FLIGHT TESTING SMALL UAVS FOR AERODYNAMIC PARAMETER ESTIMATION

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Adam Chase

September 2013

COMMITTEE MEMBERSHIP

TITLE:	FLIGHT TESTING SMALL UAVS FOR DRAG POLAR ESTIMATION
AUTHOR:	Adam Chase
DATE SUBMITTED:	September 2013
COMMITTEE CHAIR:	Robert McDonald, Ph.D. Associate Professsor, Aerospace Engineering
COMMITTEE MEMBER:	Eric Mehiel, Ph.D. Associate Professor, Aerospace Engineering
COMMITTEE MEMBER:	Russell Westphal, Ph.D. Professor, Mechanical Engineering
COMMITTEE MEMBER:	Kurt Colvin, Ph.D. Professor, Industrial and Manufacturing Engineering

ABSTRACT

FLIGHT TESTING SMALL UAVS FOR AERODYNAMIC PARAMETER ESTIMATION

Adam Chase

ACKNOWLEDGMENTS

Thank you...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 Introduction and Motivation

An accurate drag prediction is critical for conceptual aircraft design, aircraft mission planning, and predicting performance trends of comparable aircraft. To this end, industry spends an extensive amount of time and money developing wind tunnel models and executing wind tunnel tests. Additionally, it is difficult to impossible to exactly scale down a vehicle, especially when features such as rivets, servo control horns, antennas, and air data probes are included. These differences between the model and the as-built aircraft can cause accuracy of the wind tunnel test to suffer. This inaccuracy inevitably leads to aerodynamic flight tests that attempt to quantify the as-built drag and lift characteristics of the vehicle.

The flight test of full scale aircraft for drag polar prediction is generally conducted about a trimmed condition. That is, the aircraft is flown to an operating condition dictated by the test plan, and sets the control surfaces such that there are no accelerations and no moments. Data is then collected for a set amount of time, without changing the operating condition. After the data is collected, the operating point is changed, the aircraft is trimmed at this new flight condition, and data is again collected. This process is repeated at various points in the aircraft's flight envelope until enough data is collected to estimate a drag polar.

Unfortunately, for many R/C aircraft and small UAVs, this procedure isn't feasible. First, these aircraft typically operate close to ground level, meaning there could potentially be both unsteady and turbulent winds, and a steady wind. R/C aircraft and small UAVs typically have much lower moments of inertias and mass than their full-scale counterparts, which means they will be affected much more by atmospheric disturbances than full-scale vehicles. Second, many of these aircraft have a line-of-sight communication link, and R/C aircraft in particular are flown in small patterns at a flight field. Even in the case of a steady atmosphere, R/C aircraft usually are not well trimmed, because by the time the pilot can see if the vehicle is trimmed, he has to turn around in the pattern. This thesis attempts to fix these problems, by

allowing the pilot to fly in a more generic flight path, and not relying on a still atmosphere assumption.

2.0 Method

Some basic assumptions will apply throughout the modeling of dynamics in thesis. They are as follows:

1. The vehicle is a fixed mass.
2. Coriolis effects are negligible.
3. Thrust will be assumed to be 0.

Note that a stationary atmosphere is not assumed. The fixed mass assumption is consistent with the electric aircraft used to test the system. At the altitude and speed at which the vehicles will be tested, Coriolis effects can be ignored[1]. The zero-thrust assumption is in place to minimize drag error. Any error in a measured state will decrease the accuracy of the drag measurement, and the accuracy of the drag measurement can be no better than the worst state measurement error. In-flight thrust is difficult to measure accurately, so a folding propeller will be used, and the motor will be turned off during data acquisition. This will allow the propeller to fold back, eliminating most of the wind-mill drag associated with a stalled propeller.

2.1 Reference Frames

For this thesis, the reference frames used will follow those described in [1], and will be repeated here for clarity.

North-East-Down (NED) Axes (x_{ned}, y_{ned}, z_{ned})

The NED axis system defines a local tangent plane on the Earth's surface, with the origin coinciding with the vehicle's center of gravity. The $\hat{i}$ vector points due north, the $\hat{j}$ vector

points due east, and the $\hat{k}$ vector points towards the center of the earth, in accordance with the right-hand rule. This coordinate system is vehicle carried, meaning the origin is fixed to the aircraft, but the axis directions are independent of vehicle orientation.

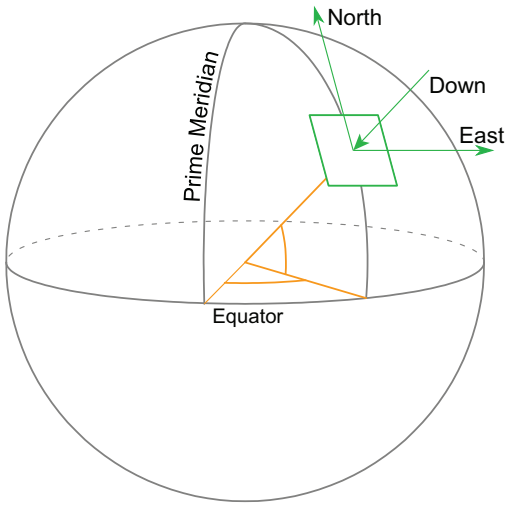


Figure 2.1: NED Frame of Reference[2]

Body Axes (x_b, y_b, z_b)

The body axis system has its origin at the vehicle's center of gravity, with the $\hat{i}$ direction pointing out the vehicle's nose, the $\hat{j}$ direction pointing out the right wing, and the $\hat{k}$ direction pointing out the belly of the aircraft, in accordance with the right-hand rule. This coordinate frame is fixed to the body, meaning the aircraft's spatial orientation does not change the direction of the axes.

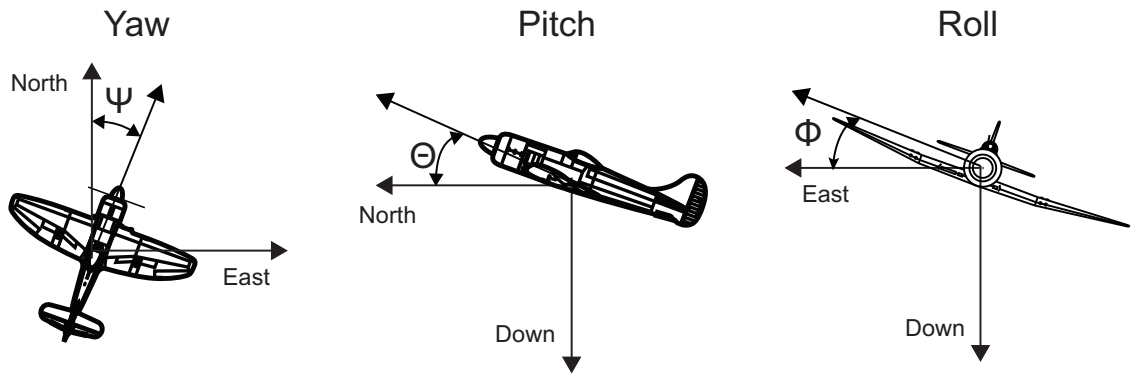


Figure 2.2: Body Axes Definition[3]

Stability Axes (x_s, y_s, z_s)

The stability axes are defined with its origin coinciding with the center of gravity of the vehicle. This axis system has essentially the same directions as the body axes, except rotated about the body axis $\hat{j}$ through an initial angle-of-attack, α_0. This inital angle-of-attack is defined at the beginning of a test maneuver and is then set for the remainder of the test, making it a body-fixed coordinate system. This system assumes no initial sideslip angle [4]. In Figure 2.3, only the $\vec{i}$ stability vector is shown, for clarity.

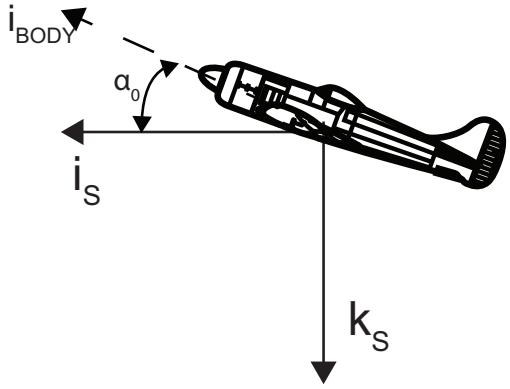


Figure 2.3: Stability Axes Definition

Wind Axes (x_w, y_w, z_w)

The wind axes are, again, a vehicle-carried coordinate system, meaning the origin of the wind axis also coincides with the center of gravity of the vehicle. However, the wind axes are not a body-fixed coordinate frame. The $\hat{i}$ direction points into the oncoming air, as seen from the vehicle. The $\hat{k}$ direction lies in the x-z plane of the body reference frame. The $\hat{j}$ direction is then defined to be out the right side of the vehicle, in order to follow the right hand rule.

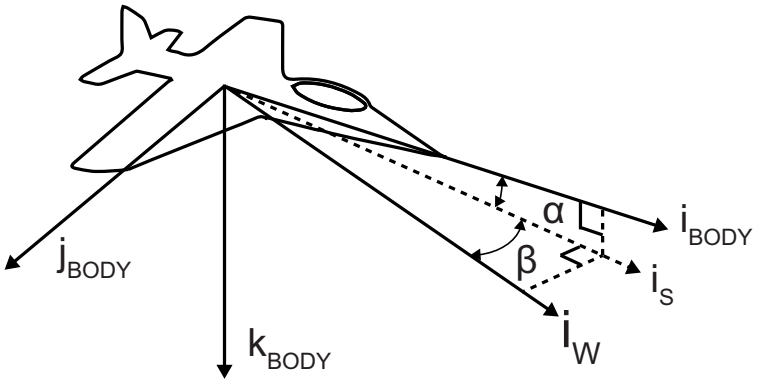


Figure 2.4: Wind Axes Definition

In Figure 2.4, only the $\vec{i}$ wind vector is shown, for clarity.

2.2 Equations of Motion

Newton's 2nd Law of Motion states

$$\vec{F} = \frac{d}{dt}(m\vec{V}) \tag{2.2.1}$$

where $\vec{F}$ is the sum of all applied forces, $\vec{m}$ is the mass of the vehicle, and $\vec{V}$ is the vehicle's velocity. Using the fixed mass assumption, this reduces to

$$\vec{F} = m\frac{d\vec{v}}{dt} \tag{2.2.2}$$

$$= m\vec{a} \tag{2.2.3}$$

The applied forces on the vehicle are

$$\vec{F} = \vec{F}_A + \vec{F}_G + \vec{F}_T \tag{2.2.4}$$

where $\vec{F_A}$ accounts for all aerodynamic forces acting on the vehicle, $\vec{F_G}$ is the force due to gravity, and $\vec{F_T}$ accounts for forces from the propulsion system.
Aerodynamic forces are described in the stability reference frame. In general, they are defined as

$$\vec{F}_{A_S} = D\hat{i}_s + Y\hat{j}_s + L\hat{k}_s \tag{2.2.5}$$

where D is drag force, Y is side force, and L is lift force.
The gravitational force on the vehicle acts in the $+z_{ned}$ direction and is equal in magnitude to the vehicle's weight W, leading to

$$\vec{F}_{G_{ned}} = 0\hat{i}_{ned} + 0\hat{j}_{ned} + W\hat{k}_{ned} \tag{2.2.6}$$

In general, propulsive forces are modeled as

$$\vec{F}_{T_b} = T_x\hat{i}_b + T_y\hat{j}_b + T_z\hat{k}_b \tag{2.2.7}$$

where T_x, T_y, and T_z are components of thrust in their respective body axis directions. However, as previously mentioned, propulsive forces are assumed to be $\vec{0}$ for this thesis.

The forces are combined and transformed into the body axes reference frame so that they align with the output of body mounted accelerometers. The combined equations of motion are then

$$\vec{F}_{AERO_W} = DCM_{bw}^{-1}(m\vec{a} - DCM_{ib}\vec{F}_{G_{ned}}) \tag{2.2.8}$$

The first term in Equation 2.2.8, DCM_{bw}, is a rotation matrix that rotates a vector in body axes into wind axes, through the wind angles α and β. The vehicle mass and weight

are m and $\vec{F}_G$, respectfully. Accelerations are shown by the variable $\vec{a}$. The rotation matrix DCM_{ib} requires the Euler angles between the NED frame and the body frame. These states, when taken together, are the minimum required states for drag polar estimation.

2.3 Kalman Filter Usage

This thesis utilizes multiple Kalman filters to estimate both regression coefficients and improved states. As a brief overview, the Kalman filter combines the measured state with a predicted state to give an optimal[5] estimate of the actual system state.

2.3.1 Linear Kalman Filter

A linear Kalman filter can be applied[6] where the system in question can be described in the form

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.3.1}$$

where A is the state transition matrix, x_{k-1} is the previous state, B is the input matrix, u_{k-1} is the input vector, and w_{k-1} is random process noise.

The measured state is then

$$z_k = Hx_k + v_k \tag{2.3.2}$$

where H is the output matrix and v_k is measurement noise.

The Kalman filter operates in a predictor-corrector manner, where the predictor step is often called the *a priori* estimate, and the corrector step is often called the *a posteriori* estimate. The *a priori* state estimate is calculated using prior states and inputs, while assuming no process noise

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{2.3.3}$$

The *a priori* estimate of the covariance matrix is projected in a similar manner

$$P_k^- = AP_{k-1}A^T + Q \tag{2.3.4}$$

where Q the process noise covariance matrix.

The Kalman gain is calculated by combining the predicted, *a priori* covariance matrix with the measurement noise covariance matrix R

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.3.5}$$

This optimal Kalman gain is then used to estimate the *a posteriori* estimate of the state and covariance matrix

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - y_k) \tag{2.3.6}$$

$$P_k = (I - K_k H)P_k^- \tag{2.3.7}$$

where y_k is the predicted value of z_k found using the output matrix H and the *a priori* state estimate

$$y_k = H\hat{x}_k^- \tag{2.3.8}$$

Note that Equation 2.3.6 is essentially a weighted average of a measured state and an expected state. The weighting is the Kalman gain, which is related to the ratio of confidence in the measured state and the expected state. For a 1-D case with equal confidence between the measured state and the expected state, the Kalman gain $K_k = 0.5$, and the Kalman Filter becomes a simple and straight-forward average.

2.3.2 Extended Kalman Filter

The Extended Kalman filter is used for a non-linear system and is essentially a linearization of a nonlinear plant. A non-linear system can be described as [6]

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{2.3.9}$$

$$z_k = h(x_k, v_k) \tag{2.3.10}$$

The process noise w_{k-1} and measurement noise v_k are not known (or the Kalman filter would not be necessary), so the states are approximated assuming both noise sources are 0

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{2.3.11}$$

$$\tilde{z}_k = h(\tilde{x}_k, 0) \tag{2.3.12}$$

The actual states are related to the approximate states by

$$x_k \approx \tilde{x}_k + A(x_k - \hat{x}_{k-1}) + W w_{k-1} \tag{2.3.13}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_{k-1}) + V v_k \tag{2.3.14}$$

where the matrices A, W, H, and V represent the different Jacobians matrices:

$$A = \frac{\partial f_i}{\partial x_j}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{2.3.15}$$

$$W = \frac{\partial f_i}{\partial w_j}(\hat{x}_{k-1}, u_{k-1}, 0) \tag{2.3.16}$$

$$H = \frac{\partial h_i}{\partial x_j}(\hat{x}_k, 0) \tag{2.3.17}$$

$$V = \frac{\partial h_i}{\partial v_j}(\hat{x}_k, 0) \tag{2.3.18}$$

The Extended Kalman filter uses these linearized equations to perform the same process as the linear Kalman filter. Again, the first step is to calculate the *a priori* estimate of the state and the covariance matrix

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{2.3.19}$$

$$P_k^- = A_k P_{k-1} A_{k-1}^T + W_k Q_{k-1} W_k^T \tag{2.3.20}$$

Next, the Kalman gain is calculated

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{2.3.21}$$

The Kalman gain is then used to calculate the *a posteriori* estimate of the state and covariance matrix

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - y_k) \tag{2.3.22}$$

$$P_k = (I - K_k H_k)P_k^- \tag{2.3.23}$$

where y_k is, as in the linear case, the predicted value of z_k, but calculated using the nonlinear output function and the *a priori* state estimate

$$y_k = h(\hat{x}_k^-, 0) \tag{2.3.24}$$

Unlike the linear Kalman filter, the Extended Kalman filter is not proven to be optimal. However, it has been utilized for a wide range of applications with excellent results.

3.0 Drag Meta-Modeling

Drag force comes from many different contributions, but can generally be split into drag that is independent of lift, and drag that is due to lift [7]. The summation of all sources of drag that are independent of lift is often called minimum drag ($C_{D_{min}}$ often used for the coefficient)[8], and is roughly constant, for a given Reynold's number and Mach number. The drag due to lift can be split into viscous drag-due-to-lift and inviscid drag-due-to-lift.

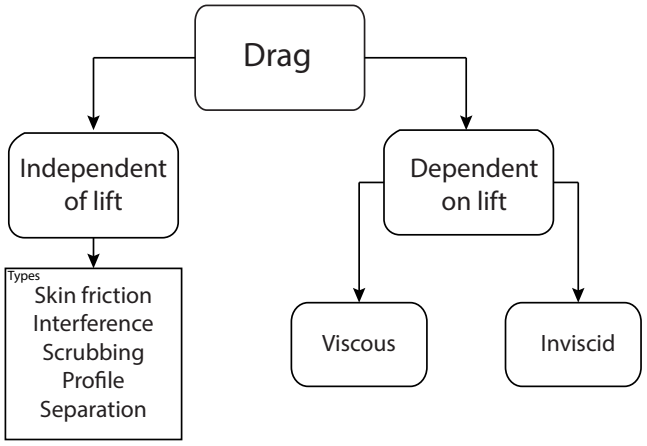


Figure 3.1: Drag Contribution Types

The viscous drag-due-to-lift is a type pressure drag that increases when the angle of attack of the wing increases, therefore generating more lift. It is a function of airfoil geometry, such as leading edge radius, thickness distribution, and camber. This type of drag is independent of finite wing vortices, and can be seen on two-dimensional airfoil data charts. To show an example of viscous drag-due-to-lift, a NACA 4412 was analyzed using XFOIL[9].

Figure 3.3 shows the airfoils viscous drag-due-to-lift. Since the shape is roughly parabolic through the linear region of the lift curve, the contribution to the total aircraft drag is

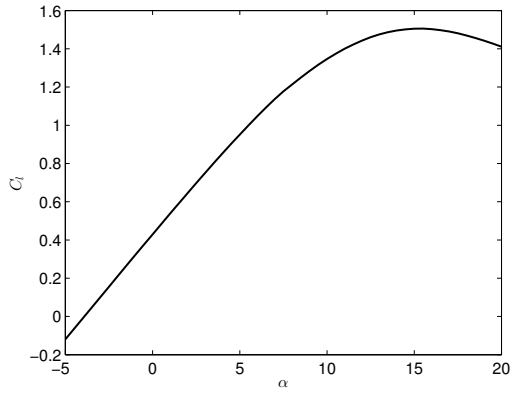


Figure 3.2: NACA 4412 Lift Curve

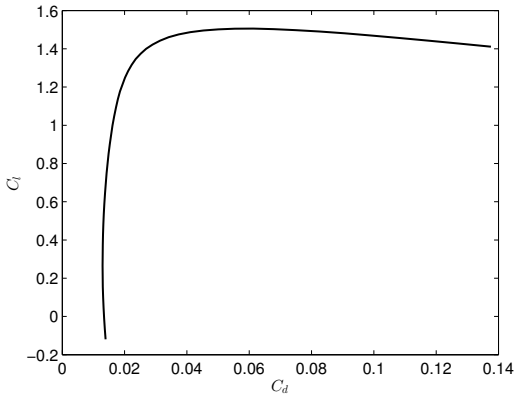


Figure 3.3: NACA 4412 Drag Polar

approximated using the Equation 3.0.1[7].

$$C_{D_{Visc.Lift}} = K_1 * (C_L - C_{L_{MinDrag}})^2 \tag{3.0.1}$$

where K_1 is the slope of the line shown in Figure 3.4, and $C_{L_{MinDrag}}$ is the lift coefficient at which minimum drag occurs (roughly 0.25 for this airfoil).

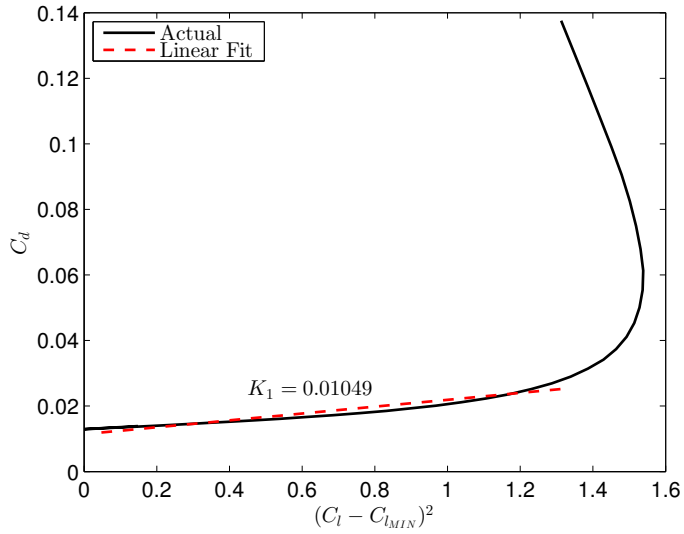


Figure 3.4: NACA 4412 K_1 Estimation

Inviscid drag-due-to-lift occurs because of the pressure difference between the top and bottom of a finite wing. This pressure difference causes the high pressure flow underneath

the wing to slip over to the top of the wing, causing a downwash velocity on the free stream velocity, shown in Figure 3.5.

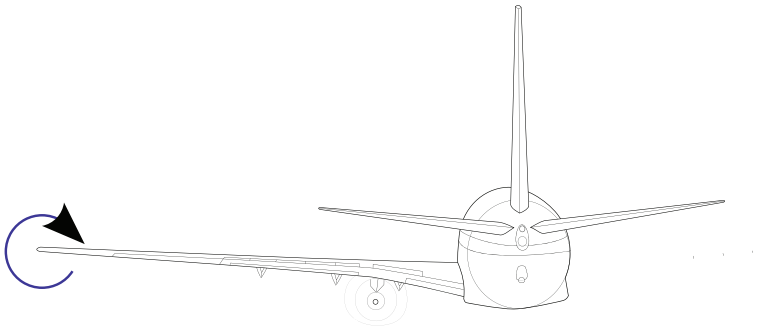


Figure 3.5: Downwash Caused By Wingtip Vortices[10]

This downwash changes the direction of the flow going into the airfoil. Since lift acts perpendicularly to the flow going into airfoil, there will be an induced angle between the free-stream lift vector (perpendicular to V_∞) and the airfoil's lift vector.

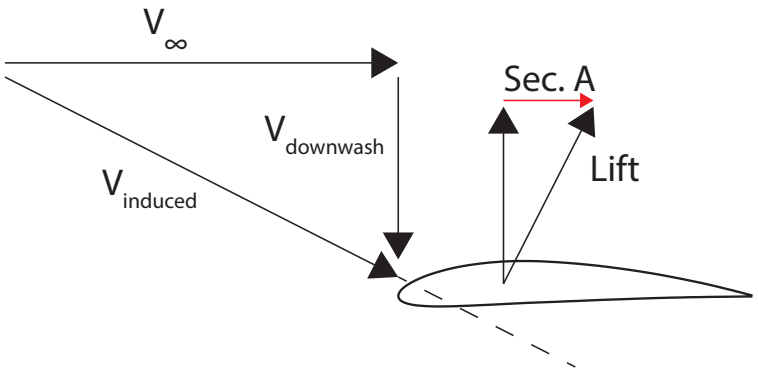


Figure 3.6: Induced Drag Free Body Diagram

The red section, labeled Sec. A in Figure 3.6, is a component of the airfoil's lift which is parallel to V_∞, meaning the airfoil's lift causes drag on the vehicle.

Induced drag is affected by the span lift distribution and the wing aspect ratio[11], and is governed by Equation 3.0.2.

$$C_{D_i} = \frac{C_L^2}{\pi e AR} = K_2 C_L^2 \tag{3.0.2}$$

These three coefficients (C_{D_0}, K_1, and K_2) are combined to result in a parabolic drag polar of the form

$$C_D = C_{D_{min}} + K_1(C_L - C_{L_{MIN}})^2 + K_2 C_L^2 \tag{3.0.3}$$

This drag polar gives valuable insight into how a vehicle will perform. The complete drag polar can be found using various regression techniques, discussed in the following sections.

3.1 Regression Model - Least Squares Fit

The standard model for a polynomial regression is

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + ... \tag{3.1.1}$$

When Equation 3.0.3 is expanded and like-terms of C_L are combined, equation 3.1.1 becomes

$$C_D = (C_{D_{min}} + K_1 C_{L_{min}}^2) - 2K_1 C_{L_{min}} C_L + (K_1 + K_2) C_L^2 \tag{3.1.2}$$

The value ($C_{D_{min}} + K_1 C_{L_{min}}^2$) is referred to as the parasite drag coefficient, and is represented by C_{D_0}[8]. These coefficients can be estimated using an Ordinary Least Squares fit. The Ordinary Least Squares problem statement is as follows

$$\bar{A}\vec{x} = \vec{b} \tag{3.1.3}$$

If $\bar{A}$ is an mxn matrix of measured state data, $\vec{x}$ is an nx1 vector of correlation coefficients, and $\vec{b}$ is an mx1 vector of measured function data, the solution to the Ordinary Least Squares problem is

$$\bar{A}\vec{x} = \vec{b} \tag{3.1.4}$$

$$\bar{A}^T\bar{A}\vec{x} = \bar{A}^T\vec{b} \tag{3.1.5}$$

$$(\bar{A}^T\bar{A})^{-1}(\bar{A}^T\bar{A})\vec{x} = (\bar{A}^T\bar{A})^{-1}\bar{A}^T\vec{b} \tag{3.1.6}$$

$$\bar{I}\vec{x} = (\bar{A}^T\bar{A})^{-1}\bar{A}^T\vec{b} \tag{3.1.7}$$

When applied to estimating a parabolic drag polar, the $\bar{A}$ matrix becomes

$$\bar{A}_{i,:} = \begin{bmatrix} 1 & C_{L_i} & C_{L_i}^2 \end{bmatrix} \tag{3.1.8}$$

the $\vec{b}$ vector becomes

$$\vec{b}_i = \begin{bmatrix} C_{D_i} \end{bmatrix} \tag{3.1.9}$$

and the $\vec{x}$ vector, which is the vector of interest, becomes

$$\vec{x} = \begin{bmatrix} C_{D_0} & -2K_1C_{L_{min}} & (K_1 + K_2) \end{bmatrix}^T \tag{3.1.10}$$

The coefficients C_{D_0}, K_1, and K_2 can then be found, assuming $C_{L_{min}}$ is known through wind tunnel testing, XFOIL, CFD, or other means.

3.2 Regression Model - Kalman Filter

The coefficients in question can also be estimated using an Extended Kalman filter. The system can again be described as

$$C_D = C_{D_0} - 2K_1C_{L_{min}}C_L + (K_1 + K_2)C_L^2 \tag{3.2.1}$$

For the Kalman filter regression, the state to be estimated are the coefficients C_{D_0}, $-2K_1C_{L_{min}}$, and $K_1 + K_2$. For ease of notation, substitute

$$C_1 = -2K_1C_{L_{min}} \tag{3.2.2}$$

$$C_2 = K_1 + K_2 \tag{3.2.3}$$

Since the regression coefficients should not change, the state transition matrix is an identity matrix, leading to

$$\hat{x}_k = \begin{bmatrix} C_{D_0} & C_1 & C_2 \end{bmatrix} \tag{3.2.4}$$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2.5}$$

The measured data z_k is a vector containing the lift and drag coefficients at the k-th instant in time

$$z_k = \begin{bmatrix} C_D \\ C_L \end{bmatrix} = h(x_k, 0) \tag{3.2.6}$$

The vector y_k contains the estimates of C_D and C_L found using the *a priori* state vector $\hat{x}_k^-$ and is equal to

$$y_k = \begin{bmatrix} C_{D_{0_k}}^- + C_{1_k}^- C_L + C_{2_k}^- C_L^2 \\ C_L \end{bmatrix} = h(x_k^-, z_k, 0) \tag{3.2.7}$$

To implement into the Extended Kalman filter, the Jacobian of $h(x_k^-, z_k, 0)$ with respect to x_k needs to be calculated. Once done, the H matrix in the EKF becomes

$$H_k = \begin{bmatrix} 1 & C_L & C_L^2 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.2.8}$$

With the H matrix calculated, the EKF algorithim can be implemented as described in Section 2.3.2. The measurement noise covariance at each instant was calculated by doing error propagation as described in Section 4.1 and the process noise covariance was set to zero because the parabolic regression coefficients should be exactly constant.

4.0 Error Analysis

There are two main types of error: model regression error and the error of a single data point. The error of a single data point comes from random noise in sensors, and can be decreased by improving the accuracy of the sensor or filtering the results. The model regression error comes from the fact that every aspect of the dynamics of the system are not precisely modeled. This type of error can be reduced by improving the accuracy of the dynamics being modeled or by increasing the amount of data being taken, as discussed in Section 4.2.

4.1 Random Error

Without estimates of the error in data, the data itself is fairly meaningless. The equations of motion can be used to propagate uncertainty in a signal, and allows the uncertainty in the coefficients to be estimated based on sensor noise. The noise can be either systematic or random. Systematic error can be caused by many things, including poorly modeled physics and a steady-state offset of a sensor. Random error is due to the inherent inaccuracy of the sensors, and is generally assumed to be normally distributed.

The error of a single point is assumed to be random error and normally distributed. The first step in error propagation is to define the y_i to be the i-th entry of the true function vector, $\hat{y_i}$ to be the i-th entry of the measured function vector, and to then do a Taylor series expansion about the operating point.

$$\hat{y_i} = y_i + \frac{\partial y_i}{\partial x_j} dx_j + HOT \tag{4.1.1}$$

where x_j is the j-th element of the state vector and HOT represents the truncated **H**igher **O**rder **T**erms of the Taylor series. Note that the term $\frac{\partial y_i}{\partial x_j}$ is the Jacobian ($\bar{J_{ij}}$) matrix of the

state transition function. The error can then be defined as (dropping $HOTs$)

$$dy_i = \hat{y_i} - y_i = \bar{J_{ij}} dx_j \tag{4.1.2}$$

If the error is then interpreted as a discrete difference instead of a continuous difference, equation 4.1.2 becomes

$$\Delta y_i = \bar{J_{ij}} \Delta x_j \tag{4.1.3}$$

If the Δ values are further assumed to represent standard deviations of normally distributed error, equation 4.1.3 becomes

$$\sigma_{y_i} = \bar{J_{ij}} \sigma_{x_j} \tag{4.1.4}$$

For the purposes of this thesis, σ_{y_i} is a vector of the standard deviations of the aerodynamic force coefficients,

$$\sigma_{y_i} = \begin{bmatrix} \sigma_{C_{D_i}} & \sigma_{C_{Y_i}} & \sigma_{C_{L_i}} \end{bmatrix}^T \tag{4.1.5}$$

and σ_{x_j} is a vector of the standard deviations of the state values,

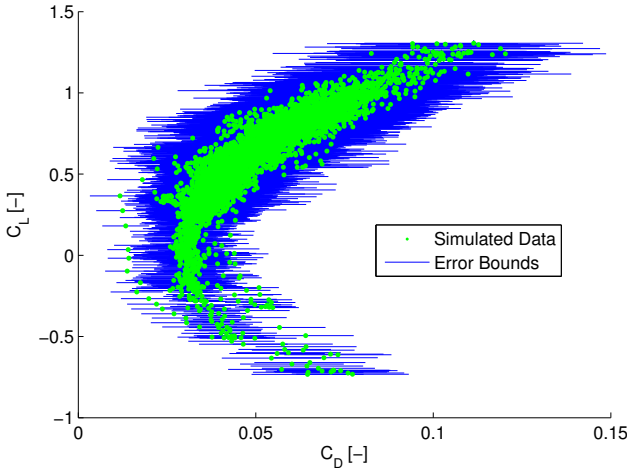
$$\sigma_{x_j} = \begin{bmatrix} \sigma_{\bar{a}_j} & \sigma_{\alpha_j} & \sigma_{\beta_j} & \sigma_{\bar{\omega}_j} & \sigma_{\phi_j} & \sigma_{\theta_j} & \sigma_{\psi_j} \end{bmatrix}^T \tag{4.1.6}$$

For initial error estimation, the noise levels reported by instrument manufacturers was assumed to be one standard deviation. The Jacobian matrix was calculated at each observed data point and combined with the sensors standard deviation to produce estimates of the standard deviations of the aerodynamic coefficients.

One of the key findings of this section is that the error of the coefficient depends on the coefficient itself. This means that the error varies from data point to point, which is called *heteroskedasticity* (as opposed to homoskedasticity, which means the error is independent of the state itself). To account for this, an error estimate function was created in MATLAB, which used the error propogation outlined in this section. This function was used whenever

an estimate of point error was required, such as the variance matrix P_k used in the Kalman filters utilized for state estimation. A plot of simulated flight data is shown in Figure 4.1, where the error bounds shown were calculated using the error estimate function. Note that this figure also shows the heteroskedastic nature of the error.

Figure 4.1: Heteroskedastic Error from Simulated Flight



Additionally, while heteroskedasticity does not color the estimate of $\hat{\beta}_i$, it does color the confidence intervals. The method of dealing with this problem is discussed in Section 4.2.

4.2 Least Squares Model Error

The error of a single data point is not the main driving factor in the accuracy of a regression model. The important factors in the model are how accurate the coefficients are known, which is a function of the accuracy of each point, as well as the number of points sampled. The main parameter that describes the accuracy of the regression coefficients are the confidence intervals. A confidence interval is a range of values such that, if the experiment were repeated, the parameter calculated would be within the range some percentage of the time. So, a parameter can be represented as an estimated value, with a confidence bound

$$\beta = \beta_{EST} \pm t\frac{\sigma}{\sqrt{n}} \tag{4.2.1}$$

where β is the parameter in question, β_{EST} is the estimated value of the parameter, t is the t value based on the number of samples and the desired confidence interval, σ is the standard deviation of the sample, and n is the number of data points collected. Since the number of data points collected will be large (> 100) in this thesis, the t value will be taken as 1.96 for a 95% confidence interval.

As previously mentioned in Section 4.1, one of the assumptions made in a Least Squares regression is homoskedasticity. However, the error using standard uncertainty propogation is heteroskedastic. This becomes a problem in estimating confidence intervals, because the standard error can be driven by outliers. If each data point had the same error, these outliers could be valid. However, if the data is heteroskedastic, the outlier may have a larger error bound, meaning the data point isn't as unlikely as it first appears. This fact can drive the standard error estimate to be larger than is appropriate, which leads to a larger confidence interval and possibly a false acceptance of the confidence interval's hypothesis test. To account for this, the `robustfit` function in MATLAB was used to estimate both the coefficients and robust standard error estimates. The `robustfit` function calculates robust standard error estimates by doing a weighted average, where the weighting is based on a radial basis function. This means that the farther away a data point is from the estimated regression model, the less impact it has on the standard error of the model. Some weighting functions for the weight w that are available in `robustfit` in MATLAB are shown below.

$$\text{“Cauchy”} : w = \frac{1}{1 + r^2} \tag{4.2.2}$$

$$\text{“Bisquare”} : w = \text{logical}(|r| < 1) * (1 - r^2)^2 \tag{4.2.3}$$

$$\text{“Fair”} : w = \frac{1}{1 + |r|} \tag{4.2.4}$$

$$\text{“Welsch”} : w = e^{-r^2} \tag{4.2.5}$$

The default weighting function used by `robustfit` is bisquare, so it was used for this thesis.

4.3 Kalman Filter Error

Kalman filters are often used to propagate states. The filter does this by combining the system dynamics with a measured state. The variance is propagated using Equation 2.3.23. When estimating coefficients using the Kalman filter, the $\bar{A}$ state transition matrix is an identity matrix, which is due to the fact that the coefficients stay constant. When propagated through the filter, this means the variance estimate P_k is essentially a variance-weighted-average of the coefficient estimates. The resulting matrix P_k contains the variance of the coefficient estimates. Equation 4.2.1 calculates the confidence interval of regression coefficients and needs the standard deviation of the mean, also called the standard error. The matrix P_k can be used to calculate the confidence intervals by noting

$$P_k = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_i^2 \end{bmatrix} \tag{4.3.1}$$

The confidence interval for coefficient β_i can be calculated using σ_i in Equation 4.2.1.

5.0 Simulation

A 6-DOF flight simulator was used to validated the drag prediction method before hardware was purchased. The main utility of the simulator was to provide simulated flight test data with signals that contained no errors. The actual sensors used for flight testing contain noise, and this noise can be added onto the pure simulator signals to test the sensitivity of the drag polar regression to sensor accuracy.

5.1 Simulation Environment

The flight simulator used was a model of the de Haviland Beaver that comes as a demo in the Aerospace Toolbox of Simulink. The Simulink model was modified to output required signals to the workspace, which essentially created a sensor with zero noise. The mass, moments of inertia, and reference lengths were then scaled to those of a Zagi R/C aircraft found in [12]. The original Simulink model was already connected to a Flight Gear visualization engine, but the model was slightly altered to make flight gauges function properly.

5.2 Simulation Inputs

The engine forces and moments were set to zero in the simulator, to match the assumption of a folding propeller. The drag force calculation built into the Beaver Simulink model were replaced with a parabolic drag polar of the form

$$C_D = C_{D_0} + K_1(C_L(\alpha) - C_{L_{min}})^2 + \frac{(C_L(\alpha))^2}{\pi e A R} \tag{5.2.1}$$

Airfoil data, including K_1, $C_{L_{min}}$, and $C_L(\alpha)$, was taken from nonlinear aerodynamic data of a NACA 4412. While this approximation to a drag polar does not capture the nonlinear section of profile drag rise due to stall, it does represent the limited lifting capability of a real wing, making it more realistic than assuming the wing does not stall.

5.3 Simulation Results

The first goal of the simulation testing was a verification of the correct equations of motion, and that the data analysis routines developed in Matlab did in fact match inputs to outputs. The simulation was initialized with various initial states to ensure there was no dependency on initial conditions. The vehicle was then flown by an R/C aircraft pilot using a joystick attached to the simulation. It was noted early in the simulation testing that flying a sweep of speeds was beneficial, as a wider range of the drag polar was flown. This result was included in much of the flight test planning. After adequate data had been taken, the data was analyzed without adding simulated sensor noise. The results are shown in Figure 5.1.

Figure 5.1: Data Analysis Verification (No Noise)

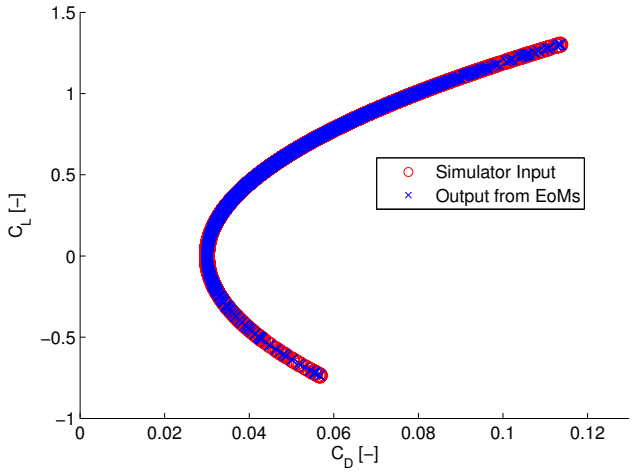
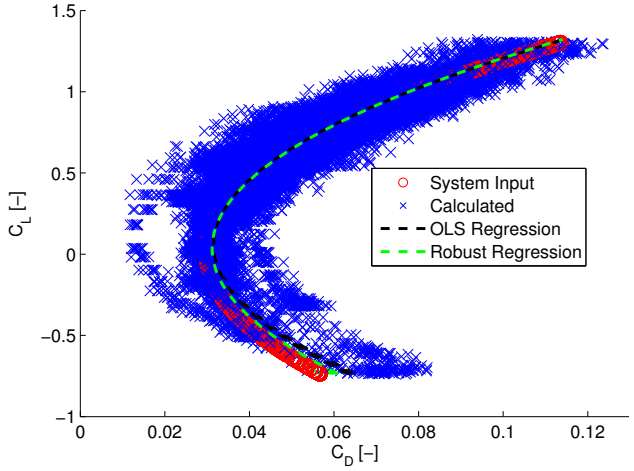


Figure 5.1 shows that the equations of motion used in the data analysis functions properly calculate the coefficients being passed into the system. With this result, noise was added to the system to see how sensitive coefficient estimation was to noise in each sensor. This process was a balancing act between available sensor accuracy and the desired accuracy of the final solution. The final result guided sensor selection to those discussed in Section 6. To check if the final sensors chosen were acceptable, Gaussian noise was added to each state, with a mean of 0 and a standard deviation equal to the root-mean-squared error listed in the manufacturer's data sheet for each sensor.

Figure 5.2: Drag Polar Prediction of Simulated Test Flight



For the particular simulated test flight shown in Figure 5.2, the estimated drag polar coefficients had error coefficients outlined in Table 5.1.

Table 5.1: Nonlinear Model Results

	C_{D_0}	K_1	K_2
System Inputs	0.0493	0	0.03
OLS Estimate	0.0516	-0.0056	0.0317
Robust LS Estimate	0.0500	-0.0035	0.0313

The results of this simulated flight test showed that the measurement system outlined in Section 6 predicted the simulated drag polar with a reasonable error.

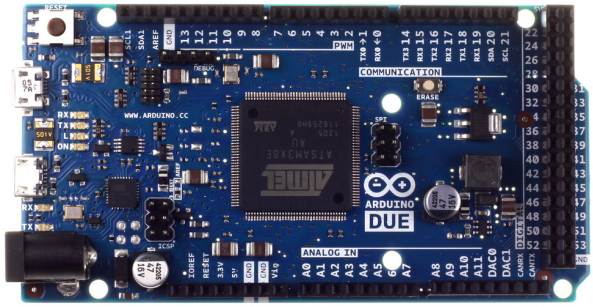
6.0 Hardware

Hardware selection was approached from a "minimum sensor" perspective. One of the main goals of this thesis was to reduce the number of sensors as much as possible to increase the variety of aircraft the hardware can fly on. To accomplish this, sensors were chosen based on which state they could estimate, with a main goal being to reduce overall size. The sensors were then combined into a single printed-circuit-board (PCB) that eliminated loose wiring and the potential for error that comes with it.

6.1 Flight Computer

The flight computer chosen was an Arduino Due. This board has a 32-bit ARM processor, 54 digital I/O pins, 12 analog input pins, and 2 analog output pins. The main driver in the decision to use an Arduino-based platform was the vast support community, which allowed quicker code development. The Arduino also offers a package that integrates well into most of the available airframes, and the stackable head pins allowed for easy integration with other boards. The Due in particular was chosen as it is (at the time of writing) the most advanced Arduino available. The main advantages it has over the comparable Arduino Mega is its increased clock speed (84 MHz for the Due[13] vs 16 MHz for the Arduino Mega[14]) and its 32-bit architecture (vs. 8-bit for the Arduino Mega). The Arduino Due is also capable of some single cycle floating point arithmetic[13], which is another advantage over the Mega, and was taken advantage of during GPS parsing.

Figure 6.1: Arduino Due Flight Computer



The Arduino Due uses a 3.3V architecture instead of the usual Arduino architecture, which uses a 5V operating voltage. This was mainly beneficial, since most of the selected sensors used 3.3V as both supply and logic voltage. An oscilloscope was used to check logic levels of the 5V sensors, and the logical levels were within the Due's voltage limits, so no logic level converters were required. The Due also supplies regulated 5V and passes through the supply power, which must be between 6V and 15V. The board will be powered through the 3.5mm barrel jack, using a 3-cell LiPo battery, with a nominal voltage of 11.1V.

6.2 Accelerations

The accelerometer chosen for this thesis was the ADXL-362 from Analog Devices. It has a noise error of $175\mu g/\sqrt{Hz}$ and uses a 3.3V digital SPI interface[15]. The accelerometer is in a QFN package and was surface mounted to the main PCB, with a decoupling capacitor between power and ground. The circuit was modeled after Sparkfun's ADXL-362 Breakout Board[16].

Each axis of the accelerometer was calibrated using a two step method. The first step was to orient the axis being calibrated in a direction not influenced by gravity. Data was acquired for 10 seconds and a simple average was taken. This value was then subtracted from every reading to correct for zero offset. The second step was to take two different readings: first, with the positive axis in the gravity direction; and second, with the negative axis in the

Figure 6.2: ADXL-362 Schematic

gravity direction. Both readings were taken after removing the zero offset using step one. A linear line was fit between a nominal $\pm -32.2\frac{ft}{s^2}$ gravity value.

6.3 Vehicle Mass

All test vehicles were weighed using a U-Line H-1650 counting scale. The scale has an accuracy of 0.001 lbs and a maximum capacity of 30 lbs. The minimum capacity of the scale is 10 grams [17].

6.4 Euler Angles

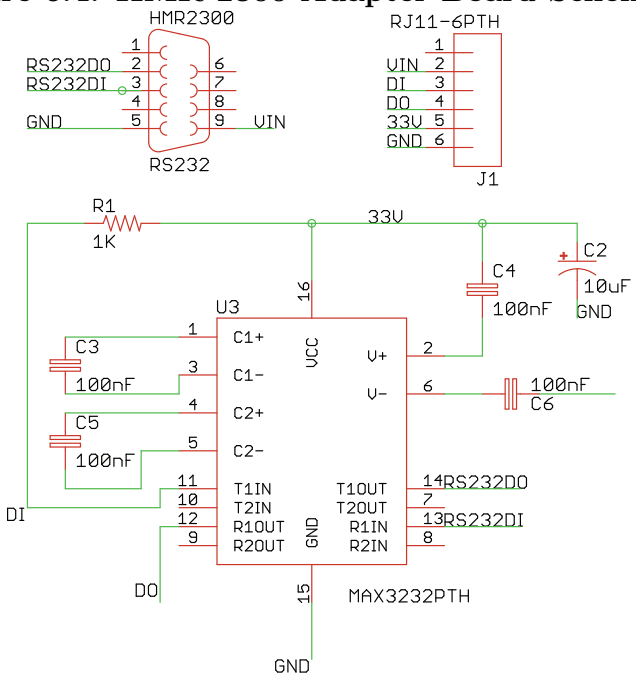
Euler angles in general can be estimated using either an 3-d electronic compass or a 3-d magnetometer. A magnetometer reads the direction and magnitude of a magnetic field, while a compass combines a magnetometer with accelerometer readings to produce a more accurate estimate of the true angles. However, if the vehicle is maneuvering, the craft's acceleration will distort the angle estimation, making compasses unsuitable for this application. Two separate magnetometers were used for separate purposes. A Honeywell HMR-2300 3-d magnetometer is the main magnetometer and is used when the most accurate sensing is desired.

Figure 6.3: Honeywell HMR-2300 3-d Magnetometer

This magnetometer provides a RMS error of 0.1 milliGauss for all axes, using the 1 Gauss full-scale setting[18]. This corresponds to an angle accuracy of .

The HMR-2300 can be supplied with power between 6V and 15V, so the 3-cell 11.1V nominal LiPo battery that powers the Arduino also passes through to power the magnetometer. Additionally, the HMR-2300 operates using an RS-232 serial interface. To properly interface with the Arduino Due, which uses 3.3V TTL logic levels, a Max-3232 integrated circuit (IC) was used. This IC, when combined with charge pump capacitors, translates TTL levels between 3V and 5.5V to RS-232 logic levels of ±6V.

Figure 6.4: HMR-2300 Adapter Board Schematic



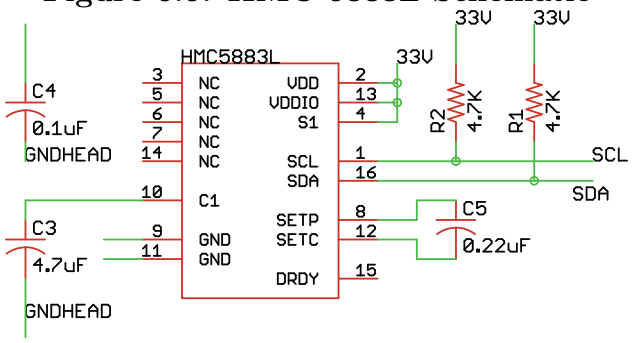
This board was connected to the magnetometer using a DB-9 connector, and was connected to the flight computer using an RJ-11 cable.

The second magnetometer is a Honeywell HMC-5883L, which comes in a QFN package that was surface mounted to the main sensor board. It was added to the system for two main reason: it is much smaller for applications where size is critical, and it is much less expensive for testing with unproven vehicles. It communicates with the Arduino using an I^2C interface and uses a 3.3V operating voltage[19]. The circuit used was similar to that used by Sparkfun on their HMC-5583L breakout board[20].

The HMC-5883L has an accuracy of 2 milliGauss on each axis, which corresponds to an accuracy.

Both magnetometers were calibrated for bot soft-iron and hard-iron effects using a method documented in[21]. To do this, data was acquired for one minute with the magnetometer

Figure 6.5: HMC-5883L Schematic



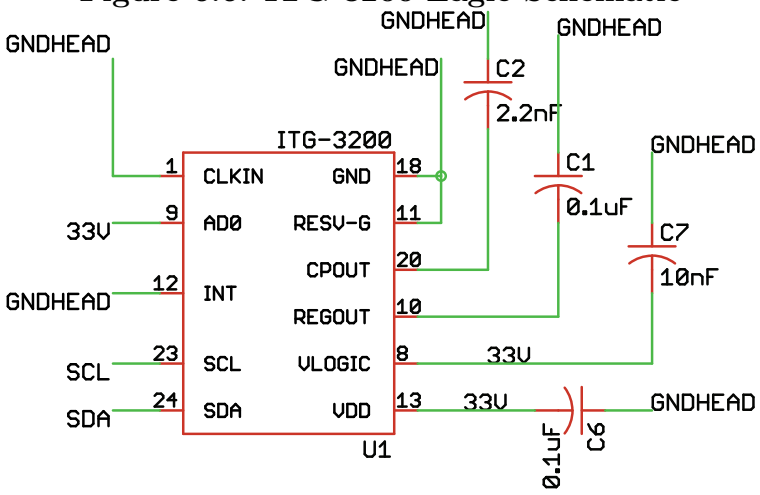
being swept through all directions. An ellipsoid was fit to the data using a ordinary least squares method available from the MATLAB file exchange[22]. The least squares fit estimates the center and radius of each axis. The center values for each axis was subtracted from the readings to remove hard-iron effects. Each i-th axis is then scaled by $\frac{1}{R_i}$ to "squish" the ellipse into a circle, which removes soft-iron effects.

Euler Angle Kalman Filter

For increased accuracy, a discrete linear Kalman filter is applied. This necessitated adding a three-axis gyroscope to the system. The gyroscope chosen was the Invensense ITG-3200, which comes in a QFN package. This gyroscope has a total error of 0.38°/s-rms[23], and uses a digital I^2C interface on a 3.3V operating voltage. The gyroscope has a full-scale span of ±2000°/s. The gyroscope was integrated into the main sensor board using a circuit based on that of Sparkfun's ITG-3200 breakout board[24].

The gyroscope was calibrated using a two-step process similar to that of the accelerometer. First, zero-offset noise was removed from each axis by acquiring data for 10 seconds and subtracting the simple average from future readings. Second, the gyro was placed on a turn table which was set to $33\frac{1}{3}$ RPM. Each axis was first lined up in the positive direction and next in the negative direction, and a linear fit was applied between the two data points.

Figure 6.6: ITG-3200 Eagle Schematic



With the angular rates available from the gyroscope, the system equations for the Kalman filter are

$$\begin{bmatrix} \phi_k \\ \theta_k \\ \psi_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_{k-1} \\ \theta_{k-1} \\ \psi_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta T & 0 & 0 \\ 0 & \Delta T & 0 \\ 0 & 0 & \Delta T \end{bmatrix} \begin{bmatrix} p_k \\ q_k \\ r_k \end{bmatrix} + \hat{w}_{k-1} \tag{6.4.1}$$

$$z_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \phi_k \\ \theta_k \\ \psi_k \end{bmatrix} + \hat{v}_{k-1} \tag{6.4.2}$$

The measurement covariance noise matrix was calculated using the standard deviation of the error between the magnetometer's ellipsoid fit and the acquired, calibrated data itself. The process noise covariance matrix was calculated using the standard deviation of the zero offset calibration data for the rate gyroscope.

6.5 Wind Angles

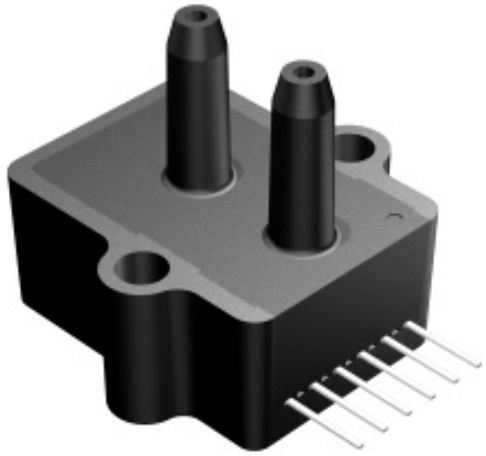
The accuracy at which the aerodynamic wind angles are calculated is critical to the overall prediction accuracy. In keeping with the goal of minimizing the number of required sensors, an attempt was made to estimate aerodynamic angles without directly measuring

them. There has been a plethora of work conducted on this subject. One of the first available papers on the subject was from the Air Force Institute of Technology [25]. In that paper, two methods were developed: one for in flight estimation, and the other for post-flight estimation. Both methods relied on either estimated or known stability derivatives. Since the purpose of this thesis is to estimate part of the dynamics, this estimating scheme will not work. Other techniques assume linearization about an operating condition [26]. For R/C aircraft this assumption generally cannot be made due to visual flight rules. Other work [27] combined a dynamics model and a no-vertical-wind assumption. The overarching theme of this previous estimation work was that too many assumptions needed to be made to get results that were not accurate enough ($\approx 1^\circ - 2^\circ$) for drag polar estimation. For this reason, it was necessary to directly measure airflow angles.

A five-hole probe was chosen to measure aerodynamic angles as they do not contain moving parts and can provide very accurate, repeatable data. The five-hole probe selected was the Aeroprobe Air Data probe. It has a length of 6 inches and a diameter of 1/8 inch and comes calibrated to pressures.

Each pair of lines of the air data probe is connected to an All Sensors digital differential pressure sensor with a full scale range of $\pm$5"-H_2O[28].

Figure 6.7: All Sensors 5-INCH-D-DO Pressure Sensor



These pressure sensors have a total error band of 0.25% of full scale. They use a UART serial interface that operates on a 5V logic level, but the logic levels are compatible with the 3.3V interface of the Arduino Due. The serial interface includes addressable read commands, which allows multiples devices on a single bus, and ensures all devices record pressure at the same time. The sensor can output both a 14-bit pressure reading and a 12-bit temperature reading, which the devices uses to correct its pressure measurement.

The pressure sensors were calibrated using the same two step process as the gyroscope and accelerometer. The zero offset was obtained by connecting port A to port B and taking data for 10 seconds. Pressure was applied for the linear fit using a Paroscientific Model 745 Pressure Standard, capable of 0.008% pressure accuracy[29].

6.5.1 Wind Angle Kalman Filter

To improve the accuracy of the wind angle estimation, a discrete Extended Kalman filter was used. The state transition functions are nonlinear and are

$$\dot{\alpha} = \frac{1}{V\cos\beta}(-a_x \sin\alpha + a_z \cos\alpha) + q - (p\cos\alpha + r\sin\alpha)\tan\beta \tag{6.5.1}$$

$$\dot{\beta} = \frac{1}{V}(-a_x \cos\alpha \sin\beta + a_y \cos\beta - a_z \sin\alpha \sin\beta) + p\sin\alpha - r\cos\alpha \tag{6.5.2}$$

These state transition equations come from solving for the vehicle forces in wind axes instead of body axes[1]. The equations for the Kalman filter for the wind angles are

$$\begin{bmatrix}\alpha_k \\ \beta_k\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha_{k-1} \\ \beta_{k-1}\end{bmatrix} + \begin{bmatrix}\Delta T & 0 \\ 0 & \Delta T\end{bmatrix}\begin{bmatrix}\dot{\alpha_k} \\ \dot{\beta_k}\end{bmatrix} + \hat{w}_{k-1} \tag{6.5.3}$$

$$z_k = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\alpha_k \\ \beta_k\end{bmatrix} + \hat{v}_{k-1} \tag{6.5.4}$$

The process covariance matrix was calculated using the error propagation discussed in Section 4.1 and the standard deviation data from the zero offset calibration of each of the sensors. The measurement noise covariance matrix was calculated based on the standard deviation data for the zero offset pressure transducers connected to the five-hole probe.

6.6 Additional Sensors

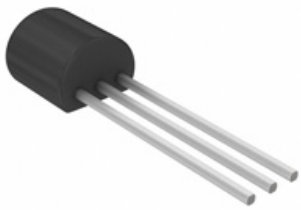
A uBlox LEA-6T GPS receiver was included in the data acquisition system to aid in mission visualization. This model was selected for its ability to output raw timing data, which can potentially be used to get an extremely accurate inertial velocity estimate[30]. The receiver itself was integrated onto a board sold by CSG Shop and has UART, USB, and I^2C interface options.

Figure 6.8: CGS Shop Board for uBlox LEA-6T

A barometric altitude sensor was also included the data acquisiton system. The model chosen was an All Sensors BARO-DO, which has a range of 600 mBar to 1100 mBar[31]. It has a nominal error of 1.0 mBar. It has the same packaging and communication protocol as the All Sensors 5-INCH-D-DO used for wind angle measurement, which simplified integration. It was calibrated to remove zero offset, but was not checked for linearity. The zero offset was removed by comparing the output data to the Paroscientific Model 745 Pressure Standard.

A digital temperature sensor was combined with the barometric altitude sensor to estimate the air density.

Figure 6.9: Dallas Semiconductors' DS18B20 Digital Temperature Sensors



The DS18B20 from Dallas Semiconductors was chosen for its relatively simple One-Wire interface. The device can be powered with the communication line and has a $\pm 0.5°C$ nominal accuracy[32]. It was calibrated across multiple measurements using the Paroscientific Model

745 Pressure Standard.

Header pins capable of reading commanded PWM signals to servos were also added. Future work could include stability derivative estimation, and the header pins provide PWM measurement, which can map to servo angles, if it is assumed the servo is not stalled.

6.7 Data Acquisition Integration

The hardware was initially built using a breadboard to ensure proper connections and to develop software. Once completed, sensors were packaged into a shield for the Arduino. This shield plugs directly into the Arduino, eliminating the need to disconnect and reconnect wiring. A board responsible for interfacing the main sensor board with the HMR-2300 was also developed, and connects to the main board with an RJ-11 cable. This allows the magnetometer to be placed with relative freedom within a vehicles fuselage. A third board was developed to integrate the pressure sensors with the main sensor board. This pressure board can be located near a wing tip and provides expandability should additional sensors be desired in the future. It also interfaces the temperature sensor. Finally, all data was saved to a microSD card attached to the main sensor board. Data was saved in binary format for both increased speed and file size reductions. Once on the ground, the data is converted to meaningful values using a MATLAB data parser.

7.0 Flight Test

Flight testing was done at Cal Poly's Educational Flight Range (EFR). The goal of flight testing was to validate the performance of the final system design. The drag polar estimation was chosen as the validation case, and was approached from each of the three coefficients. To this end, the final system was flown on multiple vehicles which each had a different role in the validation routine.

7.1 C_{D_0} Validation

Validation of the parasite drag coefficient was completed using a Finwing Universal Penguin FPV R/C aircraft[33].

This model was selected because it has a large internal payload bay which has plenty of room for the system and had enough excess power to overcome additional drag. The validation method involved flying the base model and measuring the drag polar. Then, parasite drag was added in the form of streamers, similar to those used in amateur rocketry recovery. The benefit of this technique is that it does not modify either of the other two drag polar coefficients. A power law fit to the expected drag coefficient of the streamers was given in [34]. The validation routine was flying the vehicle with and without streamers and seeing the horizontal shift in the drag polar.

7.2 K_1 Validation

The K_1 term of the drag polar is mainly driven by the C_L for minimum drag. To validate this coefficient, a custom test vehicle was manufactured with two different wings of equal area. One wing had a NACA 63-014 airfoil cross section, which is a symmetric airfoil, meaning the vehicles K_1 value should be zero. The next wing had a NACA 63-614 airfoil, which has a

design C_L for minimum drag of 0.6. The validation method was flying both of these wings on the same vehicle, and seeing the vertical shift in the drag polar.

7.3 K_2 Validation

The K_2 drag polar coefficient is the inviscid drag due to lift term. This form of drag comes from the wing tip vortices of a finite wing, and is affected by both wing aspect ratio and lift distribution. Prandtl showed experimentally[11] that, for rectangular wings with aspect ratios between one and seven, the drag coefficient corresponding to one aspect ratio can be scaled to that of a different aspect ratio using the equation

$$C_{D,1} = C_{D,2} + \frac{C_L^2}{\pi e}(\frac{1}{AR_1} - \frac{1}{AR_2}) \tag{7.3.1}$$

The K_2 validation uses this fact by flying the same test aircraft with two different sets of wings with the same area. The first wing has an aspect ratio of three, while the second wing has an aspect ratio of seven. The $AR = 3$ wing was then corrected to the $AR = 7$ wing and the resulting drag polar should fall on top of each other.

8.0 Results

9.0 Summary

This is the summary chapter.

A.0 System Usage

This section documents the steps required for correct usage of the data acquisition system.

A.1 Zero Offset Calibration

Before installing the data acqusition system in the aircraft, wind-off zero calibration should be acquired for the pressure sensors. To do this, connect one port to the other on each of the differential pressure sensors, then acquire data for some amount of time. Take note of these values.

A.2 Post Flight Zero Offset Calibration

After the flight is complete, a second round of zero offset calibration data should be acquired. This set ensures that any drift that occurred during the flight is accounted for. Repeat the steps outlined in Section A.1.

A.3 Software Usage

Let's talk about how to use that sweet GUI of yours.

B.0 Flight Test Procedure

This section documents the flight test procedure for using the data acquisition system. It is split into three time periods

1. Pre-Flight Preparation
2. Flying Field Procedure
3. Post-Flight

The testing procedure is split into these time categories to ensure the testing is efficient and that any unforseen circumstances can be dealt with quickly. Specifically , this guide applies to the Cal Poly Flight Lab testing a vehicle at the Cal Poly Educational Flight Range. In general, it is better to flight test early in the morning, since there will be less people at the field, winds will be calmer, and the sun won't be as harsh in the pilot's eyes.

B.1 Pre-Flight Preparation

The preparation work required for a test flight is often overlooked, and this section will documents how to effectively flight test a vehicle. The main purpose of pre-flight preparation is to minimize the possibility of problems that might force a test to be canceled after already going to the field.

B.1.1 Day Before Test

The following should be done the day before a flight test:

1. Charge all flight battery packs, including any receiver or auxillary packs.
2. Charge the transmitter battery.

3. Ensure the airframe is structurally sound (wing tip test minimum.)

4. Verify radio system communicates properly, and the correct fail-safe is in place. **Important**: If the receiver has been used by Design/Build/Fly, the fail-safe should be changed to normal mode.

5. Verify control surface deflections matches desired directions, and all radio mixes work.

6. Verify the motor/propeller spin in the correct direction.

7. Pack a flight box, containing any necessary tools (recommend: screw drivers, various tapes, razor blades, CA glue, spare propeller, paper and pencil, as a minimum.)

8. Check the weather. The closest monitoring station to the field is KCASANLU17.

9. Check the SLO Flyers' flight schedule. Some days are reserved for certain events (glider competitions, etc.) and these need to be worked around.

10. Make sure all required personnel know when and where to meet, and there are sufficient rides to get to the field.

11. Create flight documentation that clearly lays out the test goals and how they will be accomplished. Print copies for all personnel.

B.1.2 Day Of Test

The following should be done the morning of a flight test:

1. Pack flight batteries into flight box, including receiver and auxillary packs.

2. Pack battery charging equipment, with adapters and leads, if necessary.

3. Pack transmitter into box.

4. Double check control surface deflections and radio link.

5. Do a full system check, potentially including a short taxi test in the quad.

6. Do a final check that all equipment made it into vehicles, before leaving the lab.

B.2 Flying Field Procedure

With the pre-flight preparation completed, the testing at the flying field should be fairly event free. Any problems that occur should be either fixable with the minimum supplies in the flight box, or the test should be canceled to minimize risk, and repairs done at the lab. Specific procedures at the flight field will depend on the test being conducted, but below are steps that apply to nearly all tests before flight.

1. Verify structural integrity using a wing tip test.

2. Verify motor/propeller are spinning in the correct direction.

3. Verify radio link and fail safe mode.

4. Verify control surface deflections match desired directions.

5. Verify center of gravity is at an appropriate location.

6. Create flight timer on the transmitter so the pilot knows how long the aircraft has been flying.

C.0 C_{D_0} Flight Test Card

D.0 K_2 Flight Test Card

E.0 Sample System Ouput

This appendix contains graphs representing typical outputs from the data acquisition system, shown in strip chart format.

E.1 Sample System Ouput - Raw Data

Figure E.1: accelX vs. Time

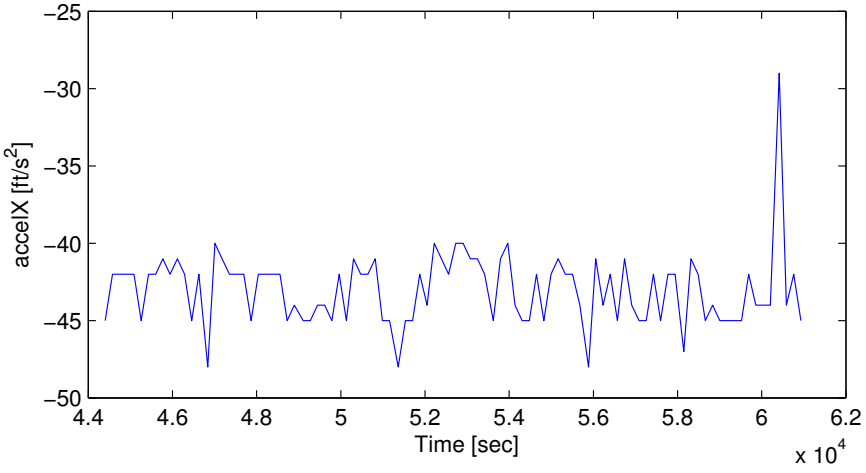


Figure E.2: accelY vs. Time

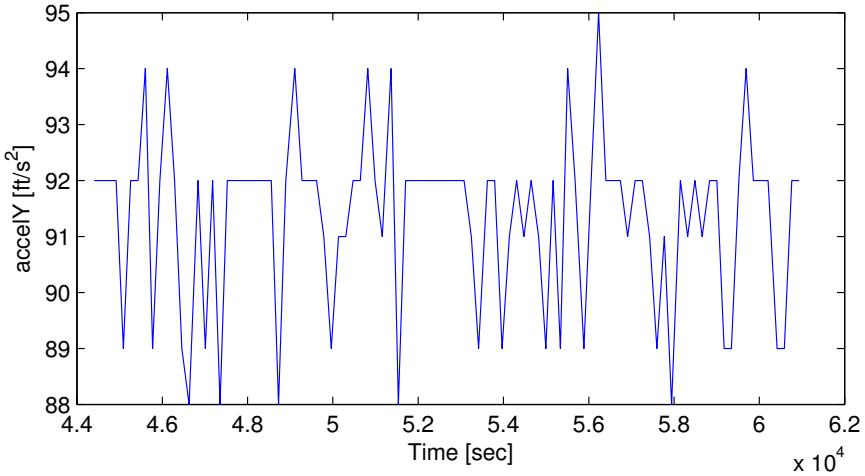


Figure E.3: accelZ vs. Time

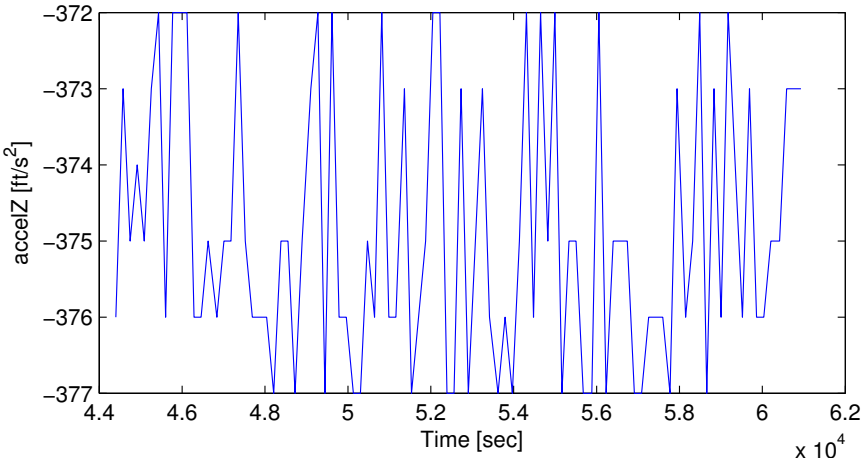


Figure E.4: gyroX vs. Time

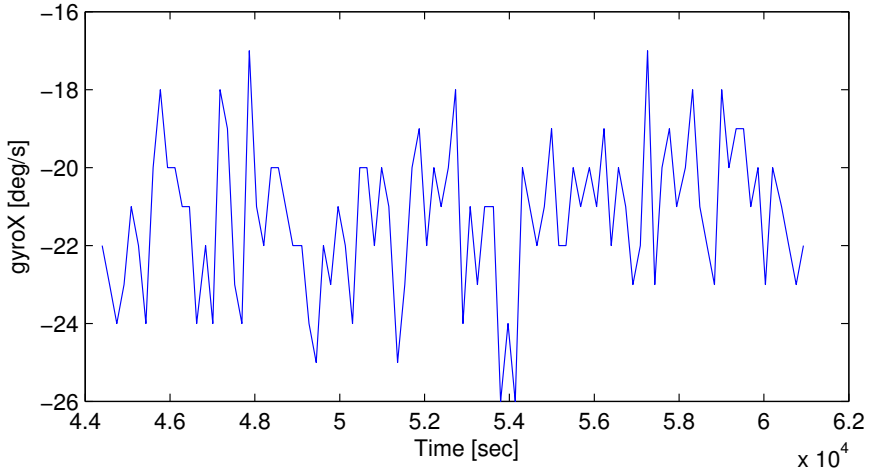


Figure E.5: gyroY vs. Time

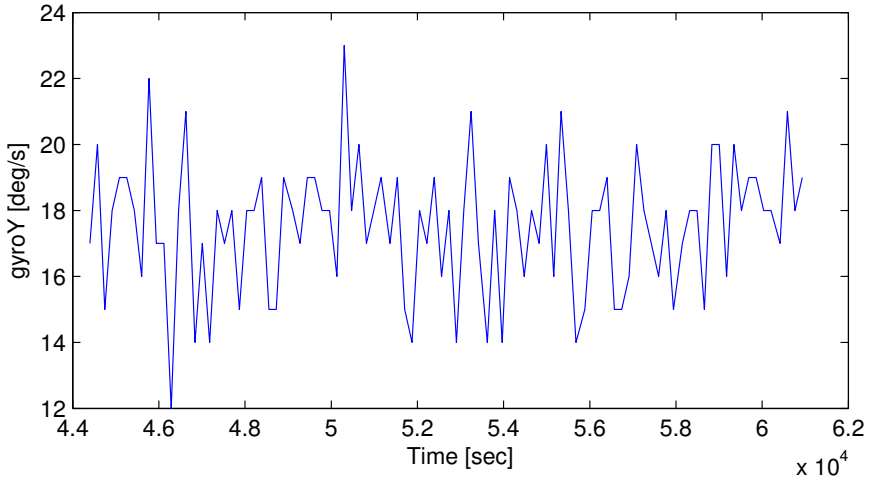


Figure E.6: gyroZ vs. Time

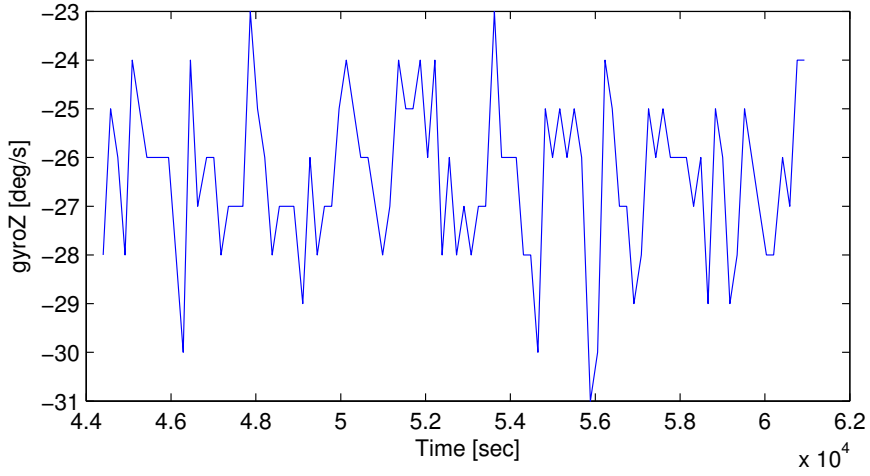
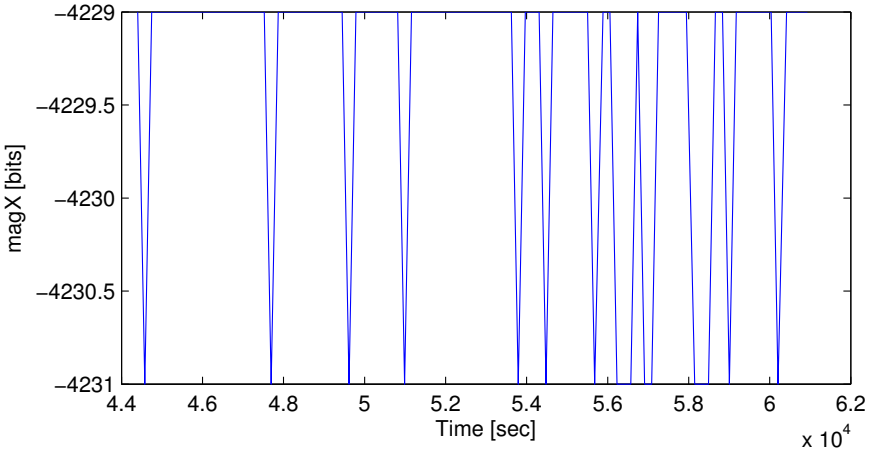


Figure E.7: magX vs. Time



E.2 Sample System Ouput - Units Data

Figure E.8: magY vs. Time

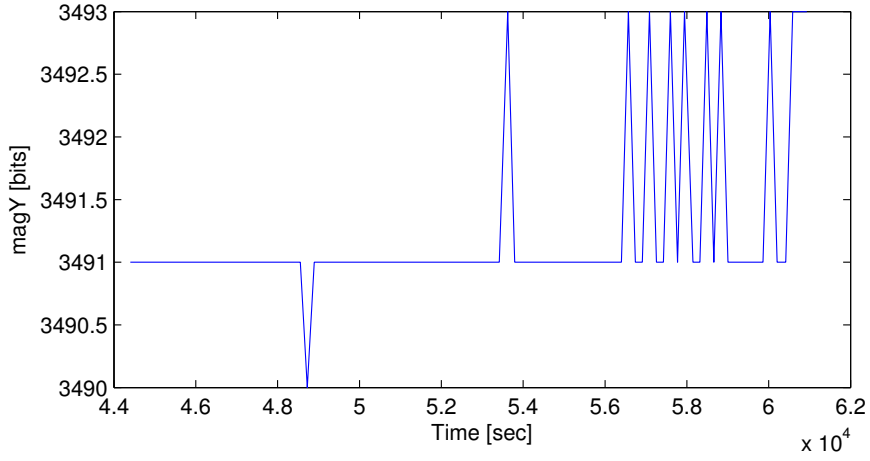


Figure E.9: magZ vs. Time

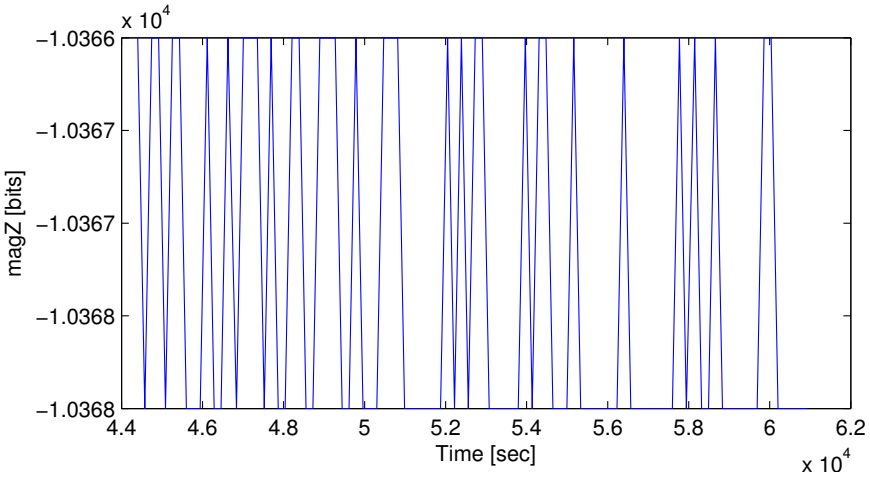


Figure E.10: press0 vs. Time

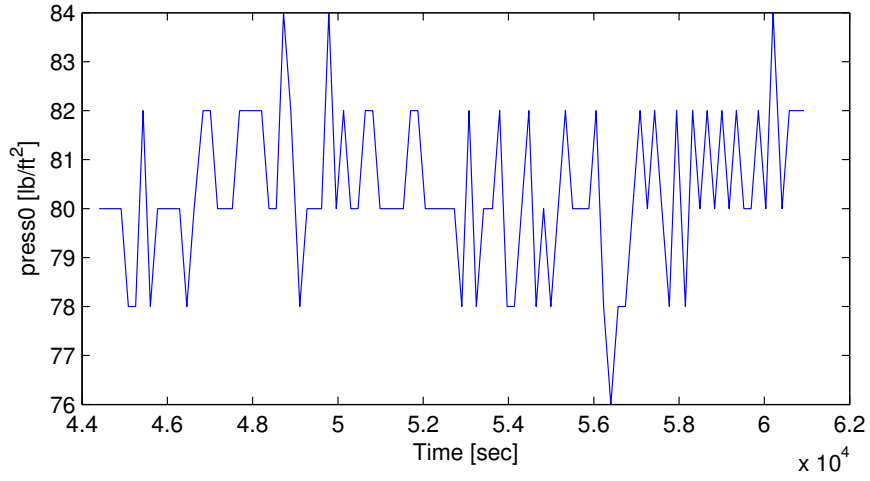


Figure E.11: press1 vs. Time

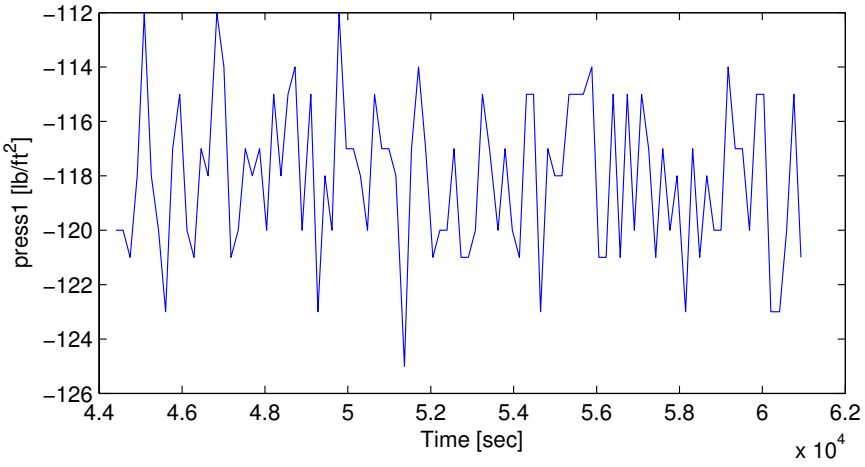


Figure E.12: press2 vs. Time

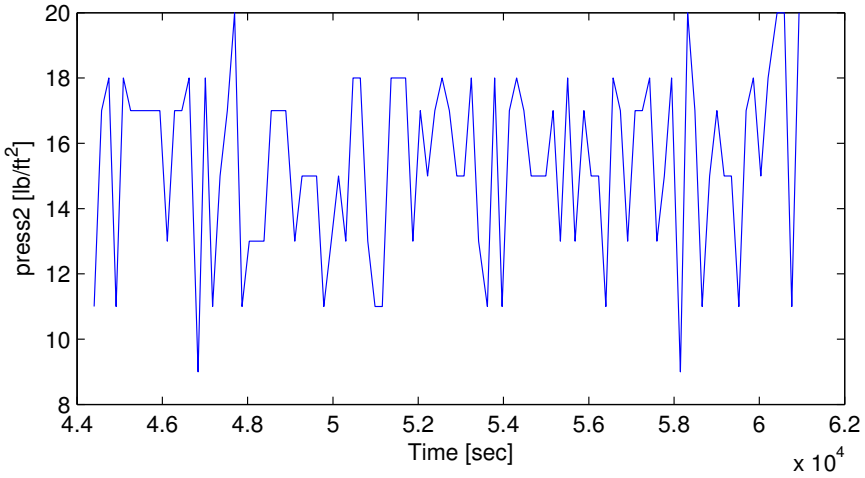


Figure E.13: press3 vs. Time

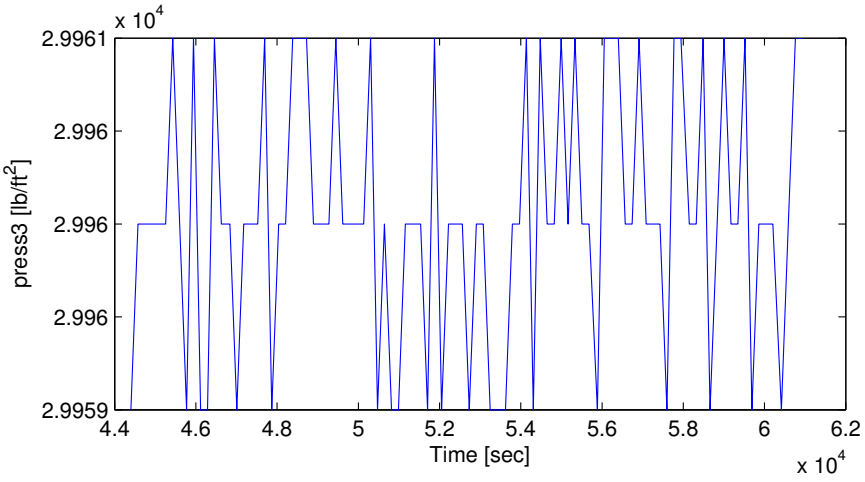


Figure E.14: gpsLat vs. Time

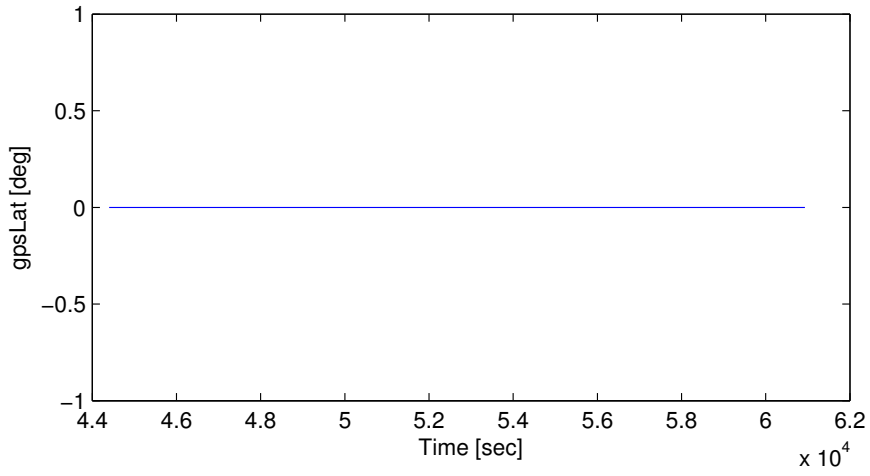


Figure E.15: gpsLong vs. Time

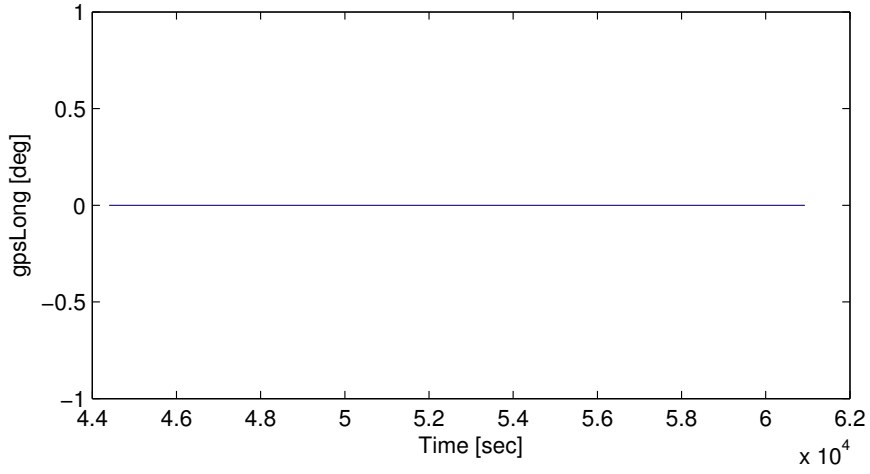


Figure E.16: gpsSpd vs. Time

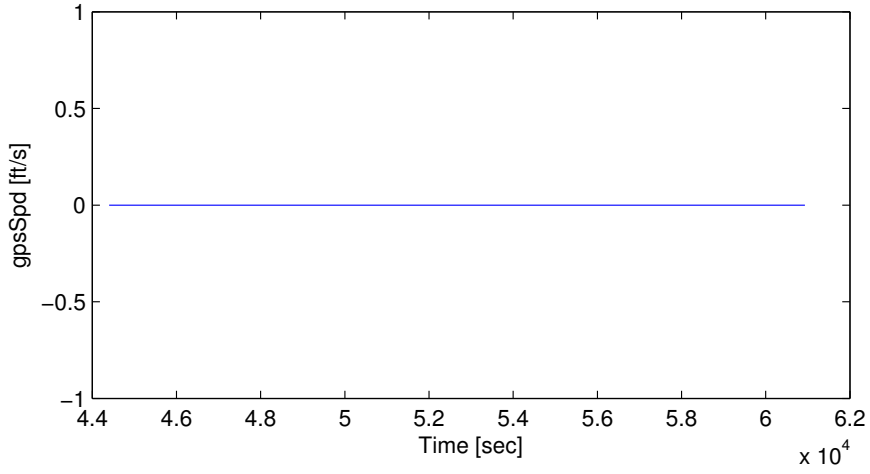


Figure E.17: gpsCrs vs. Time

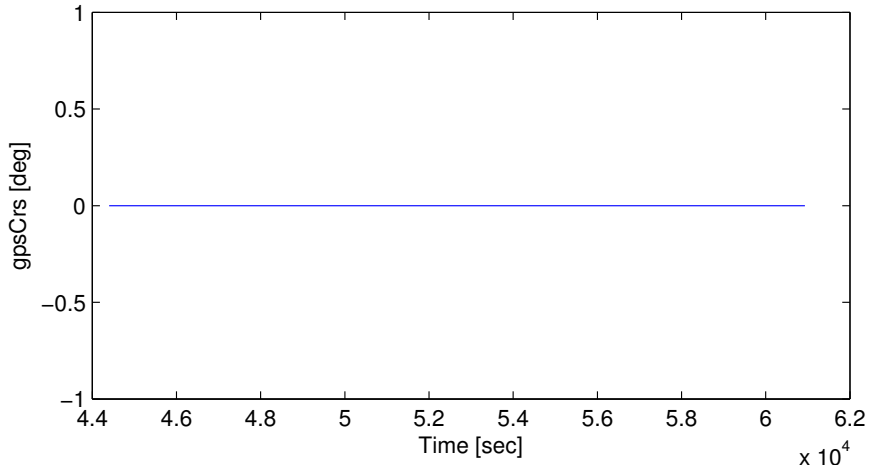


Figure E.18: date vs. Time

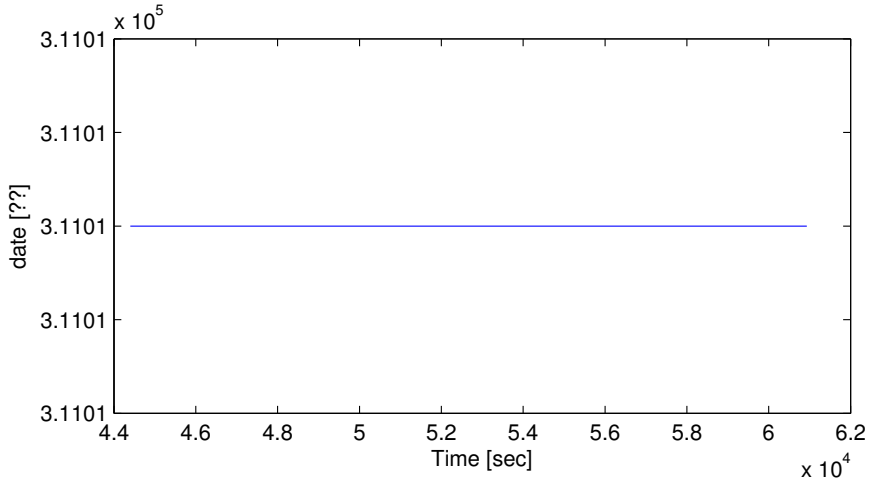


Figure E.19: CS vs. Time

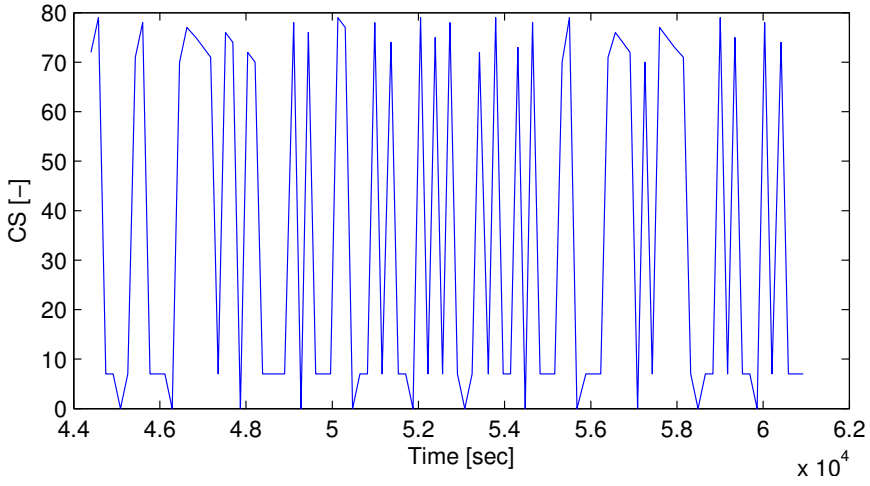


Figure E.20: temperature vs. Time

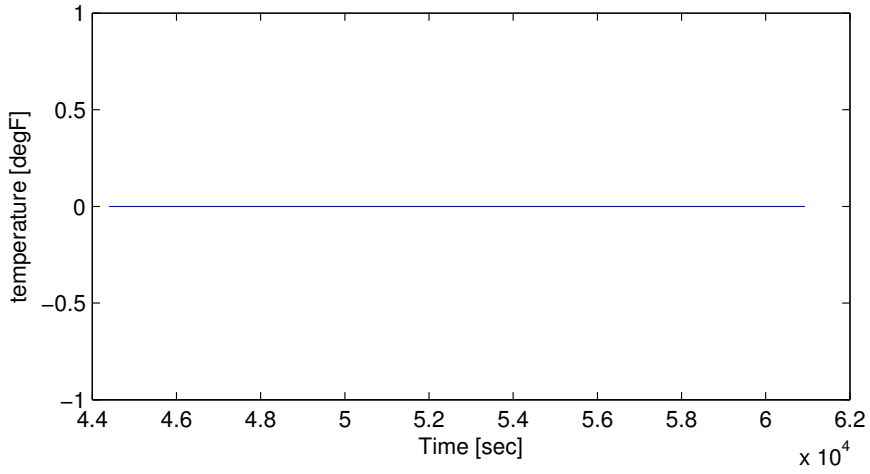


Figure E.21: deltaT vs. Time

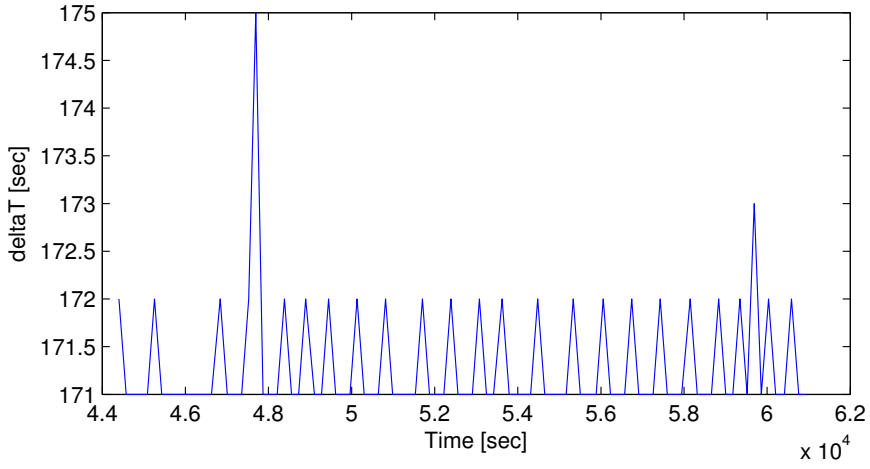


Figure E.22: rpmPwm vs. Time

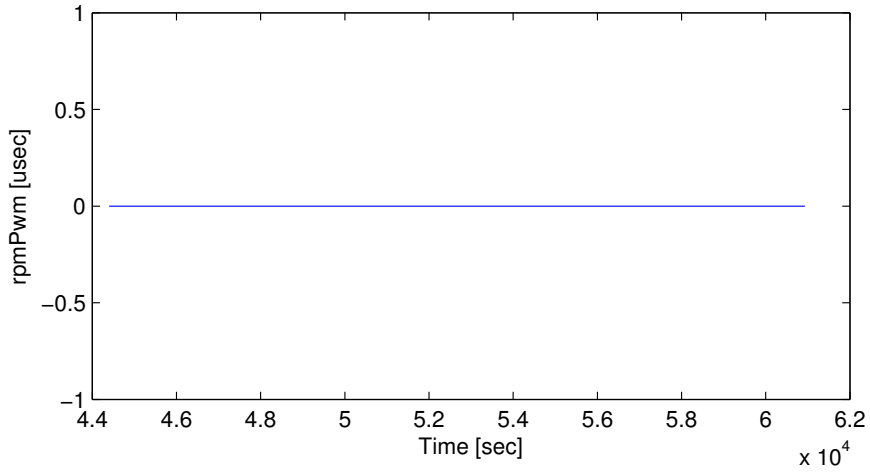


Figure E.23: accelX vs. Time

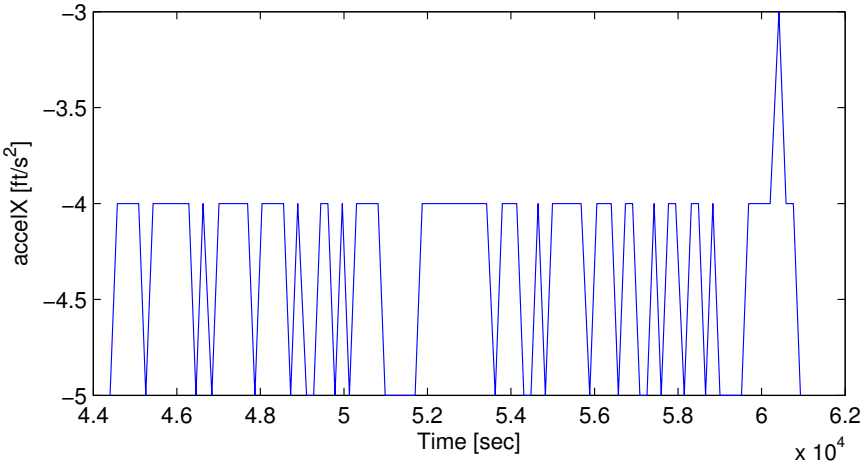


Figure E.24: accelY vs. Time

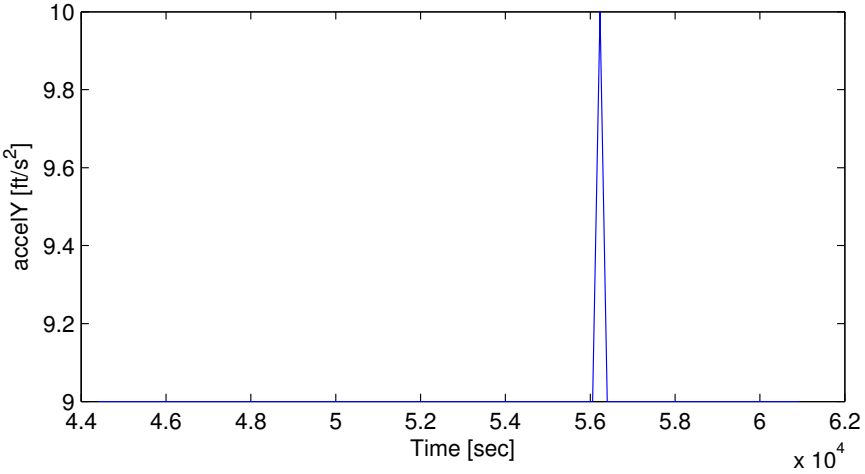


Figure E.25: accelZ vs. Time

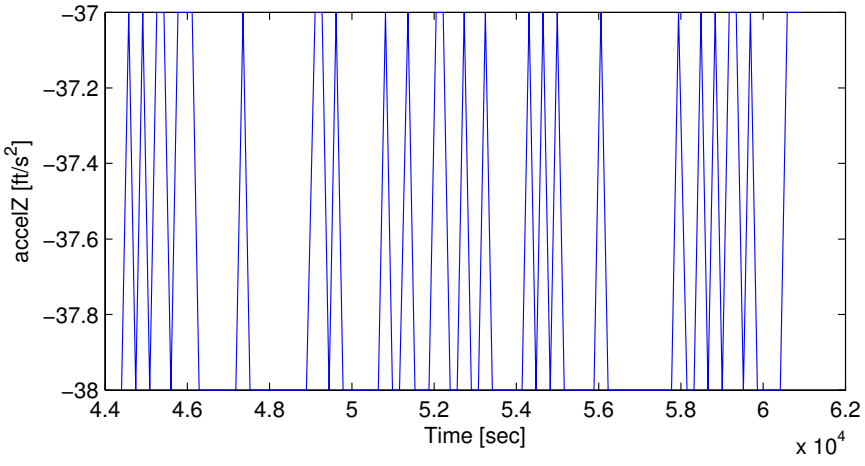


Figure E.26: gyroX vs. Time

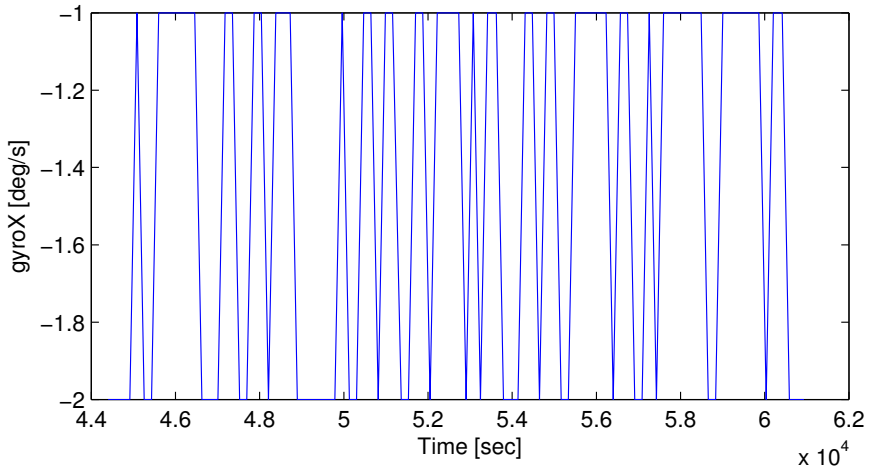


Figure E.27: gyroY vs. Time

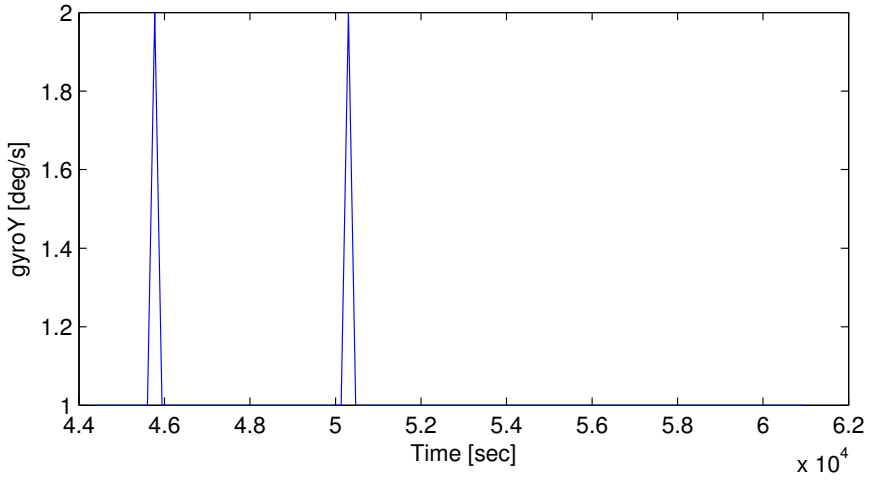


Figure E.28: gyroZ vs. Time

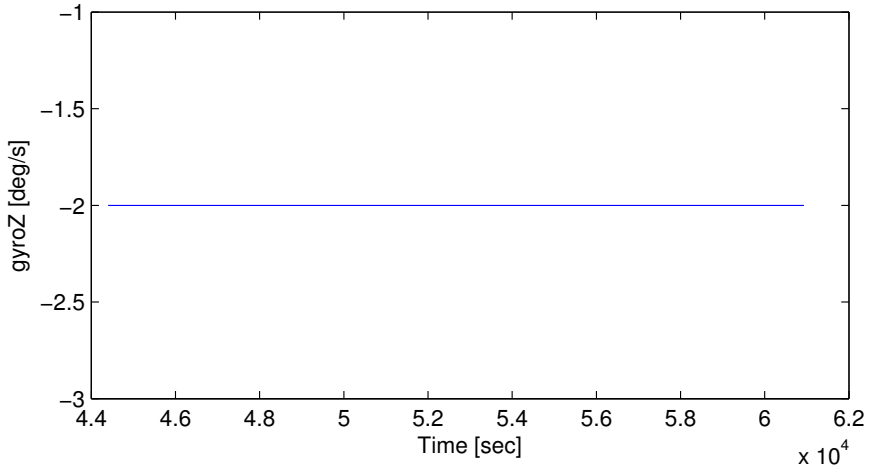


Figure E.29: magX vs. Time

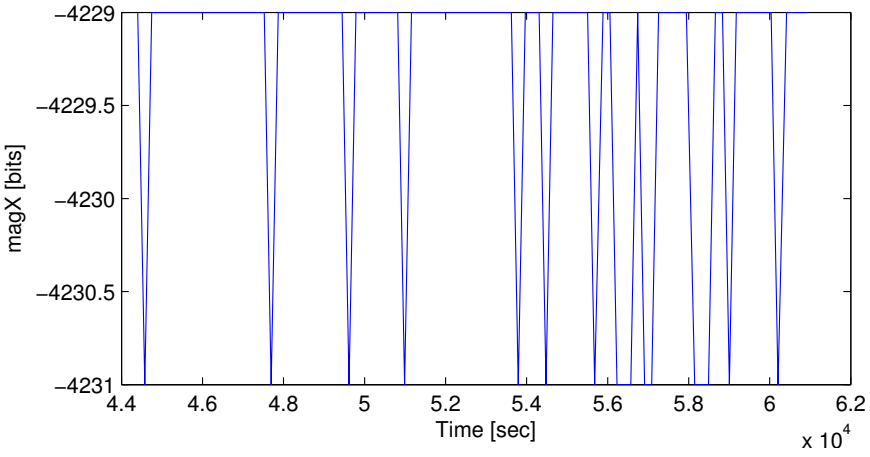
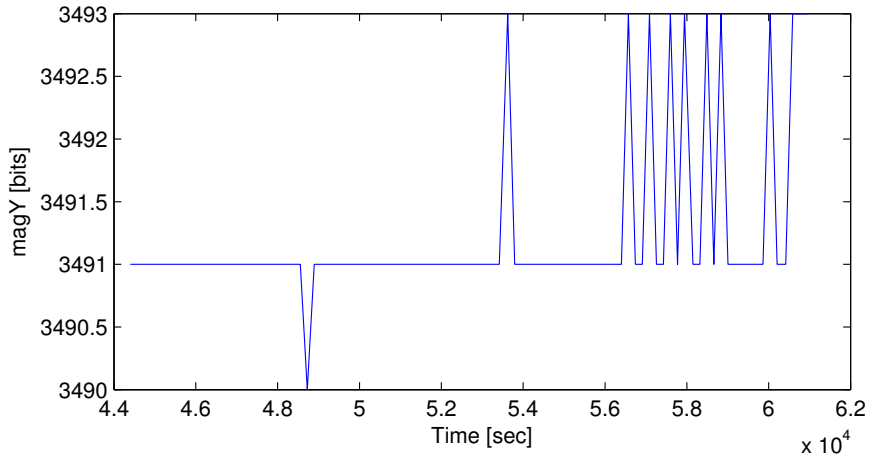


Figure E.30: magY vs. Time



E.3 Sample System Ouput - State Data

Figure E.31: magZ vs. Time

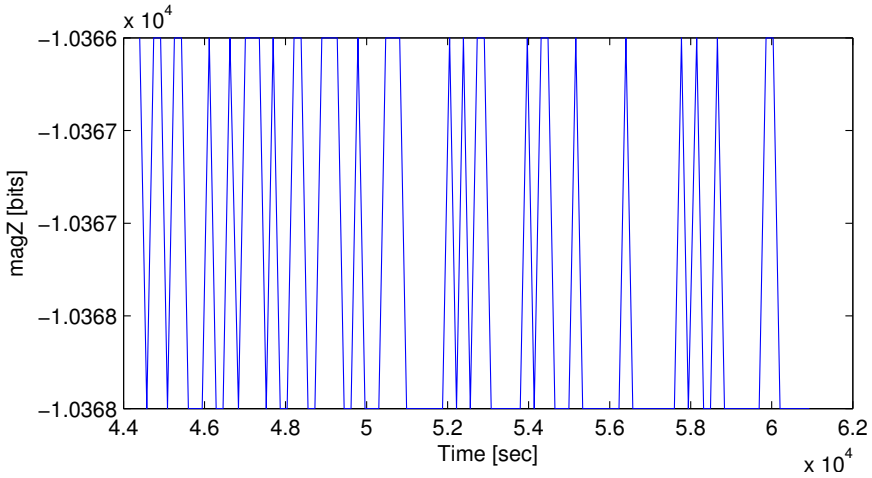


Figure E.32: press0 vs. Time

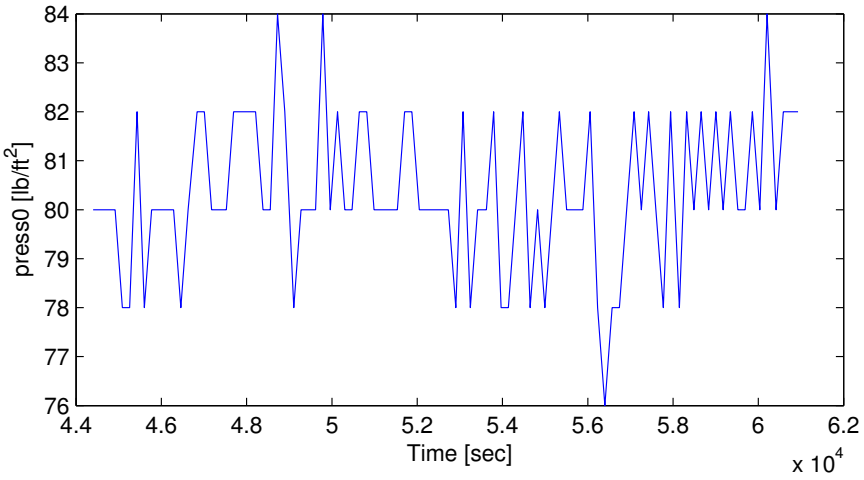


Figure E.33: press1 vs. Time

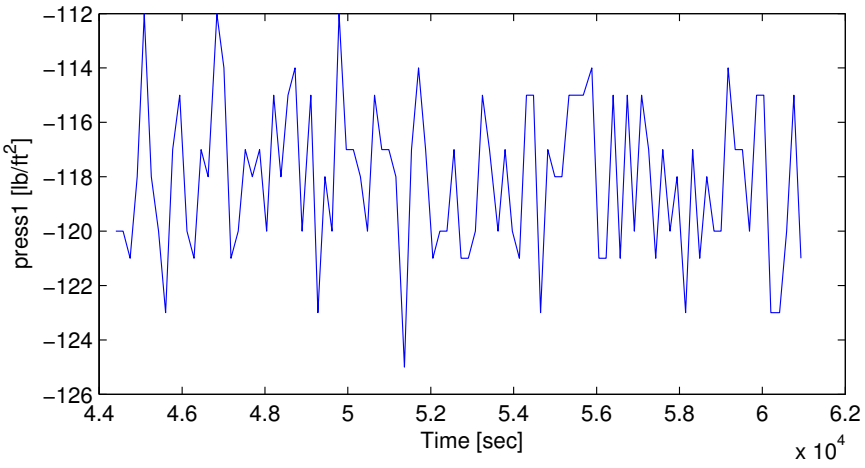


Figure E.34: press2 vs. Time

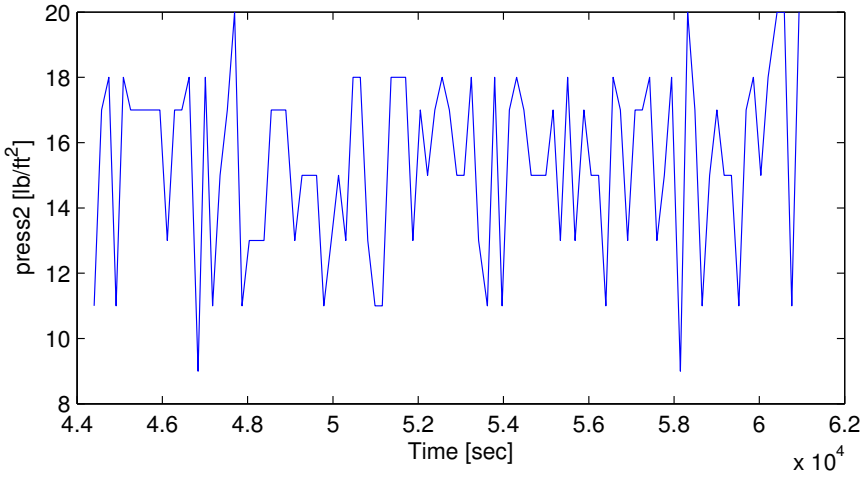


Figure E.35: press3 vs. Time

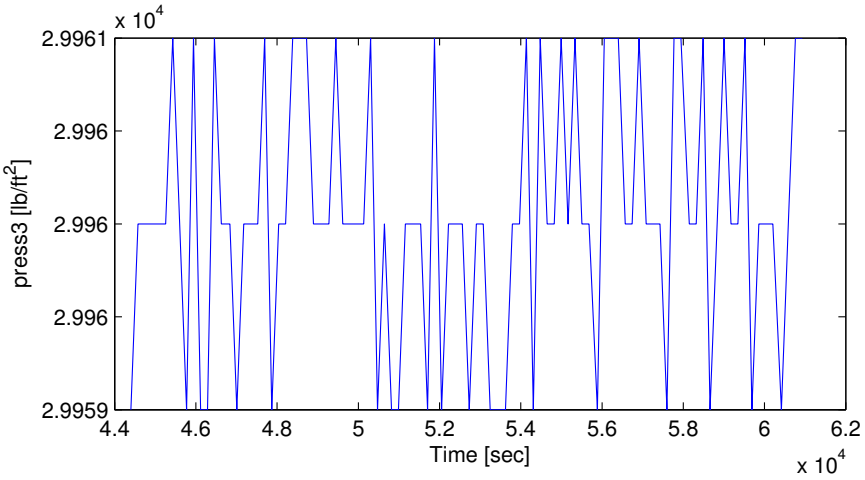


Figure E.36: gpsLat vs. Time

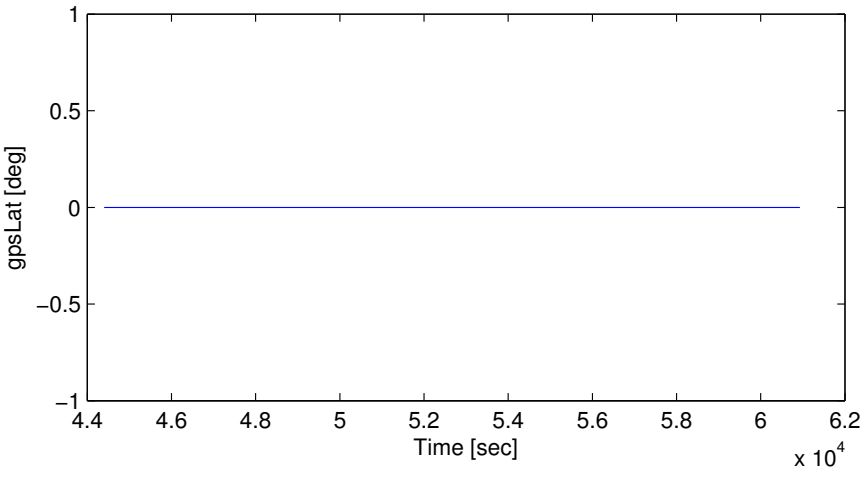


Figure E.37: gpsLong vs. Time

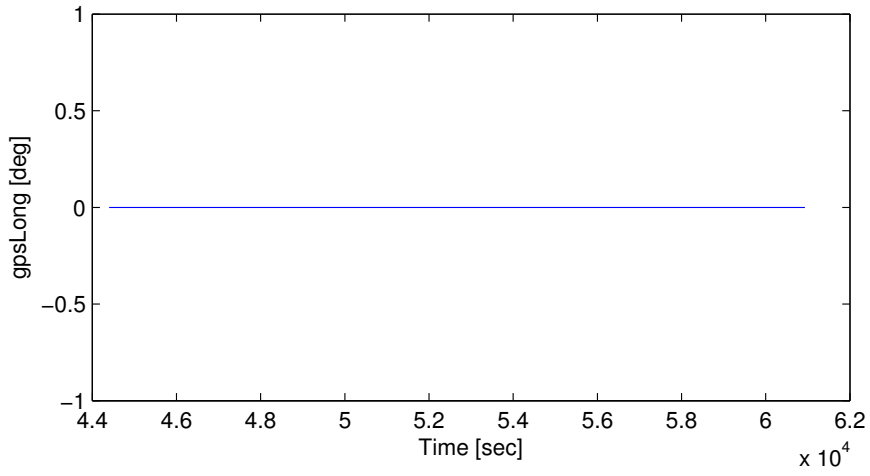


Figure E.38: gpsSpd vs. Time

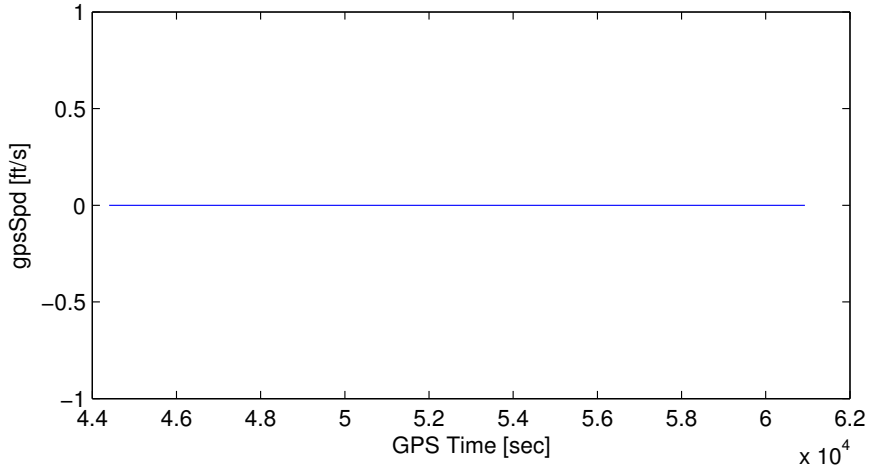


Figure E.39: gpsCrs vs. Time

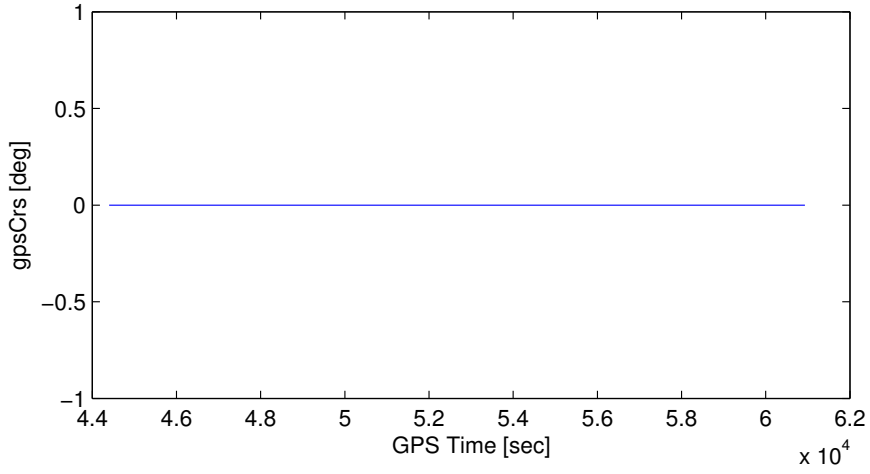


Figure E.40: date vs. Time

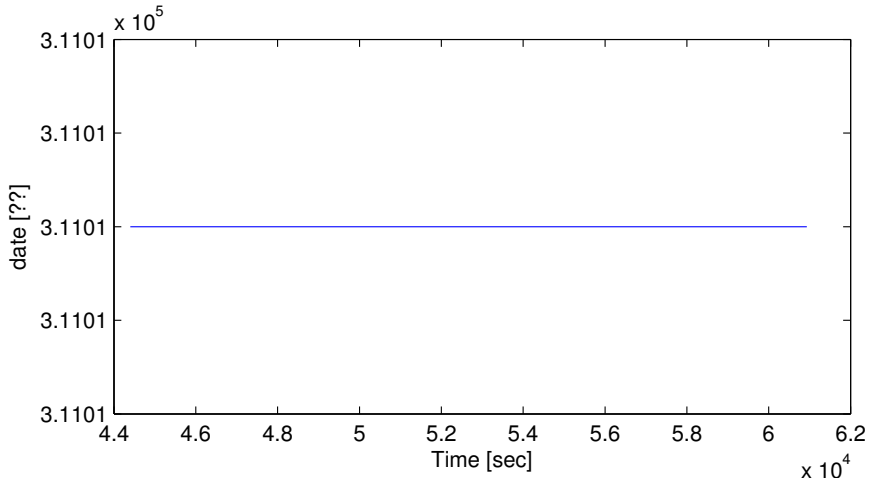


Figure E.41: CS vs. Time

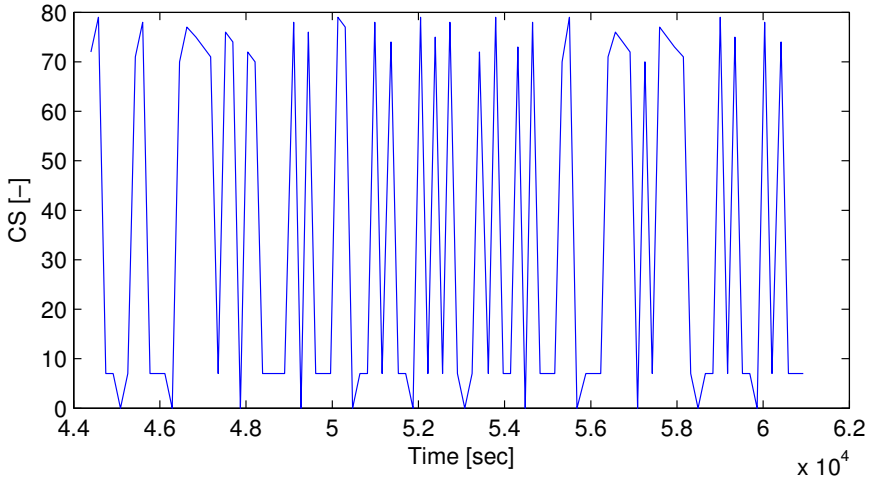


Figure E.42: temperature vs. Time

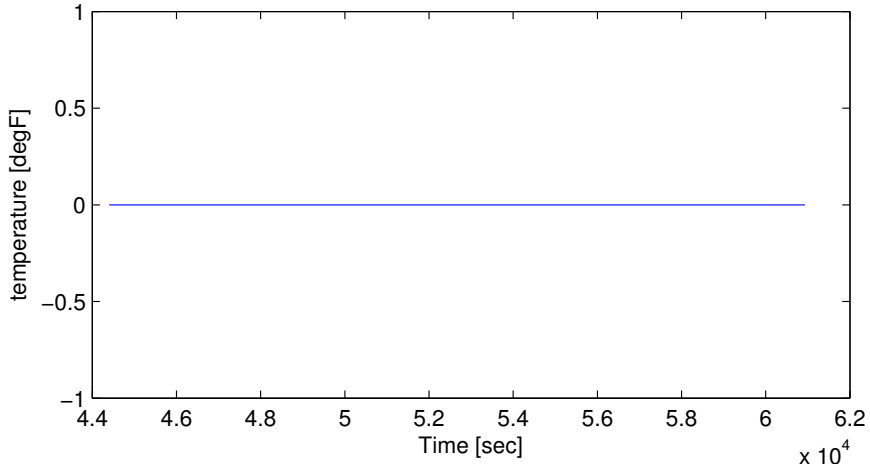


Figure E.43: deltaT vs. Time

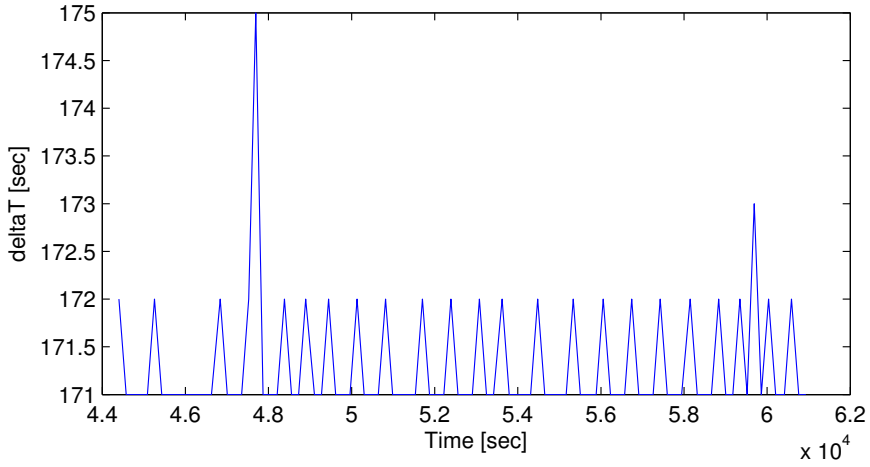


Figure E.44: rpmPwm vs. Time

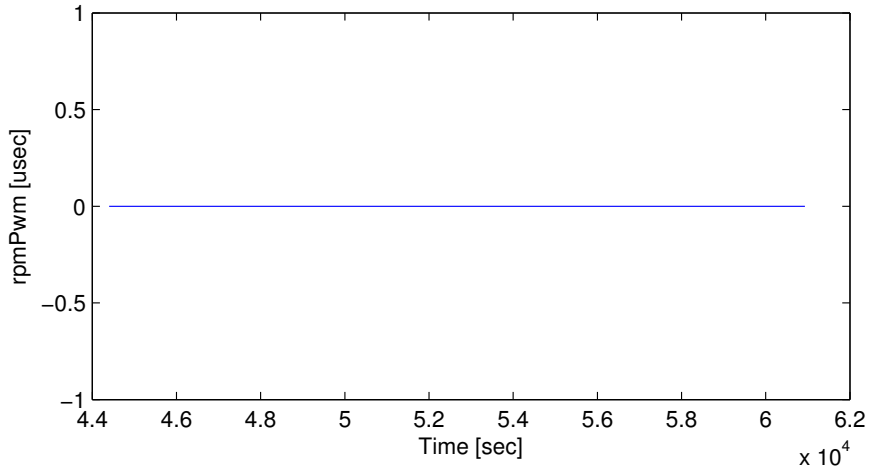


Figure E.45: ang2300X vs. Time

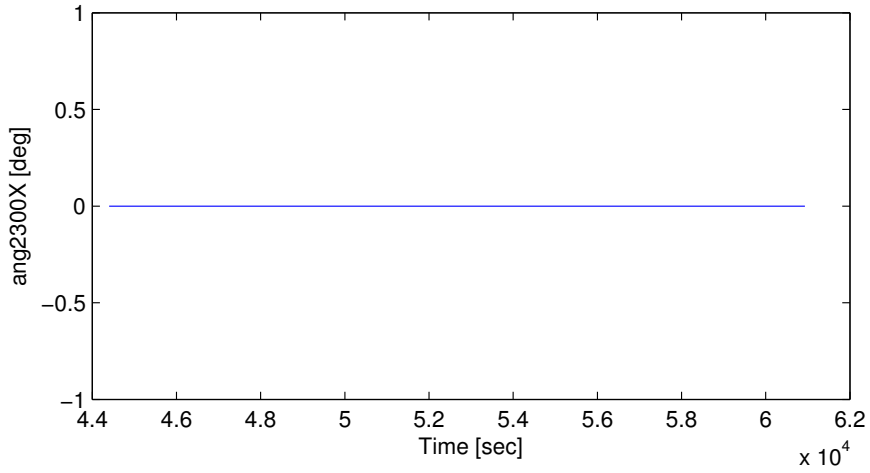


Figure E.46: ang2300Y vs. Time

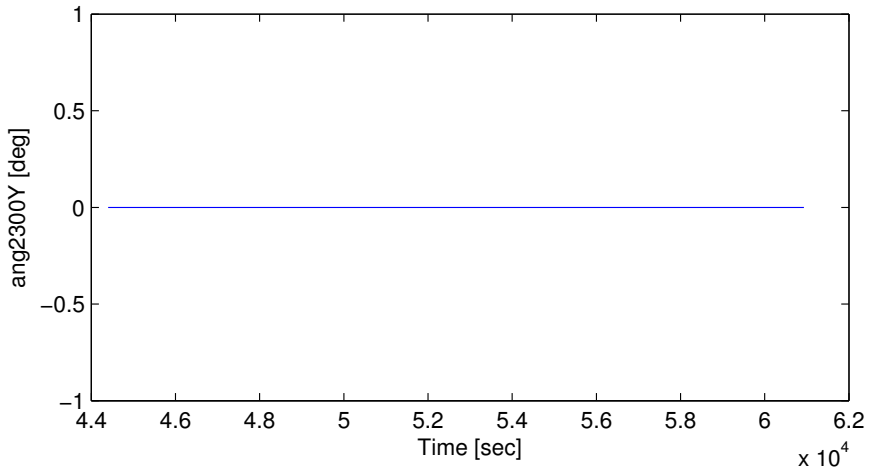


Figure E.47: ang2300Z vs. Time

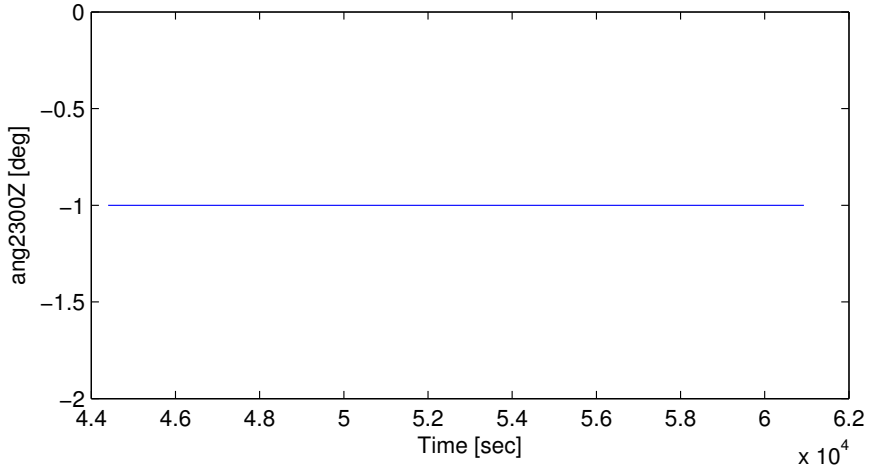


Figure E.48: ang5883X vs. Time

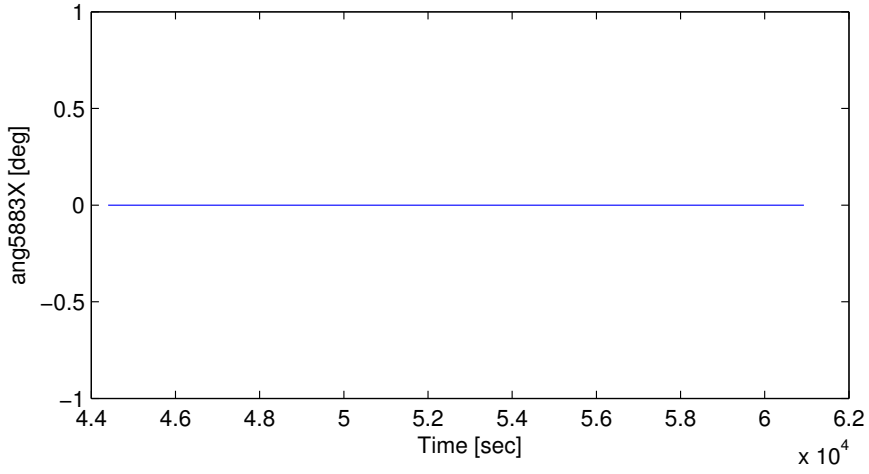


Figure E.49: ang5883Y vs. Time

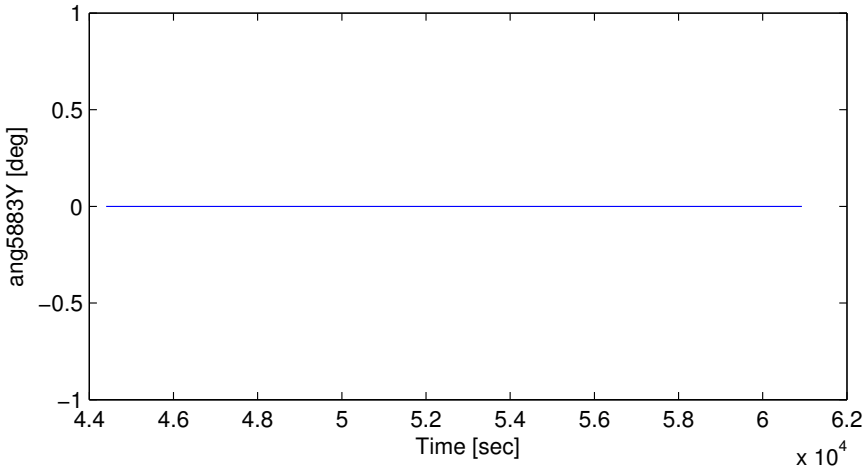


Figure E.50: ang5883Z vs. Time

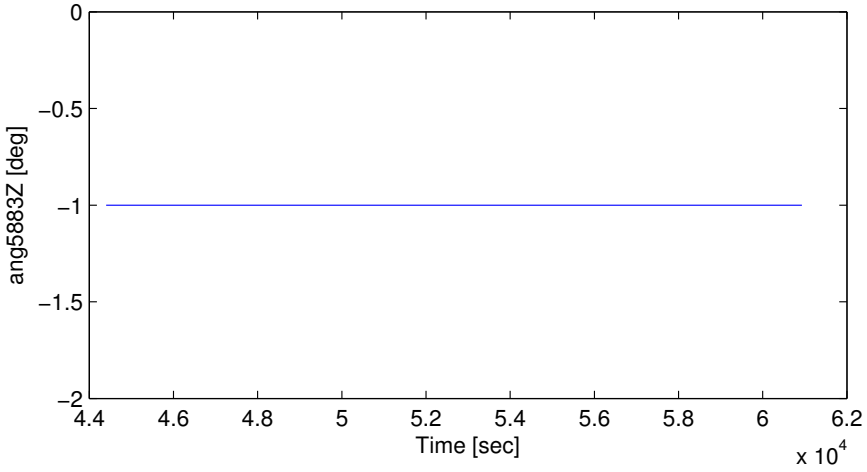


Figure E.51: qbar vs. Time

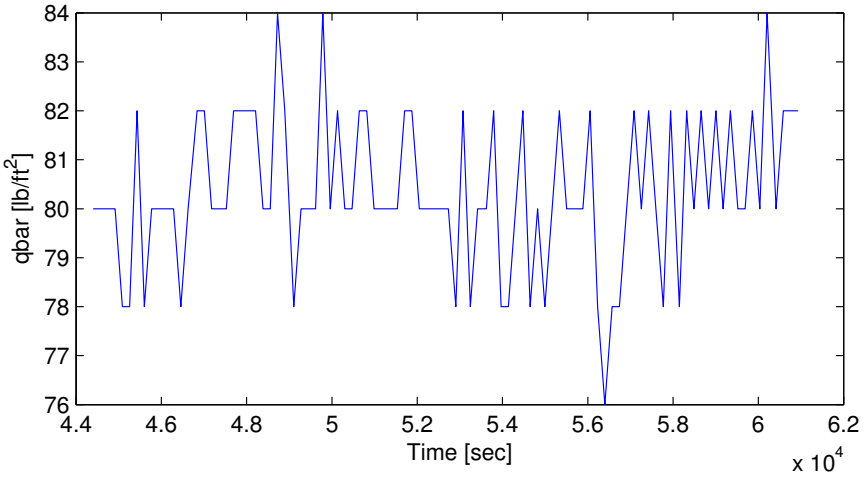


Figure E.52: rho vs. Time

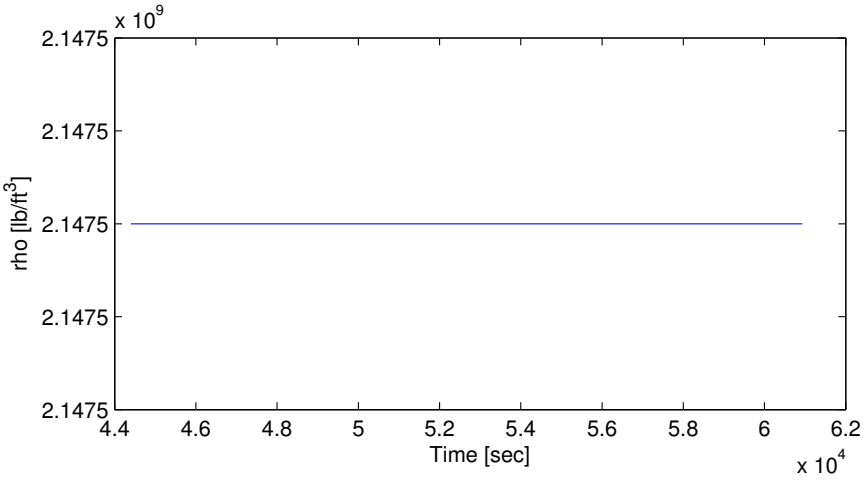


Figure E.53: alpha vs. Time

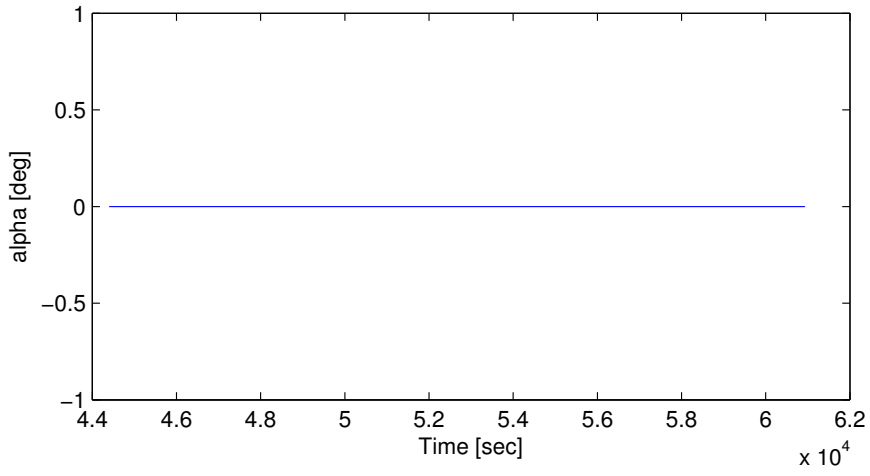
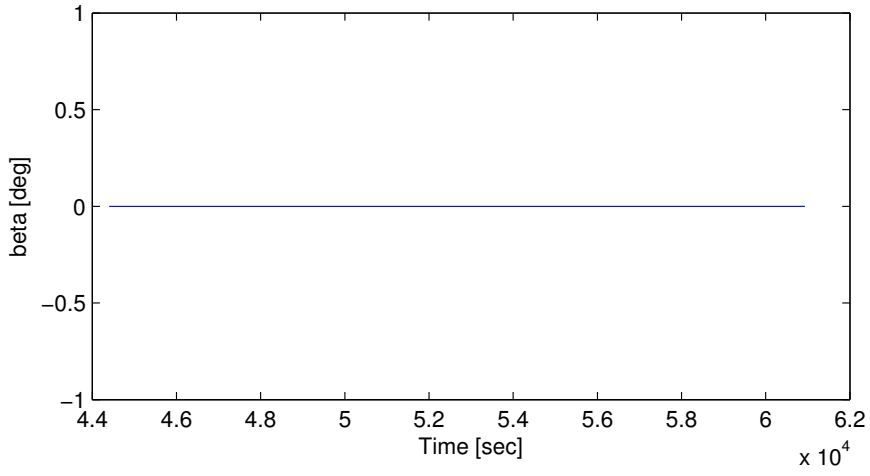


Figure E.54: beta vs. Time



E.4 Sample System Ouput - Filtered Data

Figure E.55: roll vs. Time

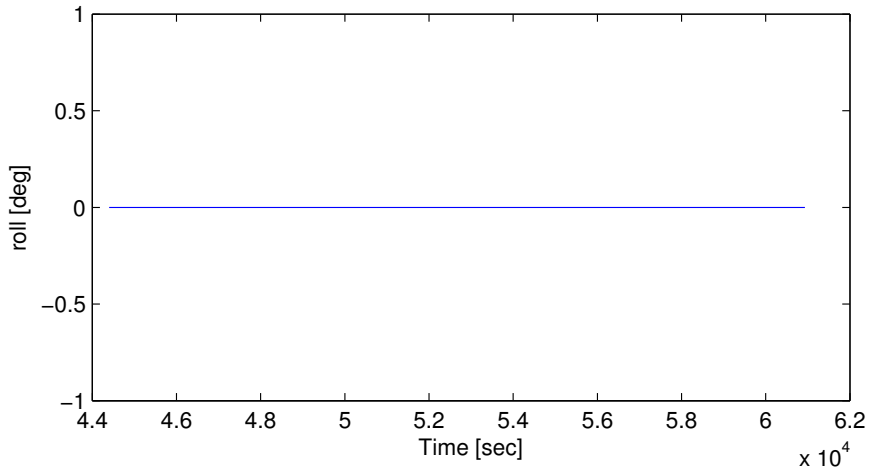


Figure E.56: pitch vs. Time

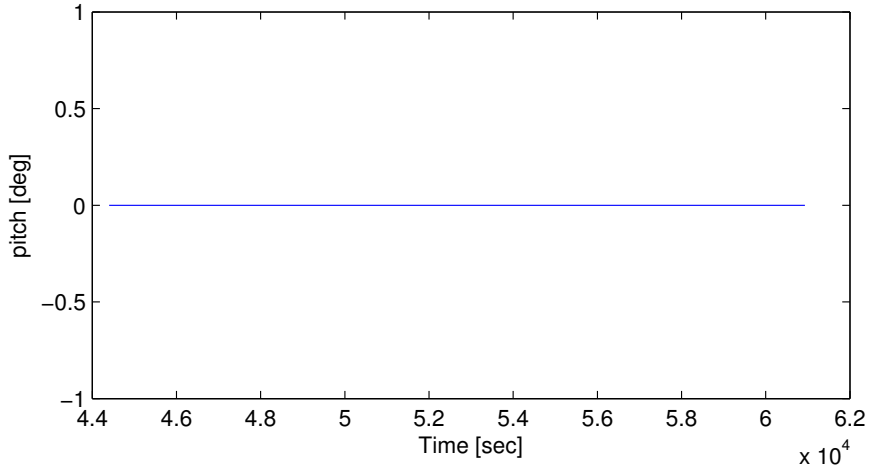


Figure E.57: yaw vs. Time

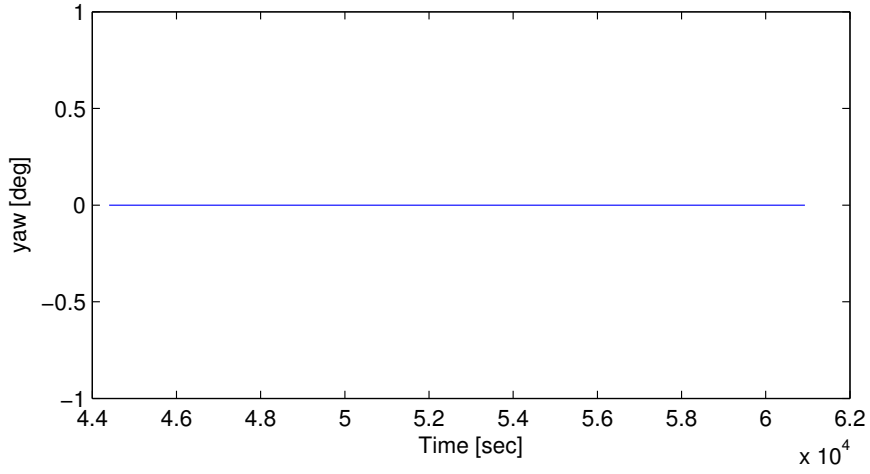


Figure E.58: rollRate vs. Time

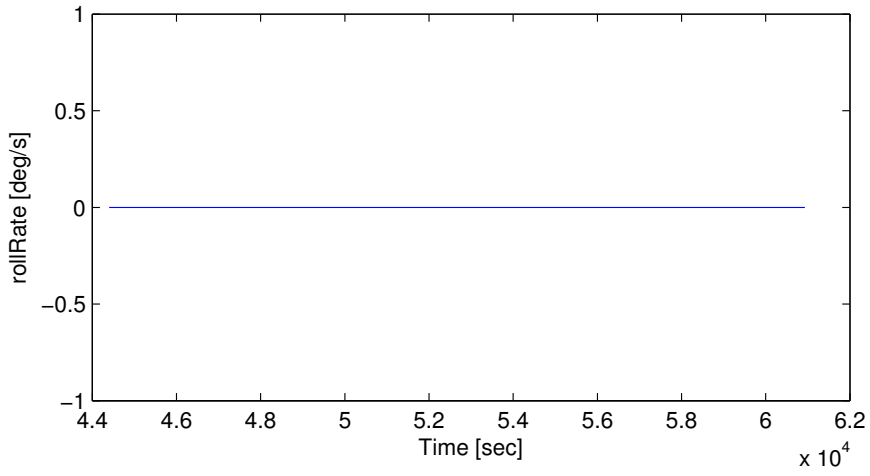


Figure E.59: pitchRate vs. Time

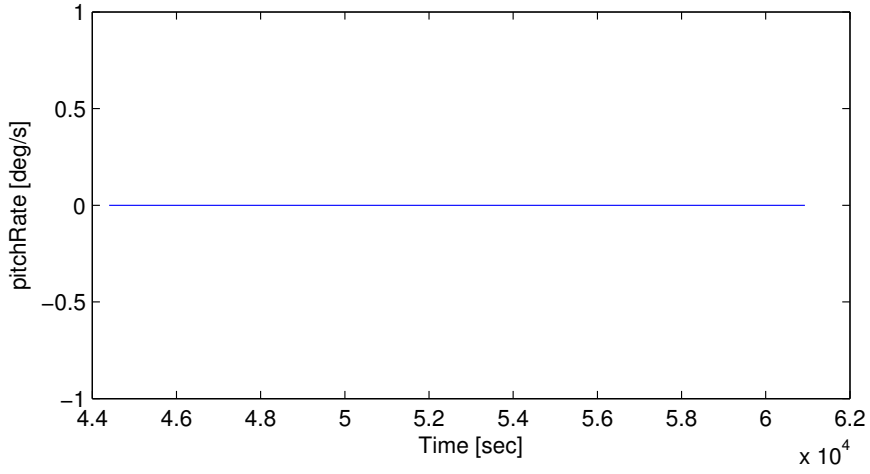


Figure E.60: yawRate vs. Time

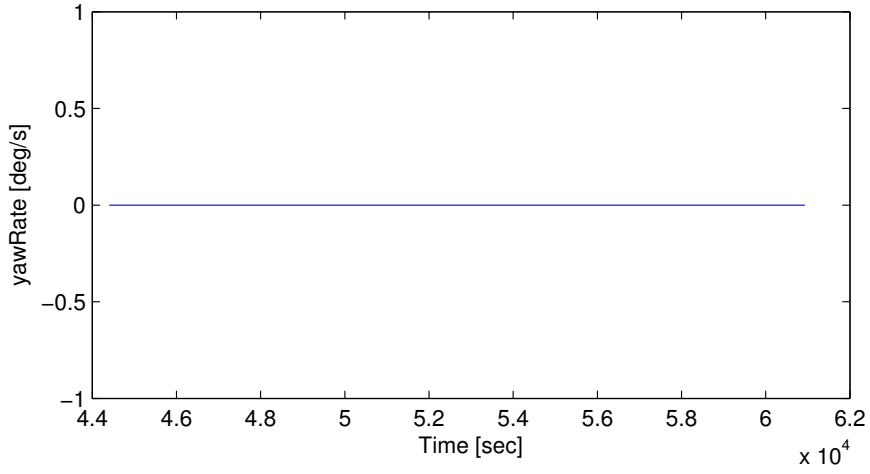


Figure E.61: accelX vs. Time

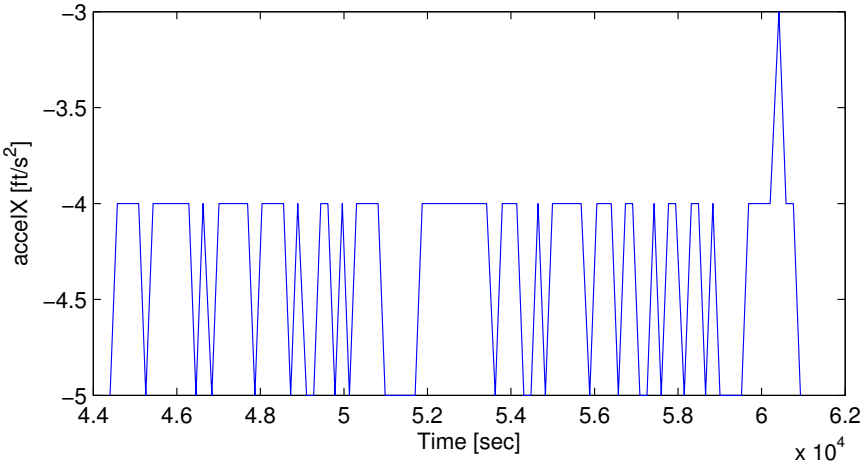


Figure E.62: accelY vs. Time

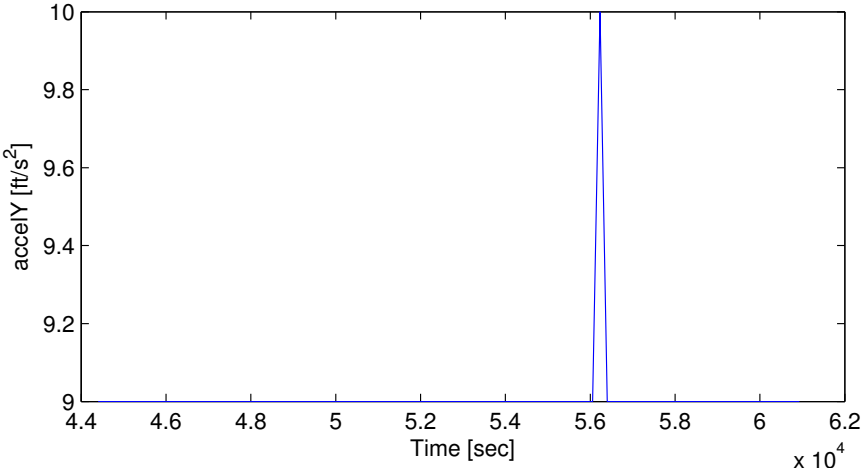


Figure E.63: accelZ vs. Time

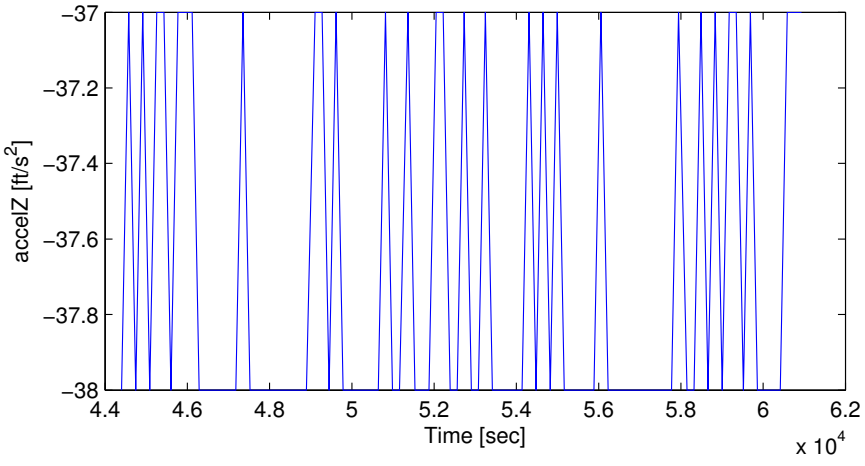


Figure E.64: qbar vs. Time

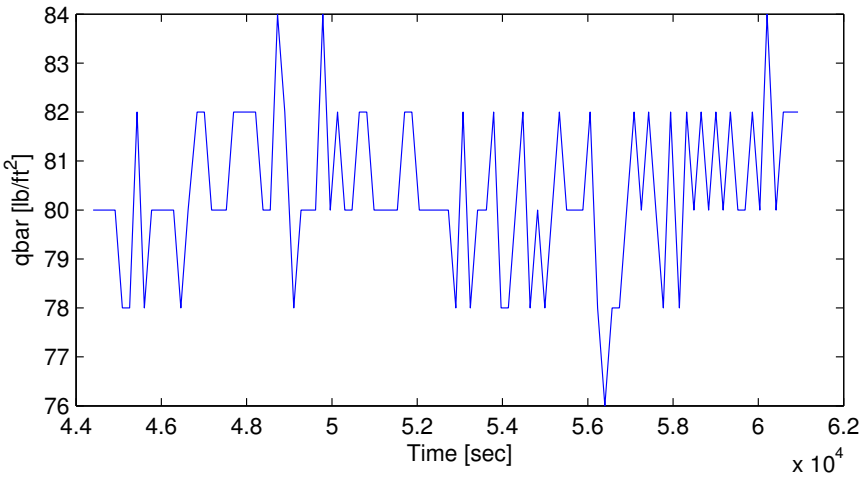


Figure E.65: rho vs. Time

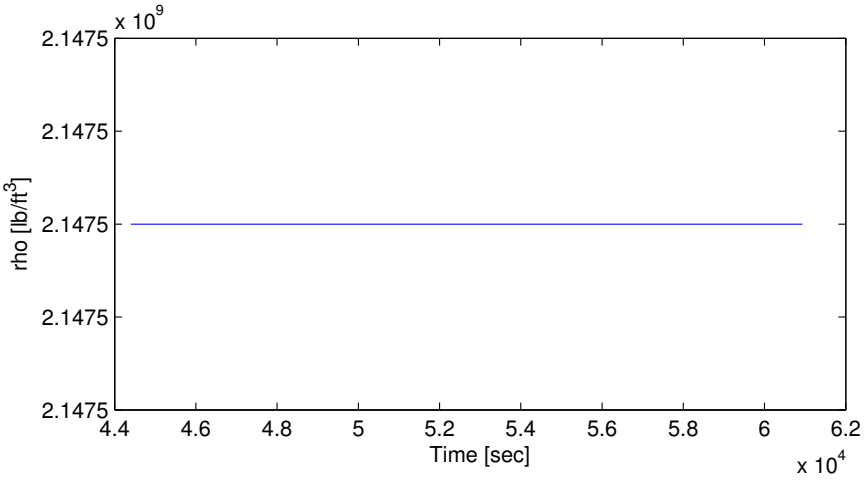


Figure E.66: alpha vs. Time

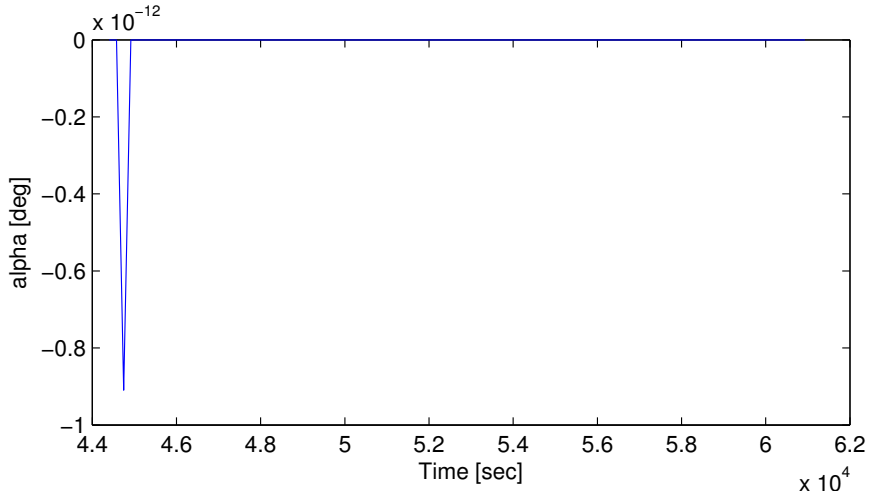


Figure E.67: beta vs. Time

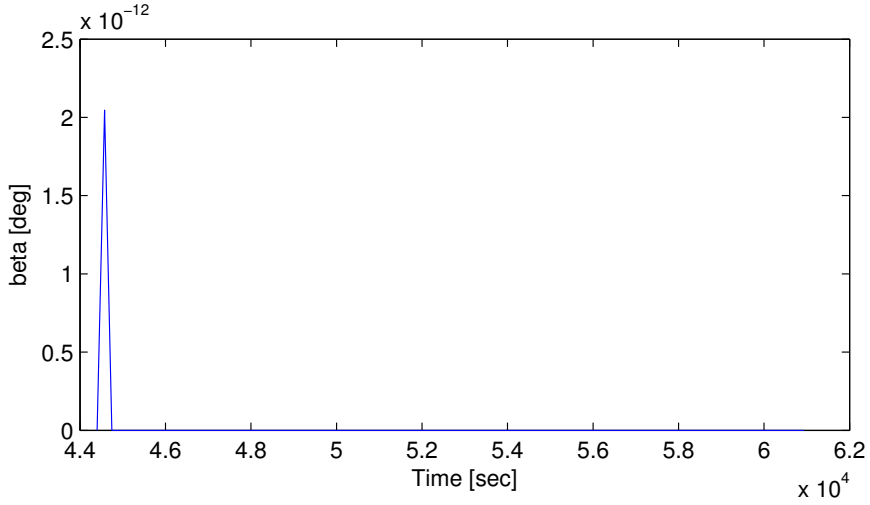


Figure E.68: roll vs. Time

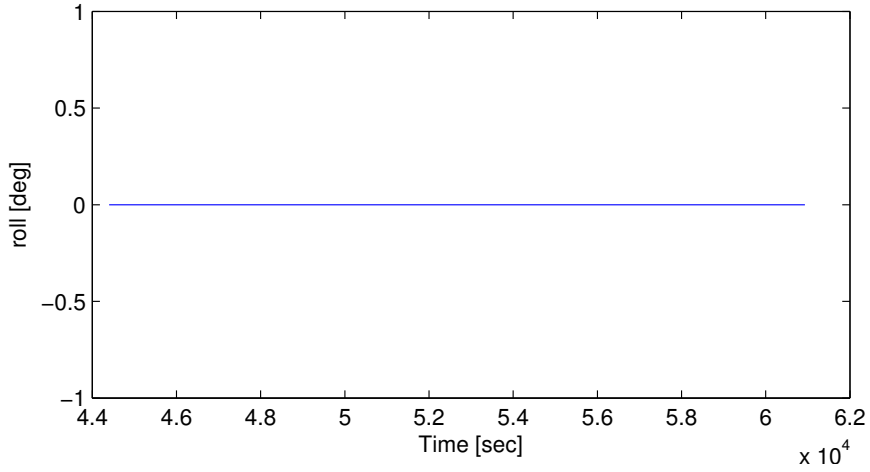


Figure E.69: pitch vs. Time

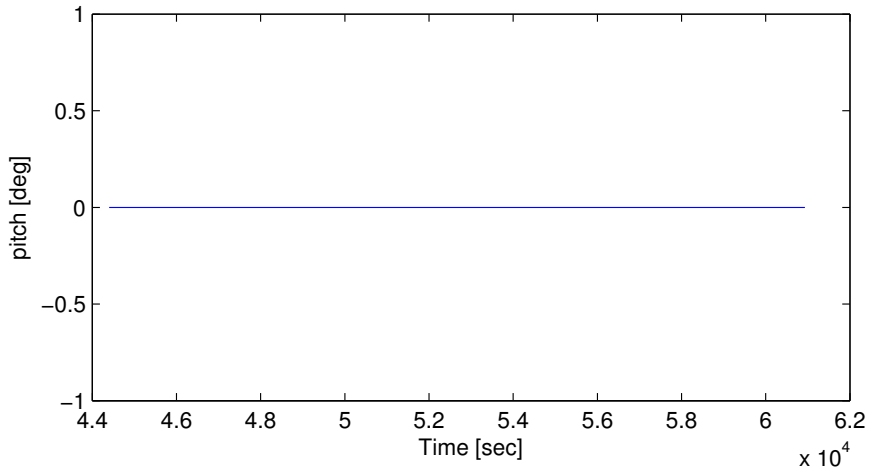


Figure E.70: yaw vs. Time

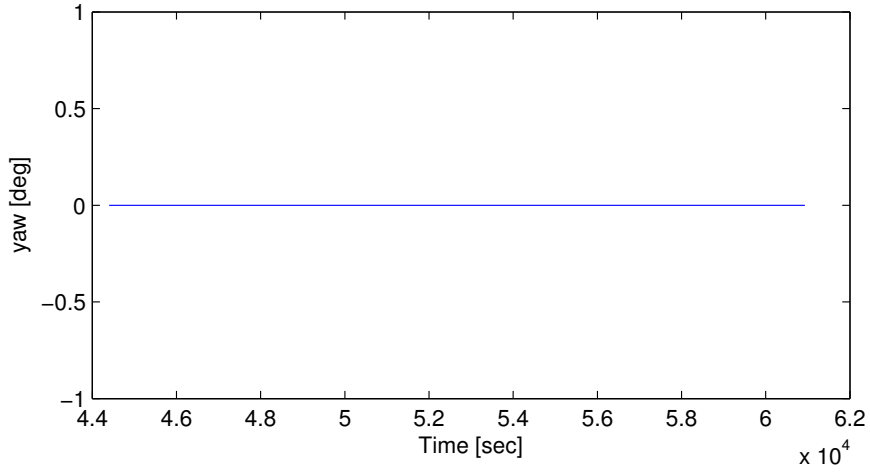


Figure E.71: rollRate vs. Time

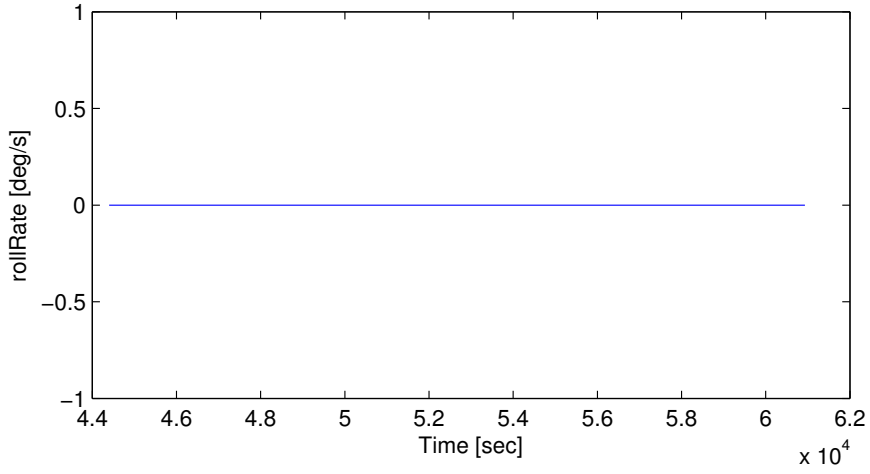


Figure E.72: pitchRate vs. Time

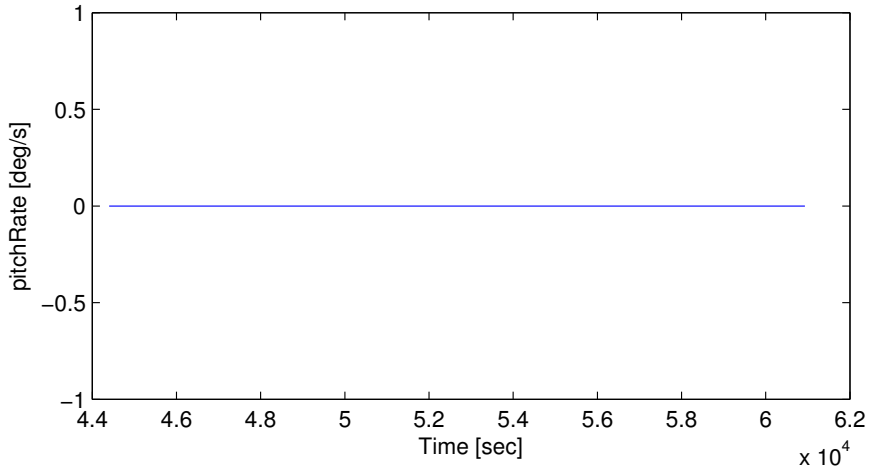


Figure E.73: yawRate vs. Time

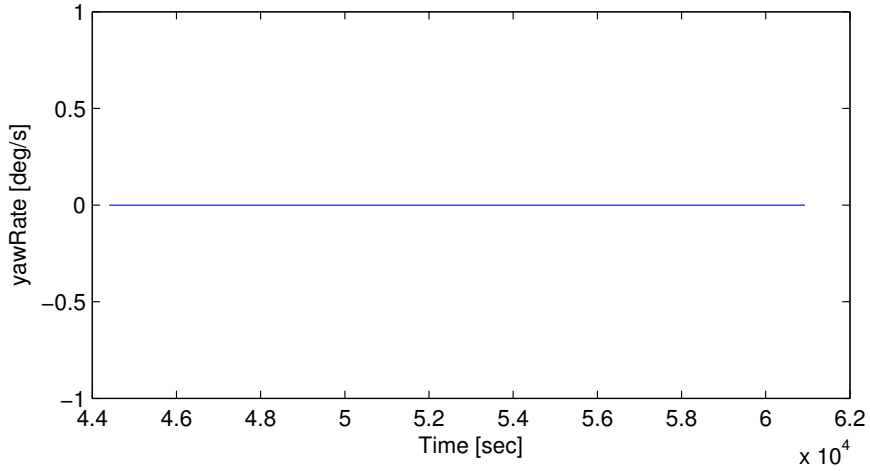


Figure E.74: accelX vs. Time

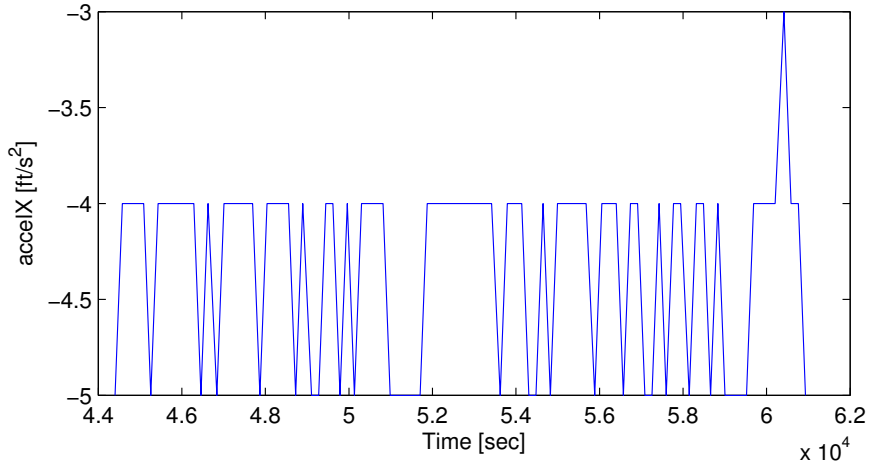


Figure E.75: accelY vs. Time

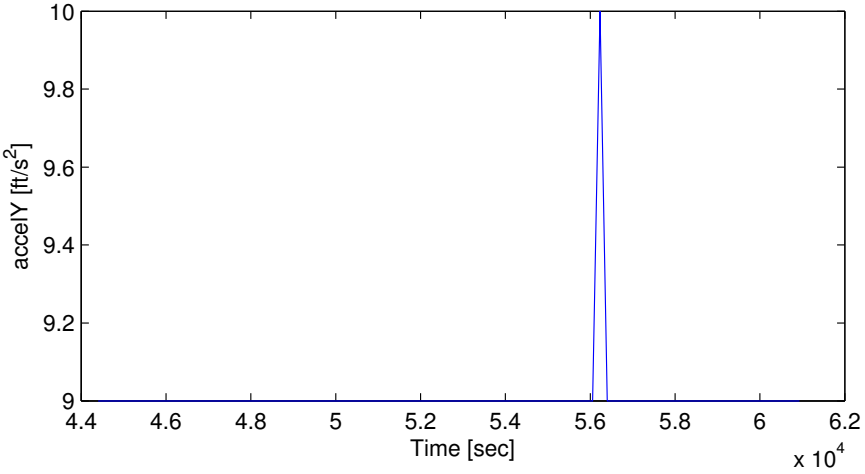
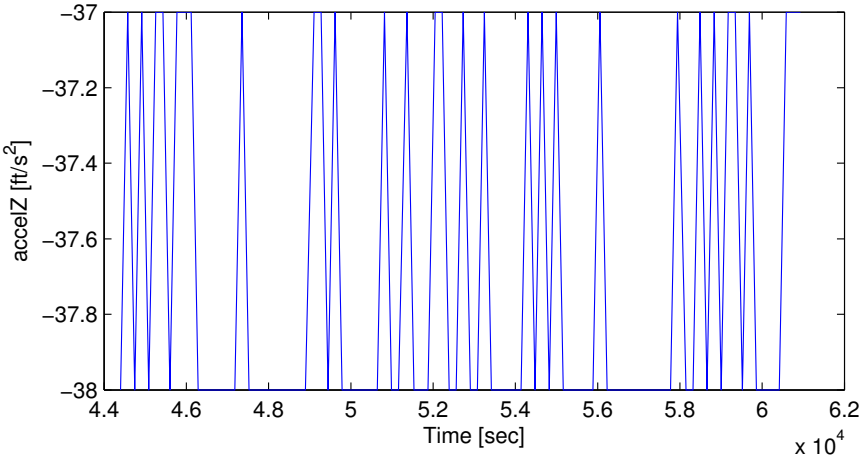


Figure E.76: accelZ vs. Time



F.0 Wiring Schematics

Figure F.1: Arduino Due Flight Data Recorder Shield

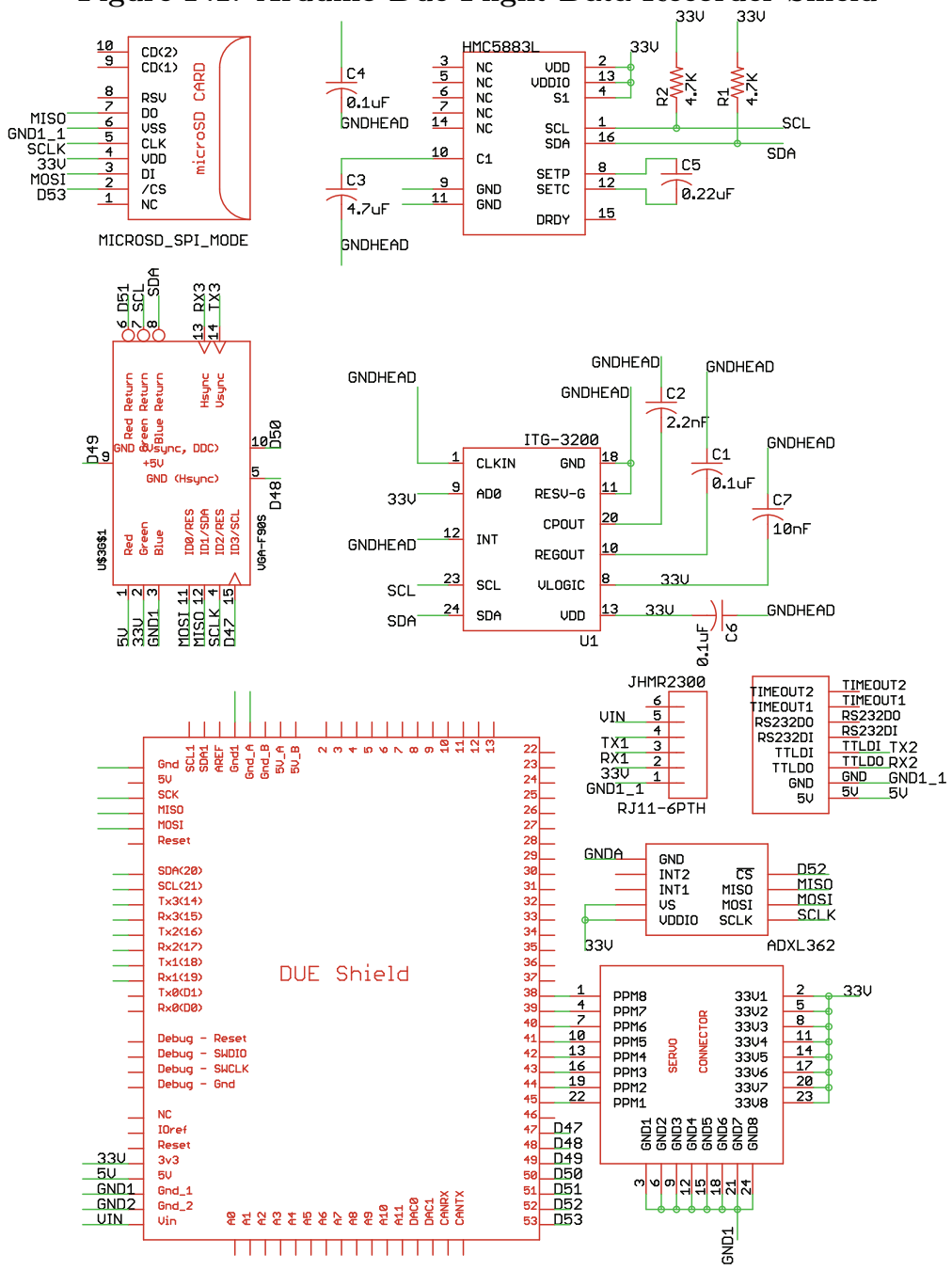


Figure F.2: Pressure Satellite Board Schematic

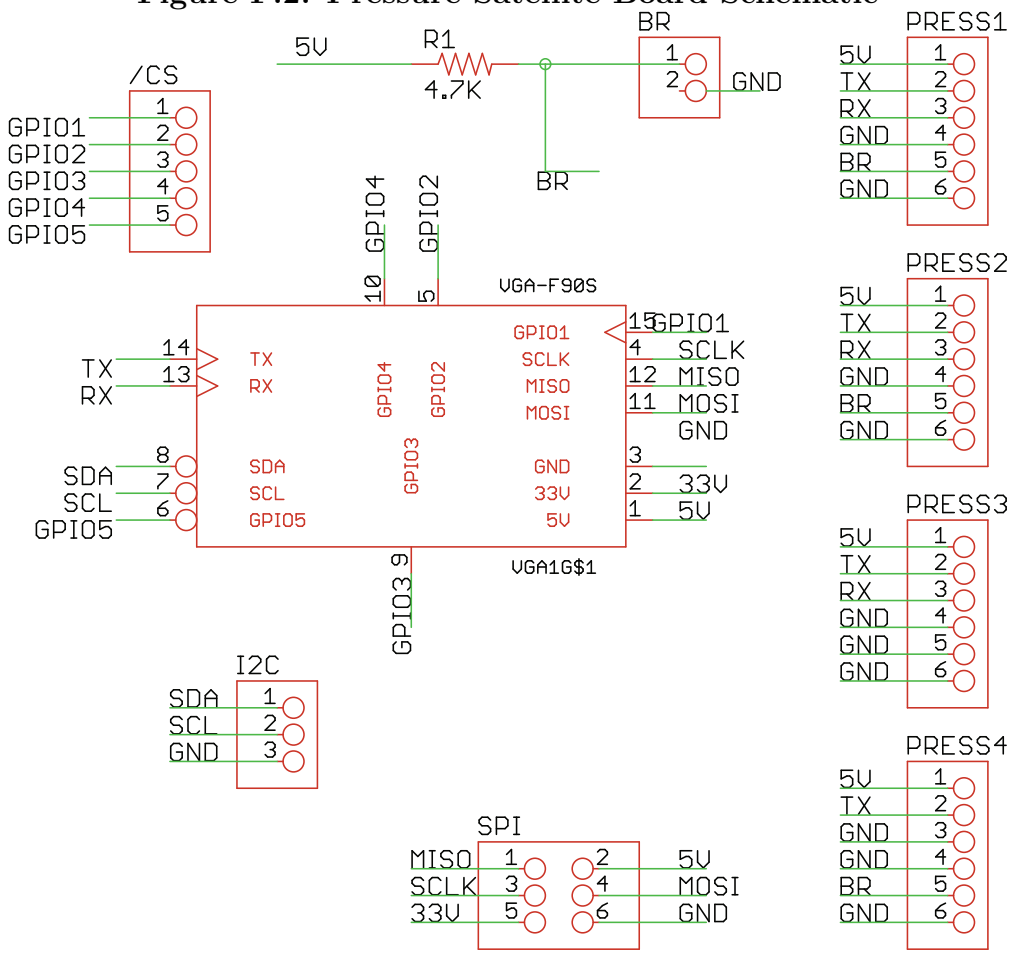
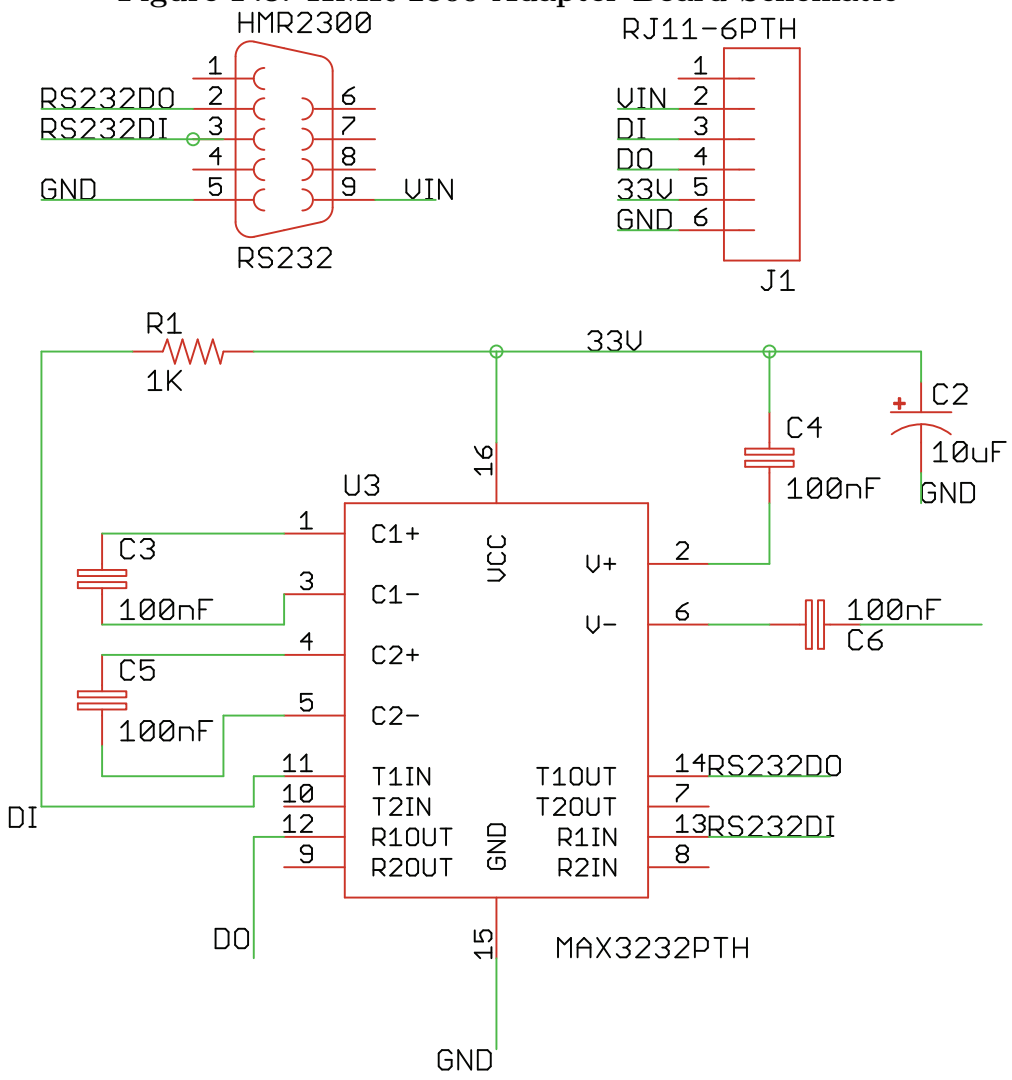


Figure F.3: HMR-2300 Adapter Board Schematic



1. Static and dynamic plant estimation
2. General trends for “are wheel pants worth it?”, etc
3. Incorporate thrust into model
4. Better integrate gps receiver
5. Redesign main board to include MAX3232 RS-232 to TTL translator
6. Redesign pressure board to surface mount headers so the size is cut in half
7. Look into smaller, less accurate pressure sensors and make a ”mini” version
8. re-write parser in python or something, and make dedicated program
9. Switch to NewSoftSerial for pressure and don’t issue unique commands
10. re-write to have everything sampled at different rates, then interpolate

Bibliography

[1] Vladislav Klein and Eugene A Morelli. *Aircraft system identification: theory and practice*. American Institute of Aeronautics and Astronautics Reston, VA, USA, 2006.

[2] Mike1024. Ecec enu longitude latitude relationships. Wikipedia, February 2010.

[3] MLWatts. Hawker tempest mark ii three view. Wikipedia, September 2012.

[4] Jan Roskam. Airplane flight dynamics and automatic flight controls, design. *Analysis and Research Corporation, Lawrence, KS*, 2001.

[5] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[6] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[7] Leeland Nicolai. Estimating r/c model aerodynamics and performance. White paper, Society of Automotive Engineers, 2009.

[8] Daniel P Raymer et al. *Aircraft design: a conceptual approach*, volume 3. American Institute of Aeronautics and Astronautics, 1999.

[9] Mark Drela. Xfoil: An analysis and design system for low reynolds number airfoils. In *Low Reynolds number aerodynamics*, pages 1–12. Springer, 1989.

[10] Olivier Clynen. 737 ng winglet effect simplified. Wikipedia, June 2012.

[11] Ludwig Prandtl. *Applications of modern hydrodynamics to aeronautics*. National Advisory Committee for Aeronautics, 1923.

[12] Brian L Stevens and Frank L Lewis. Aircraft control and simulation. 2003.

[13] Atmel. Sam3x-sam3a series summary. Data Sheet 11057BSATARM13-Jul-12, Atmel, 2012.

[14] Atmel. 8-bit atmel microcontroller with 64k/128k/256k bytes in-systemprogrammable flash. Data Sheet 2549PAVR10/2012, Atmel, 2012.

[15] Analog Devices. Micropower, 3-axis,2 g/4 g/8 g digital output mems accelerometer. Technical report, 2012.

[16] Sparkfun. Adxl362 bob v01. Technical report, Sparkfun, 2012.

[17] U-Line. Easy-count counting scale data sheet. Technical report, U-Line.

[18] Honeywell Magnetic Sensors. Smart digital magnetometer hmr2300. Technical report, Honeywell, 2006.

[19] Honeywell. 3-axis digital compass ic hmc5883l. Technical report, Honeywell.

[20] Sparkfun. Hmc5883l breakout v11. Technical report, Sparkfun, 2011.

[21] Talat Ozyagcilar. *Calibrating an eCompass in the Presence of Hard and Soft-Iron Interference*. Freescale Semiconductors, rev 3.0 edition, 04 2013.

[22] Yury Petrov. Ellipsoid fit. Matlab File Exchange, 08 2013.

[23] InvenSense. Itg-3200 product specification revision 1.4. Technical report, 2010.

[24] Sparkfun. Itg 3200 v10. Schematic, Sparkfun, 2010.

[25] E Joseph. Angle of attack and sideslip estimation using an inertial reference platform. Technical report, DTIC Document, 1988.

[26] Eugene A Morelli. Real-time aerodynamic parameter estimation without air flow angle measurements. *Journal of Aircraft*, 49(4):1064–1074, 2012.

[27] F. Adhika Pradipta Lie and Demoz Gebre-Egziabher. Synthetic air data system. *Journal of Aircraft*, pages 1–16, May 2013.

[28] All Sensors. Ds-0012 rev a. Technical report, All Sensors.

[29] Paroscientific. Model 745 high accuracy portable pressure standard data sheet. Technical report, Paroscientific.

[30] Doug Weibel. Proof of concept test - extremely accurate 3d velocity measurement with a ublox 6t module., December 2012.

[31] All Sensors. Ds-0010. Technical report, All Sensors.

[32] Maxim Integrated. Ds18b20 programmable resolution 1-wire digital thermometer. Technical report, Maxim Integrated, 2008.

[33] Finwing Hobby. Finwing universal penguin fpv airplane. Website, 2012.

[34] Dahlke Auman and CW Dahlke. Drag characteristics of ribbons. *AIAA Paper*, 2011, 2001.