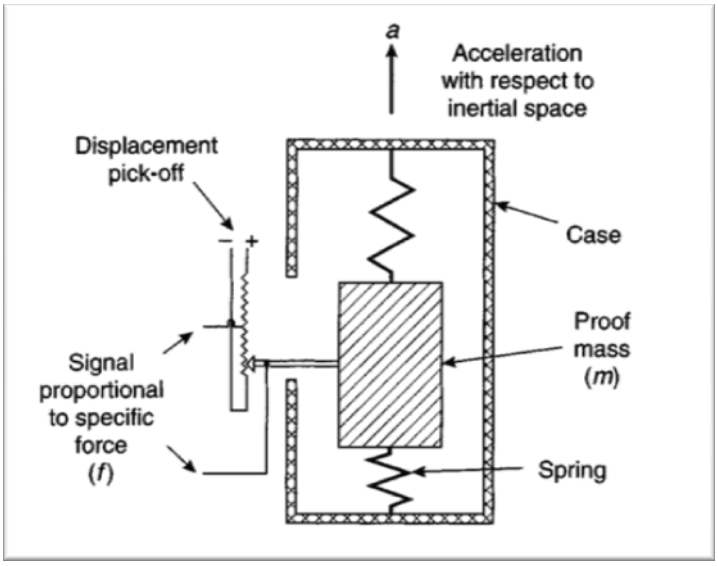


Figure 2 Accelerometer technology [8]

3. Calculation and application method

3.1 Equations and approach

With both the accelerometer and the gyroscope, a displacement angel can be calculated using the fallowing equations.

$$\theta \approx \int w(angular\ rate)dt \qquad \theta \approx \sin^{-1}\left(\frac{accel\,output}{g}\right)$$

Both of the equation can be used to determine the angle but each one has a drawback. The first equation is not suited for long periods of time because it could cause saturation in your reading for its growth characteristic through time due to the integration, but it's excellent with fast movements in short period of time. The second equation doesn't have the problem of saturation but in fast movement the accelerometer output contain the acceleration of external forces and that can cause error in readings, but its preferable in calculations involving long periods of time and slows movements. We must attempt to obtain the beneficial information from each equation whilst eliminating the erroneous data found in each equation. For that a complementary filter can be used.

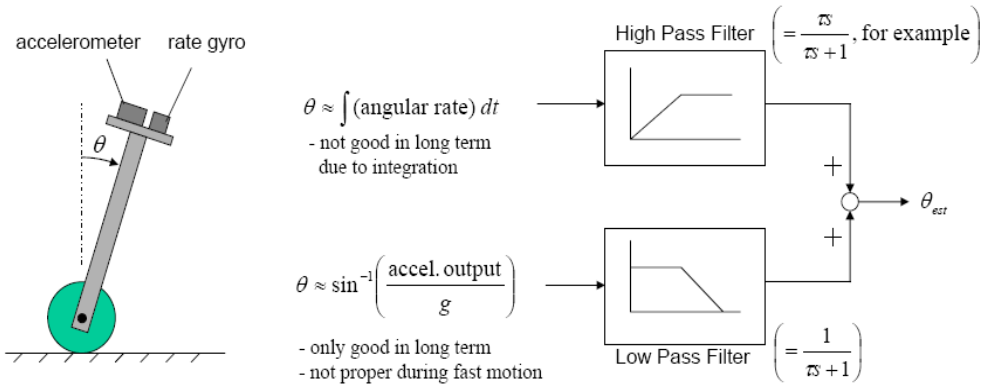


Figure 3 Complementary filter [9]

By passing the integration equation by a high pass filter we will obtain only the peaks in the graph of the integration representing the fast movement of the object in which the gyroscope is reliable and get rid of the saturation problem. When we pass the other equation through a low pass filter we will obtain only the measurements during last periods of times and get rid of momentary movement that can alter the readings for the accelerometer. Then by adding the two outputs of each filters the measurement of the accelerometers is going to be a bias reading and the gyroscope will them be the peaks that result from the fast movement, in other words, we have theoretically obtained the best of both reading and gotten rid of theirs drawbacks.

3.2 Complementary filter

A complementary filter consists of two filters, commonly a high pass and a low pass filter. If the sum of two filters in the phase is zero and the magnitude is one, they are complementary to each other. A good way to verify this is to pass some data through both filters and the sum should yield the original data. In other words, the original graph minus the sum should be zero.

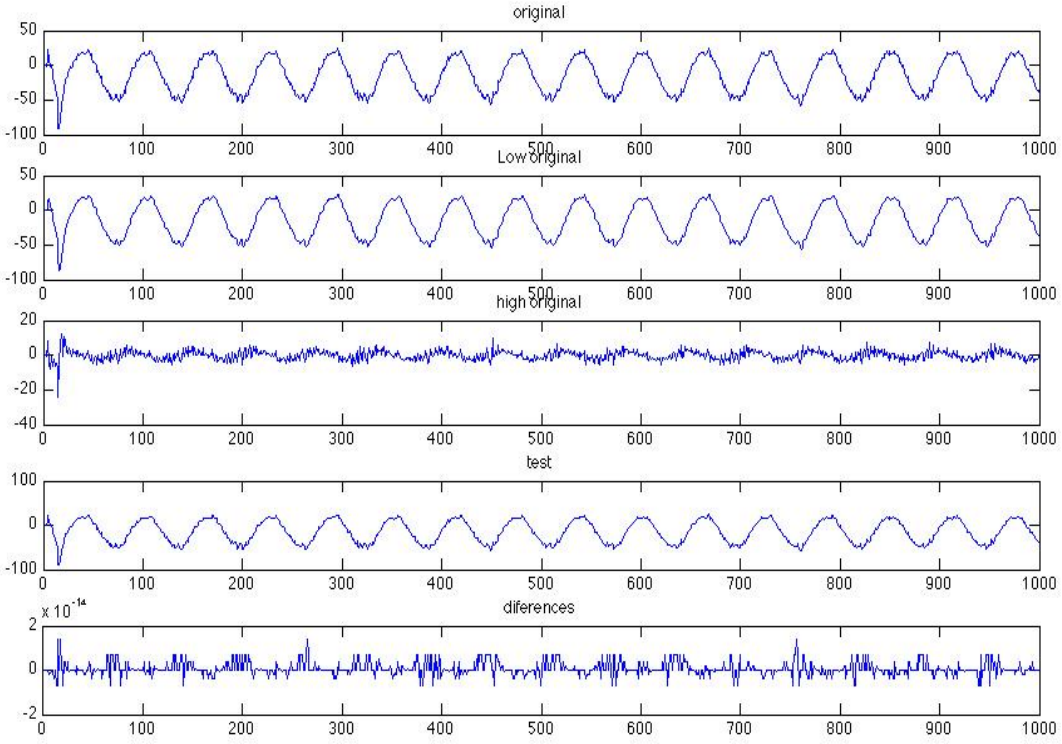
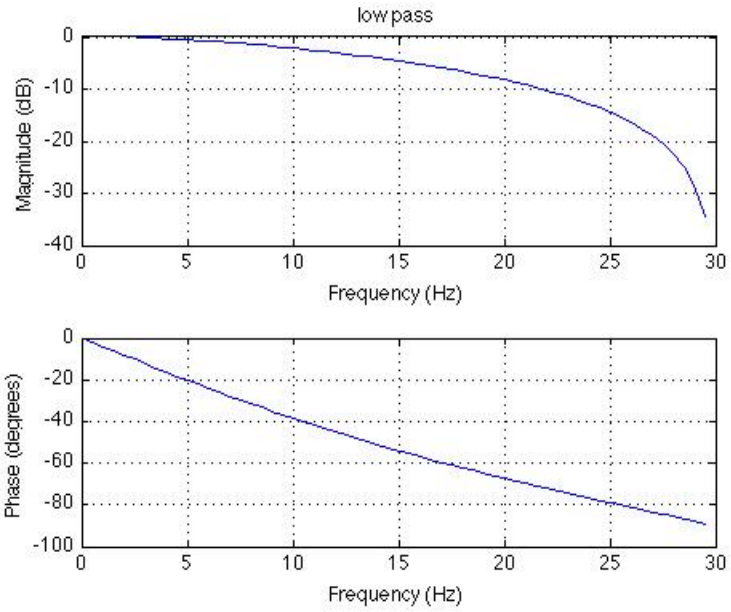


Figure 4 Complementary Filter Corroboration

In Figure 4 the data retrieved from the accelerometer can be seen in the top of it. That data was filter through both high pass and low pass filters and their sum is graph in the third plot. The differences of the original and sum is in the order of 10^-14 which can be considered to be zero. The frequency of the complementary filter was not arbitrary chosen but was chosen after some test.



The first frequency to be chosen was 25% of the sampling rate for the low pass and the same for the high pass. The one yielding better result on the measurements was approximately %12.5 percent of the sampling rate. The sampling rate was around 60 HZ. Figure 5 shows the low pass filter.

Figure 5 Low pass filters

3.3 Gyroscope and accelerometer before and after the filter

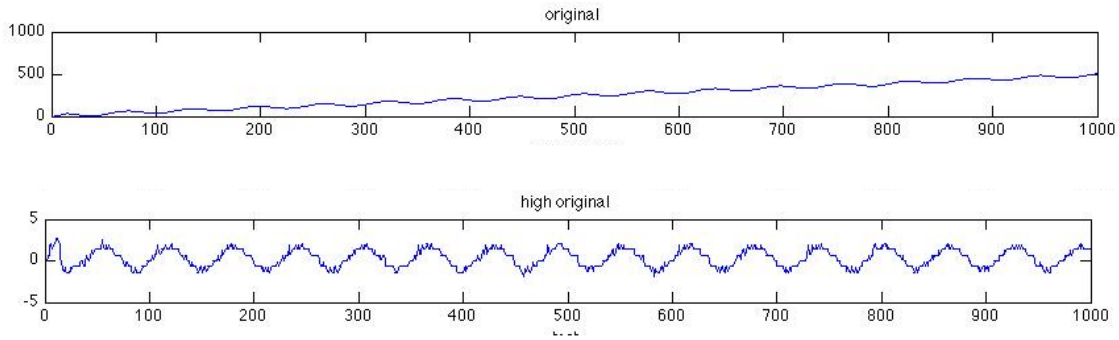


Figure 6 Gyroscope response

In figure 6 we can see the response to a sine wave movement by the motors in the readings of the gyroscope. In the first plot the saturation problems with the gyroscope are visible when the measurement continues to increase over time after performing the

integration. The second plot shows the actual respond after passing the data through the high pass filter.

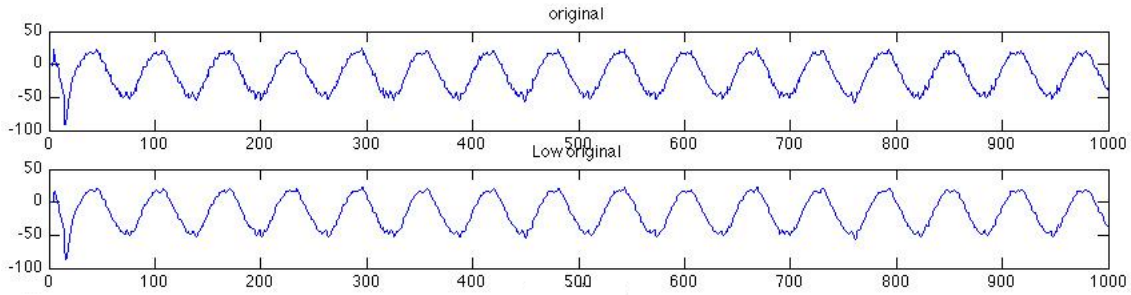


Figure 6 accelerometer response

In figure 6 we can see the response to a sine wave movement by the motors in the readings of the accelerometer. After passing the data through the low pass filter the sine wave becomes smoother and the graph gets rid of the high frequency noise it once had.

3.3 Gyroscope plus accelerometer and tracking of angle movement

After the filters adding the two signals yield the estimated angle of the (IMU). which translate to the angle of the system where the IMU have been strap down. To be able to check if the measurements are correct we need a point of references. The motors that we are using are the Dynamixel AX-12, which have an angle range from 0 degree to 300 degree, depending on the signal position that goes from 0 to 0x3ff (1023). The theoretical angle can be calculated from this relationship assuming a linear equation with an intercept in 0. The equation turns out to be y=m*x where y is the angle x is the signal and m is (300/1023). If we assume that the angle can vary from (150 to -150) making the 0 perpendicular to the surface of the motor platform. The equation turns out to be y=m*(x-511). The number 511 is the theoretical value of the signal to 150 degrees. It is important to notices that the equation y=(300/1023)*x does not hold truth for both angles 300 and 150 degrees with 1023 and 511 as respective signals hence m is not constant, but it doesn't vary that much and it is a good approximation. Now the text was to send a sine wave signal and see how well does the reading follows the theoretical response.

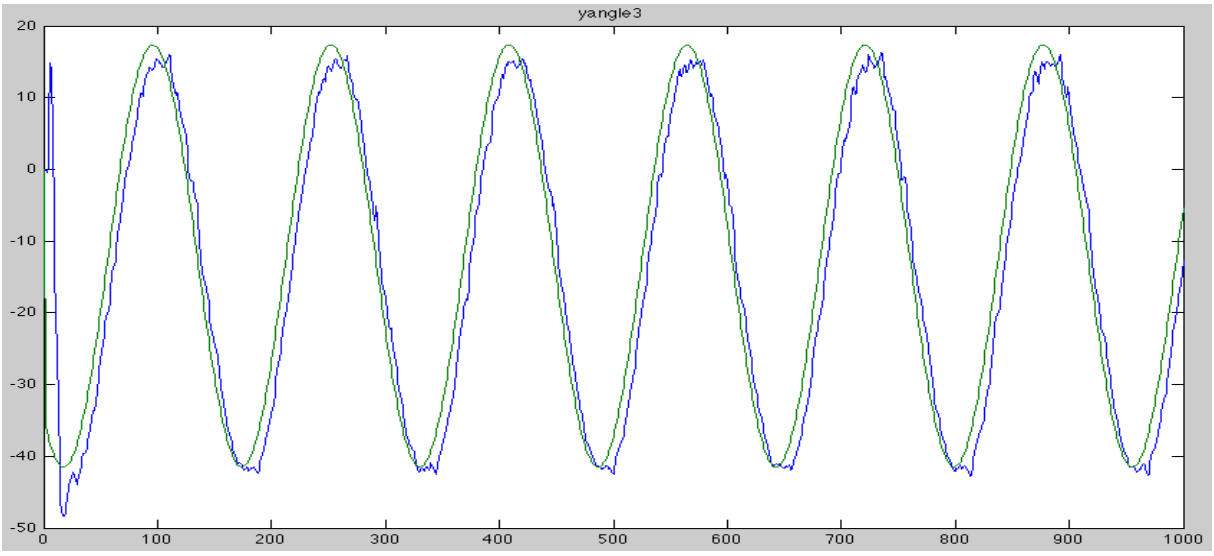


Figure 7: Accelerometer Reading VS. Theoretical Response

On figure 7 we can see how the angle reading (blue line) follows the theoretical response with some lag. The source of the lag is a result of two things; a lag caused by the filters because it's averaging data, and the delay that the motors have on getting to the designated position when the signal is sent. This was achieved for low frequencies, from .1 to 1 HZ beyond that the measurements don't follows the theoretical responses for two mayor reasons. For one, the accelerometers readings are not reliable because of the constant fast movement making it so that even if you pass a whole set of data that is wrong through the filter, the outcome will not be reliable, Also at some point the motors can't keep up with the sine wave, the other major reason for it's failure.

3.3 Feedback loop

A simple proportional (P) feedback was created for the platform. This kind of feedback takes the differences between the riding and the settling point, the error, and multiples it by a constant p to send the signal to the system. In this case this constant was not calculated but rather it was found experimentally.

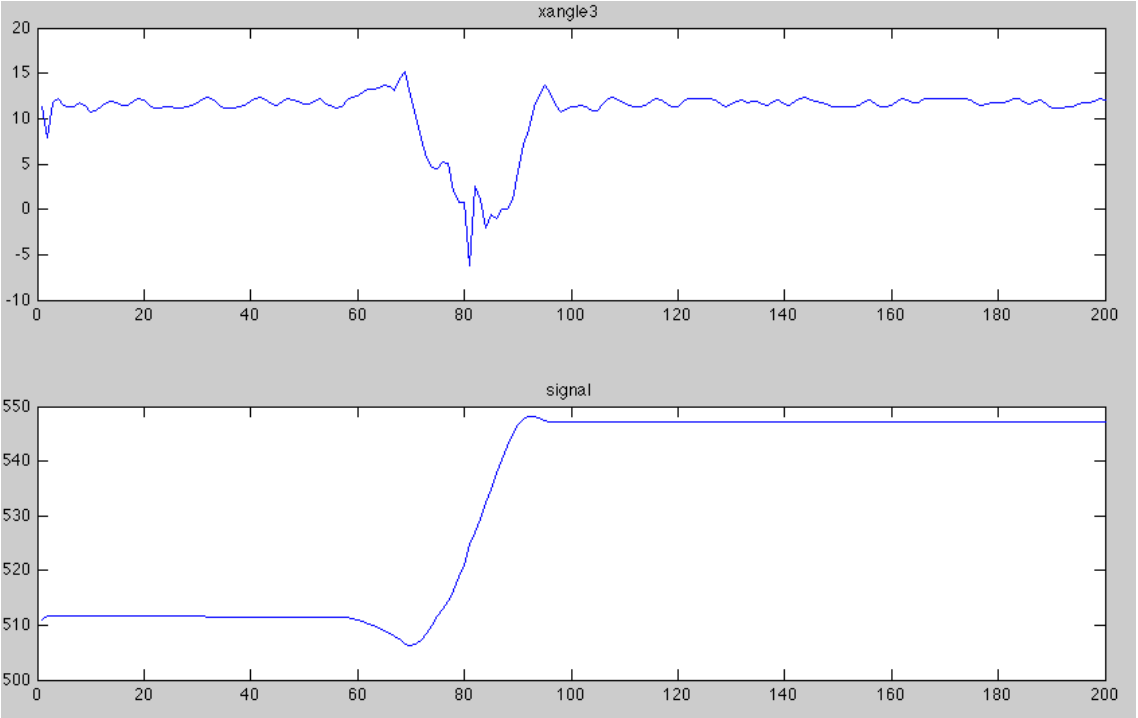
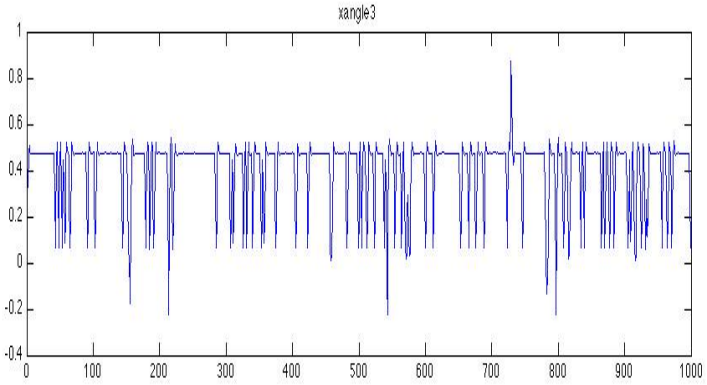


Figure 8 control loop response



In figure 8 we can see the control loop response to disturbances and how a change on the angle reading affects the control signal. Notice that even if the angle reading is not a flat line in

some part, the control signal doesn't change. This is due to the implementation of a threshold on the error or differences of 1 degree. That threshold was implemented because during a stationary state without any movements, the angle reading is not a flat line (as can be seen on figure 9).

Figure 9 shows the readings of the sensors when there is no movement on the platform. There is an offset or noise of approximately 0. 4 degrees on the upper bound and 0.7 degrees to the lower bound, assuming the center of the line is exactly at .5 degrees. For that reason a threshold of 1 degree to each bound was implemented.

Figure 9: Noise In The Reading

Future work

1) Try to improve the calculation of the angle readings. Currently working on a way to determine whether the accelerometer readings are reliable or not.

   Approach

   The readings of a 3 axis accelerometer are basically vectors in each direction; the value of each vector should go up to around 9.81. Because the only acceleration that counts is the gravity, higher numbers imply that the acceleration has another component due to external forces. Nevertheless there are three vectors, one on each axis and they are all measuring gravity. Hence all of them are a component of the G vector of which the magnitude should be 9.81. If denominated, each vector gx, gy and gz a component of G then:

   $$|G| = \sqrt{gx^2 + gy^2 + gz^2} \cong 9.81$$

   With these criteria we can determine if the readings are reliable or not. Also, there will be a margin of error for the readings in which case a threshold around 9.81 on which the readings can be considered reliable.

   These equations have been already implemented in the code, but the question of what will be better to do if the readings of the accelerometer are not reliable is still without an answer.

2) Implement a PD or PID feedback loop to improve the response of the control system.

References

[1] D. Titterton and J. Weston, "Introduction," in *Strapdown Inertial Navigation Technology, 2*th ed.The American Institute of Aeronautics and Astronautics, 2004. pp1-5

[2] G.Dudek and M. Jenkin"Nonvisual Sensor Algorithms",in Computational Principle of Mobilea Robotics. 1th ed.Cambrige University Press 2000. 56-58

[3] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology, 2*th ed.The American Institute of Aeronautics and Astronautics, 2004.

[4] Shaw, Brian, International Journal of Electrical Engineering Education, Jan 1999 Mhttp://findarticles.com/p/articles/mi_qa3792/is_199901/ai_n8842342/

[5] D. Titterton and J. Weston, "Gyroscope technology," in *Strapdown Inertial Navigation Technology, 2*th ed.The American Institute of Aeronautics and Astronautics, 2004. pp1-26(of the 4 chapter)

[6] D. Titterton and J. Weston, "Accelerometer and multi-sensor technology," in *Strapdown Inertial Navigation Technology, 2*th ed.The American Institute of Aeronautics and Astronautics, 2004. pp 21(of the 6 chapter)

[7] D. Titterton and J. Weston, "Gyroscope technology," in *Strapdown Inertial Navigation Technology, 2*th ed.The American Institute of Aeronautics and Astronautics, 2004. Pp 65 (7 of chapter 4)

[8] D. Titterton and J. Weston, "Accelerometer and multi-sensor technology," in *Strapdown Inertial Navigation Technology, 2*th ed.The American Institute of Aeronautics and Astronautics, 2004. pp 154(2 of the 6 chapter)

[9] Sanghyuk Park and Jonathan How, Examples of Estimation Filters from Recent Aircraft Projects at MIT November 2004, http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-333Fall-2004/67AC5F9B-4FC0-4F62-9A4A-3921AA27A5FB/0/l15_filter_examp.pdf

Appendixes

Control system with angle reading.

```
%{
The I\\ ID of 120

IMU Bytes
---------
#define CONTROL_FORWARDS_ACCEL_LOW 26 //z
#define CONTROL_FORWARDS_ACCEL_HIGH 27
#define CONTROL_SIDEWAYS_ACCEL_LOW 28 //y
#define CONTROL_SIDEWAYS_ACCEL_HIGH 29
#define CONTROL_VERTICAL_ACCEL_LOW 30 //x
#define CONTROL_VERTICAL_ACCEL_HIGH 31
#define CONTROL_PITCH_RATE_LOW 32 //y rate
#define CONTROL_PITCH_RATE_HIGH 33
#define CONTROL_ROLL_RATE_LOW 34 //x rate
#define CONTROL_ROLL_RATE_HIGH 35
#define CONTROL_YAW_RATE_LOW 36
#define CONTROL_YAW_RATE_HIGH 37
%}

clc

imu_id = 120;
timeout = .1; %Really arbitrary
x_addr = 30;
y_addr = 28;
z_addr = 26;
x_rate_addr = 34;
y_rate_addr = 32;
phi = 0;
zeta = 0;
%Reading bounds
start_addr = z_addr;
stop_addr = x_rate_addr+1;
length = stop_addr-start_addr+1;
```

```
num_points = 200;



factor=1024/300;
alfa=511;
---------------------------------------------------
x=(factor)*(0)+alfa;     %set the initial angle


y=(factor)*(0)+alfa;
-----------------------------------------------


if ~exist('fid'),
   fid = dynamixelOpen();      %open the comunication to the motors
end


[b,a] = butter(2,.3999,'low');
[d,c] = butter(2,.3999,'high');

for i=1:500
dynamixelMove(16,y, 1023);
dynamixelMove(18,x, 1023);
end

 % initializes all vectors to prevent boundaries problems

x_accel = zeros(num_points,1);
y_accel = zeros(num_points,1);
z_accel = zeros(num_points,1);
x_rate = zeros(num_points,1);
y_rate = zeros(num_points,1);
timestamp = zeros(num_points,1);

x_phi1 =  zeros(num_points,1);
y_phi1 =  zeros(num_points,1);

x_phi2 = zeros(num_points,1);
y_phi2 = zeros(num_points,1);

x_phi2a =  zeros(num_points,1);
y_phi2a =  zeros(num_points,1);

x_phi1a =  zeros(num_points,1);
y_phi1a =  zeros(num_points,1);

x_angle =  zeros(num_points,1);
```

```
y_angle =  zeros(num_points,1);

x_angle1 = zeros(num_points,1);
y_angle1 =  zeros(num_points,1);


x_rate_2 =   zeros(num_points,1);
y_rate_2 =  zeros(num_points,1);

xangle =zeros(num_points,1);
yangle =zeros(num_points,1);
pre_angle=zeros(num_points,1);
pre_angley=zeros(num_points,1);
signal =zeros(num_points,1);


x_phi2ab = zeros(num_points,1);
y_phi2ab = zeros(num_points,1);

x_phi1ab = zeros(num_points,1);
y_phi1ab =zeros(num_points,1);

x_angle2 = zeros(num_points,1);
y_angle2 =zeros(num_points,1);

x_angle12 = zeros(num_points,1);
y_angle12 = zeros(num_points,1);
xangle2=zeros(num_points,1);
yangle2=zeros(num_points,1);
x_phi1b=zeros(num_points,1);
y_phi1b=zeros(num_points,1);
x_phi2b=zeros(num_points,1);
y_phi2b=zeros(num_points,1);
xangle3= zeros(num_points,1);
yangle3= zeros(num_points,1);

xsignal=zeros(num_points,1);
ysignal=zeros(num_points,1);

    x1= 0;
    y1= 0;
    z1= 0;
    xp1= 0;
    yp1= 0;

    j=1;
    k=1;
    count=0;

tic;
```

```
%profile on
for i=1:num_points

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

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

   imu = dynamixelReadData(imu_id, start_addr, length);
   if(size(imu,2)==length)
      x_accel(i) = (imu(5)+imu(6)*256);
      y_accel(i) = (imu(3)+imu(4)*256);
      z_accel(i) = (imu(1)+imu(2)*256);
      x_rate(i) = (imu(9)+imu(10)*256);
      y_rate(i) = (imu(7)+imu(8)*256);
      timestamp(i) = toc;

   end
   while ( x_accel(i)==0) && ( y_accel(i)==0)  && (z_accel(i)==0) && ( x_rate(i)==0
)&& (y_rate(i)==0)
     count=count+1;

    imu = dynamixelReadData(imu_id, start_addr, length);
   if(size(imu,2)==length)
      x_accel(i) = (imu(5)+imu(6)*256);
      y_accel(i) = (imu(3)+imu(4)*256);
      z_accel(i) = (imu(1)+imu(2)*256);
      x_rate(i) = (imu(9)+imu(10)*256);
      y_rate(i) = (imu(7)+imu(8)*256);
      timestamp(i) = toc;
   end


   end
%if i>1


        g=gravity( x_accel(i), y_accel(i),z_accel(i));
        g1(j:j+3,1)=g;
        g4=g(1,1);
        if j==1
        g3=g
        end
if i>1
  if (g4>9.87||g4<9.5)

     g=g1(j-4:j-1,1)  ;

  end
end
    g1(j:j+3,1)=g;
```

```
g7(i)=g(1,1);


  %-----------------------------------------------------------------
  % its supose tu calculate the angle from the gyroscope reading


  if i==1

   x_phi1(i) = x_rate(i)-x_rate(1);
   y_phi1(i) = y_rate(i)-y_rate(1);

   x_phi2(i) =atan2( g(2,1),g(4,1))*180/pi;
   y_phi2(i) =atan2( g(3,1),g(4,1))*180/pi;

 %  x_phi2(i) =real(asin( g(2,1)/9.81)*180/pi);
 %  y_phi2(i) =real(asin( g(3,1)/9.81)*180/pi);




   x_angle2(i) = (x_phi2(i));
   y_angle2(i) = (y_phi2(i));

   x_angle12(i) = (x_phi1(i))/3.41;
   y_angle12(i) = (y_phi1(i))/3.41;


   xangle3(i)= x_angle2(i)+y_angle12(i);
   yangle3(i)=  y_angle2(i)+x_angle12(i);


  else

  x_rate_2(i) =  x_rate(i)-x_rate(1);
  y_rate_2(i) =  y_rate(i)-y_rate(1);


   x_phi1(i) = x_rate_2(i)+ x_phi1(i-1);  %perform integral
   y_phi1(i) = y_rate_2(i)+ y_phi1(i-1);


   x_phi2(i) =atan2( g(2,1),g(4,1))*180/pi;
   y_phi2(i) =atan2( g(3,1),g(4,1))*180/pi;


 %   x_phi2(i) =real(asin( g(2,1)/9.81)*180/pi);
 %   y_phi2(i) =real(asin( g(3,1)/9.81)*180/pi);
```

```
%-------------------------------------------------------------
        x_phi2ab = filter(b,a,x_phi2);
        y_phi2ab = filter(b,a,y_phi2);

        x_phi1ab = filter(d,c,x_phi1);
        y_phi1ab = filter(d,c,y_phi1);


        x_angle2(i) = (x_phi2ab(i));
        y_angle2(i) = (y_phi2ab(i));

        x_angle12(i) = (x_phi1ab(i))/3.41;
        y_angle12(i) = (y_phi1ab(i))/3.41;


        xangle3(i)= x_angle2(i)+y_angle12(i);
        yangle3(i)=  y_angle2(i)+x_angle12(i);

     end

%-----------------------------------------------------------------------------------------------------
¨%{Control loop/ part that change on the sine wave code
    if i>1


         if (xangle3(i)>(xangle3(1)+1))||(xangle3(i)<(xangle3(1)-1))
             landa=(xangle3(i)-xangle3(1));
             delta=(xangle3(i)-xangle3(i-1));
              x=(x-((landa)/4.5));

             if x<325
                x=325;
             end
           if x>625
             x=625;
           end

            % x=560;
             dynamixelMove(18,x, 1023);
        end

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

         if (yangle3(i)>(yangle3(1)+1))||(yangle3(i)<(yangle3(1)-1))
             landa=(yangle3(i)-yangle3(1));
            % delta=(yangle3(i)-yangle3(i-1));
              y=(y-((landa)/4.5));
```

```
            if y<240
               y=240;
            end
          if y>599
            y=599;
          end

           %  y=511;
            dynamixelMove(16,y, 1023);
        end
    end
}
 %-------------------------------------------------------------------------------
    xsignal(i)=x;

    ysignal(i)=y;

   pre_angle(i)=((x-alfa)/factor);
   pre_angley(i)=((y-alfa)/factor);
   j=j+4;
   k=k+10;

end
toc;

 ---------------------------------------------------------
       xint= x_phi1;
       yint= y_phi1;          %record data to simulate in filters without roning the entire
program


       xatan= x_phi2;
       yatan=y_phi2;
-----------------------------------------------------------------
%profile viewer

% Plot the angles readings


figure(1)
subplot(2,1,1);
plot(yangle3);
title('yangle3')



%subplot(4,1,2);
%plot(yangle3);
%title('yangle3')
```

```
subplot(2,1,2);
plot(ysignal);
title('signal')


%subplot(3,1,3);
%plot(pre_angley);
%title('spre_angley')


figure(2)
subplot(2,1,1);
plot(xangle3);
title('xangle3')

%subplot(4,1,2);
%plot(xangle3);
%title('xangle3')


subplot(2,1,2);
plot(xsignal);
title('signal')


%subplot(3,1,3);
%plot( pre_angle);
%title(' pre_angle')
```

```
% sine wave for either y axis or x axis
t=clock;


 pre_angle(i)=((x-alfa)/factor);
 pre_angley(i)=((y-alfa)/factor);


%x=100*sin(w*t(6)*f)+511;
%signal(i)=x;
%dynamixelMove(18,x, 1023);

y=100*sin(w*t(6)*f)+470;
signal(i)=y;
dynamixelMove(16,y, 1023);


   j=j+4;
   k=k+10;
```

```
%things that needs to be initializes for the sin wave code
f=.5;
F=f*ones(num_points,1);
w = 2*pi;
-------------------------------------------------------------------------------------------------------
--
```