


Fall 2017

Development of a Data Acquisition System for Unmanned Aerial Vehicle (UAV) System Identification

Donald Joseph Lear
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/mae_etds

 Part of the [Aerospace Engineering Commons](#), [Applied Mechanics Commons](#), and the [Computer Engineering Commons](#)

Recommended Citation

Lear, Donald J.. "Development of a Data Acquisition System for Unmanned Aerial Vehicle (UAV) System Identification" (2017).
Master of Science (MS), thesis, Mechanical & Aerospace Engineering, Old Dominion University, DOI: 10.25777/xqe7-gs03
https://digitalcommons.odu.edu/mae_etds/30

This Thesis is brought to you for free and open access by the Mechanical & Aerospace Engineering at ODU Digital Commons. It has been accepted for inclusion in Mechanical & Aerospace Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

**DEVELOPMENT OF A DATA ACQUISITION SYSTEM FOR
UNMANNED AERIAL VEHICLE (UAV)
SYSTEM IDENTIFICATION**

by

Donald Joseph Lear
B.A. December 2010, University of North Carolina at Asheville

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

AEROSPACE ENGINEERING

OLD DOMINION UNIVERSITY
December 2017

Approved by:

Drew Landman (Director)

Brett Newman (Member)

Thomas Alberts (Member)

ABSTRACT

DEVELOPMENT OF A DATA ACQUISITION SYSTEM FOR UNMANNED AERIAL VEHICLE (UAV) SYSTEM IDENTIFICATION

Donald Joseph Lear
Old Dominion University, 2017
Director: Dr. Drew Landman

Aircraft system identification techniques are developed for fixed wing Unmanned Aerial Vehicles (UAV). The use of a designed flight experiment with measured system inputs/outputs can be used to derive aircraft stability derivatives. This project set out to develop a methodology to support an experiment to model pitch damping in the longitudinal short-period mode of a UAV. A Central Composite Response Surface Design was formed using angle of attack and power levels as factors to test for the pitching moment coefficient response induced by a multistep pitching maneuver.

Selecting a high-quality data acquisition platform was critical to the success of the project. This system was designed to support fixed wing research through the addition of a custom air data vane capable of measuring angle of attack and sideslip, as well as an airspeed sensor. A Pixhawk autopilot system serves as the core and modification of the device firmware allowed for the integration of custom sensors and custom RC channels dedicated to performing system identification maneuvers. Tests were performed on all existing Pixhawk sensors to validate stated uncertainty values. The air data system was calibrated in a low speed wind tunnel and dynamic performance was verified. The assembled system was then installed in a commercially available UAV known as an Air Titan FPV in order to test the Pixhawk's automated flight maneuvers and determine the final performance of each sensor.

Flight testing showed all the critical sensors produced acceptable data for further research. The Air Titan FPV airframe was found to be very flexible and did not lend itself well to accurate measurement of inertial properties. This realization prohibited the construction of the required math models for longitudinal dynamics. It is recommended that future projects using the developed methods choose an aircraft with a more rigid airframe.

Copyright, 2017, by Donald Joseph Lear, All Rights Reserved.

DEDICATION

To my older sister, Robyn, whose persistent nature broke me out of my shell and who always encouraged me to further my academic pursuits, from grade to grad school. To Mary, my younger sister, from whose story is what good movies are made. And to my brother, Bryan, without whom I might still be laying crayons and trains in straight lines to sate some primal need for unbroken patterns. I only hope that wherever my brother may be that we might one day reunite.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Drew Landman, whose patience and dedication smoothed my transition from a Bachelor of Science in physics into the field of aerospace engineering. For the past three years, learning and understanding the science of RC aircraft and UAV design became a reality for me because of Dr. Landman's dedication to my education. Many thanks to Daniel and Keith, who worked alongside me and helped navigate some of the more electrical intensive aspects of this project. I am indebted to Brian Duvall for his assistance throughout this experience, and whose knowledge and guidance helped me overcome problems that arose in the electronic systems. He was also the invaluable pilot who expertly flew the aircraft housing my original sensor design. I must also give thanks to my family for their encouragement and for giving me the opportunity to initiate my engineering studies.

NOMENCLATURE

F	force
L, M, N	X, Y, Z axis moment in N
P, Q, R	X, Y, Z angular velocity in rad/s
ω_N	natural frequency rad or deg/s
ω_D	damped frequency rad or deg/s
ζ	damping ratio
f_n	natural frequency
α	angle of attack, deg or rad
β	regressor coefficient vector
δe	elevator deflection, deg or rad
ε	residual
μ	advance ratio
σ	standard deviation
σ^2	variance
Σ	summation
Θ	pitching angle, deg or rad
Φ	rolling angle, deg or rad
Ψ	yawing angle, deg or rad
Ω	revolutions per minute
a	acceleration
b	estimated regressor coefficient vector
C	covariance matrix

\bar{c}	reference chord, ft
C_m	pitch moment coefficient
C_{m_o}	pitch moment coefficient bias
$C_{m\alpha}$	pitch moment stability derivative, 1/deg or 1/rad
$C_{m\dot{\alpha}}$	quasi-steady pitch moment derivative, 1/deg/s or 1/rad/s
$C_{m\delta_e}$	pitch moment control derivative, 1/deg or 1/rad
C_{mq}	pitch moment damping derivative, 1/deg or 1/rad
D	aerodynamic damping
DF	degrees of freedom
$E(.)$	expectation operator
F	test statistic
g	force of gravity constant
I	moment of inertia
L	least squares function; rolling moment
k	number of regressor
m	mass
MS_R	mean square of regression
MS_E	mean square of error
n	number of observations
p	roll rate, rad/sec; number of regressor coefficients
PSE	predicted standard error
q	pitch rate, rad/sec

$\hat{q} = \frac{q\bar{c}}{2u}$	nondimensional pitch rate
\bar{q}	dynamic pressure, lb/ft ²
r	yaw rate, rad/sec
R	propeller radius, ft
R^2	coefficient of multiple determination
R^2_{adj}	coefficient of multiple determination normalized to degrees of freedom
S	reference area, ft ²
se	standard error
SS_E	error sum of squares
SS_R	regression sum of squares
SS_T	total sum of squares
T	period of natural frequency, sec
v_∞	Free-stream velocity, ft/s
VIF	Variance Inflation Factor
x	regressor variable
X	regressor matrix
y	response
L,D	lift, drag
C_L, C_D	lift, drag coefficient
AR	aspect ratio
e	efficiency
α_v	effective angle of attack
N	Torque

t_d	damped oscillation period
t_0	natural oscillation period
K_v	vane quality factor

Subscripts

i, j	indexing variables
x, y, z	axis of reference

Superscripts

\sim	vector or matrix transpose
-1	inverse matrix
\wedge	estimated (excluding $\hat{q} = \frac{q\bar{c}}{2u}$)
\cdot	time derivative
∞	infinity

TABLE OF CONTENTS

Chapter	Page
1. THEORY	1
1.1 AIRCRAFT LONGITUDINAL DYNAMICS.....	1
1.2 FLIGHT TESTING	5
1.3 DESIGN OF EXPERIMENTS AND RESPONSE SURFACE METHODOLOGY	7
<u>2. LITERATURE REVIEW</u>	16
2.1 SYSTEM IDENTIFICATION	16
2.2 DESIGN OF EXPERIMENTS AND RESPONSE SURFACE METHODOLOGY	18
<u>3. SYSTEM OVERVIEW</u>	20
3.1 PIXHAWK OVERVIEW.....	21
3.2 AIR DATA PROBE	30
3.3 PX4 FIRMWARE MODIFICATIONS	42
3.4 AIR TITAN FPV	53
3.5 DATA ACQUISITION SYSTEM MOUNTING.....	55
3.6 GROUND STATION SOFTWARE	57
<u>4. TESTING METHODOLOGY</u>	68
4.1 BENCH TEST METHODOLOGY	68
<u>5. DATA ANALYSIS</u>	79
5.1 BENCH TEST RESULTS.....	79

6. CONCLUSIONS	127
REFERENCES	131
APPENDICES	
A. MICROCONTROLLER AND SOFTWARE OPTIONS	134
A1. ARDUPILOT MEGA 2.8.....	134
A2. CUSTOM ARDUINO.....	135
A3. BEAGLEBONE BLACK.....	136
B. ADC LOGGING	142
A1. Firmware/src/modules/sdlog2/sdlog.c.....	142
A2. PARAMETER ID MANEUVERS.....	145
C. SDLOG FILE CONTENTS	166
VITA.....	170

LIST OF TABLES

Table	Page
1. Aircraft Nomenclature	3
2. Excel OFFSET Formula	61
3. VLookup	64
4. Nested IF Function.....	65
5. Name Chart Format, "X-Axis Formula Name"	65
6. Zoom Chart Code.....	67
7. Sensor Performance Overview	90
8. Fin Size Optimal ANOVA Table.....	101
9. Vane Damping Results	107
10. ADC Report Subscription	142
11. AOAS Source Structure.....	143
12. ADC Data Variable.....	143
13. ADC Data Structure.....	143
14. Sdlog Family Structure	144
15. Message Names	145
16. Log Format.....	145
17. SysID Switch	146
18. SysID Channel Creation	147
19. SysID Channel Mapping Parameter.....	148
20. SysID Channel Threshold Parameter.....	150
21. SysID Maneuver Parameter	151

	Page
22. SysID Altitude Parameter	152
23. SysID Execution Time Parameter.....	153
24. SysID Trim Time before Parameter.....	154
25. SysID Trim Time after Parameter.....	155
26. SysID Ramp Slope Parameter.....	156
27. Remove Vehicle Control Mode Structure.....	157
28. Data Structure Assignment	158
29. Parameter Tree Assignment.....	159
30. Abort Script Check	160
31. Assigning Parameters to Variables	161
32. Warning Message.....	161
33. Fully Define SysID Channel.....	162
34. Switch Position	163
35. Identify Desired Maneuver	163
36. System Identification Maneuvers	164

LIST OF FIGURES

Figure	Page
1. Aircraft Force, Moment, and Body Axes.....	2
2. 2 Factor CCD	14
3. Face Centered Design	15
4. Typical Pitching Moment Coefficient Results from 2-1-1-2 Maneuver.....	18
5. Pixhawk.....	21
6. UBlox-LEA-6H GPS	23
7. 3D Robotics 915MHZ Telemetry Radio.....	24
8. PPM Encoder	25
9. Buzzer	26
10. Arming Switch.....	26
11. Voltage Regulator XT60.....	27
12. Robotics Pitot Tube.....	28
13. MUX Board	28
14. Spektrum AR9020 Receiver	29
15. Multi Hole Probe.....	30
16. 4-Vane Air Data Boom Design.....	31
17. Air Data Vane Section View.....	32
18. Air Data Vane Free Body Diagram Top View	33
19. Air Data Vane Free Body Diagram Side View.....	34
20. Vane Quality Factor vs. Angle of Attack.....	39
21. Damping Ratio and Vane Quality Factor vs. Angle of Attack	39

	Page
22. Vishay 357 Series Potentiometer	40
23. Shroud Design.....	41
24. Operational Flowchart	43
25. PX4 Firmware Hierarchy	45
26. MAVlink Header Packet.....	46
27. Teraterm Command Window	48
28. QT Creator GUI.....	49
29. Github GUI	51
30. Air Titan FPV Airframe.....	53
31. Wiring Guide	55
32. Air Titan Final Assembly.....	57
33. Qgroundcontrol	58
34. Excel Dashboard	60
35. Dropdown Menu	62
36. Glossary Worksheet.....	63
37. Formula Worksheet.....	64
38. Zoom Chart	66
39. Wind Tunnel Test Stand	71
40. Vane Damping Test Stand	73
41. Ideal 2-1-1-2 Maneuver	76
42. Ideal Chirp Maneuver	77
43. X-Axis Accelerometer Noise Test.....	79

	Page
44. Y-Axis Acceleration Bench Test	80
45. Z-Axis Acceleration Bench Test.....	80
46. Secondary Accelerometer Bench Tests	81
47. Secondary Y-Axis Accelerometer Bench Tests.....	82
48. Secondary Z-Axis Acceleration Bench Tests	82
49. Board Temperature Bench Test I.....	83
50. Board Temperature Bench Test II.....	84
51. Board Temperature vs Acceleration	85
52. Gyroscope Noise Test I.....	86
53: Gyroscope Noise Test II	86
54. Gyroscope Noise Test III.....	87
55. ADC Noise Angle of Attack.....	88
56. Sideslip Angle vs. Time.....	89
57. 1/4 Hz Wave 27 M/S (60 MPH)	92
58. 1/4 Hz Wave 13.4 M/S (30 MPH)	93
59. 1/4 Hz Wave 8 M/S (17 MPH)	93
60. Vane Tracking 1/2 Hz 27 M/S (60 MPH).....	94
61. Vane Tracking 1/2 Hz 27 M/S (60 MPH).....	94
62. Vane Tracking 3/4 Hz 27 M/S (60 MPH).....	95
63. Vane Tracking 1 Hz 27 M/S (60 MPH).....	95
64. Vane Tracking 1/4 Hz 13.4 M/S (30 MPH).....	96
65. Vane Tracking 1/2 Hz 13.4 M/S (30 MPH).....	96

	Page
66. Vane Tracking 3/4 Hz 13.4 M/S (30 MPH).....	97
67. Vane Tracking 1 Hz 13.4 M/S (30 MPH).....	97
68. Vane Tracking 1/4 Hz 8 M/S (17 MPH).....	98
69. Vane Tracking 1/2 Hz 8 M/S (17 MPH).....	98
70. Vane Tracking 3/4 Hz 8 M/S (17 MPH).....	99
71. Vane Tracking 1 Hz 8 M/S (17 MPH).....	99
72. Fin Size Optimization Diagnostic Plots	102
73. Fin Optimal Surface Plot	103
74. Fin Size Optimization	104
75. Damping Test 15 Degrees.....	105
76. Damping Test 10 Degrees.....	106
77. Airspeed Calibration	109
78. Altitude vs. Time	110
79. True Airspeed (TAS) vs. Time	111
80. Air Titan Relative Flight Path.....	111
81. Received Signal Strength Indicator (RSSI)	112
82. ω_y vs. Time.....	113
83. Angle of Attack vs. Time.....	114
84. 2-1-1 Az vs. Time	114
85. 2-1-1 Pitch Rate	115
86. 2-1-1-2 Pitch Velocity.....	116
87. 2-1-1-2 Angle of Attack.....	116

	Page
88. 2-1-1-2 Z-Axis Acceleration.....	117
89. 2-1-1-2 Pitch Rate.....	117
90. Chirp Pitching Velocity	118
91. Chirp Angle of Attack.....	119
92. Chirp Z-Axis Acceleration.....	119
93. Chirp Pitch Rate.....	120
94. Sinusoid Pitching Velocity	121
95. Sinusoid Angle of Attack.....	121
96. Sinusoid Z-Axis Acceleration.....	122
97. Sinusoid Pitch Rate.....	122
98. Sinusoid Pitching Velocity	123
99. Sinusoid Angle of Attack.....	124
100. Sinusoid Z-Axis Acceleration.....	124
101. Sinusoid Pitch Rate.....	125
102. S-Turn Yaw Rotational Velocity	126
103. S-Turn Sideslip Angle.....	126
104. Ardupilot Mega 2.8.....	135
105. Custom Arduino.....	136
106. Beaglebone Black	137
107. Field Programmable Gate Array (FPGA)	137
108. Flight Pilot GUI	138
109. Log Muncher Web Page	139

	Page
110. PX4 Tools	141
111. sdlog2 File Contents	166

CHAPTER I

THEORY

A specific goal for this experiment is to design an instrument that will find an aircraft's short-period mode performance. The mathematical background in flight mechanics is explained in this section. In addition, Design of Experiments (DoE) methodology principles are explained, as well as why these are proven to be vital in high-noise environments.

1.1 AIRCRAFT LONGITUDINAL DYNAMICS

1.1.1 Aircraft Equations of Motion

The motion of an aircraft is, in its most general form, a combination of three translations and three rotation forces. Translation is defined as linear movement along the x , y , or z axis, where x for aircraft is defined along the body of the longitudinal axis of the fuselage, y faces orthogonally along the wing, and z faces down due to the right-hand rule. Rotation is measured by the aircraft's revolution along any of these three defined axes. All aircraft motion can be defined as a combination of translation and rotation about these axes [1].

$$\begin{aligned}
 F_x &= m(\dot{U} + qW - rV) \\
 F_y &= m(\dot{V} + rU - pW) \\
 F_z &= m(\dot{W} + pV - qU) \\
 L &= \dot{P}I_{xx} + QR(I_{zz} + I_{yy}) - (\dot{R} + PQ)I_{xz} \\
 M &= \dot{Q}I_{yy} + PR(I_{zz} + I_{xx}) - (P^2 - R^2)I_{xz} \\
 N &= \dot{R}I_{zz} + PQ(I_{yy} + I_{xx}) - (QR - \dot{P})I_{xz}
 \end{aligned} \tag{Eq. 1}$$

Figure 1 illustrates the body axes system used for the equations of motion of the aircraft.

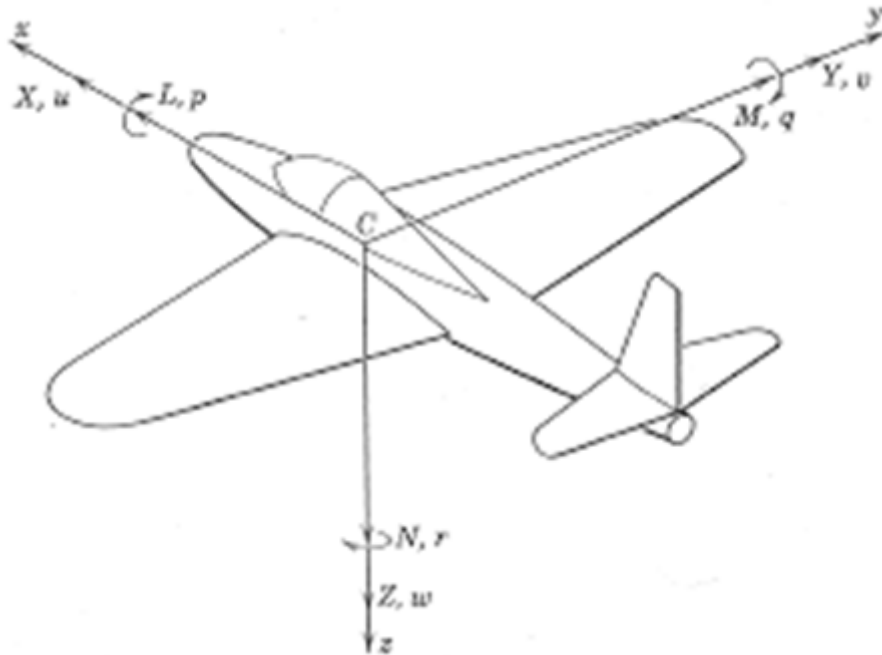


Figure 1: Aircraft Force, Moment, and Body Axes

When all forces acting on the aircraft are equal and opposed, the aircraft is in a steady-state (trimmed) flight condition [1]. Any control surface deflection, throttle change, change in weight, or shift in the free stream velocity will break the equilibrium. Perturbed flight then can be defined as the change in aircraft motion parameters, forces, and moments relative to the steady-state flight condition. Table 2 organizes the nomenclature used in the small perturbation approach where upper case letters signify steady state values, and lower case letters represent perturbed values. A subscript of “j” is added to distinguish these perturbed terms from the angular rate terms used in the equations of motion.

Axis	Name	Velocity	Angle	Angular Velocity	Moment
X_j, x_1	Roll	U_j, u_1	Φ_j, ϕ_1	P_j, p_1	L_j, l_1
Y_j, y_1	Pitch	V_j, v_1	Ξ_j, θ_1	Q_j, q_1	M_j, m_1
Z_j, z_1	Yaw	W_j, w_1	Ψ_j, ψ_1	R_j, r_1	N_j, n_1

Table 1: Aircraft Nomenclature

The translational forces are derived from Newton's second law, summing forces and setting them equal to the product of mass and acceleration, here in the body axis system. Rotational forces (moments) are derived by summing the moments about each axis and equating them to the product of angular acceleration and moment of inertia. It is worth noting that l_{xy} and l_{yz} do not contribute, as the aircraft is assumed to have a xz plane of symmetry [1].

1.1.2 Longitudinal Mode: Description of Response (Short and Long Periods)

The period of oscillation due to perturbation is classified as either a short-period or phugoid (long period) oscillation mode. These expressions are defined by their damping ratio, natural frequency, and damped frequency [1].

$$\omega_N = \sqrt{\frac{K}{M}}$$

$$\omega_D = \frac{2\pi}{T}$$

$$\zeta = \sqrt{1 - \left(\frac{\omega_D}{\omega_N}\right)^2} \quad (\text{Eq. 2})$$

Short-period modes are characterized by a moderate to relatively high damping ratio and a high natural and a moderate to relatively high damped frequency [1]. This motion is easily excited

with a sequence of forward-aft-neutral elevator deflection. The motion causes variations in the angle of attack (α) and pitch attitude (θ). Trim conditions are typically regained after a few seconds.

“The phugoid mode is characterized by complex conjugate roots with a relatively low damping ratio, natural/damped frequency” [1]. This is demonstrated by placing the aircraft in level flight, shifting the stick aft for a few seconds, and then returning it to the neutral position. This results in variations in θ and airspeed with little change to α . These oscillations tend to last much longer than the short-period before the plane returns to trim conditions.

1.1.3 Short-Period

To calculate the motion, the only equation needed is the pitching moment because of the uncoupled nature of short-period oscillation.

$$M = I_{yy}\dot{Q} - PR(I_{xx} - I_{zz}) + I_{xz}(P^2 - R^2) \quad (\text{Eq. 3})$$

Dynamic pressure, wing reference area, and mean aerodynamic chord are used to non-dimensionalize the pitching moment:

$$M = \bar{q}S\bar{c}C_m$$

$$C_m = \frac{I_y\dot{q} - PR(I_x - I_z) + I_{xz}(P^2 - R^2)}{\bar{q}S\bar{c}} \quad (\text{Eq. 4})$$

It is assumed that the pitching moment coefficient equation can be linearized with the aircraft angle of attack, non-dimensional pitch rate, and elevator deflection, assuming small perturbations:

$$C_m = C_{m_0} + C_{m_u}u + C_{m_\alpha}\alpha + C_{m_{\dot{\alpha}}}\dot{\alpha} + C_{m_q}q + C_{m_{\delta_e}}\delta_e \quad (\text{Eq. 5})$$

The thrust acts approximately through the center of gravity of the aircraft and therefore its contribution to the pitching moment coefficient is negligible. The angular acceleration term is absorbed by q . It is safe to assume that the value is about $1/3q$ for most aircraft [1]. C_{m_0} is the pitching moment coefficient bias. Thus, the remaining terms are:

$$C_m = C_{m_0} + C_{m_\alpha} \alpha + C_{m_q} q + C_{m_{\delta_e}} \delta_e \quad (\text{Eq. 6})$$

Flight data collected from the acquisition system are substituted into the equation and a model can be made using an Ordinary Least Squares (OLS) approach, which provides an estimate for each point [2].

1.2 FLIGHT TESTING

Flight testing is performed to excite the short-period longitudinal mode. The testing is based on the adjustments of two parameters: angle of attack and power. When the aircraft reaches the desired angle of attack and power for the specified run, a series of elevator deflections are employed to induce a short-period mode oscillation. The aircraft is then left to return to trim under its own power.

1.2.1 Maneuvers to Excite the Short-Period Mode

The maneuver used to excite the short-period mode is known as a multistep maneuver. The input can be viewed as a square wave approximation [3] with pulses of varying width. This is considered the poor man's sine waves as the alternating pulses are much easier to execute than a frequency sweep. The "2-1-1-2" used in this project gets its name for the length of each pulse. In this case, the elevator is held up for two periods, down for one period, up for one, and down for two. Other common inputs are the "3-2-1-1" and the "2-1-1". The length of one period or pulse

width is determined based on the expected natural frequency for the dominant mode using equation 7 [2, 3].

$$T = \frac{0.7}{2f_n} \quad (\text{Eq. 7})$$

Further experimentation was performed to test the validity of sinusoid and chirp stick inputs as facilitated by an autopilot system. Given the proper instructions, the autopilot can execute sinusoid and chirp maneuvers with frequencies dictated by the pilot. Allowing the autopilot to perform these maneuvers ensures a more consistent response than is possible from a timed manual input. (For a brief summary of the functionality of this system, see Section 3.3.11. A guide for adjusting parameters to perform specific maneuvers with the desired frequency and time is provided in Section 4.4). A step-by-step guide on adding any of these changes to the PX4 flight stack can be found in Appendix B.

1.2.2 Assumptions/Conditions

Flight tests are performed below 400 ft. sea level, meaning changes in air properties such as density and viscosity, are negligible. Propeller efficiency is assumed constant during flight tests. Given the test aircraft is powered by a Lithium Polymer (LiPo) battery pack, any relationship between power and weight is negated because there is no need to calculate for fuel weight since the weight does not change during flight time.

Since the angle of attack is a crucial factor in this experiment, slight permutations in altitude are expected. However, these changes will be insignificant because the altitude adjustments caused by angle of attack adjustments will not be enough to significantly alter air properties such that there is a measurable effect on the overall data. Due to the change in angle of attack, there

will be slight climbs and dives to maintain this angle. These altitude changes are neglected during analysis.

1.3 DESIGN OF EXPERIMENTS AND RESPONSE SURFACE METHODOLOGY

1.3.1 Foundations

Design of Experiments is a methodology that guides the user through experimental tests while developing statistically defensible mathematical models and estimates of uncertainty using a minimal expenditure of resources. This discipline is anchored on three basic principles: replication, randomization, and blocking. These foundational concepts give designed experiments a more concrete and defensible set of results than the traditional One-Factor-At-A-Time (OFAT) methods.

Randomization is the cornerstone behind the use of statistical methods in experimental design [4]. Simply put, it is required that the order of runs is randomly determined. This assists in averaging out elements that are not considered factors in the experiment, yet affect the overall response, and cannot be controlled. An example of these “nuisance factors” in outdoor testing is atmospheric turbulence, which changes over the testing period.

The concept of replication serves to provide a model-free estimate of systematic error. Data points are collected at identical factor settings such that a variance estimate may be obtained which is solely due to the random error. “Replication of design points allows the researcher to determine an internal estimate of system noise and uncertainty” [4]. Increased sample size improves estimates. This can be seen in the computation of the standard error of the mean, as the overall value will decrease as the number of runs (n) increases. Sample variance follows this same model but is designated S^2 [5]:

$$\sigma_y^2 = \frac{\sigma^2}{n} \quad (\text{Eq. 8})$$

Blocking is a defense used in DoE against nuisance factors and their effect on overall noise. It is possible that an experiment, due to time constraints, weather, or lack of resources, is best divided into multiple testing sessions. The problem with this is that nuisance factors, such as temperature and humidity, might change between test sessions. Blocking serves to isolate these different testing periods from one another, so that any changes between them can be distinguished [5].

1.3.2 Model Building Using Ordinary Least Squares

Ordinary Least-Squares Regression (OLS) is a mathematical model-fitting method which minimizes the error associated with responses. Certain assumptions are made to execute this technique when significance testing through the Analysis of Variance is employed. The data are assumed to have a normal distribution and has a constant variance. Each point is independent, meaning that the responses collected are independent of the order in which they are collected. (Residual analysis will be explained later to verify these assumptions.) This approach is often used for estimating regression coefficient terms in a linear regression model. A first order multiple linear regression model takes the following form [6].

$$y_i = \beta_0 + \sum_{j=1}^k \beta_j x_{ij} + \epsilon_i \quad (\text{Eq. 9})$$

Let n be the number of observations and k be the number of regressors, with $n > k$. Observations are available on the response variable y_1, y_2, \dots, y_n . Each regressor variable will

have a value x_{ij} at the i th observation and the j th level. The term ϵ represents the general error that arises as a consequence of real world testing [6].

For OLS fitting, the error term (ϵ) is assumed to have a mean of zero, and a variance of σ^2 : OLS is used to solve for the regression coefficients while minimizing error in the response (y). This is done by solving for the error as a function of the response and regression coefficients.

$$L = \sum_{i=1}^n \epsilon_i^2 = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^k \beta_j x_{ij} \right)^2 \quad (\text{Eq. 10})$$

Each observation is used in a separate equation that together creates a system of equations that can be expressed in matrix form.

$$y = X\beta + \epsilon \quad (\text{Eq. 11})$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & x_{21} & \cdots & x_{1k} \\ 1 & x_{12} & x_{22} & \cdots & x_{2k} \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix} \quad (\text{Eq. 12})$$

$$\beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}, \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_k \end{bmatrix}$$

To find the minimal error, a partial derivative of L is taken with respect to β and set to zero.

$$\frac{dL}{d\beta} = -2X'y + 2X'Xb = 0 \quad (\text{Eq. 13})$$

$$X'Xb = X'y$$

$$b = (X'X)^{-1}X'y \quad (\text{Eq. 14})$$

The term b represents the least squares estimate of β . The regression terms in b correspond to the terms of the linearized pitching moment equation (Equation 6). With these regression terms, the model can be created.

$$\begin{bmatrix} C_{m_1} \\ C_{m_2} \\ \vdots \\ C_{m_n} \end{bmatrix} = \begin{bmatrix} 1 & \alpha_1 & \hat{q}_1 & \delta e_1 \\ 1 & \alpha_2 & \hat{q}_2 & \delta e_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_n & \hat{q}_n & \delta e_n \end{bmatrix} * \begin{bmatrix} C_{m_o} \\ C_{m_\alpha} \\ C_{m_q} \\ C_{m_{\delta e}} \end{bmatrix} \quad (\text{Eq. 15})$$

With the 2-1-1-2 maneuver, the short-period oscillation is induced.

1.3.3 Response Surface Methodology

In the previous section, the concept of OLS was demonstrated for a first order model. Because of the nature of aerodynamics, a linear fit model cannot well represent the collected data. A second-order model is required for a proper fit [2]:

$$y = \beta_0 + \sum_{i=1}^k \beta_i x_i + \sum_{i=1}^k \beta_{ii} x_i^2 + \sum_{i < j=2}^k \beta_{ij} x_i x_j + \epsilon \quad (\text{Eq. 16})$$

This is the second-order response model. Data showing quadratic curvature normally provides a strong model fit. If the data cannot be modeled using this equation, the data can be transformed to help adapt the data to the model. For example, the regression model can be transformed by taking the square root of all responses. This can also be applied to individual design factors or a combination of both approaches. Second-order models also require a design point for each repressor term, including all main factors, second-order terms, and interactions. Each design variable must also contain three levels to estimate the second-order terms.

1.3.4 Model Metrics

RSM provides several tools to evaluate model adequacy. The most notable for this purpose is the coefficient of multiple determination, or the R -squared number. This shows the portion of variation in the response that is explained by the model. The range of values falls between zero to one. The closer the value is to one, the better the model fits the data. Calculating R -squared first requires calculating the sums of the squares regression term and sums of squares error.

$$\begin{aligned}
 SS_E &= y'y - b'X'y \\
 SS_R &= b'X'y - \frac{\sum_{i=1}^n y_i}{n} \\
 SS_T &= SS_E + SS_R
 \end{aligned}
 \tag{Eq. 17}$$

Once calculated, the values are then used to compute R -squared.

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_E}{SS_T}
 \tag{Eq. 18}$$

The value of R -squared will always increase with additional factors even if those factors are found insignificant. R -squared adjusted compensates for the additional factors by considering the available degrees of freedom allocated to error and the model. R -squared adjusted will be in close agreement with R -squared if the model fits well.

$$R_{adj}^2 = 1 - \frac{\frac{SS_E}{n-p}}{\frac{SS_T}{n-1}} = 1 - \frac{n-1}{n-p}(1 - R^2)
 \tag{Eq. 19}$$

Mean square error (MS_e) is used as an estimate for the variance (σ^2). MS_e is obtained by dividing the error sum of squares by its degrees of freedom, $n-k-1$. This value and the mean square of regression (MS_r) are independently distributed chi-square random variables [5], meaning the hypothesis test for overall regression model significance can be evaluated using the F-test statistic.

$$F_0 = \frac{MS_r}{MS_E} = \frac{\frac{SS_f}{k}}{\frac{SS_E}{n-k-1}} \quad (\text{Eq. 20})$$

The value F_0 is evaluated against the F distribution for the available degrees of freedom. If the value is greater than the critical value, then the null hypothesis is rejected and the alternative is accepted or as expressed in Equation 20.

$$F_0 > F_{\alpha, k-1, N-k-1} \quad \text{If } (H_0 \neq 0) \quad (\text{Eq. 21})$$

The standard error for each regression coefficient in the model can be used to establish confidence intervals for the individual regression coefficients. To solve for standard error, the expression for the variance-covariance matrix must be derived:

$$C = (X'X)^{-1} \quad (\text{Eq. 22})$$

The diagonal component of the variance-covariance matrix, along with the unbiased variance estimate, is used to solve for the standard error:

$$se(b_j) = \sqrt{\sigma^2 C_{jj}} \quad (\text{Eq. 23})$$

where the standard variance is:

$$\sigma^2 = \frac{SS_E}{n-p} \quad (\text{Eq. 24})$$

For a desired confidence of 95% using the t -statistic, an interval on the regression coefficient may be established.

$$b_j - t_{\frac{\alpha}{2}, n-p} se(b_j) \leq \beta_j \leq b_j + t_{\frac{\alpha}{2}, n-p} se(\beta_j) \quad (\text{Eq. 25})$$

Variance Inflation Factors (VIF) are used to determine the degree of correlation in regression coefficients in the model. VIF provides a metric to see how severe this problem is for the current experiment. An accepted rule of thumb is that the value should be no higher than ten, but is desired to be less than five.

1.3.5 Residual Analysis

Residual analysis is based on the difference between the model's predicted response and the actual response [5]. Residuals are:

$$e = y - \hat{y} \quad (\text{Eq. 26})$$

Residuals are used to diagnose the concept of normality, constant variance, and time independence. Normality is checked informally by plotting a normal probability plot of the residuals. If the points all fall within the general line without curvature, then it passes the normality check. Constant variance checks use plots of residuals against the predicted response and factors to determine if scattering is consistent and variance is bounded. Residuals versus run number illustrates the time dependence of the design. If the residuals are randomly distributed, show no trends, and relatively bounded over time, then it passes the time independence tests.

Failure on each of these tests could indicate a problem with how the tests were conducted. For instance, a centrifuge might need to warm up before experimentation, thus if it is used before it is primed, the results will change over time. A design that fails these checks may involve conducting the test again or rerunning problem runs to provide a more stable model.

1.3.6 Central Composite Design

For the two factors, angle of attack and power, The Central Composite Design (CCD) uses 2^2 factorial design with $2k$ axial points positioned a distance α from the center to estimate pure quadratic effects, and somewhere between 3 and 5 center points to provide an internal estimate of error. The response generated by these factors, the pitching moment coefficient, is measured at each individual point and placed in a regression model for analysis. This design yields attractive prediction variance characteristics and a built-in robustness to “off design” points [7]. Figure 32 provides an illustration of the structure of the CCD design.

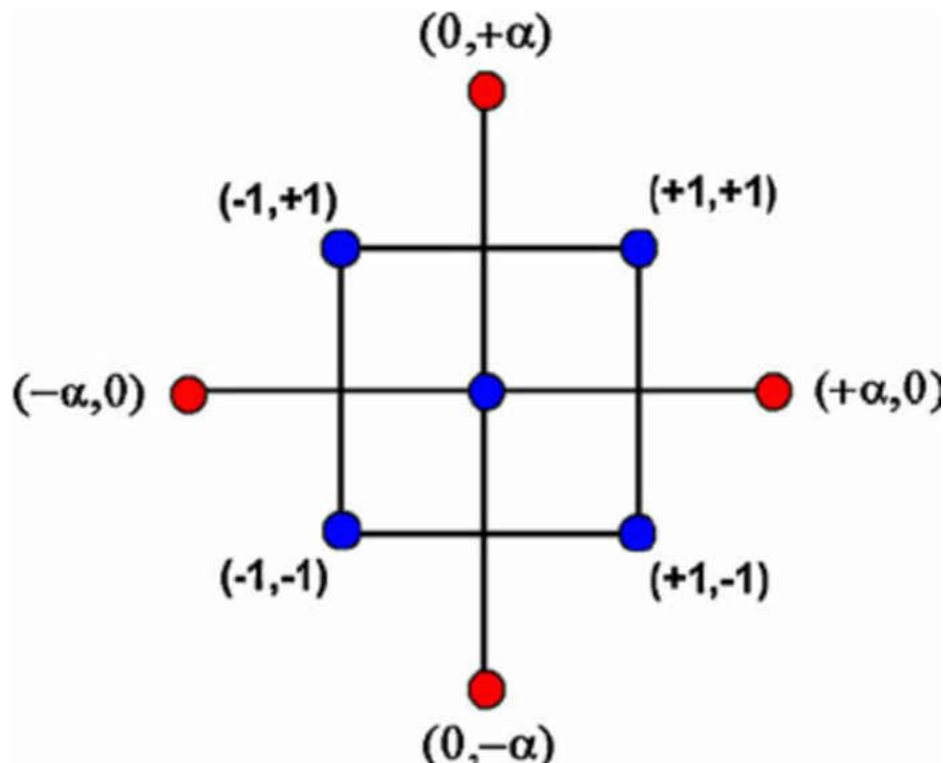


Figure 2: 2 Factor CCD

1.3.7 Face Centered Design

A Face Centered Central Composite Designed experiment (FCD) was chosen to test the overall performance of the fins compared against their dimensions. The Central Composite Design (CCD) is the most popular second-order design in use by practitioners, whose benefits are elaborated in the upcoming section [6]. The FCD makes for an effective second-order design for designs whose factors have hard limits, or when achieving the rotatable distance α is difficult. Placing these axial points to the edges of the design space, results in the most attractive variance distribution. The design is not rotatable, but this is not an important priority for a clearly cuboidal design like the FCD [6]. In the figure below, section a represents the factorial points, section b the axial points, and c is the full model with centers included.

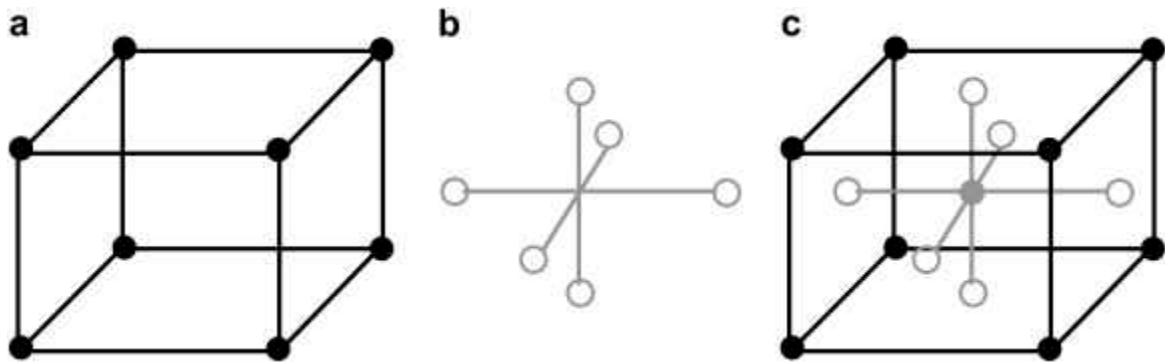


Figure 3: Face Centered Design

CHAPTER II

LITERATURE REVIEW

The choice of both methodology and technique is vital to produce a working system. The maneuvers discussed in this project are inspired by Morelli's[3] research into aircraft system identification, which has been applied successfully in several projects since his initial research. Likewise, the application of Design of Experiments and Response Surface Methodology (DoE/RSM) experiment strategies toward aerospace research has demonstrated success in recent years. Details on the hardware used will be discussed in the system overview section of this thesis.

2.1 SYSTEM IDENTIFICATION

This thesis was inspired by the work of several aerospace engineering scientists who integrated DoE/RSM together with System Identification methods. These methods enable the construction of regression models, leading to robust collection of data during flight testing. A review of system ID methods provided a concise list of desired parameters to be measured by the flight data system. Required flight maneuvers were also identified.

Dr. Vladislav Klein [8] provides a comprehensive guide to estimate an aircraft's aerodynamic parameters from flight data. In his essay, Klein details how he used certain mathematical models to characterize an aircraft's force and moment coefficients through several techniques, including Ordinary Least Squares, Stepwise regression, data partitioning, spline generation and Maximum Likelihood estimation. These same techniques were later used by Michael Lensi [9] to identify the stability and control derivatives for the TU-144LL Supersonic Transport Aircraft in 2000. Many of the theoretical obstacles faced in this thesis' experiment were solved by applying Klein's models, such as estimating the pitching moment of the aircraft.

Eugene A. Morelli [2, 3] studied the system identification cycle to improve flight test efficiency, provide immediate feedback for flight test results and enhance data quality through consistent input maneuvers. According to Morelli:

“The approach to aircraft system identification proposed here changes the philosophy of aircraft dynamic modeling, experimentation from designing test maneuvers, based on *a priori* predictions of the dynamic characteristics and evaluating the data quality post-flight, to an in-flight adaptive approach.”

These concepts are more thoroughly explored in Klein and Morelli’s *Aircraft System Identification: Theory and Practice* [3], in which they provide further details about modeling, in-flight testing methods, regression methods, interpolation, Maximum Likelihood methods, and Frequency Domain-based approaches. The data analysis techniques outlined in the book are utilized in this thesis’ experiment, as well as adopting their method of multi-step system identification.

In Ralph D. Kimberlin’s *Flight Testing of Fixed-Wing Aircraft* [10], he details the means by which to safeguard against noise factors when testing the performance of fixed wing aircraft. The means to calibrate electronic systems, avoid pressure effects, and other potential sources of error are explained, along with steps to minimize errors from various sources, many of which were used for this experiment.

In 2006, Noah Favaregh [7] proposed to use system identification maneuvers in conjunction with DoE methods to create a regression surface model using Ordinary Least Squares. By inputting the resulting design space into a regression model, it could then be used to map the force and moment coefficients of the aircraft. The systems developed in this thesis are meant to facilitate this experiment’s design approach through the construction of an air-data probe, a unique data acquisition system based on an autopilot, and a system identification-centric data-visualization computer program.

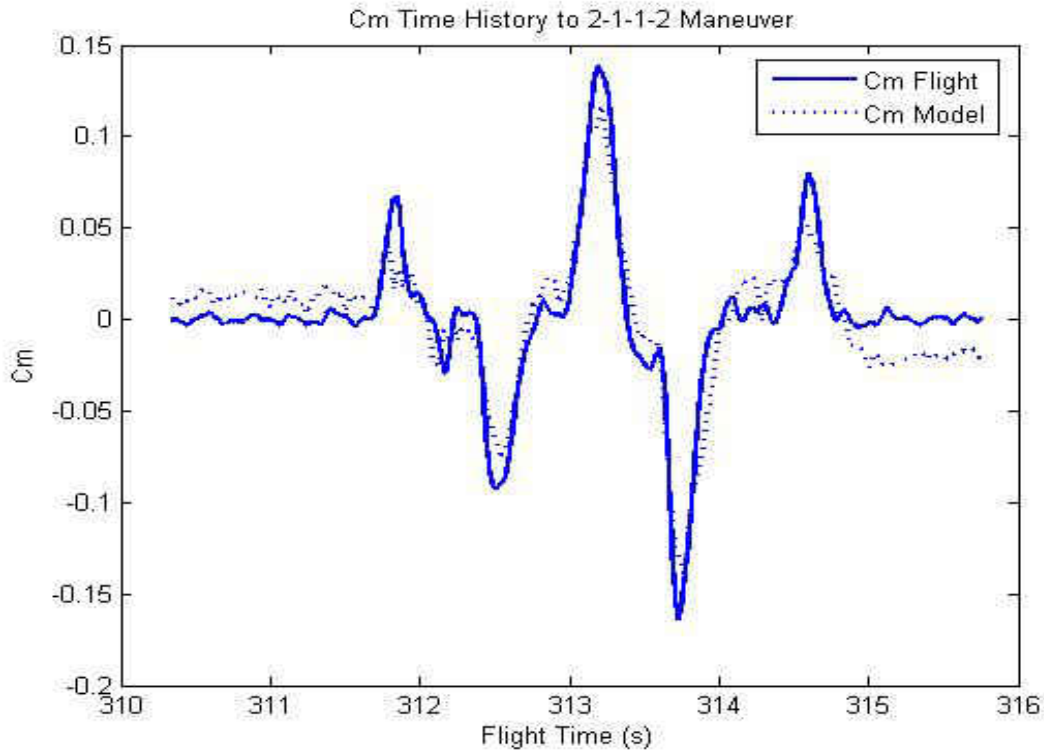


Figure 4: Typical Pitching Moment Coefficient Results From 2-1-1-2 Maneuver

2.2 DESIGN OF EXPERIMENTS AND RESPONSE SURFACE METHODOLOGY

In 2001, Morelli used DoE techniques while working with Richard Deloach on a non-linear modeling technique to characterize the response surfaces of aerodynamic force and moment coefficients. Data outputs based on the DoE used “multivariate orthogonal functions generated from the independent variable data as modeling functions in a least squares context to characterize the response surface” [2]. Results from this modeling were compared against those from the more traditional OFAT approach, “drag coefficient model fit and predictions were significantly degraded using OFAT data compared to using MDOE data” [2]. This research was a significant factor in choosing testing methods for this thesis’ experiment.

Deloach's "MDOE Perspectives on Wind Tunnel Testing Objectives [11]" details certain weaknesses in the traditional data-centric approaches to wind tunnel testing that make them less useful for the scientific research environment. His approach using DoE methods, as opposed to those used in OFAT research, allows for a more knowledge- vs. data-centric approach to wind tunnel testing. While OFAT is limited to one factor varied during each test, DoE creates a design space that not only tests the individual factors, but their various interactions as well. Later, Deloach [12] used DoE techniques to defend against systemic variation in wind tunnel testing. The systemic variation was shown to occur often enough to create doubt about the independence of random observations. In 2004, Deloach and Bobby L. Berrier [13] used the DoE approach to reduce the cycle time and operating costs in aerodynamic testing, when they performed an experiment at the 16-ft transonic tunnel to illustrate the advantages of the DoE method.

To better characterize a high-performance aircraft's non-linear aerodynamic behavior, Drew Landman et al. [4] investigated the use of RSM on a X-31 model at the Langley Full-Scale Tunnel. A five-level nested face centered central composite design space was constructed using the model's attitude and control surface inputs as the main factors. In 2007, Landman [14] conducted an exploratory study using RSM versus OFAT methods to map the aerodynamic behavior of the X-48B blended wing body model, again using attitude and control surface inputs as factors. When comparing the results, the RSM data were found more robust than the OFAT's, especially when compared to high noise levels related to the set point errors of the BWB's control surfaces.

CHAPTER III

SYSTEM OVERVIEW

An autopilot system is used to both collect flight characteristic data and execute specified maneuvers. This section provides details on the Pixhawk's hardware capabilities and a brief overview of the system's components. The Pixhawk strikes an ideal balance between ease of use and computational power for this project. Other hardware options are discussed in Appendix A.

A detailed look into the PX4 flight stack is also included, as several contributions were added to the firmware throughout this project. A brief description of the PX4 architecture is provided, along with descriptions of the various software tools needed to find and modify the open source code. Each firmware modification is given a brief summary describing its overall function. A more detailed breakdown of these changes can be found in Appendix B. A suitable ground station is needed to transmit commands to the aircraft in flight, and visualize collected data. Information on changing maneuvers in flight are discussed in the testing methodology section, so this section focuses on the excel document used to organize the information collected by the Pixhawk's logging application.

3.1.2 Connectivity

The board supports five Universal Asynchronous Receiver/Transmitters (UART), a Control Area Network (CAN) bus, Inter-Integrated Circuit (I2C), Serial Peripheral Interface Bus (SPI), and 8 Analog to Digital Converter (ADC) channels. The I2C comes with a I2C splitter, which allows for up to three signals to be passed through one port. This allows for multiple tools, such as GPS and Airspeed to work together. Only three of the eight ADC channels are available as exposed ports. Two pins are located in the 3.3V port and the last is the 6.6V port [15].

3.1.3 Gyroscope Specifications

The gyroscope is a three-axis MEMS sensor, part number L3GD20H. It features I2C or SPI communication, a low pass filter, and has an adjustable sensitivity. The sample rate is set to 760 Hz by the Pixhawk drivers, with a sensitivity of 2000 DPS, and a low pass filter of 50 Hz [15].

3.1.4 Accelerometer Specifications

The eCompass module part number LSM303D features a three-axis accelerometer and 3D magnetometer. The linear accelerator can measure up to +/-16g's, and be set as low as +/- 2 g's for maximum sensitivity. The magnetometer likewise has a minimum range of +/-2 and a maximum of +/- 12 gauss. The part includes a I2C serial bus interface and a SPI serial standard interface [15].

3.1.5 Secondary Inertial Measurement Unit

There is also a redundant inertial measurement unit (IMU) manufactured by InvenSense (part number MPU 6000). The chip is equipped with a three-axis gyroscope and an accelerometer. The accelerometer specifications are identical to the eCompass module, and serves mainly as a secondary failsafe. The gyroscope has full scale ranges of +/- 250, 500, and 1000 [15].

3.1.6 Analog to Digital Converter

The Analog to Digital Converter (ADC) is an 8-channel, 3.3V, 12-bit chip. Twelve bits results in 4096 potential integer values for input, meaning that each bit value represents a voltage difference of 0.0008 volts. Since the Pixhawk runs on a standard 5V supply, the 6.6V pin cannot receive maximum voltage, which reduces its overall resolution.

3.1.7 Global Positioning Sensor

The GPS sensor is an uBlox LEA-6H with a 5Hz update rate. It is equipped with an integrated ceramic patch antenna and a digital compass to provide consistency check against the accelerometer. The GPA is connected through the 12C and GPS ports on the Pixhawk. The manufacturer claims the chip can receive navigation information down to -162dBm and -148 dBm on a cold start. and a power draw of 121 mW while in operation [15].



Figure 6: UBLOX LEA-6H GPS

3.1.8 Barometric Pressure Sensor

Barometer (part number MS5611) is a high-resolution sensor made by Measurement Specialties. This barometric pressure sensor is optimized for altimeters with an altitude resolution of 10 cm. It has a digital resolution of 24 bits for pressure and temperature values. The chip also features integrated digital conversion and I2C and SPI serial connections, in addition to a second-order temperature compensation routine [15].

3.1.9 915 MHz Telemetry Radio

Produced by 3D robotics, the Telemetry Radio is designed to integrate with the Pixhawk, allowing it to connect to a ground station equipped with a paired radio. Each radio has a micro USB and a 6-wire cable connector, meaning that both radios can be purposed for both ground station transmission, and autopilot receiving. The current can be adjusted up to 100 mW to boost signal, and has a -117 dBm reception sensitivity [15].



Figure 7: 3D Robotics 915mhz Telemetry Radio

3.1.10 Ppm Encoder

The Crazypony PPM encoder module translates up to eight pulse width modulation (PWM) into one pulse position modulation (PPM) signals. The modified PPM signal is then passed onto the autopilot.

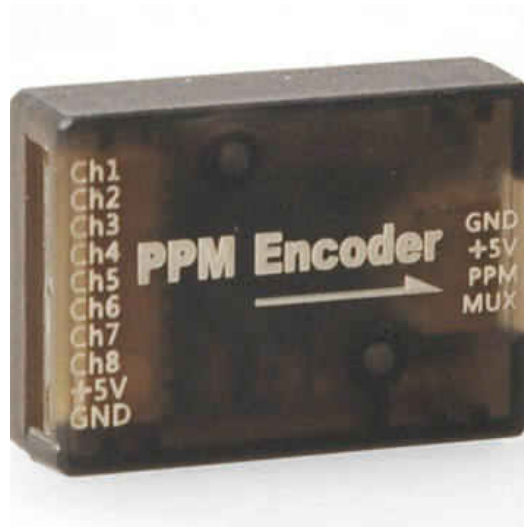


Figure 8: PPM Encoder

3.1.11 Buzzer

The Buzzer works together with the Pixhawk's multi-light LED display to communicate information about the system's state. It will communicate specific problems detected in pre-flight checks [15].



Figure 9: Buzzer

3.1.12 Arming Switch

This button serves as a secondary failsafe for arming the aircraft. Once all preflight checks are complete, the Pixhawk will blink either a blue light signaling no GPS reception, or a green light indicating the GPS is being received. Once in either of these states, by holding down the button for a few seconds will turn the blinking LED into solid red. At this point, the Pixhawk can be armed through the transmitter at any time [15].



Figure 10: Arming Switch

3.1.13 Voltage Regulator and Monitor

The default voltage regulator and monitor allows the user to connect a 4S LiPo battery through an XT60 connection, which provides the Pixhawk with a steady 5V power supply. This XT60 also tracks the battery voltage endurance and power levels during flight. For those using batteries with more cells, the default regulator is not advised. Other regulators are on the market that can handle the increased voltage from these batteries [15, 17].

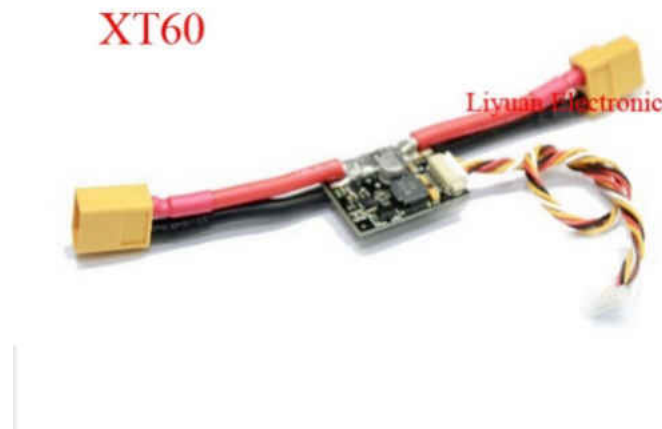


Figure 11: Voltage Regulator XT60

3.1.14 Pitot Static System

The pressure transducer was manufactured by Measurement Specialties, is a small, ceramic based, PCB mounted pressure transducer. It sports a 1 psi measurement range and a resolution of 0.84 Pa. Data from the transducer is passed as a digital signal with a 14-bit resolution. The transducer also collects temperature data from the Pixhawk's barometer to calculate the true airspeed from the indicated airspeed [15].

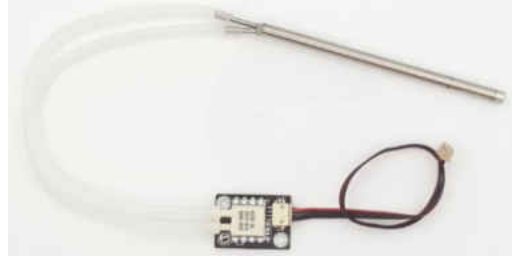


Figure 12: Robotics Pitot Tube

3.1.15 Mux Board

Produced by Cytron, the MUX board is an 8-channel RC multiplexer. This board is traditionally used for radio control systems and servo control between multiple control sources. The signals from the receiver are routed through input A, with an additional eighth channel used to toggle between the two inputs. The secondary system, in this case the Pixhawk autopilot, is connected through input B. This system's outputs are the servos for aileron, elevator, rudder, and electronic speed controller (ESC) that drives the motor.

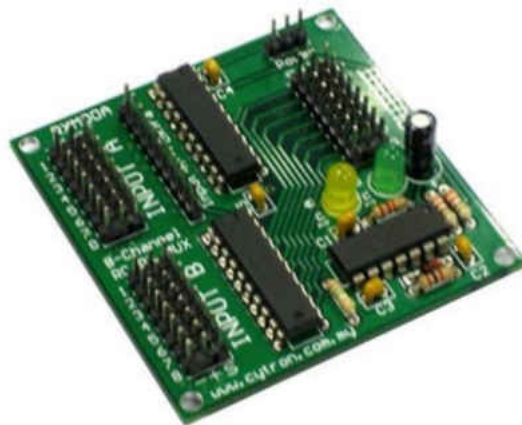


Figure 13: Mux Board

The multiplexer serves as a failsafe for the Pixhawk autopilot system. Control of the aircraft is routed through the Pixhawk during testing, yet many of issues could lead to catastrophic results. If the Pixhawk's gains are not properly tuned, for example, the system could over-compensate from its flight path. Gains set too low would result in a lack of response to external stimuli. Both situations could result in losing the aircraft. The MUX board allows the pilots to switch to direct receiver control in the event the autopilot deviates from its intended function.

3.1.16 RC Receiver

Transmitted signals are collected through the Spektrum AR9020 receiver. Manufactured by Spektrum, the receiver can support up to nine channels and can support additional channels by adding a Spektrum X-Plus8 channel expander into the SRXL port. The system also has two satellite receivers, which allow the receiver to see around conductive materials and maintain an unbroken connection regardless of position.



Figure 14: Spektrum Ar9020 Receiver

3.2 AIR DATA PROBE

3.2.1 Survey of Available Probes for AoA and Sideslip Maneuvers

Multi-hole type probes exploit variations in the coefficient of pressure C_p around a hemispherical or similar nose to provide flow angularity information [18]. The number of holes determine the sensitivity as expressed by the change in the coefficient of pressure per degree of deflection. The 5 hole design has a minimum number of ports for resolving the velocity vector while the seven hole design provides greatly increased angular range across two orthogonal planes, assuming the data can be processed [18]. The flow angle must be known for a range of coefficients, meaning that the multi-hole probe must be calibrated in a wind tunnel prior to use [16]. As can be seen in Jose C. Gonzalez and E. Allen Arrington's work [19], the change in angle was found to be mostly linear over a ten-degree range, meaning higher order regression techniques are likely unnecessary.



Figure 15: Multi Hole Probe



Figure 16: 4-Vane Air Data Boom Design

Vane based designs serve as a more direct means to determine orientation. The system works much like a weather vane, using a small aerodynamic fin that aligns itself with the relative airflow. These fins need only be connected to an optical encoder or a potentiometer to create a working system. Each of the individual components are relatively inexpensive and easy to replace if damaged, and calibration is a much simpler task that can even be completed without the use of a wind tunnel.

A major disadvantage over the multi-hole is the vane possesses its own dynamic properties in the airflow. This means special care must be given to ensure the flow dynamics of the probe do not interfere with the flow of the system. James T. Karam covers this in his paper "Dynamic Behavior of Angle-of-Attack Vane Assemblies [20]." Included is a general model, but this model makes assumptions that friction is negligible due to a relatively high minimum airspeed of 50mph. Such airspeeds are difficult to maintain in a small unmanned system meaning friction needs to be considered in the vane dynamics.

Minimizing the effects of probe dynamics also involves choosing the optimal fin shape. J. Wieringa performed tests on several fin shapes to determine an ideal vane shape. In it, he

concludes that a curved fine vane and double wedged fin vane obtained about the same magnitude of effectiveness as the flat fin, while the single wedge fin and streamlined fin showed decidedly inferior results [21]. The double fin configuration used in Wieringa's tests provides redundancy for readings, yet the lack of available ports in the Pixhawk limits the air data boom to a two-vane configuration.

3.2.2 Design Philosophy

The vane-based probe is intended to collect data measuring both angle of attack and sideslip angle, while avoiding interference due to aircraft proximity. It can be mounted either on the wingtips for puller propeller designs, or nose mounted for pusher based designs. To minimize effects on the aircraft's rotational inertia, especially for wingtip mounting, the probe must be as light as possible.

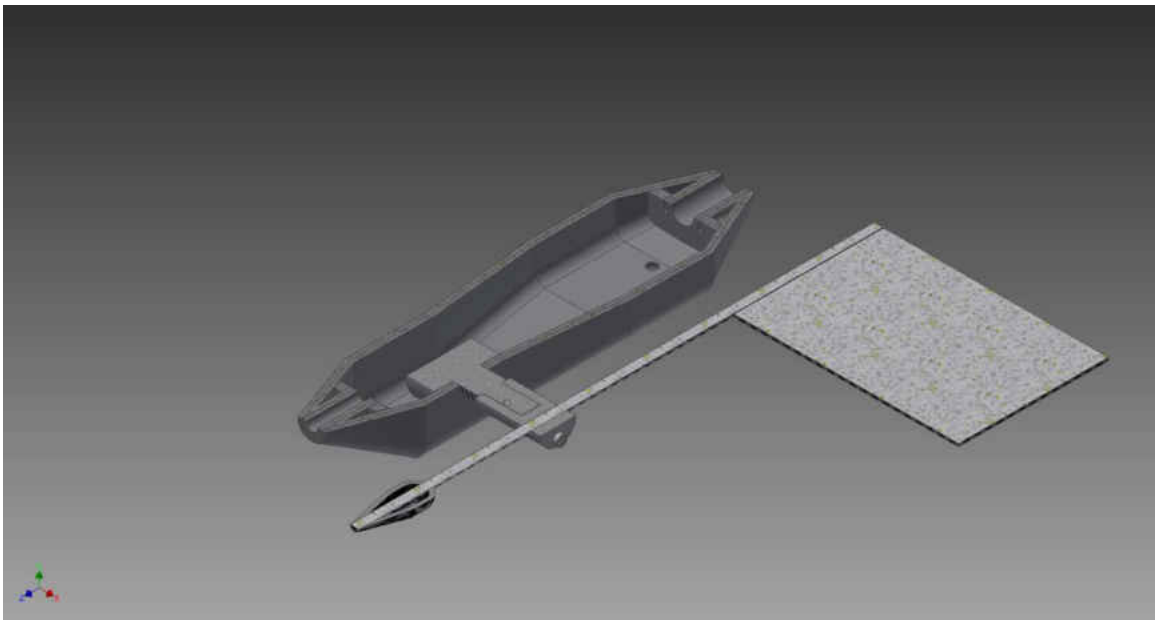


Figure 17: Air Data Vane Section View

The figure above shows a section view of the overall design. The fulcrum of the vane is attached to a rotary potentiometer, which turns as the vane orients to track the wind. Voltage readings from the potentiometer are transmitted to the Pixhawk through its 3.3V ADC channel and converted to degrees during data processing. The counterweight is a hollow streamlined pod whose weight can be adjusted after construction. The finished product features two orthogonal vanes to measure angle of attack and sideslip angle.

3.2.3 Fin Aerodynamics

The fins used in this project can be regarded as flat symmetrical airfoils with span b , chord c , and area S . Angle of attack (α) and sideslip angle (β) represent the angles between the vane axis and the freestream velocity V for each respective fin. Deviation from zero for either α or β will result in a static force F_v applied to the aerodynamic center, which sits at $1/4c$ from the leading edge. This aerodynamic center is represented as a dotted line in the free body diagram [1, 21, 22].

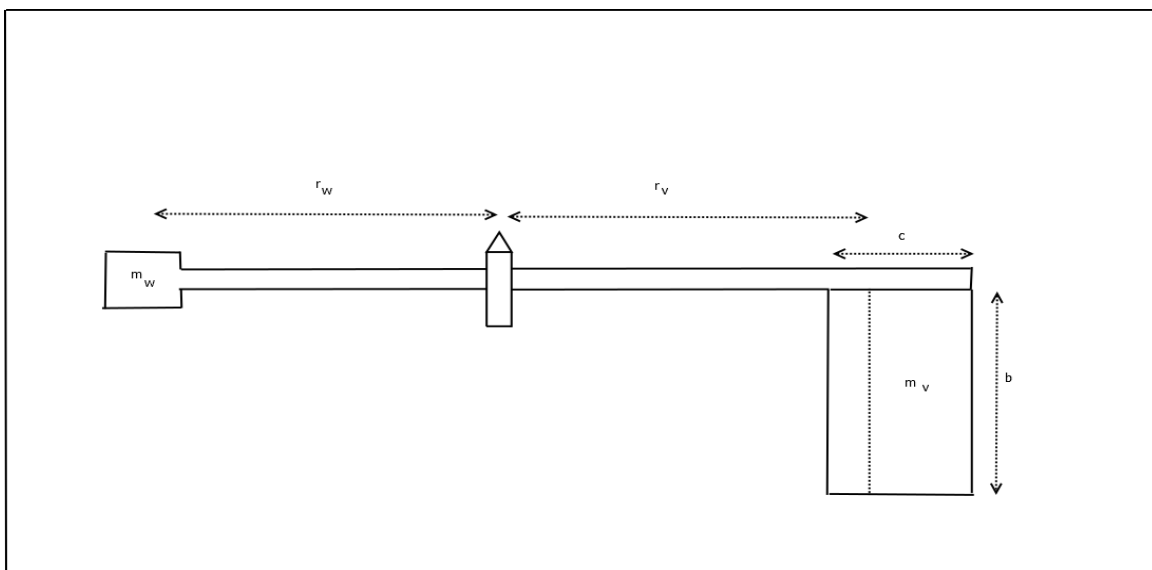


Figure 18: Air Data Vane Free Body Diagram Top View

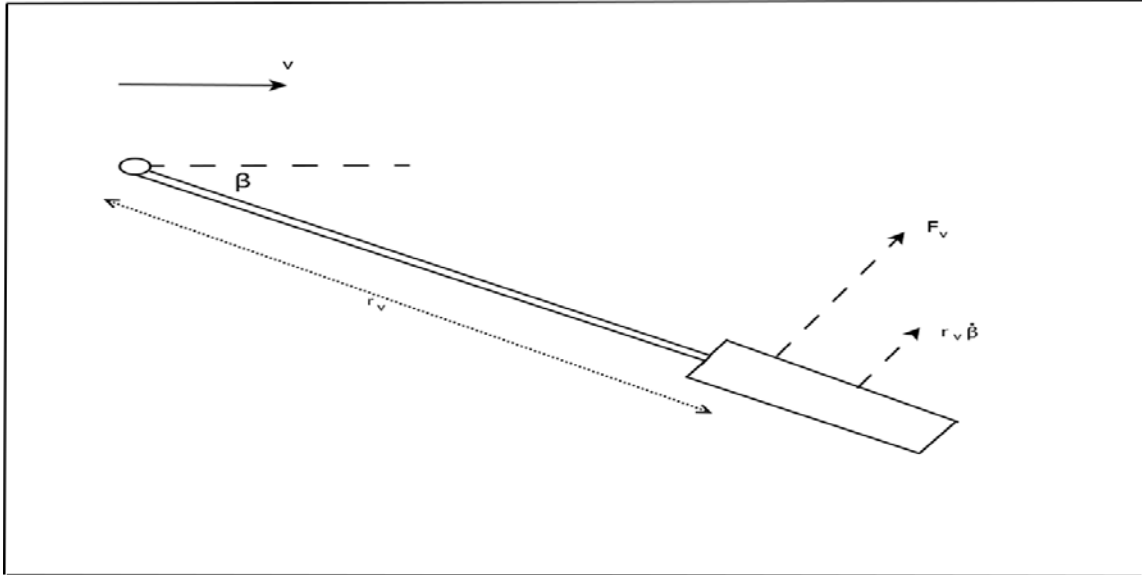


Figure 19: Air Data Vane Free Body Diagram Side View

The force F is the resultant of the lift and drag forces acting on the fin. Each of these forces depend on the dynamic wind pressure ($q=1/2\rho V^2$) and the span S and a normalized force coefficient unique to each force [22].

$$L = qSC_L \quad (\text{Eq. 27})$$

$$D = qSC_D$$

Force coefficients are dimensionless, and convenient to use as they are constant over a range of dynamic pressures.

$$C_L = \frac{L}{qS} \quad (\text{Eq. 28})$$

$$C_D = \frac{D}{qS}$$

In the case of vane fins, C_D is about an order of magnitude smaller than C_L [21]. The value of C_L for a symmetrical flat plate airfoil can be approximated by.[1]

$$C_L = C_{L\alpha}(\alpha) \quad (\text{Eq. 29})$$

$$C_{L\alpha} = \frac{C_{l\alpha}}{1 + \frac{57.3C_{l\alpha}}{\pi eAR}} \quad (\text{Eq. 30})$$

$C_{l\alpha}$ represents the lift curve slope for an airfoil with infinite span. Unless otherwise specified, 0.11/deg is a reasonable number to use in the equation above [1]. The span efficiency factor (e) has an optimal value of 1.0 and applies for airfoils with a constant downwash across the wing's span. Other planforms have a lower efficiency typically ranging between 0.95 and 1.0. The aspect ratio AR represents the general shape of the wing and is governed by this equation.

$$AR = \frac{b^2}{S} \quad (\text{Eq. 31})$$

The drag coefficient for a finite wing is calculated with the equation below. Induced drag is a function of the lift coefficient, with span efficiency and aspect ratio being constant [1].

$$C_D = C_d + \frac{C_L^2}{\pi eAR} \quad (\text{Eq. 32})$$

The flat plate airfoil drag coefficient C_d is largely dependent on the boundary layer thickness. Equation 33 assumes the boundary layer to be turbulent and gives the drag coefficient for flow on both sides of the fin [22].

$$C_{d\delta} = \frac{0.3747}{RE_c^{0.2}} \quad (\text{Eq. 33})$$

where RE is the Reynolds number for the airfoil based on chord. In the case of vane motion the effective resultant force F_v is always perpendicular to the vane arm [21]. When the vane is in motion however, a supplementary force results from the fin speed, which changes the effective angle of attack α_v . Changes to the effective wind speed are shown to be negligible for actual vane motion. This effective angle of attack is shown in Equation 34 [21].

$$\alpha_v = \arctan\left(utana\alpha + \frac{r_v\dot{\alpha}}{ucos\alpha}\right) \approx \alpha + \frac{r_v\alpha}{u} \quad (\text{Eq. 34})$$

The approximation is valid for angles of attack less than ten degrees. Using the vane's moment of inertia I, the equilibrium condition is [21]:

$$-I\ddot{\alpha} = r_v F_v = N\alpha_v = N\alpha + \left(\frac{r_v N}{u}\right)\alpha \quad (\text{Eq. 35})$$

where N is the torque

$$N = \frac{r_v F_v}{\alpha} \quad (\text{Eq. 36})$$

and

$$\frac{r_v N}{u} = D \quad (\text{Eq. 37})$$

is the dampening term for the vane. The equilibrium condition is a second order system, and if N is constant, has the solution

$$\alpha = \alpha_0 \exp\left(-\frac{D}{2J}t - 2\pi i \frac{t}{i_d}\right) \quad (\text{Eq. 38})$$

where α_0 is the vane's initial displacement and

$$t_d = \frac{2\pi}{\left[\left(\frac{N}{J}\right) - \left(\frac{D}{2J}\right)^2\right]^{\frac{1}{2}}} \quad (\text{Eq. 39})$$

is the dampened oscillation period of the vane. Excluding the damping term reduces this expression to the natural oscillation period.

$$t_0 = \frac{2\pi}{\left(\frac{N}{J}\right)^{\frac{1}{2}}} \quad (\text{Eq. 40})$$

The damping ratio provides an indication of the system damping.[1] Since vanes are intended to reach zero deflection within a finite period, the system must be sub-critically damped, meaning a damping ratio less than one. The critical damping D_0 for this vane is.[21]

$$\frac{N}{J} = \left(\frac{D_0}{2J}\right)^2 \quad (\text{Eq. 41})$$

The damping ratio is therefore

$$\zeta = \frac{D}{D_0} = \frac{r_v N}{uD_0} \quad (\text{Eq. 42})$$

Substituting the values from Equations 41 and 42 above results in this equation

$$\zeta = \frac{\pi r_v}{ut_0} \quad (\text{Eq. 43})$$

The damping ratio and natural wavelength are constant parameters of the vane motion and only these two are needed to know the full motion properties of the vane. Another useful factor to consider is the vane quality factor, which is derived on the relation of the fin and its counterweight.

$$K_v = \frac{a_v}{\mu_v \left(1 + \frac{r_w}{r_v}\right)} \quad (\text{Eq. 44})$$

where a_v is the ratio of the resultant force versus the current angle, and μ_v is the density of fin area S .

$$a_v = \frac{C_v}{\alpha} = \frac{C_L + C_D}{\alpha} \quad (\text{Eq. 45})$$

$$\mu_v = \frac{m_v}{S} \quad (\text{Eq. 46})$$

The greater the value of the vane quality factor, the greater its accuracy. It is worth noting that decreasing the ratio between the counterweight moment arm and the vane moment arm results in an increased vane quality factor. This means a heavier counterweight with a shorter moment arm will result in improved accuracy. This project sacrificed some quality to save weight, as the vane is also designed to mount on a wing tip for aircraft with tractor propeller configurations. Official W.M.O regulations request the vane finishes damping after 0.37 seconds at a speed of five knots. This requires a K_v value of at least 1.25 with a damping ratio greater than 0.3 [21]. Because these values depend on the angle of attack, the figures below map these values versus their respective angles of attack. A fin with a span of two inches and a chord of two inches was determined to be the ideal candidate for the airspeed ranges anticipated while maintaining a subcritical damping ratio and an acceptable vane quality factor.

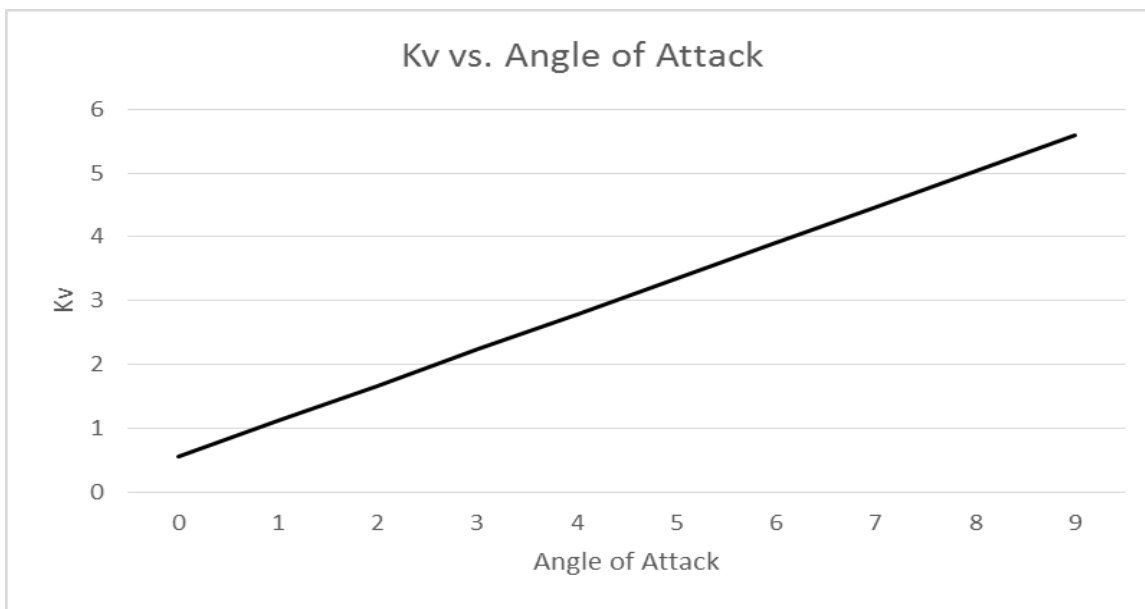


Figure 20: Vane Quality Factor vs. Angle of Attack

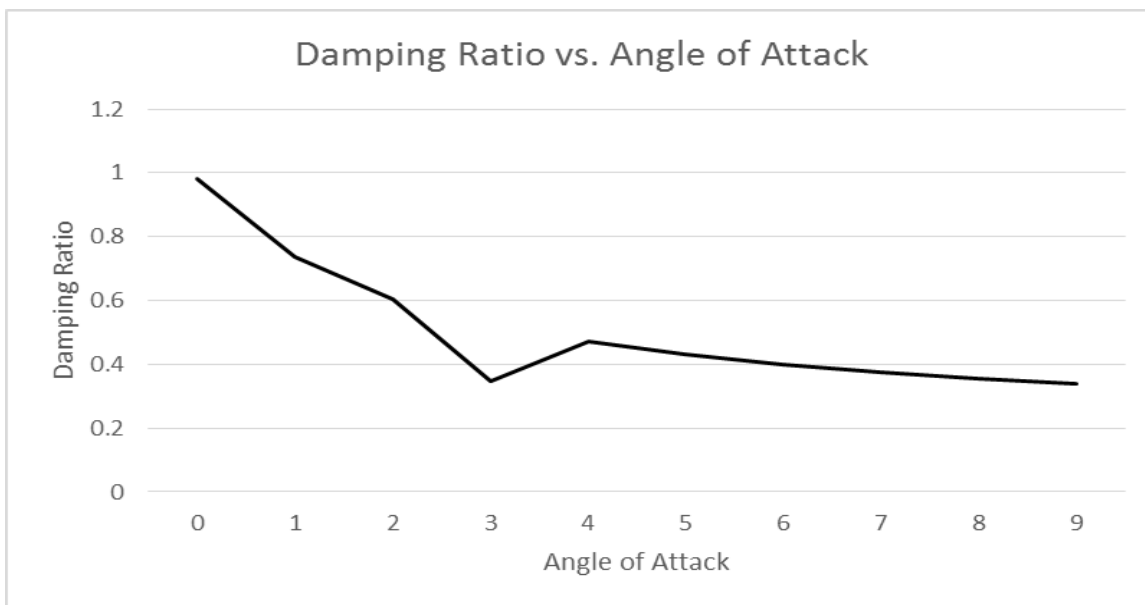


Figure 21: Damping Ratio and Vane Quality Factor vs. Angle of Attack

3.2.4 Potentiometer Selection

For measurements as sensitive as 0.3 degrees, the vanes require either a potentiometer with low resistance to torque or an optical encoder. While laser encoders are more sensitive, they are relatively expensive and add significant complexity for Pixhawk integration - as they require counter hardware. For this experiment, the Vishay 357 series potentiometer fits all the necessary criteria, including its reduced complexity and price. The Vishay diameter measures 7/8 inch, height 1.5 inches, and is capable of measuring changes in moment as small as ½ oz-in or 0.00353N-m [16].



Figure 22: Vishay 357 Series Potentiometer

3.2.5 Shroud Design

The shroud is designed to create minimal turbulence across the vanes, while containing the rotary potentiometer. Four two-dimensional graphs representing the dimensional constraints are lofted together. This allows the faces to remain symmetrical along the aircraft's y-axis, while reducing the amount of material needed [16].

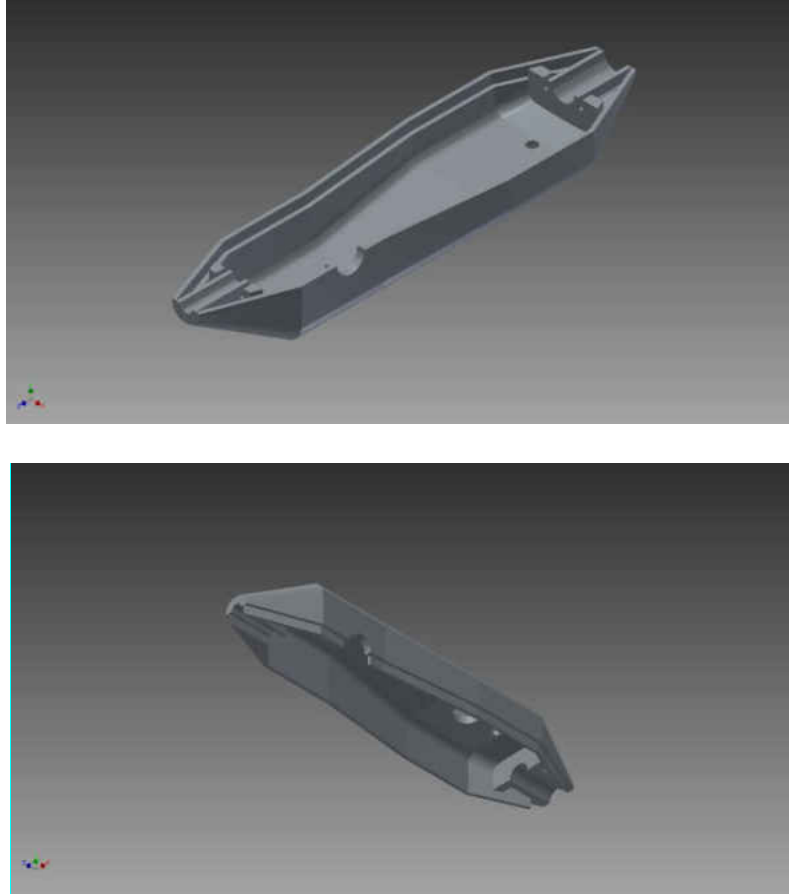


Figure 23: Shroud Design

A circular cylinder could be used, but a flat surface is preferable for maintaining orthogonality between the vanes. The design is printed in two halves in ABS plastic, both 0.1-inch-thick, with interconnecting pegs running along the body of the shroud. The final assembly is further secured by two bolts inserted through holes in the top.

3.3 PX4 FIRMWARE MODIFICATIONS

Several functions were added to the PX4 firmware package to integrate the air data vane into the Pixhawk, boost the sampling rate of the Pixhawk's logging function, and the creation of an additional RC channel for performing system identification maneuvers. Understanding how these modifications were made requires a surface understanding of the system's architecture. This section provides a brief overview of the PX4's source files and its overall structure. Software tools used for diagnostic and modification work are also elaborated on, followed by an explanation for the overall functionality of each modification. The source code for these changes can be found online at <https://github.com/Deafro/FirmwareAOASS> [23]. For those needing specific modifications, a more thorough breakdown of the code is provided in Appendix B.

3.3.1 Pixhawk Operation Flowchart

The flowchart below assists in visualizing the Pixhawk's operation. Red symbols represent user inputs, white hexagons are automatic processes, dark grey diamonds are decision points, and light grey symbols are functions accessible by the user [16].

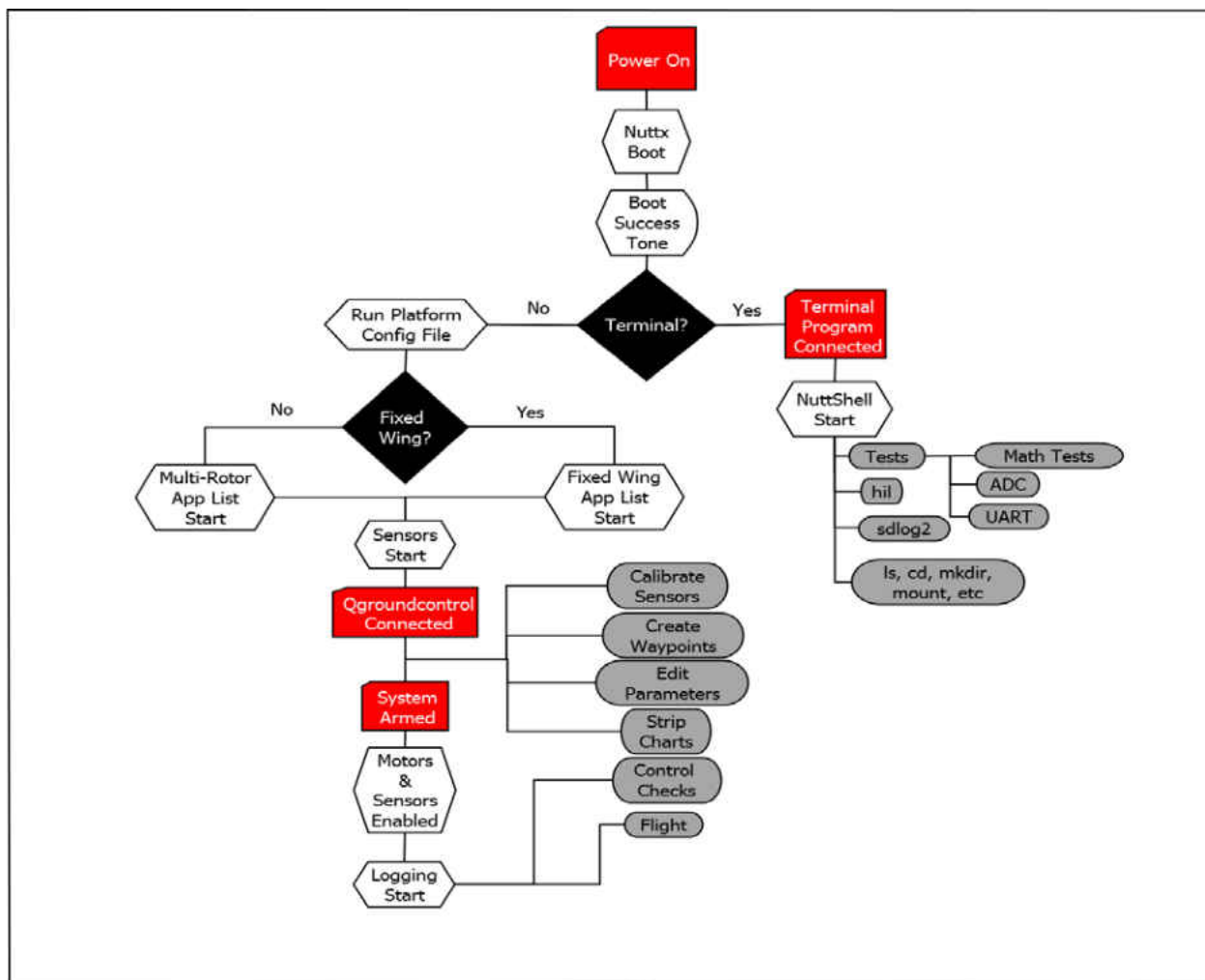


Figure 24: Operational Flowchart

The process begins when the Pixhawk is initially powered. After power is connected, the Pixhawk boots the Nuttx operating system and sounds the buzzer's success tone. Next, it checks to user connection by a terminal program, such as Teraterm. If the user is connected, it will open a NuttShell session and allow access to all commands listed after "NSH Start." This includes a suite of test applications not otherwise available.

If the user does not connect by terminal, the Pixhawk will run the platform configuration file stored in the firmware. This allows the Pixhawk to configure with the PX4 flight stack and set parameters accordingly. All sensors then start and the user has the option to interface with a ground control station. The user can then calibrate the Pixhawk's sensors, create a flight plan, edit MAVLink parameters, view parameter charts, along with any other available options.

With the system powered and linked to a ground station, the last step is to arm the system. This is done by pressing the arming switch and holding it down until the LED stops blinking. With the arming switch pressed, the transmitter's throttle stick is pushed to the bottom right corner until the light stops blinking and the buzzer sounds the confirmation tone. At this point, the Pixhawk is ready to fly.

3.3.2 Px4 Development Guide

The PX4 development team compiled a comprehensive guide for any users who want to modify the firmware. Experience using C++ is helpful but not required. Those interested in modifying the PX4 flight stack can go to <http://dev.px4.io/> for more information [24].

3.3.3 PX4 Firmware Overview

The PX4 firmware architecture used by the Pixhawk, functions much like a traditional computer. Each of its main applications listed in the src\modules directory is self-contained in

terms of dependencies and even runtime. The independence of these applications grants flexibility for development, as the modular nature makes future changes to the structure relatively simple.

The overall operating structure for the Pixhawk is linear, with the exception of the main applications. The core of the structure is the hardware itself, and just above it is the kernel of the NuttX operating system, which handles lower level functions like file IO and resource management. Drivers provided by the PX4 Middleware request sensor information from the kernel and pass the information to the publishing/subscribing message API called uORB that converts collected data to a more legible form (converting raw pin voltage to airspeed as an example), and then makes it available for the main applications. The applications are mostly independent, but can perform calculations that can be fed back into uORB to make available for other programs. A diagram of this hierarchy is shown in the figure below [16, 24].

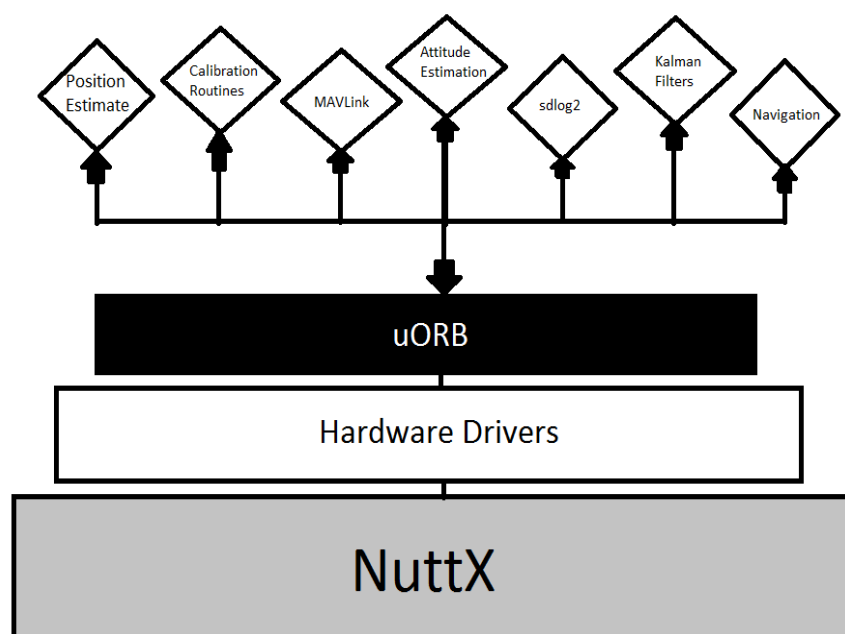


Figure 25: PX4 Firmware Hierarchy

3.3.4 MAVLink

MAVlink is a header-only message library used by Micro Aerial Vehicles (MAV) for communication between the vehicle itself and a ground station using a serial communication channel. The protocol was released by Lorenz Meier in 2009 and serves as the communication backbone for all commercial UAV/MAV models [25].

The protocol uses XML messages to generate MAVlink libraries in various coding languages, depending on the architecture of the host application. Each message is given an ID integer value ranging from 0 to 255, and houses all related fields for a specific instrument. An example of this is the GPS message, which stores the GPS timestamp, number of satellites, latitude, longitude, HDOP, and VDOP as separate fields under the same message.

Byte Index	Content	Value	Explanation
0	P a c k e t start sign	v1.0: 0xFE (v 0 . 9 :	Indicates the start of a new packet.
1	P a y l o a d length	0 - 255	Indicates length of the following payload.
2	P a c k e t sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	M e s s a g e ID	0 - 255	ID of the message - the id defines what the payload "means" and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, excluding packet start sign, so bytes 1..(n+6) Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

Figure 26: MAVlink Header Packet

The message ID is transmitted in a six-byte header packet from a ground station. The MAV specified in this header receives the packet, executes, and then returns the packet with the data attached. Details on the content of this transmission are shown in the figure below.

It was designed to handle wireless communications between an aircraft and a ground station through shared telemetry radios. It is a popular library for microcontrollers like the Pixhawk because of its small size in comparison to other networking program

3.3.5 NuttX

NuttX, designed by Gregory Nutt [26], is a real-time operating system made to work on machines like microcontrollers using lower processing power. It is open source and, based on its BSD license, only requires the name of the software developer's organization and cannot be used to promote other products without permission, which means it is legal to build upon the system.

The NuttX operating system serves as the conduit for the hardware and device drivers. It also controls other properties, including CUP load measurement, file IO, timers, and resource allocation. NuttX also includes a standard C++ library, device drivers for all ports, and various networking protocols.

NuttX can scale to low end hardware by implementing a configuration system that allows the user to specify the elements needed and remove everything else. Thus, the size of the final binary is compact enough to store in the autopilot's memory.

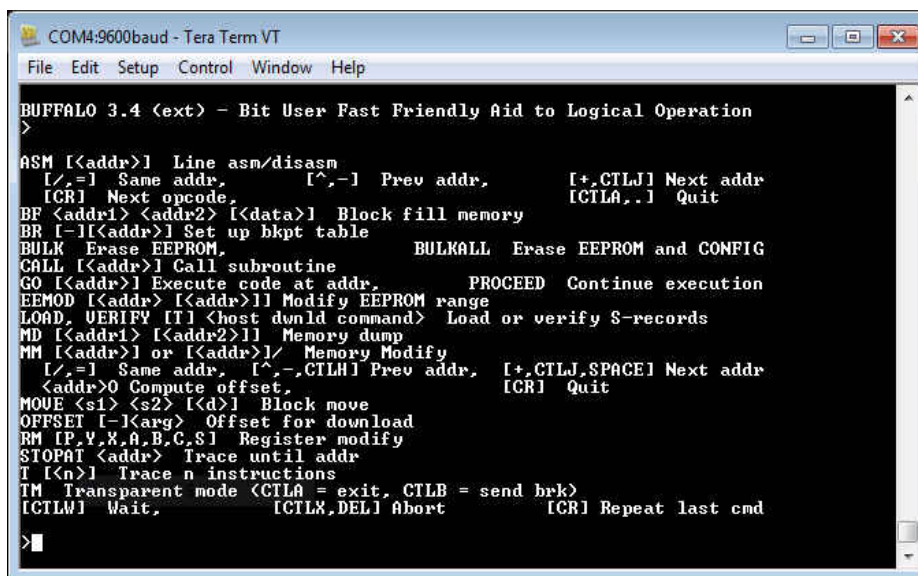
3.3.6 Nuttshell

NuttShell as its name implies is a command line shell used to interact directly with the NuttX operating system. Assessing this program requires a terminal program like TeraTerm and to trigger the program at a specific point in the Pixhawk's boot up process. The terminal program will display an "nsh>" prompt in the command line when the device is properly connected.

Typing “help” will provide a list of commands. These functions relate to file IO, script execution, and other tasks NuttX is responsible for managing. In addition to the default command, the PX4 firmware adds additional commands, including several test commands that run data quality checks on various sensors including the IMU, GPS, and the even the CPU [16, 24].

3.3.7 Teraterm

TeraTerm was initially created by Takashi Teranishi as an open source terminal. This program comes standard with the PX4 Toolchain. It transmits commands over USB to the Pixhawk. To specify the specific serial connection just go to Setup -> Serial port and select the COM port, baud rate, and size of the data packet for the Pixhawk connection. Connecting to the Pixhawk just involves selecting the proper COM port and baud rate [16].



```

COM4:9600baud - Tera Term VT
File Edit Setup Control Window Help

BUFFALO 3.4 (ext) - Bit User Fast Friendly Aid to Logical Operation
>
ASM [<addr>] Line asm/disasm
  [/,=] Same addr,      [^,-] Prev addr,      [+,.CTLJ] Next addr
  [CR] Next opcode,
BF <addr1> <addr2> [<data>] Block fill memory      [CTLA,.] Quit
BR [-] [<addr>] Set up bkpt table
BULK Erase EEPROM,          BULKALL Erase EEPROM and CONFIG
CALL [<addr>] Call subroutine
GO [<addr>] Execute code at addr,          PROCEED Continue execution
EEMOD [<addr> [<addr>] Modify EEPROM range
LOAD, VERIFY [T] <host dwnld command> Load or verify $-records
MD [<addr1> [<addr2>] Memory dump
MM [<addr>] or [<addr>] Memory Modify
  [/,=] Same addr,      [^,-.CTLH] Prev addr,      [+,.CTLJ,SPACE] Next addr
  <addr>0 Compute offset,          [CR] Quit
MOVE <s1> <s2> [<d>] Block move
OFFSET [-] [<arg>] Offset for download
RM [P,V,X,A,B,C,S] Register modify
STOPAT <addr> Trace until addr
T [<n>] Trace n instructions
TM Transparent mode (CTLA = exit, CTLB = send brk)
[CTLW] Wait,          [CTLX,DEL] Abort          [CR] Repeat last cmd
>

```

Figure 27: Teraterm Command Window

3.3.8 QT Creator

QT Creator provides a cross platform C++ integrated development environment, which includes a large range of tools for simplifying the coding process. It boasts a tabbed view of all open files, a file path tree showing the structure of the current project, and a customizable build and run interface [24].

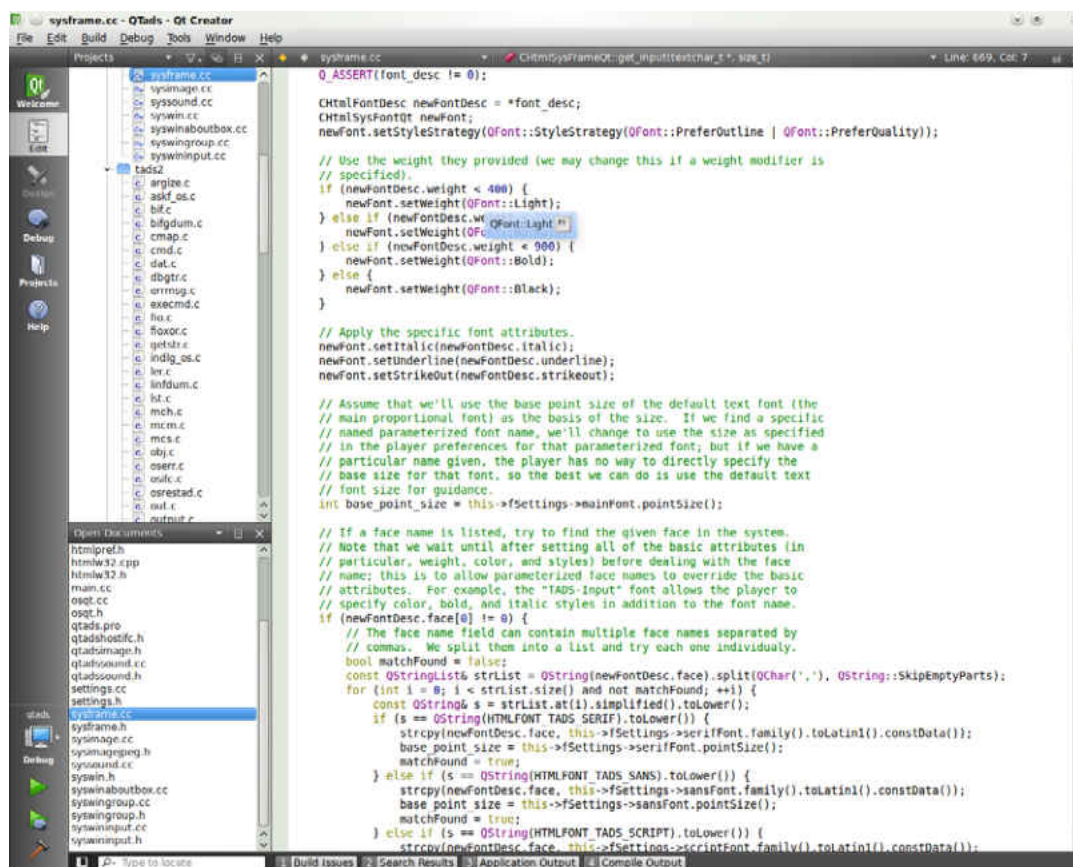


Figure 28: QT Creator GUI

The process of compiling code into package for the Pixhawk requires a specific command called “make.” Make is used to clean up specific directories, create support files, and compile the firmware. All that is needed is to create custom executable run command and place make in the

executable line. Adding “upload” as an additional argument will write the modified firmware to the Pixhawk after compiling.

3.3.9 Github

Github is a version control repository and internet hosting service. It provides the source code management functionality of Git, and includes other additional features including bug tracking, feature requests, and wiki pages for project documentation. It is the largest host for source code in the world with over 85 million repositories.

Open source repositories like the PX4 flight stack are free to access, provided the user has an account on the site, and creates a branching fork to the main repository. Forking allows a developer to modify an isolated copy and test changes without disrupting the core repository structure. Any computer can then request a clone of the modified repository to store in a local hard drive for practical application. The cloned repository will not update if further modifications to the forked repository are made, however the local computer can send a request to update the repository in a process known as “syncing” [23]. A sample screen shot is provided below.

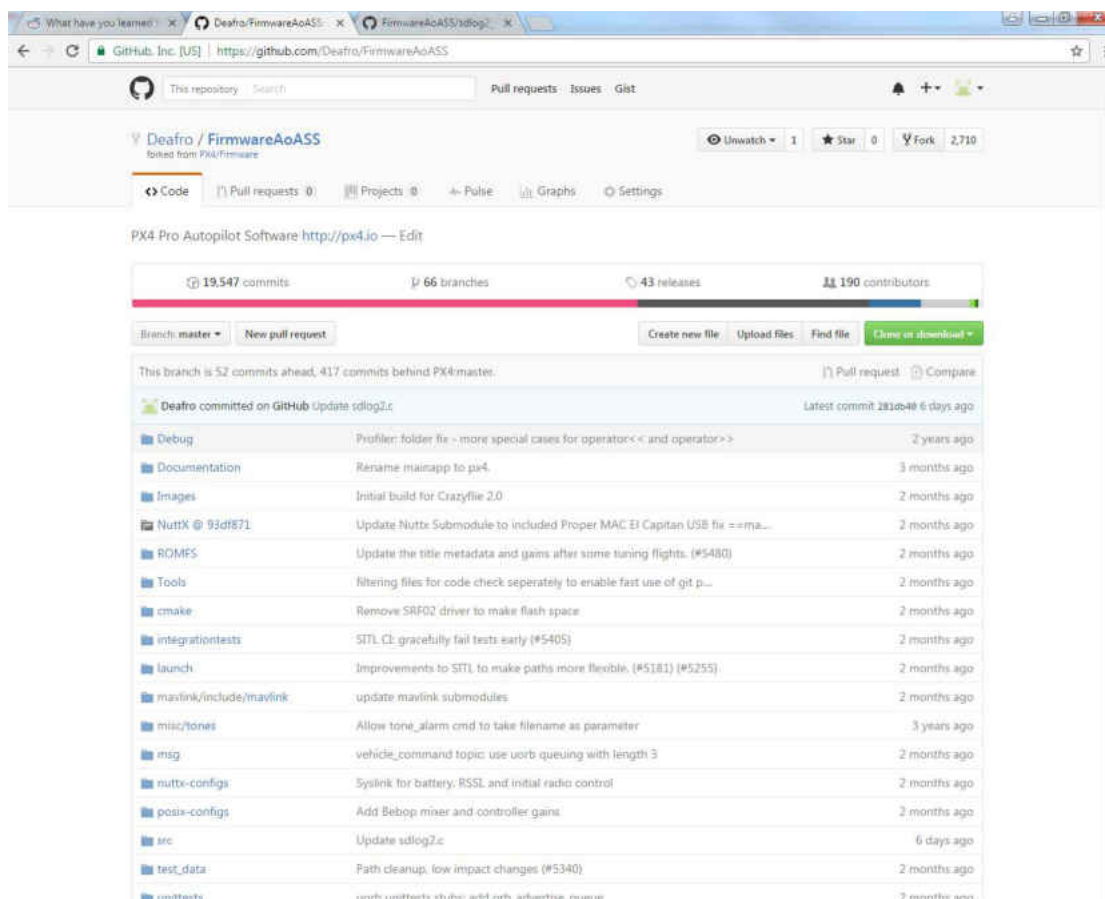


Figure 29: Github GUI

3.3.10 ADC Logging

The output signals for the Air Data vanes are directed into the Pixhawk's 3.3V ADC pins. By default, the readings from these pins are not directed to sdlog2. Modifications needed to be made to collect voltage data from adc_report, and redirect it into sdlog to be read in post analysis. The changes made are minimal and do not alter other sections of code.

3.3.11 System Identification Maneuvers

Using the autopilot to execute commanded control surface maneuvers means the excitation is consistent across all tests. This is especially useful for complex maneuvers like chirps. The

modification creates a separate channel input that can be bound to the transmitter that serves as an operation switch. When this switch is flipped, manual control over one or more control surfaces is overridden. The control surface is held at zero deflection for the time dictated by the parameter “SID_TRIM_TIME_B”. This gives the aircraft time to settle into trim flight before the maneuver is executed. Depending on the integer inputted into “SID_MANEOUVRE”, the plane will execute a variety of maneuvers, including chirps, sinusoids, step-in functions, and multistep maneuvers. This time allotted to this maneuver is determined by “SID_ON_TIME” and can range anywhere from one to fifteen seconds. Upon completion of the maneuver, the control surface(s) return to zero deflection for a time determined by the parameter “SID_TRIM_TIME_A”. Manual control is then returned to the pilot.

It is worth noting that until the maneuver has been fully executed, the pilot will have no control over the specified control surface(s). The multiplexer used in this experiment allows the pilot to override the Pixhawk’s control by switching to a direct receiver input, in the case where the maneuver threatens to destabilize the aircraft. Without a failsafe to override control, the maneuver could result in damage, or even total loss of the aircraft.

3.3.12 Sampling Rate

The current Pixhawk firmware has a default sampling rate of 100 Hz with a 16 kb buffer. This default rating can be modified by increasing the scheduling priority for the sdlog2 function to its maximum capacity. This can easily be done using the “SDLOG_PRIO_BOOST” parameter. A value of three gives maximum scheduling priority, and a boosted sampling rate of 250 Hz, while a value of zero grants no priority. It is recommended to have the default value of two, as any lower can result in the loss of information from sdlog.

3.4 AIR TITAN FPV

The Air Titan FPV is one of the largest foam based aircraft on the market, designed to house pan and tilt and first person view (FPV) camera equipment. The design features a pusher propeller, which allows for the air data vane to be mounted at the front of the aircraft. It features a wingspan of 99.2 in, providing ample lift capacity, and plenty of cabin space to house the Pixhawk. The recommended battery is a 6S Lipo 10400mAh, and can provide as much as thirty minutes to an hour of total flight time depending on power consumption.



Figure 30: Air Titan FPV Airframe

The Air Titan as an FPV platform is designed to provide stable trimmed flight while being able to carry a variety of payloads suited to various applications. It is the largest of the three Finwing FPV UAV models in production, which in conjunction with its high mounted wings provides a much lower wing loading, giving the aircraft very stable and predictable flight characteristics. The rectangular wingspan measures at 99.2 inches, with a total wing area of 1091.2 square inches and the overall craft weighs in at 5.3 pounds before electrical components.

The MK60 brushless motor provides 1500 W's of power and a static thrust of around 6 kg using the recommended 15x6 propeller and a 6S 10400 A lithium polymer battery. Three Finwing FAM052 metal gear servos control the ailerons and steerable nose gear with a rotational speed of 0.1 sec /60 deg at their operating voltage. The dual rudders are steered with two Finwing FUS017 servos that provide an identical rotation speed. The elevator servo is a 3-kg metal gear servo. The Phoenix Edge 75 A ESC was substituted for the recommended 85 A Opto for improved cooling and better mounting with near identical performance. Total flight time with these parts can last a full hour, though the various maneuvers performed in the upcoming flight tests dropped that ideal to forty minutes.

3.4.1 Air Titan Hardware Wiring Guide

To insure safety during testing, the traditional wiring of the aircraft was expanded to create system redundancy for receiver input. The modified Air Titan FPV requires four receiver channels to provide transmitter inputs to all control surfaces and the motor. These inputs are threaded into input A of the multiplexer, with A1 for the aileron, A2 for the elevator, A3 for the throttle, and A4 for the rotor. A8 is threaded to the receiver's second auxiliary channel, which will allow the pilot to toggle the MUX board between its A and B inputs.

The signal for these four channels is also directed into the Pixhawk's PPM encoder, along with a channel to control the Pixhawk's flight modes, and a channel to execute programmed system identification maneuvers. These PWM signals are then converted into a PPM pulse, which is then transmitted to the Pixhawk. The Pixhawk then interpolates this signal and outputs the information to the B input of the MUX board. Depending on channel A8's position, the A or B signal is then outputted to the servos and the ESC.

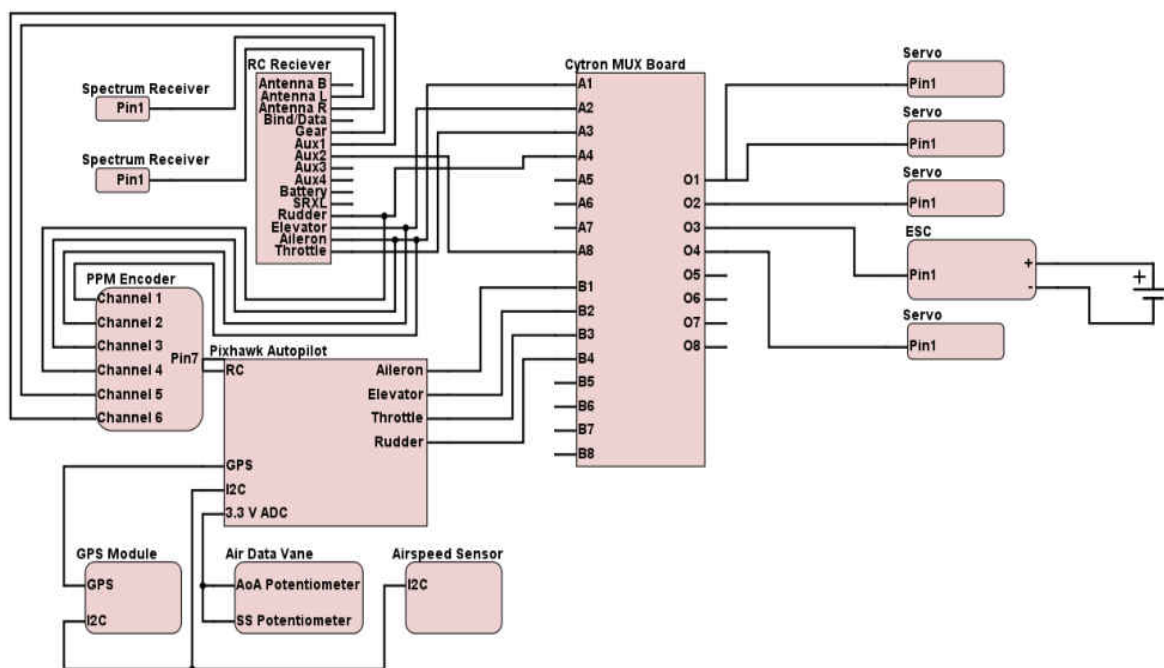


Figure 31: Wiring Guide

3.5 DATA ACQUISITION SYSTEM MOUNTING

For optimal results, the Pixhawk is secured at the aircraft center of gravity (c.g.) for the accelerometers. The air data vane is secured to a camera mount located on the roof of the aircraft, and its wiring is fed into the cabin and attached to the Pixhawk's 3.3V ADC port. For optimal

reading, the vanes must be set orthogonal to the body axis of the aircraft. The probe is set 1.5 fuselage diameters from the aircraft nose to avoid proximity effects from the aircraft [10]. A laser guide was used to ensure the vanes were properly aligned with the aircraft and orthogonal for their upcoming bench tests. A hole was drilled in the nose of the aircraft to accommodate the pitot static system. The fully assembled aircraft is displayed in Figure 24.

The airspeed probe is intended to work together with the air data vanes for the air data probe. Difficulties arose when it was discovered that the extensions to the 12C port wires produced signal reflections that corrupted the signal transmitted by the pressure transducer, which resulted in the Pixhawk's inability to register the airspeed signal.

Several methods of counteracting this problem were considered. An RC circuit could be spliced into the wire to filter out the undesired noise and reduce the damage caused by signal reflection. However, this would only provide a temporary stopgap measure and might possibly filter out other transmissions from higher signals. Another possible solution was to place a current amplifier on both ends of the circuit. These would boost the current in the circuit and improve the conductivity of the wiring and resolve degradation. The simplest and safest choice, however, was to extend the air hose, allowing signal wires to remain short. While this measure did not completely reach the Pixhawk, the hose extension allowed the Pitot tube to be placed beyond the wash of the propeller.



Figure 32: Air Titan Final Assembly

3.6 GROUND STATION SOFTWARE

A ground control station refers to all hardware used for UAV operation. Smaller UAV's like the one used in this experiment are operated through a traditional twin stick (mode 2) transmitter, and work alongside a portable computer with ground station software. These programs such as Mission Planner or Qgroundcontrol, are used for planning automated missions and receives sensor data through the UAV's telemetry radio. Only a few minor differences are present between the major software packages and all of them are free to install.

3.6.1 Qgroundcontrol

When researching ground-control software, it is important the system is easy to use and crucial that it is compatible with the Pixhawk flight-controller. Qgroundcontrol was constructed to pair with any UAV device that uses MAVlink communication. The user interface of QGC is designed to easily adapt to various screen sizes and resolutions, and its use of Qml for hardware acceleration is key for its use in low powered devices such as phones and tablets. Full set up support is available for both ArduPilot and PX4 firmware packages, including sensor calibration, flight mode configuration, and access to system parameters. Readings from the autopilot's instruments can be observed in real time by telemetry through its plotter tool. The source files are also free to clone and modify for custom needs, such as the addition of new MAVlink libraries.

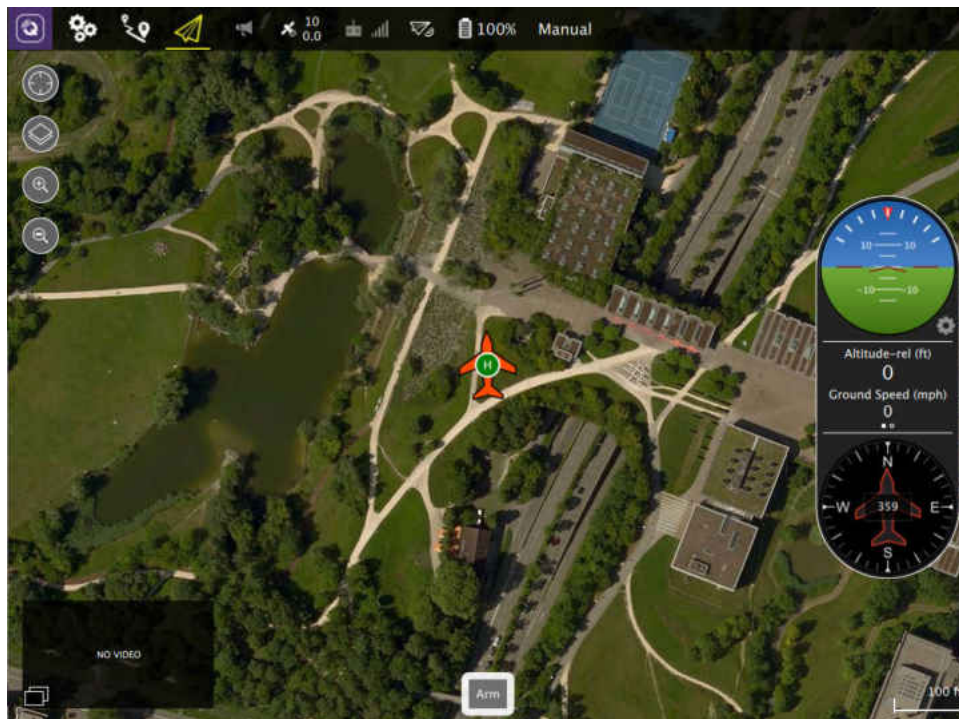


Figure 33: Qgroundcontrol

Qgroundcontrol is available on Windows, OSX, and Linux platforms in addition to iOS and Android devices [16, 25].

3.7 DATA COLLECTION

3.7.1 sdlog2

The sdlog2 program is the most reliable way to log data from the Pixhawk and it also runs in parallel with Pixhawk's other programs. sdlog2 writes data to Pixhawk's built-in microSD card whenever the Pixhawk is armed and it is entirely separate from MAVLink messages, which means clean output and organized data [23].

The program runs up to the sample rate specified by *-r* and checks what is being published by the uORB program from various sensors. uORB already publishes information for other programs to determine new data, so sdlog2 simply checks and writes the parameter at its sampling rate. sdlog2 also works in conjunction with an sdlog2_messages.h file that organizes related information as branches of related information.

sdlog2 writes the data in a binary format to reduce the number of high latency transfers to the microSD card. Using binary rather than text files allows for large quantities of written data, especially with the bandwidth constraints of low-processing microcontrollers. This ability makes sdlog2 the strongest contender for using the PX4 firmware. The Pixhawk is proven to log 143 separate parameters simultaneously at a rate of 100 Hz. Conversion of these files into readable formats is even simple with programs like Flightplot or PX4Tools, though in this case the programs could not process data from the air data vanes. Information on these alternative programs is included in Appendix A.

3.8 DATA VISUALIZATION

3.8.1 Introduction to Excel

Microsoft Excel a well-known spreadsheet program. By using Excel's formula script, a compatible template can be used to construct a data visualization tool.

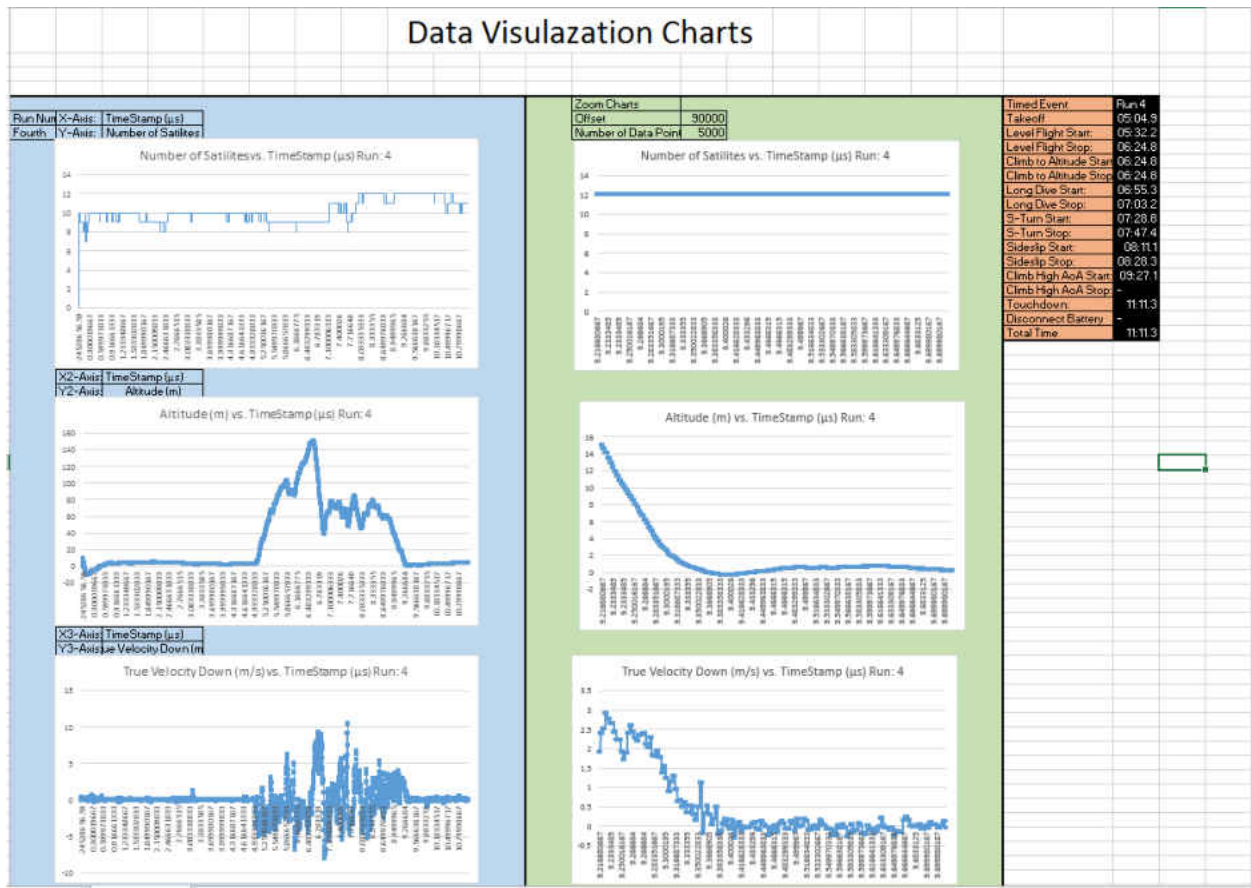


Figure 34: Excel Dashboard

Excel can catalog the parameters of the flight test, and with customized construction, it can also distinguish between individual runs. A proper use of formulas gives the end user the ability to shift through individual parameters across multiple runs with the push of a button. This section will detail the inner workings of this template and provide a guide for its construction. To avoid

redundancy, this section will cover the equations used to construct the dynamic X-axis. The same steps must be repeated for the Y-axis in each of the three plots in the figures below.

3.8.2 Assumptions

For this template, it is assumed all parameters provided in each run will be constant. This template associates the moniker heading of data with a number representing the column where the data are housed.

3.8.3 Categorizing Data

Each flight run in this program will have its raw data stored in a separate worksheet. All parameters are then highlighted and given a name based on their individual run number. The OFFSET formula is used in conjunction with the COUNT formula to occupy as many cells as there are parameters in the experiment. These formulas are stored as names under Excel's formulas tab and will not appear in any cell.

=IF(Dashboard!\$A\$7=1,OFFSET('1!\$A\$2,,,\$I\$2-1,COUNT('1!\$A\$2:\$A\$1048576)),0)
--

Table 2: Excel OFFSET Formula

For Excel to switch between runs, these formulas must be associated with a number corresponding to their run number. A two-column table is formed in a “Formulas” worksheet, with the first column being the run name and the second representing the run number. VLOOKUP is used to correlate these two values by referencing a drop-down menu stored on the “Dashboard” worksheet.

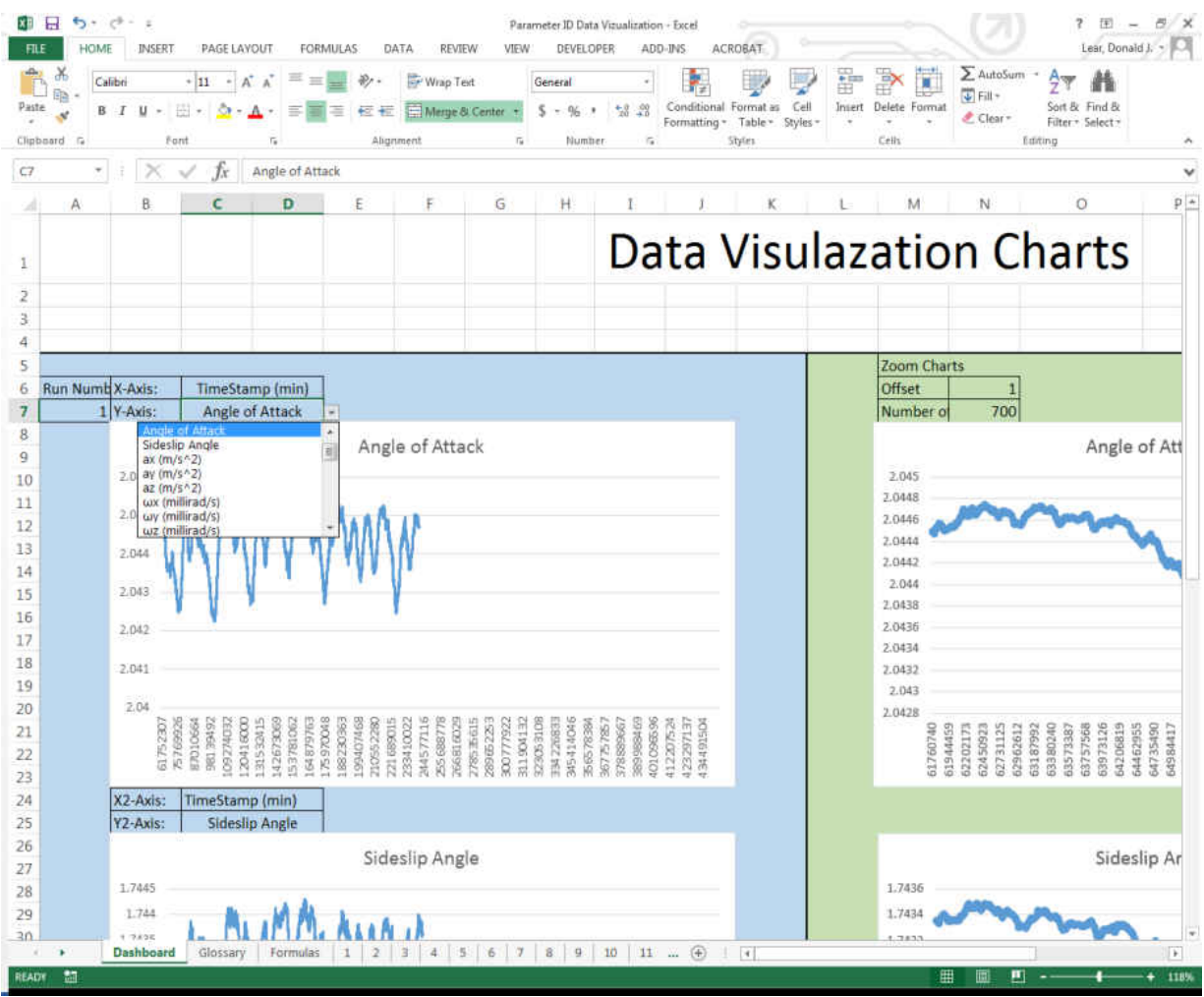


Figure 35: Dropdown Menu

The messages organizing the raw data are not intuitive for other users, so it is important to associate the monikers with their more comprehensive terms. For this, a “Glossary” page is created, detailing the mathematical term, the message term, the corresponding number, and a description from the PX4 message boards as to what this parameter means.

Meaning	Moniker	Marker	Raw Notes
Angle of Attack	ADAS_AoA	1	Angle of Attack ** Conversion: Value*3.3/4096
Sideslip Angle	AOAS_SS	2	Sideslip Angle
ax (m/s ²)	IMU_AccX	3	X Acceleration
ay (m/s ²)	IMU_AccY	4	Y Acceleration
az (m/s ²)	IMU_AccZ	5	Z Acceleration
lux (millirad/s)	IMU_GyroX	6	Angular speed around X-Axis (millirad/sec)
luy (millirad/s)	IMU_GyroY	7	Angular speed around Y-Axis (millirad / sec)
luz (millirad/s)	IMU_GyroZ	8	Angular speed around Z-Axis (millirad / sec)
Bx (milli tesla)	IMU_MagX	9	X Magnetic Field (milli tesla)
By (milli tesla)	IMU_MagY	10	Y Magnetic Field (milli tesla)
Bz (milli tesla)	IMU_MagZ	11	Z Magnetic Field (milli tesla)
P (Pa)	SENS_BaroPres	12	Barometric Pressure
T	SENS_BaroTemp	13	Barometric Temperature
ΔP (Pa)	SENS_DiffPres	14	Differential Pressure (1 hectopascal)
TimeStamp (min)	TIME_StartTime	15	Timestamp in Microseconds
GPS Fix Type	GPS_Fix	16	0-1: no fix, 2: 2D fix, 3: 3D fix, 4: DGPS, 5: RTK. Some applications will not use the value of this field unless it is at least two, so always
GPS HDOP	GPS_EPH	17	GPS HDOP horizontal dilution of position (unitless). If unknown, set to: UINT16_MAX
GPS VDOP	GPS_EPV	18	GPS VDOP vertical dilution of position (unitless) If unknown, set to: UINT16_MAX
Latitude (°)	GPS_Lat	19	Latitude (WGS84), in degrees * 1E7
Longitude (°)	GPS_Lon	20	Longitude (WGS84), in degrees * 1E7
Altitude (m)	GPS_Alt	21	Altitude (AMSL, NOT WGS84), in meters * 1000 (positive for up). Note that virtually all GPS modules provide the AMSL altitude in addit
True Velocity North (m/s)	GPS_VeIN	22	True Velocity in m/s in NORTH direction in Earth-Fixed NED Frame
True Velocity East (m/s)	GPS_VeIE	23	True Velocity in m/s in EAST direction in earth-fixed NED frame
True Velocity Down (m/s)	GPS_VeID	24	True Velocity in m/s in DOWN direction to earth-dixed NED frame
Course Over Ground (° * 100)	GPS_Cog	25	Course over ground (NOT heading, but direction of movement) in degrees * 100, 0.0:359.99 degrees. If unknown, set to: UINT16_MAX
Number of Satellites	GPS_nSat	26	Number of Satellites
Signal to Noise Ratio	GPS_SNR	27	GPS Signal to Noise Ratio
Indicated Airspeed (m/s)	AIRS_IndSpeed	28	Indicated Airspeed
True Airspeed (m/s)	AIRS_TrueSpeed	29	True Airspeed
Air Temperature (°C)	AIRS_AirTemp	30	Air Temperature
Data Points	Data Points	31	List of Data Points by Sequence.
Velocity Magnitude	Vel_Mag	32	Scalar Velocity (SQRT(Veast ² +Vnorth ² +Vdown ²))

Figure 36: Glossary Worksheet

Two VLOOKUP functions are employed to fully translate the parameter names. The first associates the mathematical term with the message term. The second associates the message term with the column in which the message data is stored. The transitive nature of this chain then gives the mathematical term used for the end user association with its respective data column. The lookup value for the first VLOOKUP is tied to the X-axis and Y-Axis drop down menus in the Dashboard worksheet. This needs to be done for three Y-Axes as it allows for the comparison of all three directions for any one given parameter.

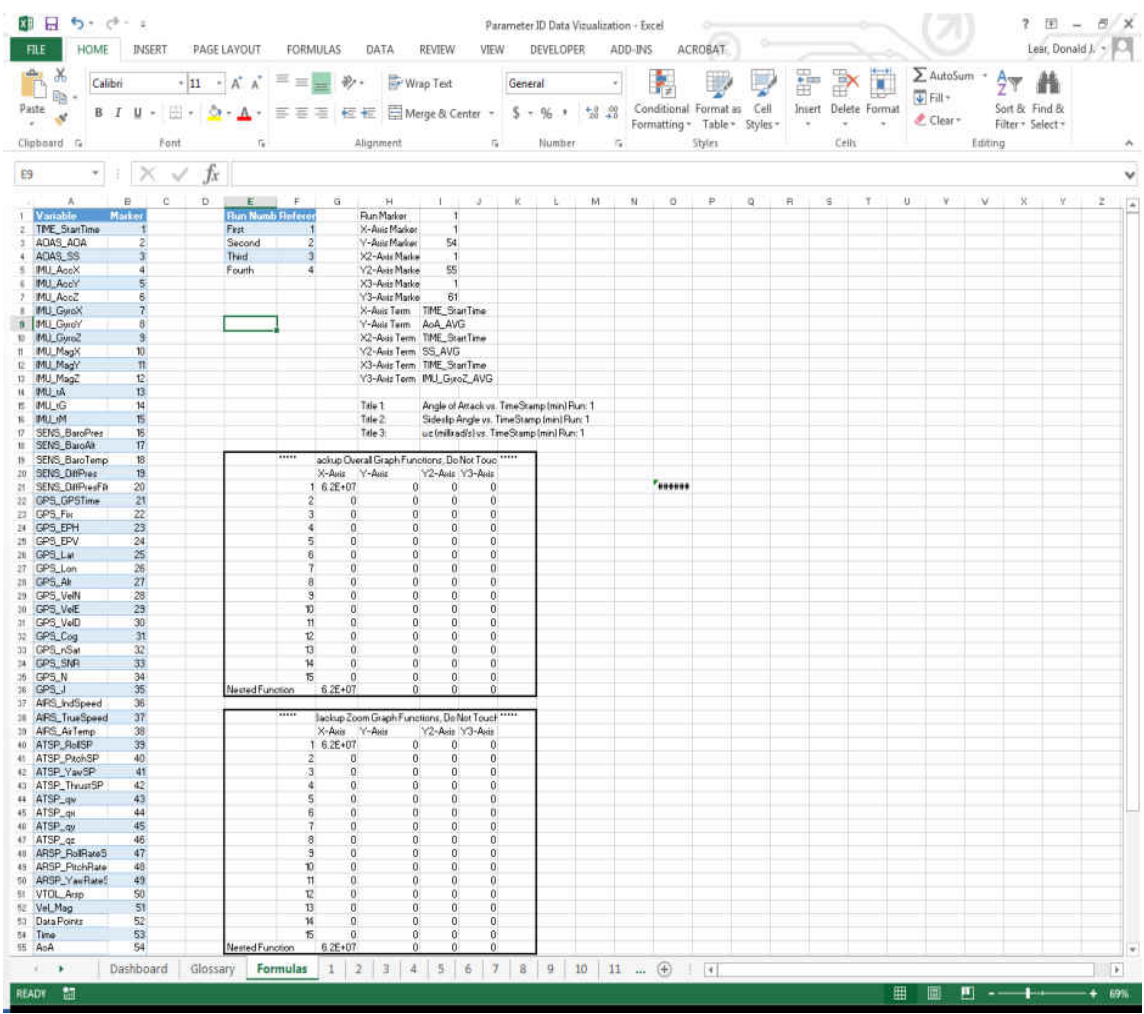


Figure 37: Formula Worksheet

```
=VLOOKUP(Dashboard!$C$6,OFFSET(Glossary!$A$1,1,,COUNTA(Glossary!$A$2:$A$202),4),2,FALSE)
=VLOOKUP(18,OFFSET($A$2,,,COUNTA($A$2:$A$202),2),2,FALSE)
```

Table 3: Vlookup

The final step is to create a nested IF function that will shift through all the runs based on their run number. Each true statement is an OFFSET function that uses COUNT to only take the cells with data in each column. A nested IF is made for the X, Y1, Y2, AND Y3 Axes. Each equation is stored as a name under name manager.

3.8.4 Equation: Nested If Function

The final step is to make a graph that relies on this Nested IF function. Right click the chart and choose “Select Data.” For the chart values, type “NameofWorksheet!Formula Name”

```
=IF(Dashboard!$A$7=1,OFFSET('1'!$A$2,,$I$2-1,COUNTA('1'!$A$2:$A$1048576)),IF(Dashboard!$A$7=2,OFFSET('2'!$A$2,,$I$2-1,COUNTA('2'!$A$2:$A$1048576)),IF(Dashboard!$A$7=3,OFFSET('3'!$A$2,,$I$2-1,COUNTA('3'!$A$2:$A$1048576)),IF(Dashboard!$A$7=4,OFFSET('4'!$A$2,,$I$2-1,COUNTA('4'!$A$2:$A$1048576)),IF(Dashboard!$A$7=5,OFFSET('5'!$A$2,,$I$2-1,COUNTA('5'!$A$2:$A$1048576)),IF(Dashboard!$A$7=6,OFFSET('6'!$A$2,,$I$2-1,COUNTA('6'!$A$2:$A$1048576)),IF(Dashboard!$A$7=7,OFFSET('7'!$A$2,,$I$2-1,COUNTA('7'!$A$2:$A$1048576)),IF(Dashboard!$A$7=8,OFFSET('8'!$A$2,,$I$2-1,COUNTA('8'!$A$2:$A$1048576)),IF(Dashboard!$A$7=9,OFFSET('9'!$A$2,,$I$2-1,COUNTA('9'!$A$2:$A$1048576)),IF(Dashboard!$A$7=15,OFFSET('15'!$A$2,,$I$2-1,COUNTA('15'!$A$2:$A$1048576)),IF(Dashboard!$A$7=10,OFFSET('10'!$A$2,,$I$2-1,COUNTA('10'!$A$2:$A$1048576)),IF(Dashboard!$A$7=11,OFFSET('11'!$A$2,,$I$2-1,COUNTA('11'!$A$2:$A$1048576)),IF(Dashboard!$A$7=12,OFFSET('12'!$A$2,,$I$2-1,COUNTA('12'!$A$2:$A$1048576)),IF(Dashboard!$A$7=13,OFFSET('13'!$A$2,,$I$2-1,COUNTA('13'!$A$2:$A$1048576)),IF(Dashboard!$A$7=14,OFFSET('14'!$A$2,,$I$2-1,COUNTA('14'!$A$2:$A$1048576)),,"Not Found"))))))))))))
```

Table 4: Nested If Function

```
Dashboard!X_AxisChart
```

Table 5: Name Chart Format, "X-Axis Formula Name"

3.8.5 Zoom Charts

Zoom charts follow the same data as the parent chart, but can adjust position and display a number of data points specified by the user. Because Excel charts can only display a limited number of data points, much of the information contained in the compressed, which hides the finer data. As the name describes, these charts allow the user to “zoom in” on trends to view more accurate charts of specific sections of the flight.

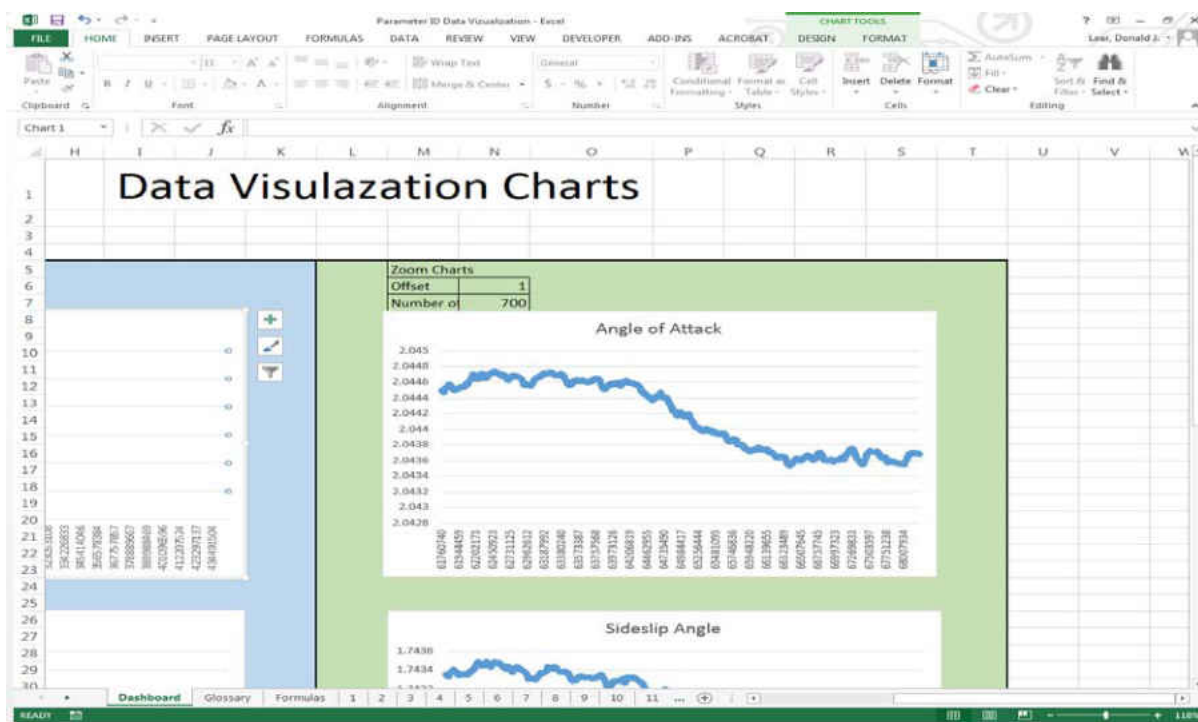


Figure 38: Zoom Chart

This chart works with the same mechanics as the compressed charts, but the height is represented by the number next to “Number of Data Points” on the dashboard and row is specified by “Offset.”

OFFSET('15!\$A\$2,Dashboard!\$N\$6,Formulas!\$I\$3-1,Dashboard!\$N\$7)
--

Table 6: Zoom Chart Code

The end product only requires the user to copy and paste log data into the run worksheets to access all the chart parameters. Additional parameters can also be added to the template by adding the proper information into the glossary and formula tables. The template can also be used to plot expressions dependent on these parameters, such as scalar velocity or lift.

CHAPTER IV

TESTING METHODOLOGY

This section details how the individual components of the data acquisition system are combined into a cohesive unit. It also details the use of this system in data acquisition using a trainer aircraft. To help organize these tests, they are broken down into three sections. Bench testing is for tools like the Pixhawk accelerometer and compass to find any hysteresis and bias in the data. The next section covers wind tunnel tests and details the calibration information regarding the air data probe. The actual flights come at the end, one testing the effectiveness of the data acquisition system, and the final flight regarding the DoE to derive the pitching moment coefficient.

4.1 BENCH TEST METHODOLOGY

These tests are meant to evaluate the performance of the individual sensors that make up the Pixhawk package. First, it is tested to see if all the individual components cooperate with the Pixhawk mainframe and to evaluate if it achieves its advertised standards. Since all the parts come from the same manufacturer, they are expected to cooperate. The exception to this is the custom air data probe, which must be calibrated and tested before integration with the Pixhawk's systems.

4.1.1 Accelerometer Noise Test

The Pixhawk is left stationary for ten minutes while the accelerometer collects data. The data points are summed to compute the standard deviation for each parameter using the equation below. This computation is performed for all accelerometer and gyroscope parameters. Any deviation from zero is treated as a bias. An exception is made in the case of the z-axis, which

reads a consistent -10 meters per second squared as the Pixhawk is affected by the Earth's gravitational pull.

$$\sigma^2 = \frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1} \quad (\text{Eq. 47})$$

4.1.2 Gyroscope Noise Test

The gyroscope is used to track the rotation of the Pixhawk. To ensure that it is measuring properly, the device is left on and remains still while the digital reader provides the positional data. Any deviation in this data should be from noise surrounding the machine, which gives a benchmark for later evaluations to compensate for this noise.

4.1.3 Analog to Digital Converter Noise Test

Given the sensitivity desired, a bench test was performed on the probe as well. The vanes are aligned using a laser level that projects orthogonal planes to be both perpendicular to themselves and the aircraft. The vanes are left to sit for ten minutes to collect data.

4.1.4 Vane Calibration

A key advantage to the use of vanes for measurement is the simplicity of calibration. The change in voltage from the potentiometers rotation is a linear relationship. A simple protractor can be used to correlate the changes in voltage with changes in degrees. The vane is moved in ten degree increments from -30 to 30 degrees. Readings were also taken at ± 90 degrees to maximize sensitivity, though it is unlikely the aircraft will experience these ranges, the data charted to excel is given in its raw voltage, and needs to be converted to degrees using a simple linear equation.

$$\theta = K_{\theta}(V - V_0) \quad (\text{Eq. 48})$$

The reference voltage V_0 is the voltage at zero degrees, which was discovered through the ADC noise bench test. The scaling factor is derived by the value that translates the collected

voltage data to degrees. This equation is later tested in the wind tunnel tests to confirm the accuracy of these tests.

4.2 WIND TUNNEL TEST METHODOLOGY

4.2.1 Test Conditions

All instrument calibration tests were performed in the ODU Low Speed Wind Tunnel's High-Speed test section. The section is 3x4x8 feet with a maximum airspeed of 55 m/s. Anticipated airspeeds are centered around 13.4 m/s though increased speeds are also tested. Data from this test are collected with National Instrument hardware and processed through LabVIEW.

4.2.2 Vane Dynamic Tracking

This test investigated the dynamic response of the vanes. A HS-755MG Giant Scale Servo was plugged into the Pixhawk and secured to a stand within the wind tunnel. The probe itself was placed in a shaft collar and secured to the servo arm. The air data vane was placed at a distance where pressure effects from the test stand would be negligible. The system identification maneuvers include an option for sinusoidal oscillations, and the frequency of those oscillations can be adjusted with the "SID_START_FREQ" and "SID_STOP_FREQ" parameters.



Figure 39: Wind Tunnel Test Stand

After the wind tunnel has reached the test velocity, the system identification switch on the transmitter is flipped, executing a $\frac{1}{4}$ Hz sinusoidal oscillation. To ensure vane consistency, the oscillation is repeated three times. The same process is repeated for $\frac{1}{2}$ Hz, $\frac{3}{4}$ Hz, and 1 Hz oscillations.

Data from the vanes are collected by a National Instruments data acquisition board, where a LabVIEW program applies the conversion to change the voltage values to their equivalent degree range. The data are then exported to Excel. An ideal sine wave with identical frequency is overlaid with the experimental results. Correlation between these two plots gives a metric for the dynamic tracking potential of the vanes.

4.2.3 Airspeed Calibration Methods

The pitot tube in the data probe is calibrated by mounting the device inside the wind tunnel and monitoring its output at various speeds. The recorded speed is compared against readings from the wind tunnel dynamic pressure measurements.

4.2.4 Fin Size Optimization

The fins used in the air data vane were derived from a theoretical baseline as established in the Air Data Probe section. A duo of tests was performed to validate the performance of these fins and ensure their damping and performance were within acceptable guidelines.

Fin span, fin chord, and moment arm r_w serve as the factors. The center of the FCD design space is established as the optimal fin calculated in the fin aerodynamics section, with a deviation of ± 1 inch for each factor to form the overall range. A vane was created for each combination of factors, and could be easily switched in between runs.

The air data probe's shroud was mounted in ODU's low speed wind tunnel and secured to a servo arm. Once the testing chamber reached the desired airspeed of 30mph, this servo arm received a command from the Pixhawk autopilot to perform a sinusoid motion with a frequency dictated by user input through Qgroundcontrol. The lab's PC data acquisition system collects the resulting position voltage data from the fin's potentiometer and compares it against the ideal sinusoid. The absolute sum of all discrepancies between the actual readings and the ideal motion serves as the response in the FCD. The response for each run are tabulated and Design Expert software is used to validate the fin size needed for optimal tracking.

$$y_i = \left| \left(\sum (V_{actual} - V_{ideal}) \right) \right| \quad (\text{Eq. 49})$$

Using the findings from the first test, the optimal fin undergoes a damping test. A solenoid is used to secure the fin at a specified angle on a fixture in the wind tunnel. When the test chamber reaches the target airspeed of 30mph, the solenoid allows the vane to drop. The Pixhawk's logging function reads the potentiometer voltage to determine the length of time needed for the fin to return to a neutral position. The overall design of the rig as mounted in the low speed wind tunnel is shown below.

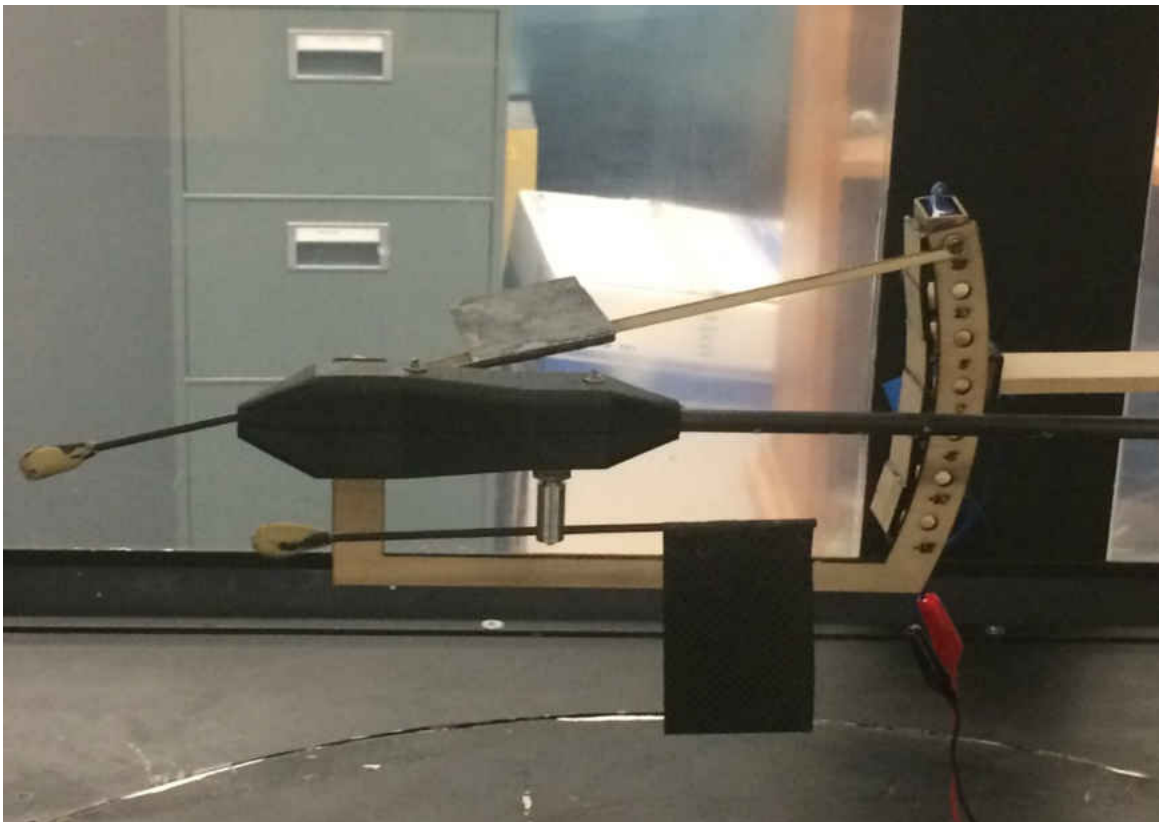


Figure 40: Vane Damping Test Stand

4.3 FLIGHT TESTING

4.3.1 Central Composite Design

The lift and drag forces acting on the aircraft are a function of angle of attack. Because these forces have a quadratic dependency on angle of attack, the resulting response across the design space is expected to demonstrate quadratic curvature. A second order response surface model is required. A full description of the CCD's structure is covered in Section 1.3.6.

4.3.2 Pilot Methodology

Part of the difficulty in dealing with unmanned aerial vehicles (UAV) flown "line of sight," is the sense of distance the pilot has from the aircraft. Unlike the traditional on-board pilot of an aircraft, the UAV pilot is a third-party observer, which means the pilot only uses one of his five senses to pilot the aircraft. A pilot of a manned aircraft constantly scans instruments, feels the inclination of the plane, and knows where the plane is going from the craft's altitude vantage point. Not only does a UAV pilot have a much more difficult time orienting the plane, but also knowing if something is not working correctly during flights [27].

An RC pilot with considerable experience, was elected to help pilot the Air Titan through its maneuvers. To prevent potential coupling with lateral dynamics it was required that the sideslip angle remain as close to zero as possible. The plane should also remain at a similar height during tests, but this was not considered to be a high priority, as the change in altitude at these ranges was assumed to be negligible.

4.4 MANEUVERS

The firmware modifications section focused on the general intent behind the additions to the PX4 firmware, and Appendix B details step by step the changes made for execution of

maneuvers. "System identification is the determination, on the basis of observation of input and output, of a system within a specified class of systems to which the system under test is equivalent [3]." In the case of aircraft, this refers to knowing its stability and control derivatives. This section details how these maneuvers work when executed. All parameters can be accessed through any ground control software, and can be changed in flight using the telemetry radio.

4.4.1 Multistep Maneuver

The 2-1-1-2 Multistep is designed to be quick and easy to perform while exciting a wide band of frequencies. The maneuver begins by setting the elevator to zero deflection for a time dictated by the parameter "SID_TRIM_TIME_B". The elevator then moves through a chain of pulses alternating between full positive and full negative deflection[3]. The total time allotted to complete the maneuver is dictated by the parameter "SID_ON_TIME". As the name suggests, the elevator is held up for 2/6 of "SID_ON_TIME", then 1/6 down, 1/6 up, and 2/6 down. The elevator then returns to zero deflection and remains for the time dictated by "SID_TRIM_TIME_A". These deflections can be reversed by changing the sign on parameter "SID_AMPLITUDE". A visual representation is provided in the figure below.

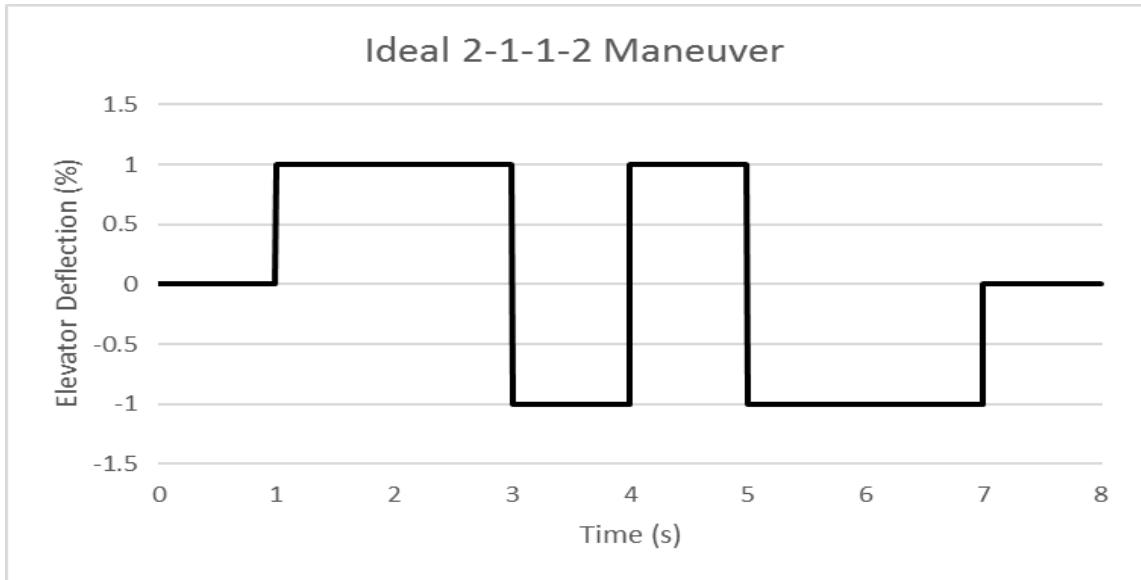


Figure 41: Ideal 2-1-1-2 Maneuver

Any series of pulses with different widths can be considered a multistep. The 3-2-1-1 maneuver uses a series of incrementing square waves to simulate a sinusoidal frequency sweep. This input can be difficult since the 3 pulse tends to drive the aircraft off flight conditions. For those cases, the 2-1-1 input can be used instead. These additional maneuvers as well as the 2-1-1-2 are available for the pitch, roll, and yaw channels.

4.4.2 Chirp and Sinusoid Maneuvers

The chirp maneuver is a sine wave that gradually changes in frequency throughout the maneuver. Inputs oscillate between maximum and minimum deflections dictated by “SID_AMPLITUDE”. Like the multistep maneuver, the plane will hold the control surface at zero deflection both before and after the maneuver as dictated by parameters “SID_TRIM_TIME_B” and “SID_TRIM_TIME_A”. When the maneuver begins, manual control of the specified control surface is overwritten, and executes the formula shown below.

$$P(t) = A * \text{Sin}(2\pi f_{start}t - \Delta f * t^2/2\Delta t) \quad (\text{Eq. 50})$$

“A” is the amplitude of the chirp as dictated by “SID_AMPLITUDE”, the starting frequency is controlled by “SID_START_FREQ” and the end frequency is controlled by “SID_STOP_FREQ”. The total time allotted to this maneuver is controlled by “SID_ON_TIME”. A plot of the ideal maneuver is shown in the figure below.

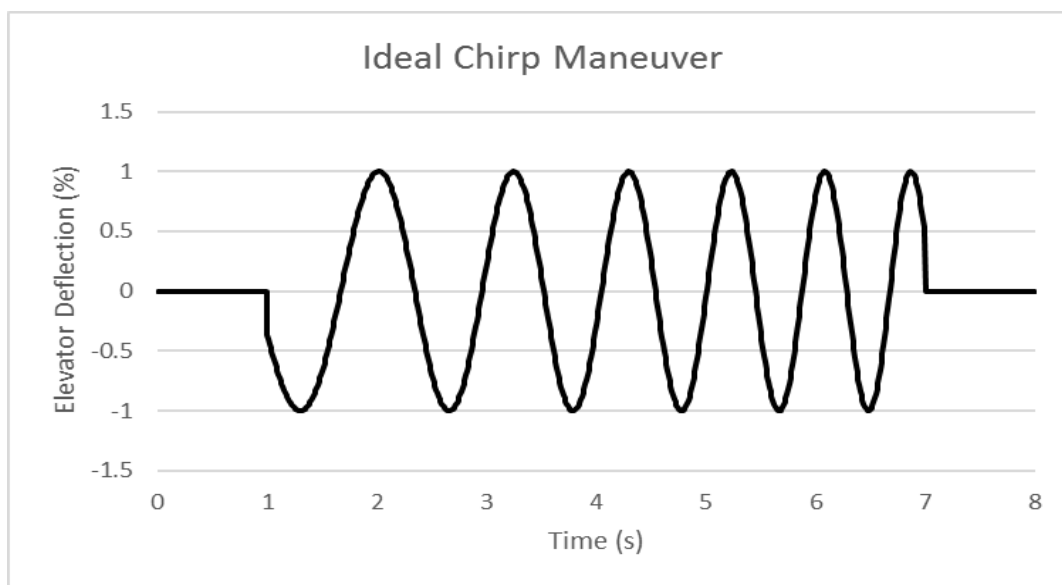


Figure 42: Ideal Chirp Maneuver

As with the multistep maneuver, the chirp maneuver is available for the pitch, roll, and yaw channels. The chirp function is also capable of performing simple sinusoid maneuvers. Simply

setting “SID_START_FREQ” and “SID_STOP_FREQ” to be equal, will reduce the chirp equation back to a traditional sine function.¹

¹ The system is also capable of step in maneuvers for aileron, elevator, rudder, and throttle channels, but these were not flight tested. For more details, see Appendix B.

CHAPTER V

DATA ANALYSIS

5.1 BENCH TEST RESULTS

5.1.1 Accelerometer Noise Analysis

The noise profile of the accelerometer provides an overall view of the sensors performance. The figures below are accelerometer data from each of the three axes, collected over a ten-minute period. The sampled data underwent signal averaging to boost the signal to noise ratio.

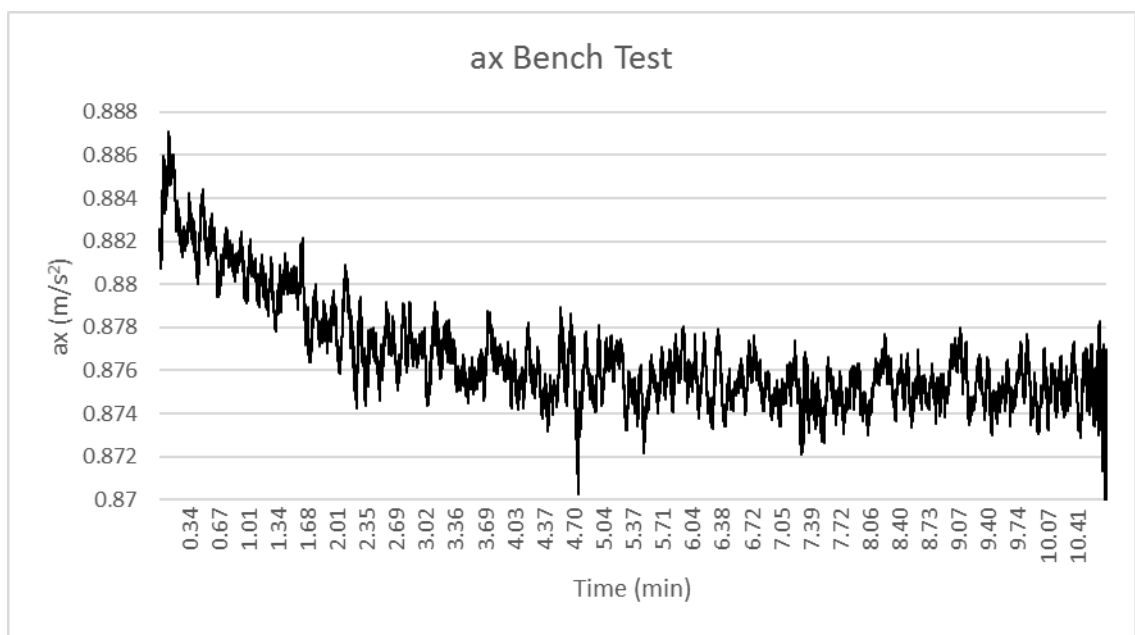


Figure 43: X-axis Accelerometer Noise Test

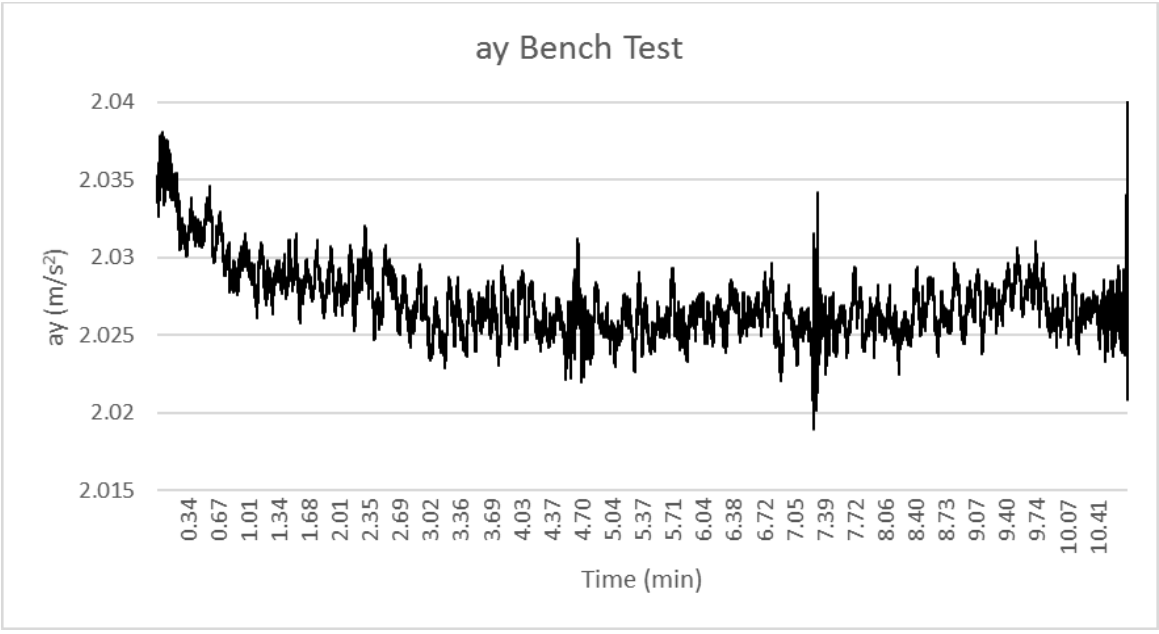


Figure 44: Y-Axis Accelerometer Bench Test

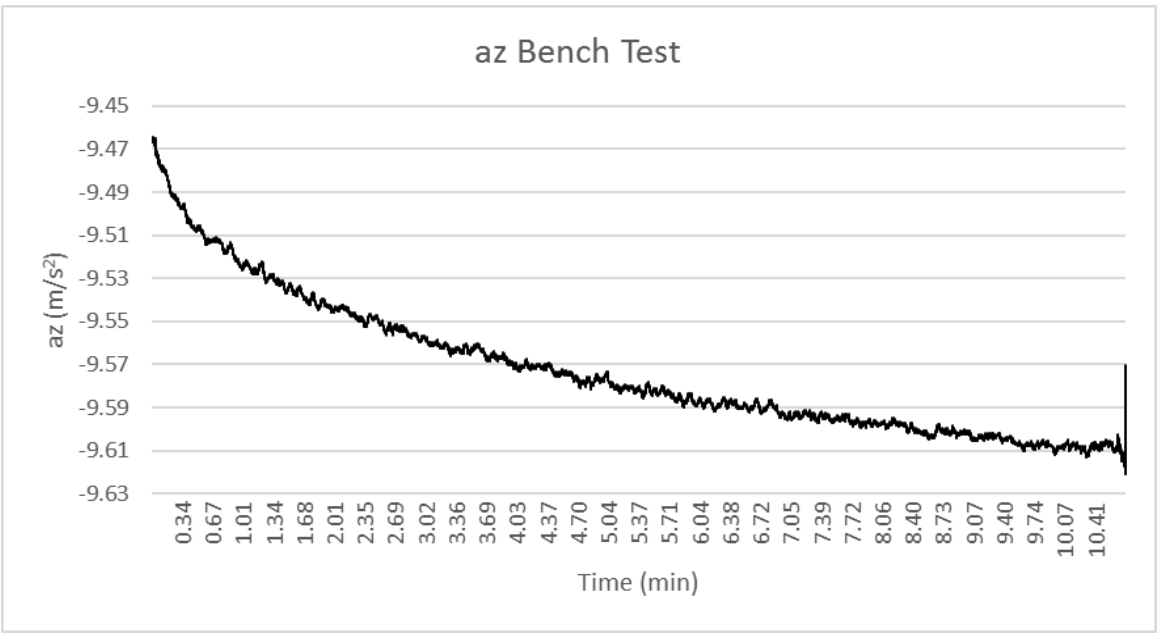


Figure 45: Z-Axis Accelerometer Bench Test

Analysis of standard deviation of the noise profiles is useful. The standard deviation across these samples is ± 0.0122 , 0.0168 , and 0.01945 m/s^2 or 1.2436 , 1.7125 , and 1.9847 mg , for a_x , a_y , and a_z respectively. The overall sensitivity of the accelerometer itself is quoted at 0.732 mg/LSB meaning after signal averaging, the noise in the accelerometer falls within 3 LSB's [28]. Somewhat more concerning is the clear logarithmic trend present on all the accelerometer data. The magnitude of this change is small, except for the a_z reading, but is clearly present. Near the end of the ten minutes, the slope begins to level out and reach a stable value. A secondary test was performed to see if these logarithmic trends were consistent between tests.

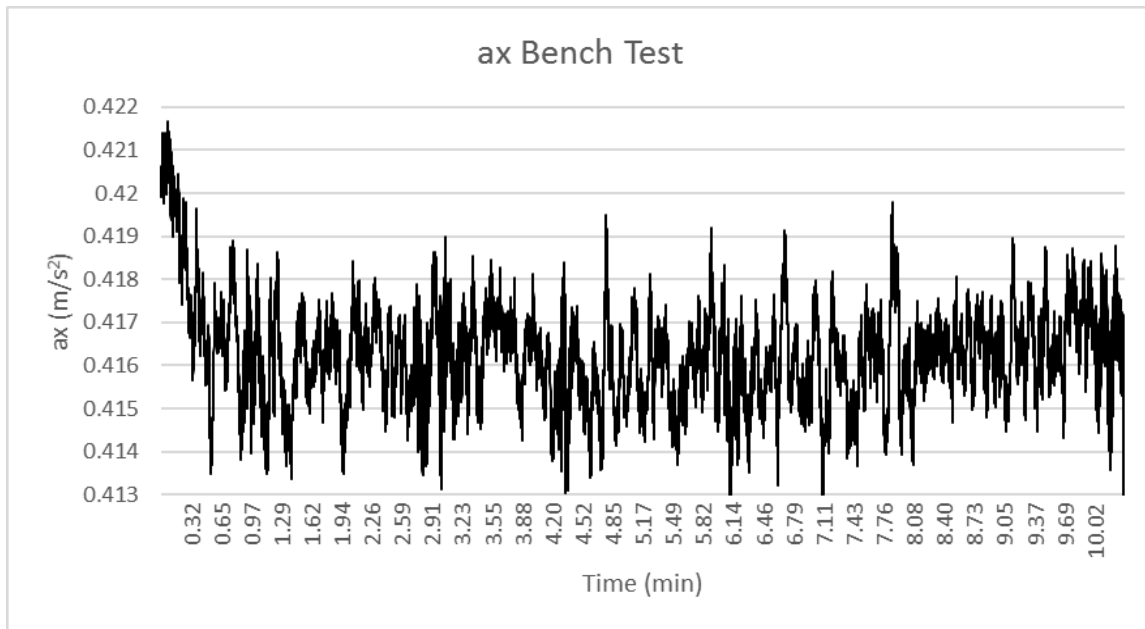


Figure 46: Secondary Accelerometer Bench Tests

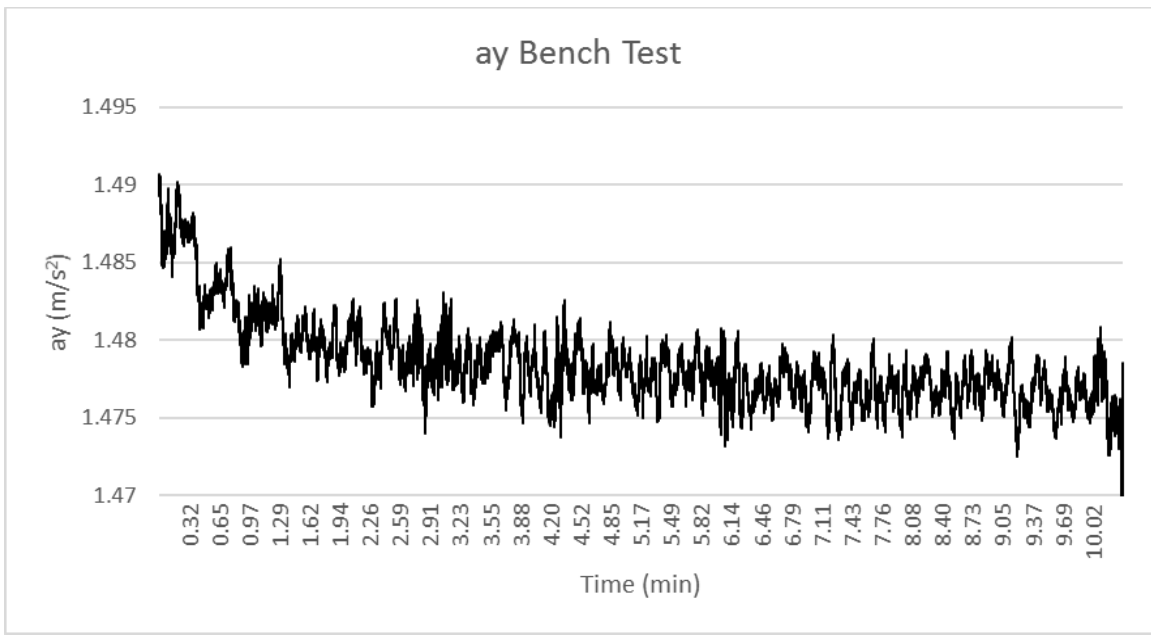


Figure 47: Secondary Y-Axis Accelerometer Bench Tests

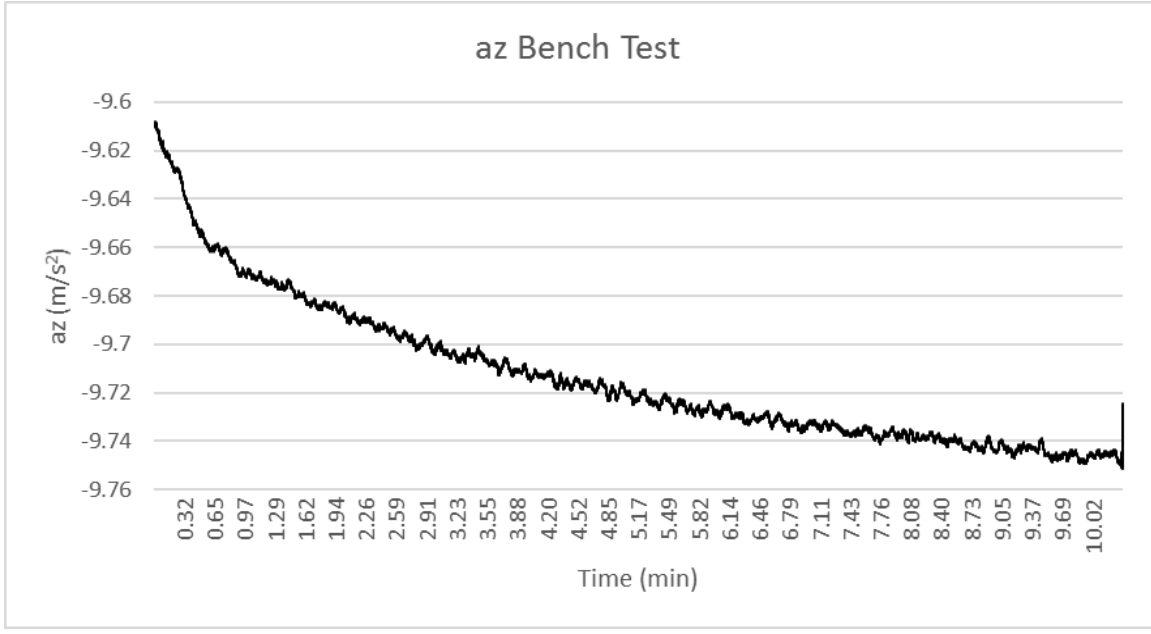


Figure 48: Secondary Z-Axis Accelerometer Bench Test

The most probable cause for this pattern is waste heat generated during the Pixhawk's operation. A figure of the Pixhawk's internal temperature as recorded by its barometric sensor is shown below for both tests.

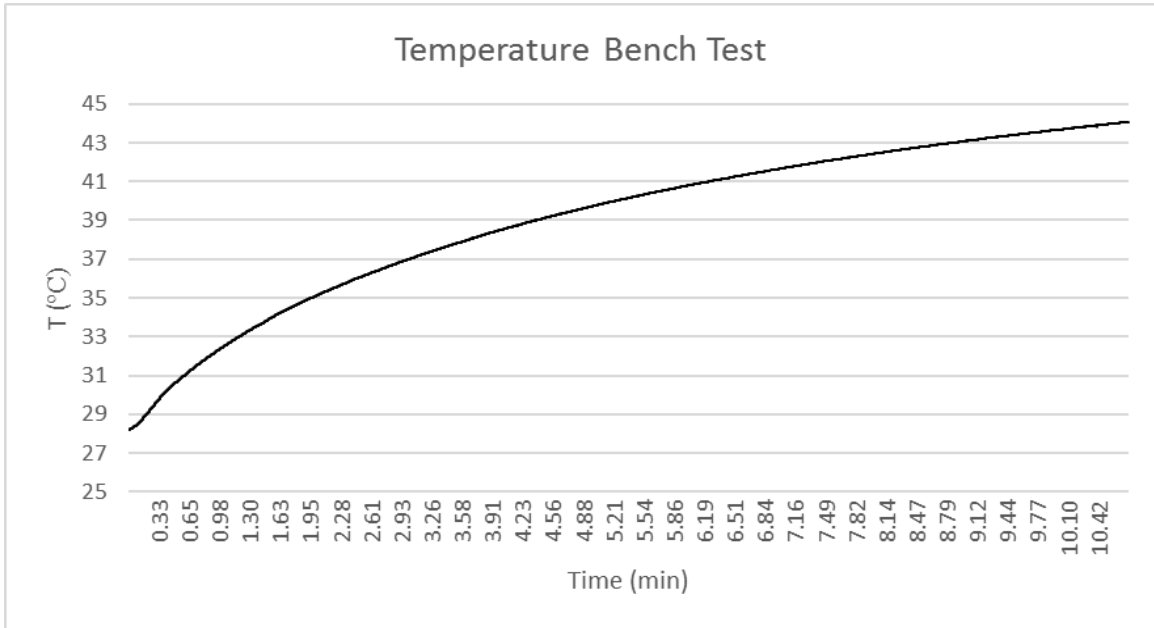


Figure 49: Board Temperature Bench Test I

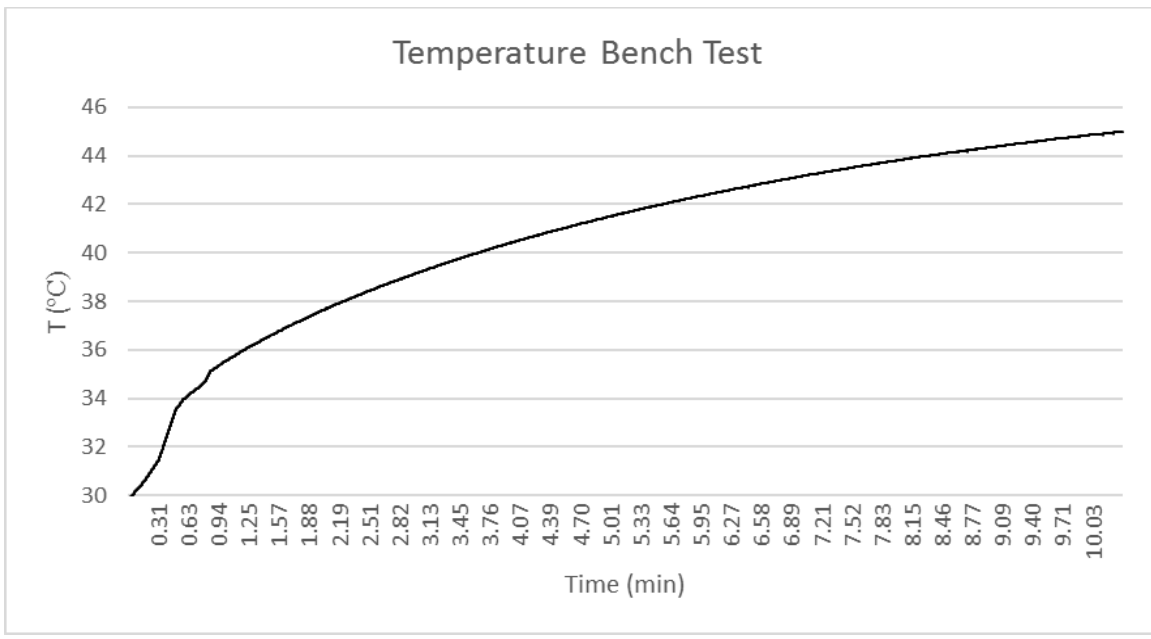


Figure 50: Board Temperature Bench Test II

The temperature curve correlates well with the accelerometer profiles. Because both temperature and the acceleration profiles were collected simultaneously, it can be calculated that the board heating affects accelerometer readings of 0.0008 m/s^2 per $^{\circ}\text{C}$. Figure 44, below, shows the correlation between the board temperature and the accelerometer data.

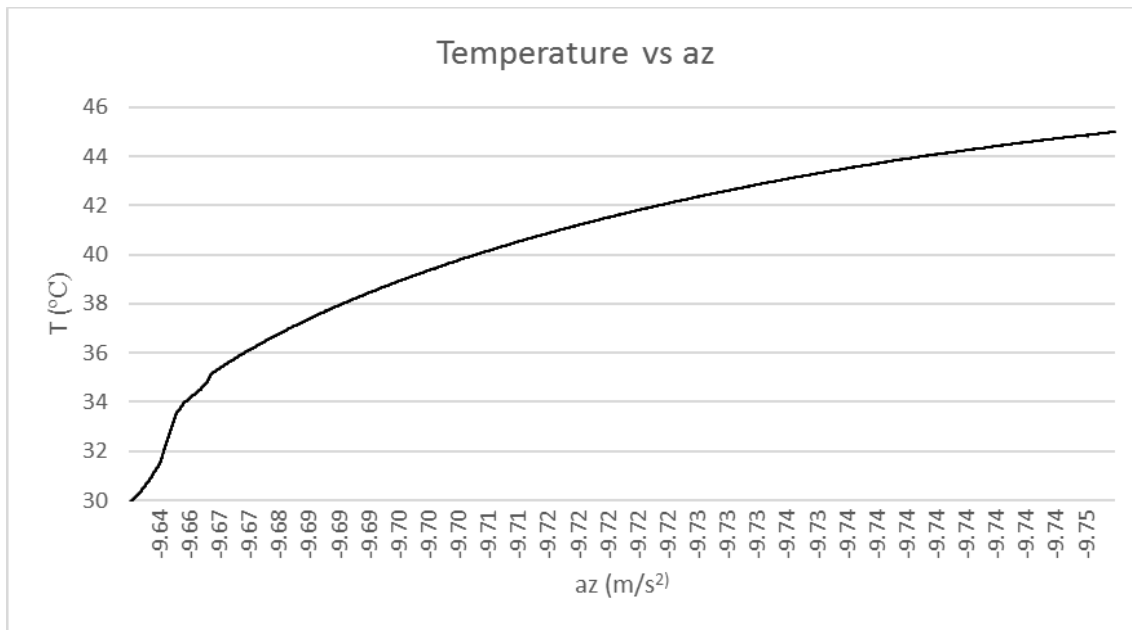


Figure 51: Board Temperature Vs Acceleration

Board heating does not explain the relatively sharp drops seen in the z-axis accelerometer profile. Calculations based on the testing locations latitude yield a local gravity value of 9.79893 m/s^2 . Even ignoring the overall drift, the accelerometer identifies a steady state value of 9.6 m/s^2 . The board heating effects may exacerbate the problems seen, but it seems that even after calibration, the accelerometer can only account for inaccuracies on the order of 0.05 m/s^2 .

5.1.2 Gyroscope Noise Analysis

The gyroscope also operates as a 3-axis sensor and its bench test is conducted alongside the accelerometer. Again, these bench tests fall over a ten-minute period, whose outputs were averaged to boost its signal to noise ratio. The resulting plots are shown in the figures below.

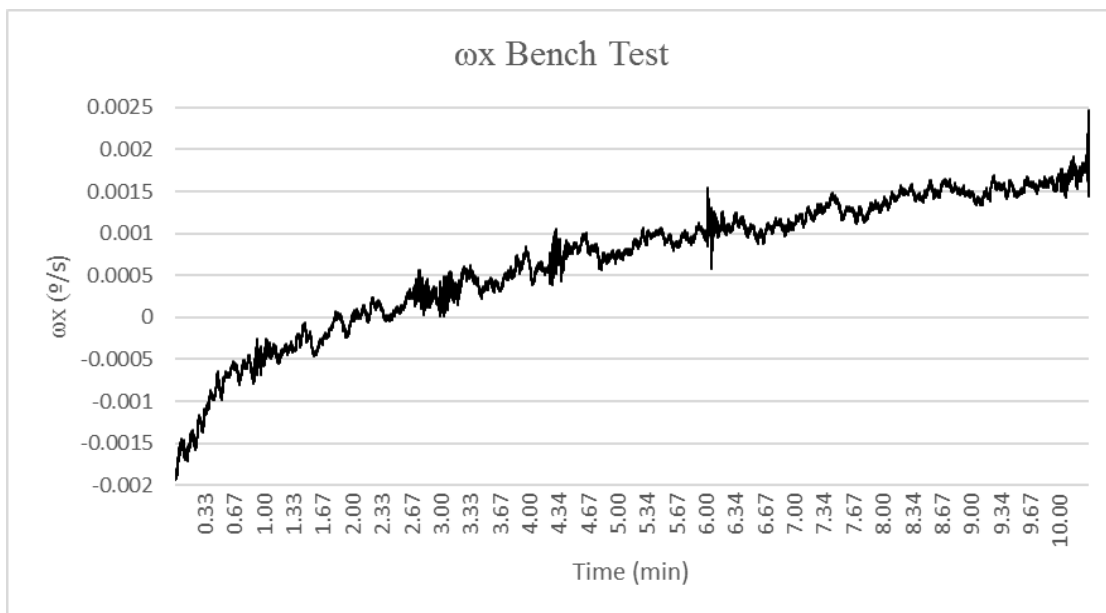


Figure 52: Gyroscope Noise Test I

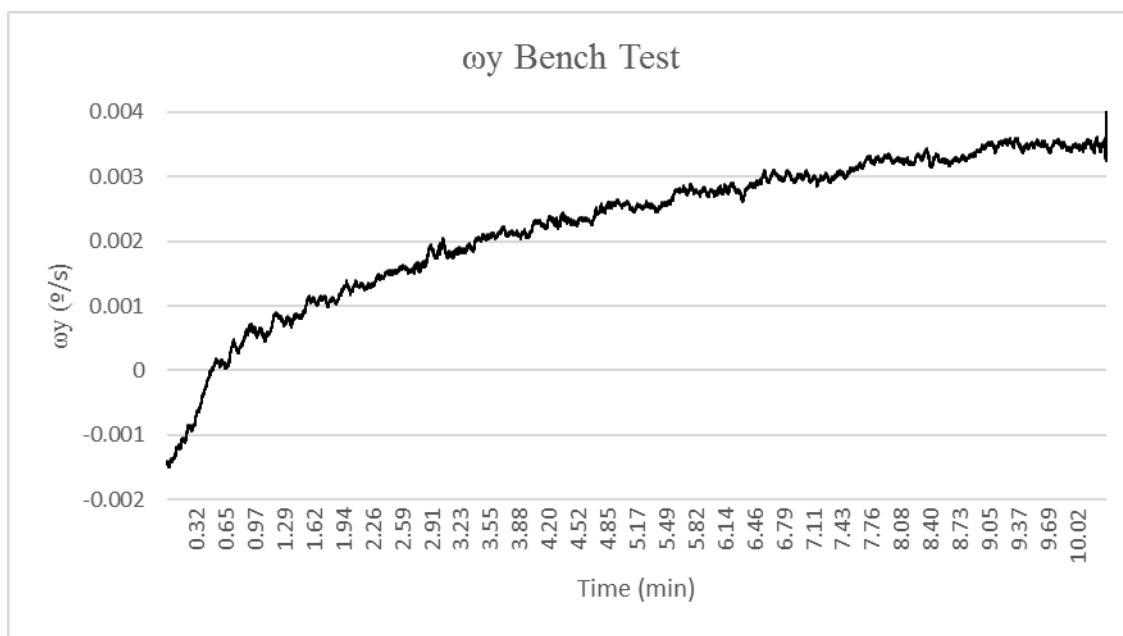


Figure 53: Gyroscope Noise Test II

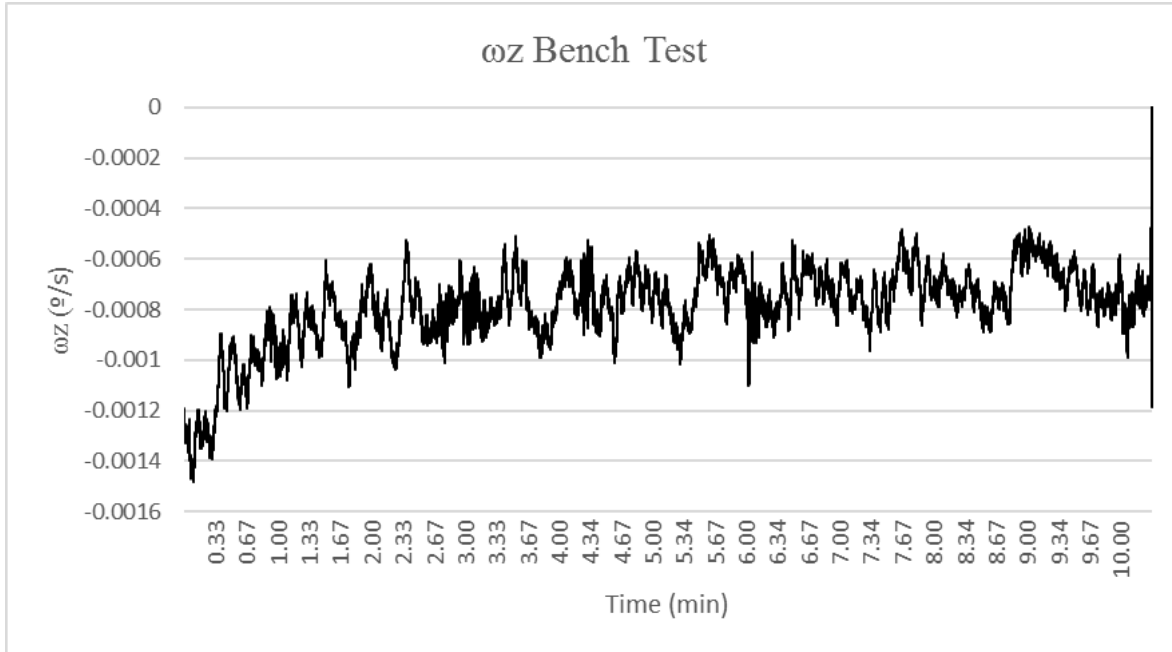


Figure 54: Gyroscope Noise Test III

Standard deviations from these samples fell to 0.0009, 0.0065, and 0.0009 deg/s for the x, y, and z axis respectively across a period of 10 minutes. This means the total effect of noise on the gyroscope is less than a hundredth of a degree per second after averaging. Specifications quote the gyroscope with a sensitivity of 8.75 mdps/digit or 0.00875 dps/LSB[29]. This means after averaging, the signal is falls far below the LSB of the device, meaning the device is operating within an acceptable range. Like the accelerometer data, there is a distinct logarithmic curve present in the data. Even so, the overall drift is less than a hundredth of a degree over a ten-minute period, meaning board heating effects are less substantial.

5.1.3 Analog to Digital Converter (ADC) Noise Analysis

Noise from the ADC channel is noise from the rotary potentiometers used in the vanes. As stated in testing methodology these voltage data from these potentiometers is passed through the Pixhawk's 3.3V ADC channel and logged using the Pixhawk's sdlog function. A planar laser

level section was used to ensure the vanes were orthogonal and parallel to the aircraft. This is not needed for characterizing the signal noise, but instead provides the reference voltage that represents zero degrees for the conversion. This bench test was also conducted alongside both the accelerometer and gyroscope noise tests.

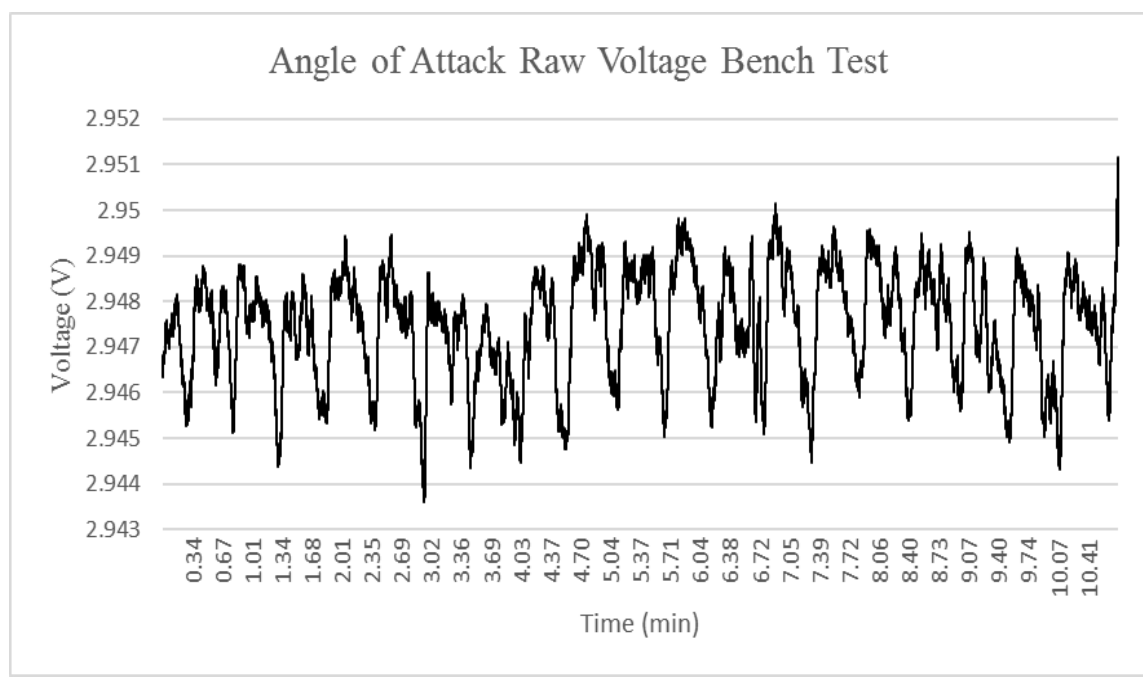


Figure 55: ADC Noise Angle of Attack

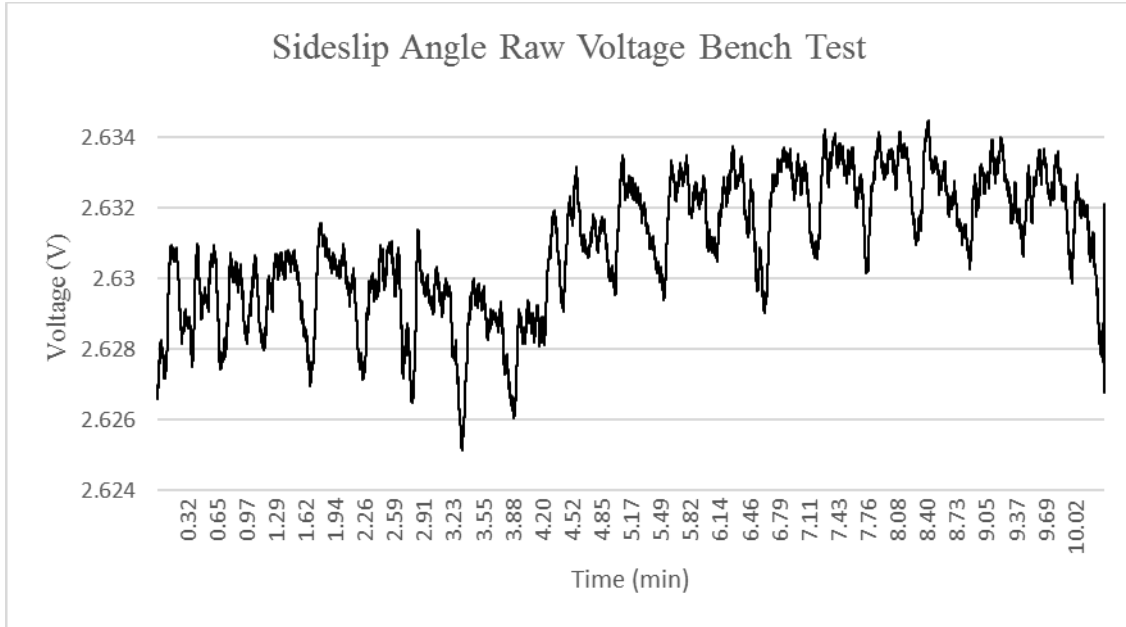


Figure 56: Sideslip Angle vs Time

The rotary potentiometer has 360 degrees of rotation with a voltage range from 0 to 3.3V. This means each degree of rotation equates to 0.0092V. Results from the vane's calibration show this conversion is closer to 0.0082V per degree. Looking at the data, it appears voltage noise falls within ± 0.003 volts, meaning the noise in degrees is approximately ± 0.366 degrees. The conversion formulas to change these raw voltages to degrees are provided in the equation below.

$$\alpha = 121.95(V - 2.948) \quad (\text{Eq. 51})$$

$$\beta = 121.95(V - 2.63)$$

5.1.4 Overview of Capabilities

Component	Random Noise	Bias Error	Worst Case Total Error
Accelerometer	$\pm 0.01 \text{ m/s}^2$	0.05 m/s^2	0.1 m/s^2
Gyroscope	$\pm 0.001 \text{ deg/s}$	0.005 deg/s	0.01 deg/s
ADC	$\pm 0.002 \text{ V}$		0.02 V
Alpha Vane	$\pm 0.25^\circ$		
Beta Vane	$\pm 0.25^\circ$		
Airspeed Sensor	± 0.01	1 m/s	1.3 m/s

Table 7: Sensor Performance Overview

The table above provides a summary of all tolerances derived from the Data Analysis chapter. Each sensor has a maximum sensitivity related to its digital resolution, and the values listed post signal averaging. Bias errors are only provided for sensors whose performance could be compared against specifications provided by the manufacturer. The flow direction of the vanes, and the ADC channel that collects their information, do not have a listed worst case error because the dynamics of the vanes make it difficult to quantify for a general case.

The comparison between the actual performance and the specifications from each sensor's datasheets is explained in more detail in the prior sections of the bench test results. To compensate for board heating, the standard deviations were taken after the first fifteen minutes of each test. The accelerometer itself demonstrated the worst expected performance. The standard deviation across these samples is ± 0.0122 , 0.0168 , and 0.01945 m/s^2 or 1.2436 , 1.7125 , and 1.9847 mg , for

ax, ay, and az respectively. Compared against the 0.732 mg/LSB benchmark in the datasheet, the overall noise falls within 2-3 LSB's even after heavy signal averaging [28].

Results from the gyroscope demonstrated a much cleaner signal, falling into the expected ranges according to its datasheet. Standard deviations from these samples fell to 0.0009, 0.0065, and 0.0009 deg/s for the x, y, and z axis respectively, meaning the total effect of noise on the gyroscope is less than a hundredth of a degree per second after averaging. Specifications quote the gyroscope with a sensitivity of 8.75 mdps/digit or 0.00875 dps/LSB [29]. After data averaging, the gyroscope falls well within the specified limits of the system.

The results show the Pixhawk while performing somewhat under par, is able to record all the critical parameters for analyzing the flight performance of an unmanned aircraft. Parameters such as airspeed could be improved with the replacement of a more sensitive pressure transducer, but the current model has a strong performance, and can serve as a reliable platform for research applications.

5.2 WIND TUNNEL TEST RESULTS

5.2.1 Vane Tracking

Three tests were performed in ODU's low-speed wind tunnel at separate airspeeds. The first tests the vane at wind speeds far exceeding the anticipated value of 13.4 meters per second or 30 mph. Increased windspeed corresponds to more accurate tracking and this test shows the vane functioning in ideal conditions. The second test is the vane operating at the expected airspeed of 13.4 meters per second. This test demonstrates errors that should be expected during wind tunnel testing. The final test is at wind speeds below anticipated values, which shows the vane at a weakened performance and provides information on pattern changes. A sine wave with identical frequency is overlaid with each excitation period to serve as the ideal path of the vane.

Each case demonstrated a trend of increasing amplitude corresponding with increased frequency. This is likely due to the sudden change in direction at the peaks and troughs of the oscillation. At higher frequencies, the probe experiences a strong deceleration at the extremes that does not translate to the free floating vanes, which are carried slightly further by their own inertia. This could also be compounded by a slight bending of the carbon rod at these extremes. Lower frequency sweeps show little to no deflection meaning the bending itself must not be significant compared to the vane's inertia.

Tracking the 60 mph and 30 mph cases demonstrates near perfect correspondance to the control at $\frac{1}{4}$ and $\frac{1}{2}$ Hz frequencies. However, the 17 mph case displays consistant patterns of degradation, illustrated where the lines are more jagged than the expected smooth contours and the waves' peaks and troughs are flattened. This is likely because the lower dynamic pressure causes the vanes to be sluggish. Figures of each case in their entirity are provided below, as well as the first $\frac{1}{4}$ Hz wave of each case.

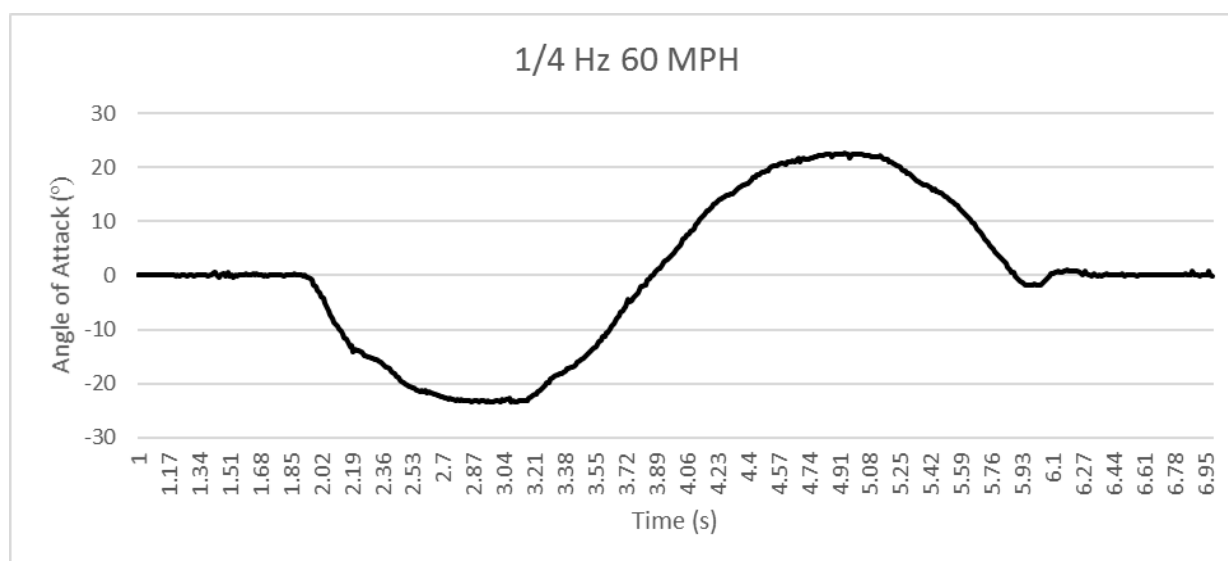


Figure 57: 1/4 Hz Wave 27 M/S (60mph)

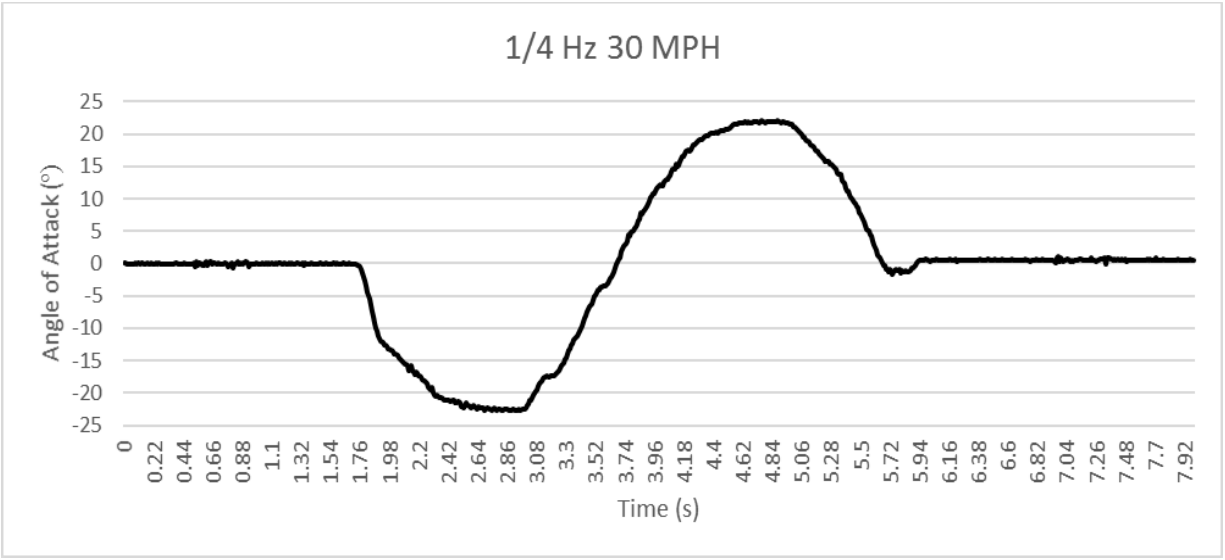


Figure 58: 1/4 Hz wave 13.4 m/s (30mph)

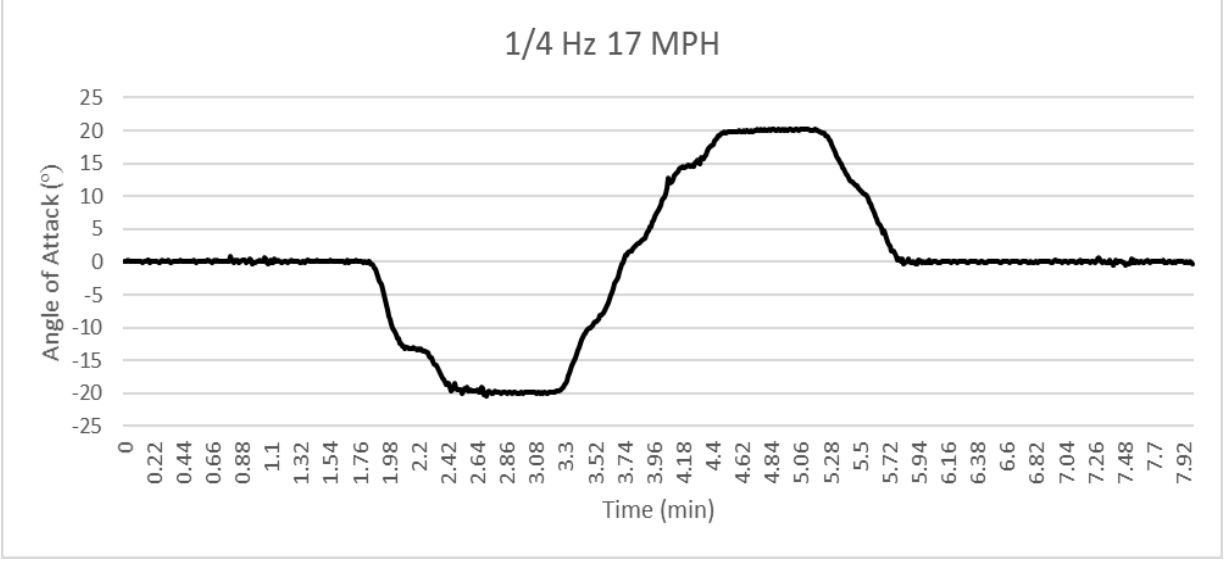


Figure 59: 1/4 Hz Wave 8 M/S (17mph)

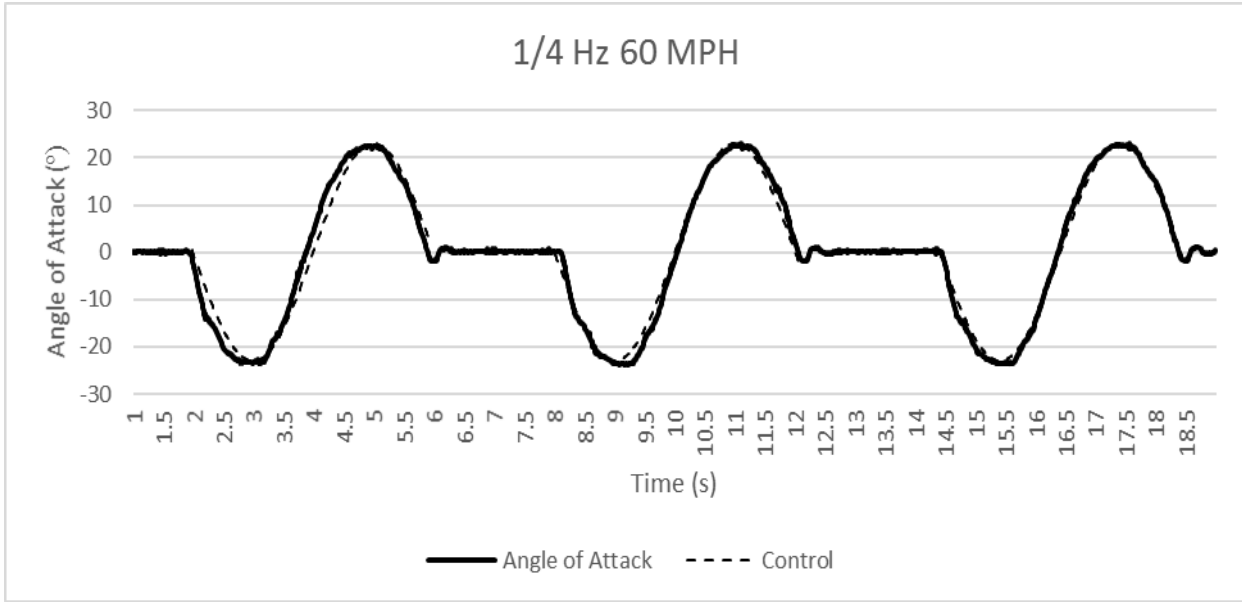


Figure 60: Vane Tracking 1/2 Hz 27 M/S (60 MPH)

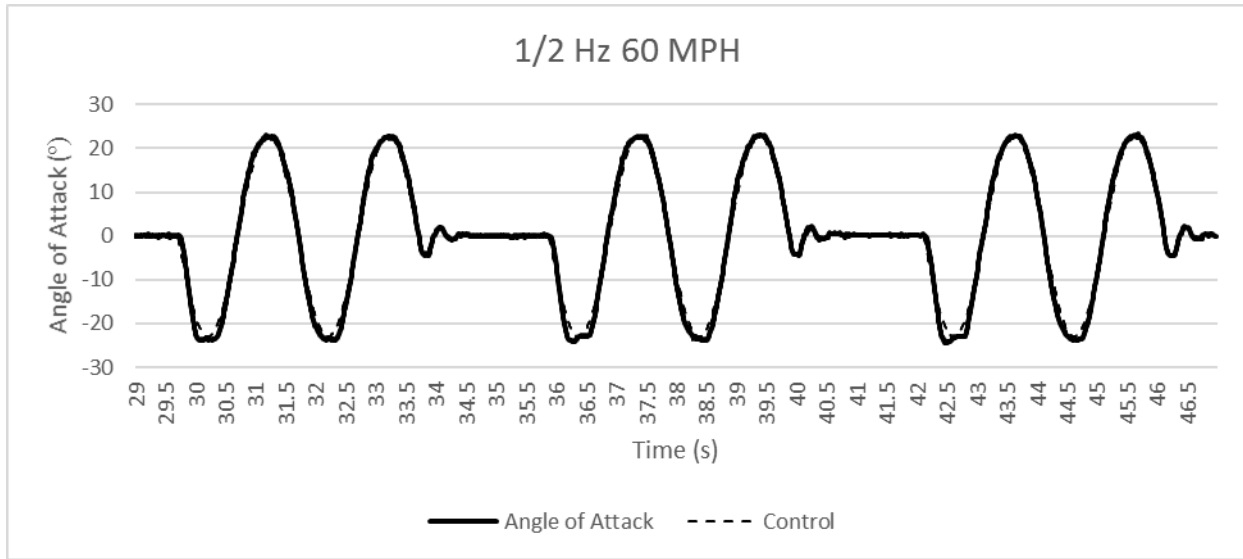


Figure 61: Vane Tracking 1/2 Hz 27 M/S (60 MPH)

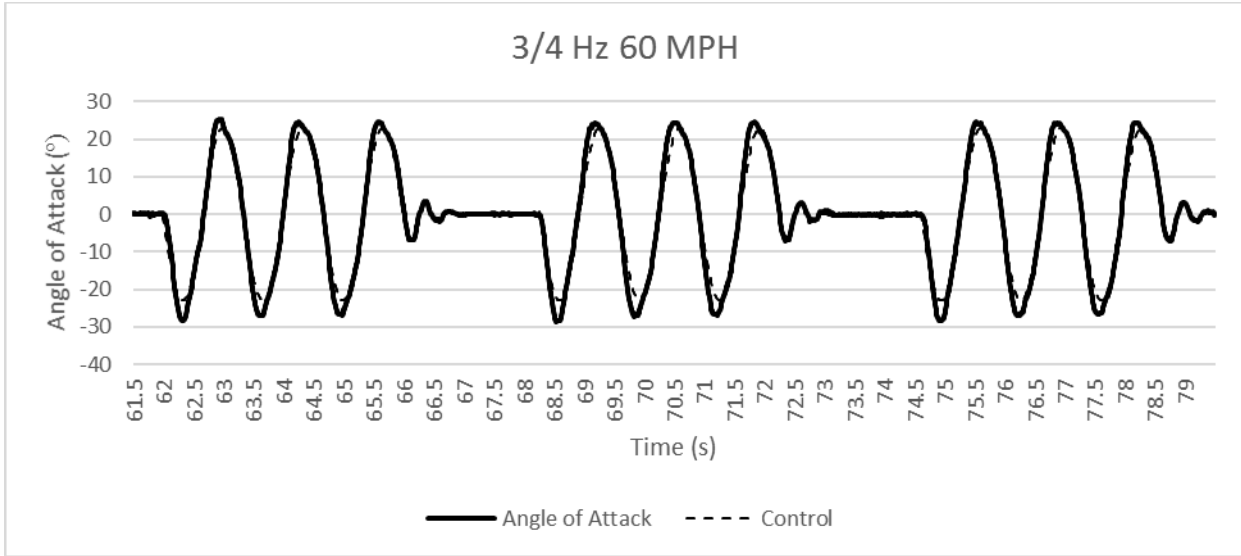


Figure 62: Vane Tracking 3/4 Hz 27 M/S (60 Mph)

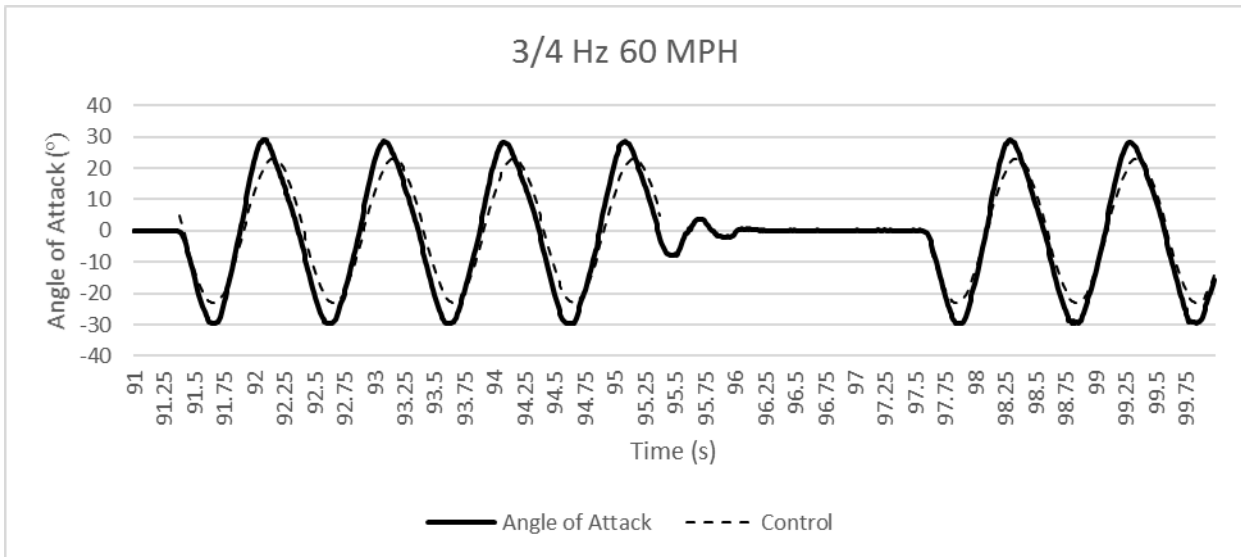


Figure 63: Vane Tracking 1 Hz 27 M/S (60 MPH)

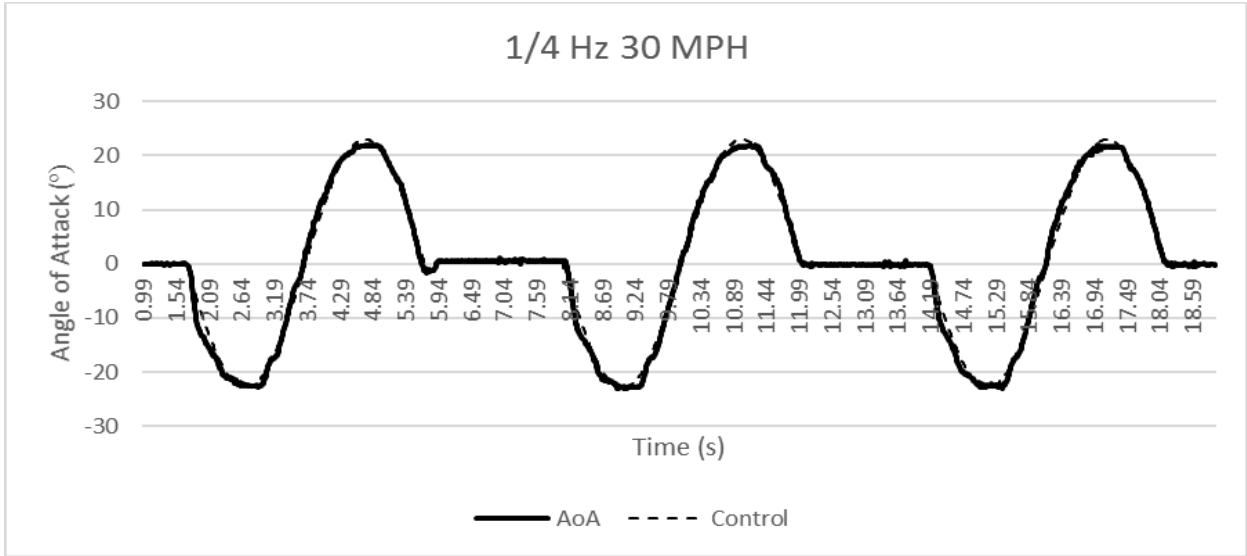


Figure 64: Vane Tracking 1/4 Hz 13.4 M/S (30 Mph)

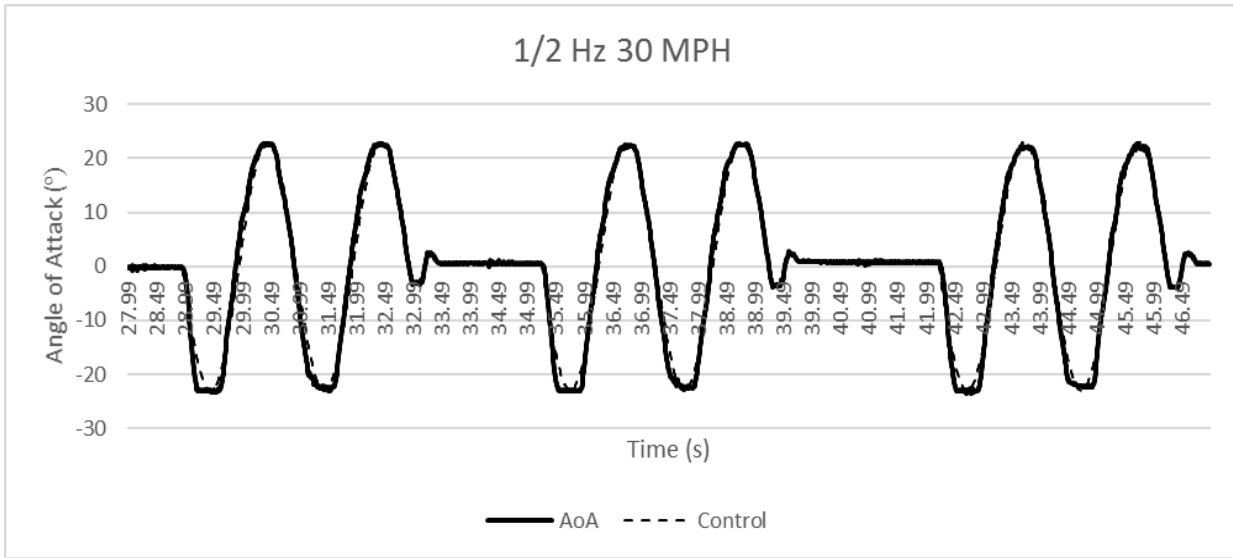


Figure 65: Vane Tracking 1/2 Hz 13.4 M/S (30 MPH)

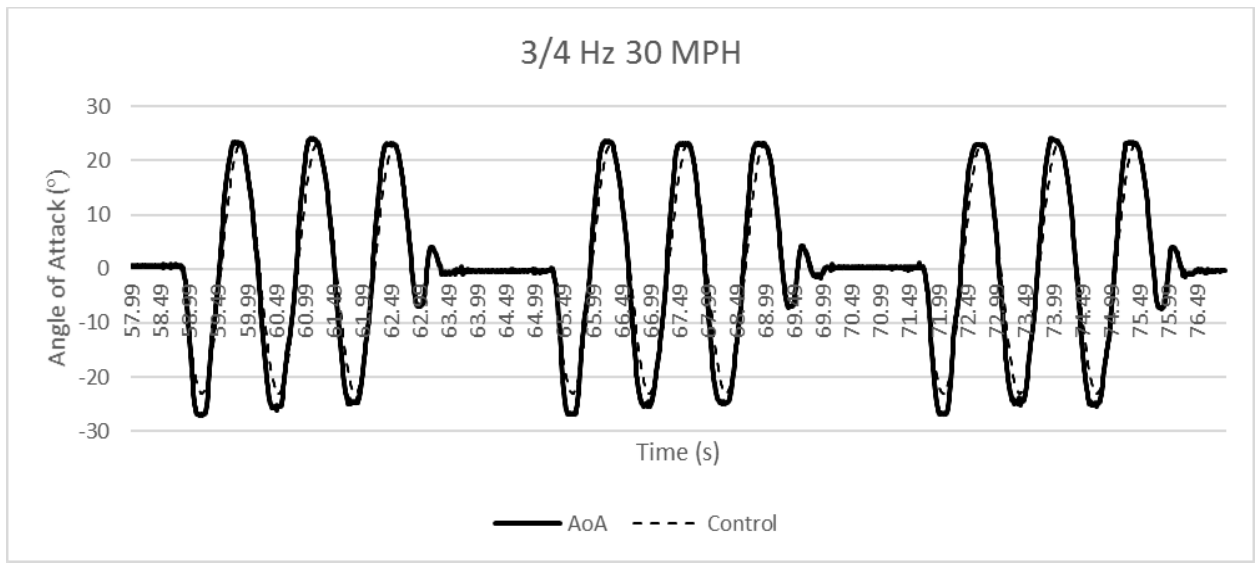


Figure 66: Vane Tracking 3/4 Hz 13.4 M/S (30 MPH)

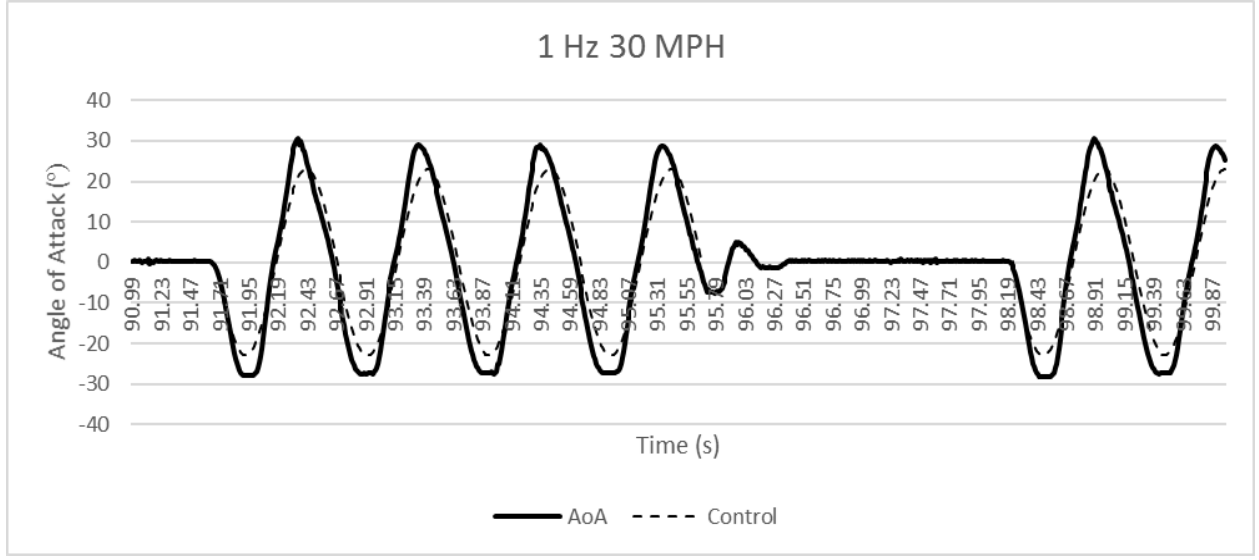


Figure 67: Vane Tracking 1 Hz 13.4 M/s (30 MPH)

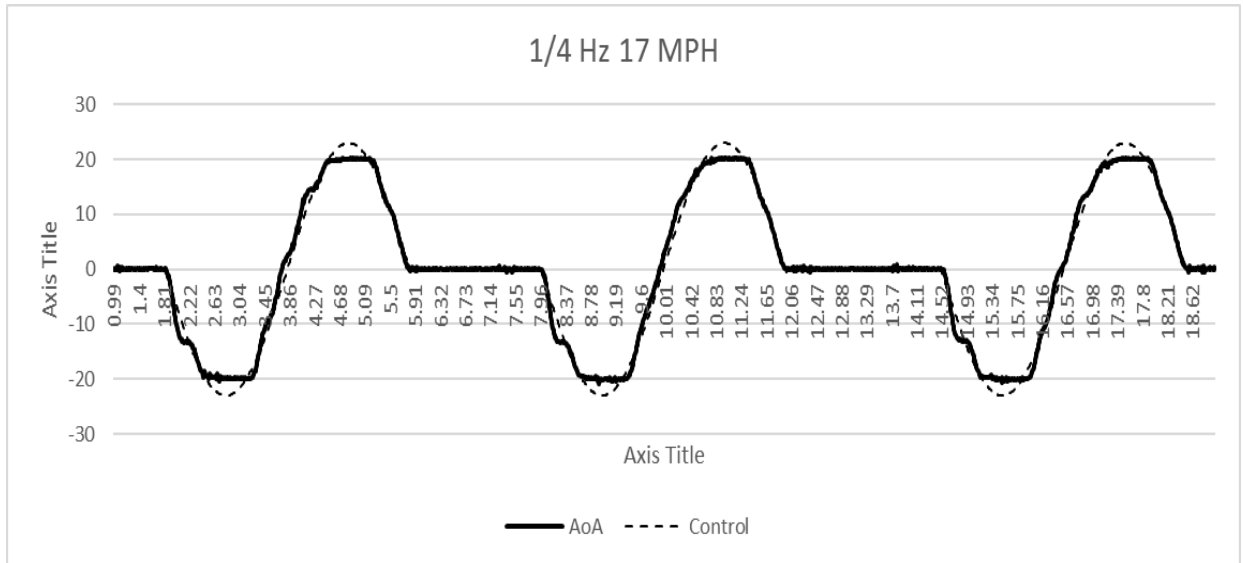


Figure 68: Vane Tracking 1/4 Hz 8 m/s (17 mph)

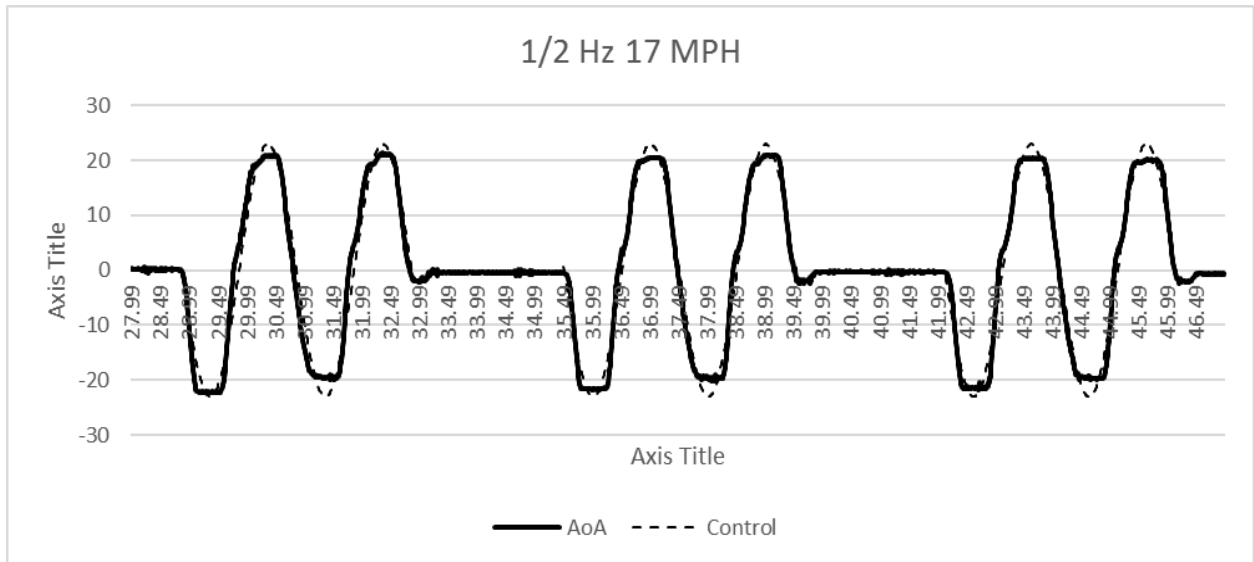


Figure 69: Vane Tracking 1/2 Hz 8 M/S (17 MPH)

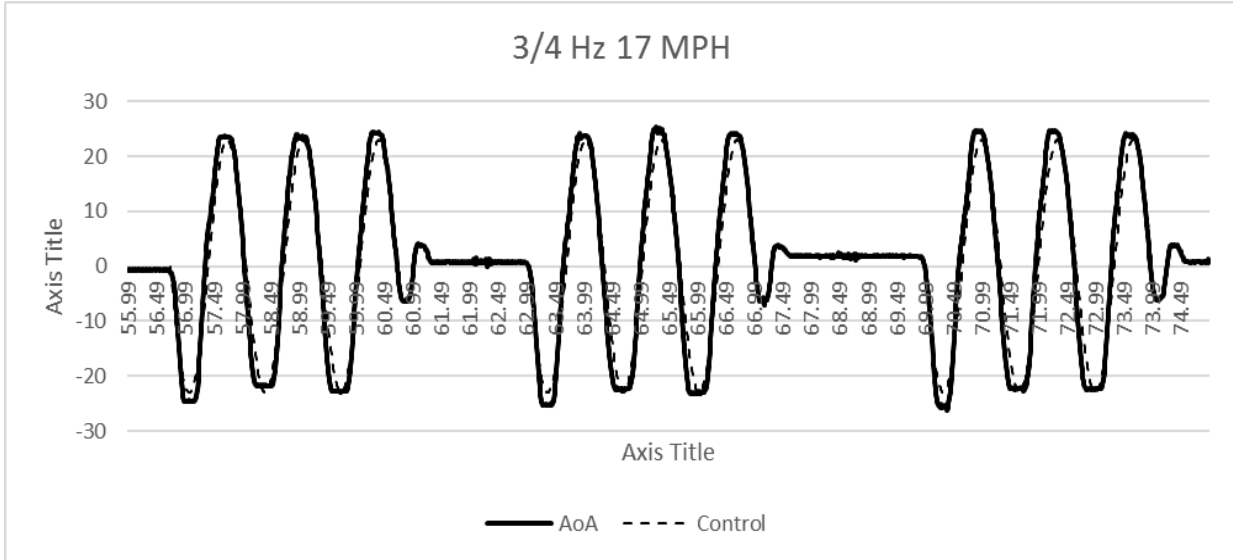


Figure 70: Vane Tracking 3/4 Hz 8 M/S (17 MPH)

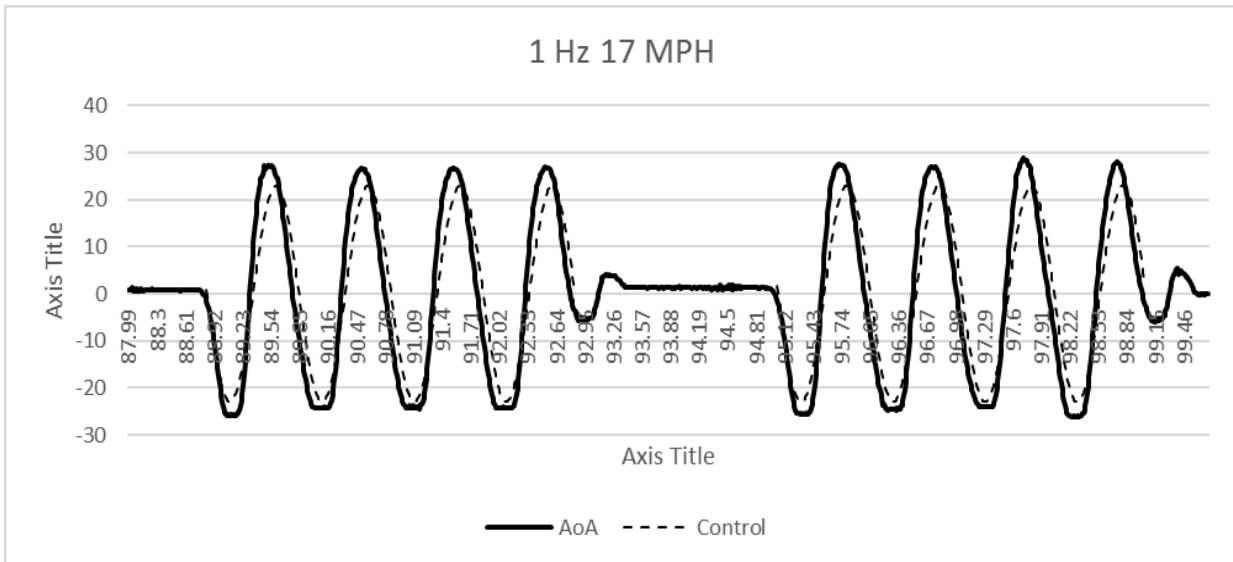


Figure 71: Vane Tracking 1 Hz 8 M/S (17 MPH)

5.2.2 Fin Size Optimization

The analysis of variance or ANOVA table provides information needed to assure a well fitted model. The table below is the ANOVA table for the fin size optimization FCD. Before coming to this final result, a candidate regression model must be selected. By default, the model is assumed to have a linear relationship, and must be transformed to best fit the design points. The second order model FCD is capable of estimating quadratic curvature. The best fit model equation is listed below and uses the letter ID's as given in the following table.

$$\ln(y_i) = 6.5166 * b - 0.47927 * c - 0.1072 * r_w + 0.6385 * c^2 \quad (\text{Eq. 52})$$

Response	1	$\left \sum y_{probe} - y_{ideal} \right $				
Transform:	Natural Log	Constant:	0			
ANOVA for Response Surface Reduced Quadratic model						
Analysis of variance table [Partial sum of squares - Type III]						
Source	Sum of Squares	df	Mean Square	F Value	p-value Prob > F	
Model	5.8431	4	1.4608	10.3515	0.0003	significant
A-Span	2.2970	1	2.2970	16.2772	0.0011	
B-Chord	0.1135	1	0.1135	0.8041	0.3840	
C-Moment Arm	1.5481	1	1.5481	10.9699	0.0047	
B^2	2.0262	1	2.0262	14.3584	0.0018	
Residual	2.1168	15	0.1411			
Lack of Fit	1.5097	10	0.1510	1.2434	0.42833097 4	not significant
Pure Error	0.6071	5	0.1214			
Cor Total	7.9600	19				

Table 8: Fin Size Optimal ANOVA Table

The diagnostic residual plots are grouped and shown in Figure 72:

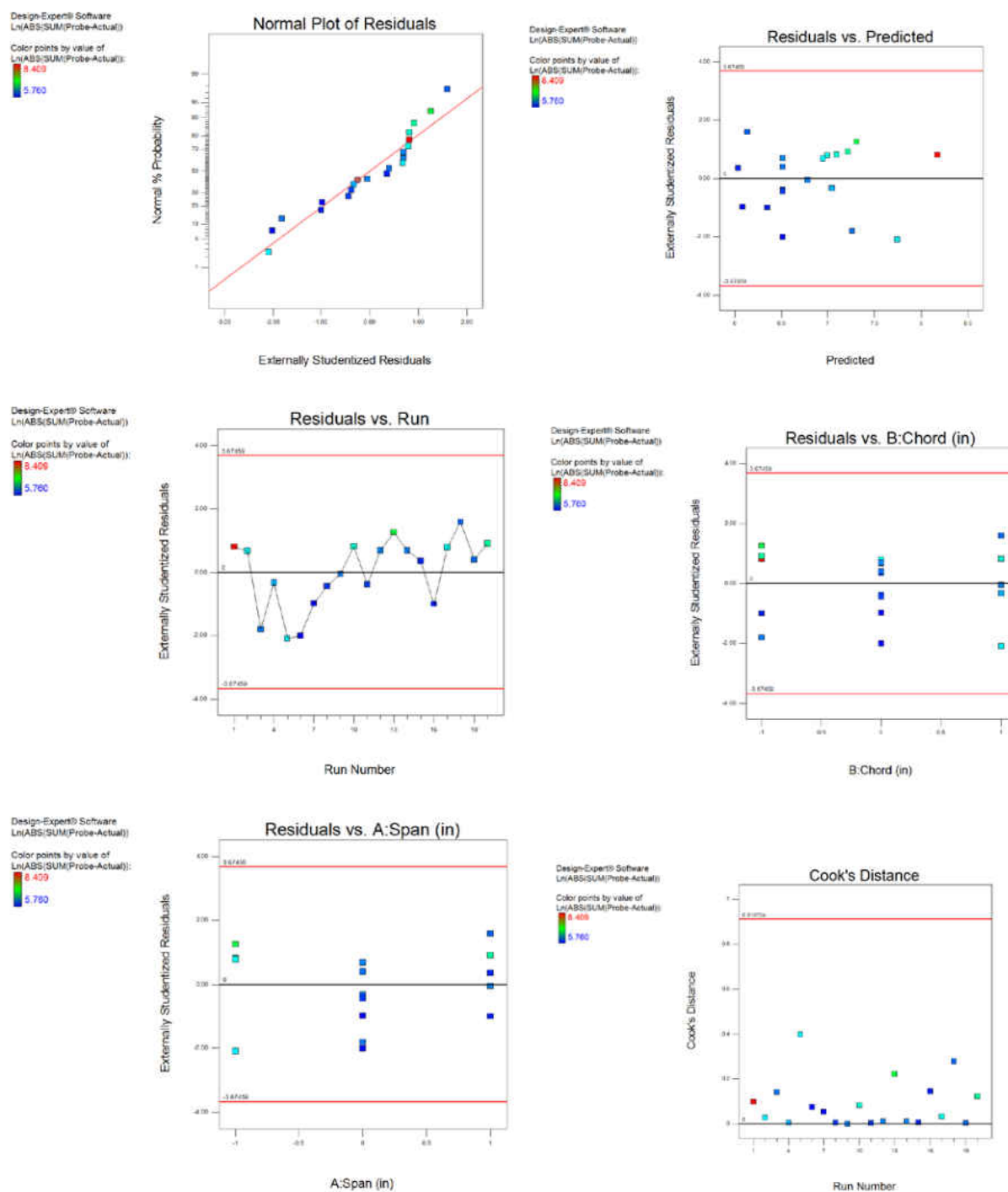


Figure 72 : Fin Size Optimization Diagnostic Plots

Normal plot of residuals checks for the normality of the data. These residual points follow along the probability line, which means the data is normally and independently distributed. Points in the Residuals vs. Predicted plots are randomly scattered, meaning the variance of each observation is considered constant. The Residuals vs. Run plot determines if an environmental variable skewed the results over time. The points are near equally distributed across the line with no drift, meaning the results of each test were independent of nuisance factors. Residuals vs. Chord and Residuals vs. Span show model adequacy and constant variance for each factor respectively, and show a desired scattered pattern with no noticeable trends. Cook's Distance is a good check for huge outliers in the batch of runs. All runs fall below a value of one, meaning the externally studentized residuals are clear of problems.

Figure 73 represents the surface model of the response against the span and chord.

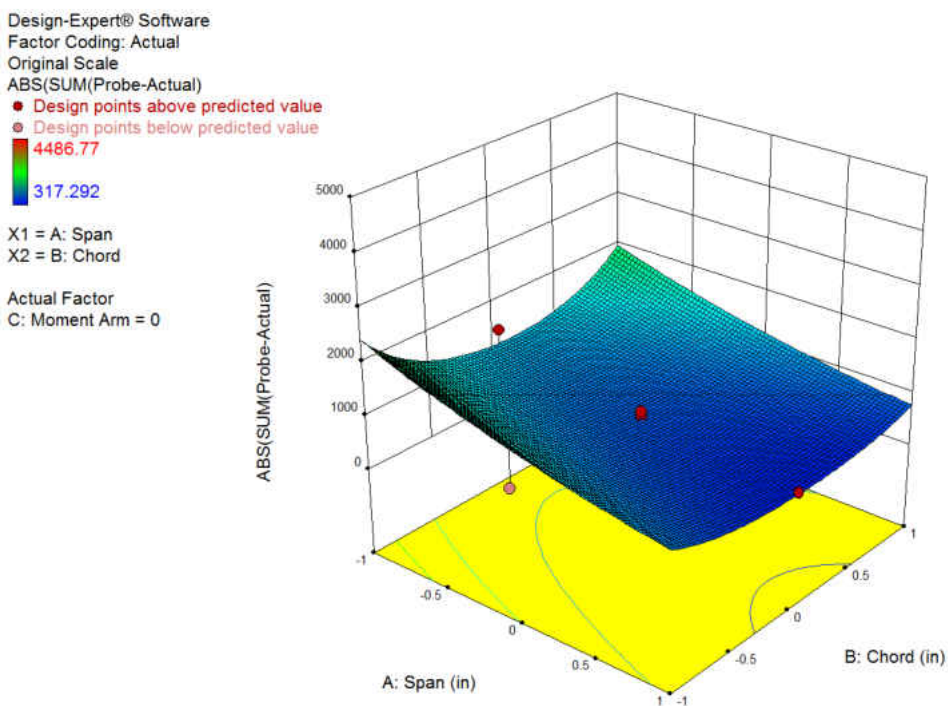


Figure 73: Fin Optimal Surface Plot

For optimizing this experiment, the response needs to be as close to zero as can be managed. The chord result is expected as its minimal arc falls around the calculated value, and yet the model states increased performance with an increase of span. This is not entirely unexpected, as the longer span results in an increased lifting force in the fins, resulting in faster response time. The experiment itself can only speak for a span up to three inches, yet it was possible that a larger span could result in increased fidelity. Further testing was performed with fins up to four inches, but results showed an instability that provided inconsistent results.

Figure 74 then provides an optimal combination of factors for maximum performance, with span maximized, and chord maintained.

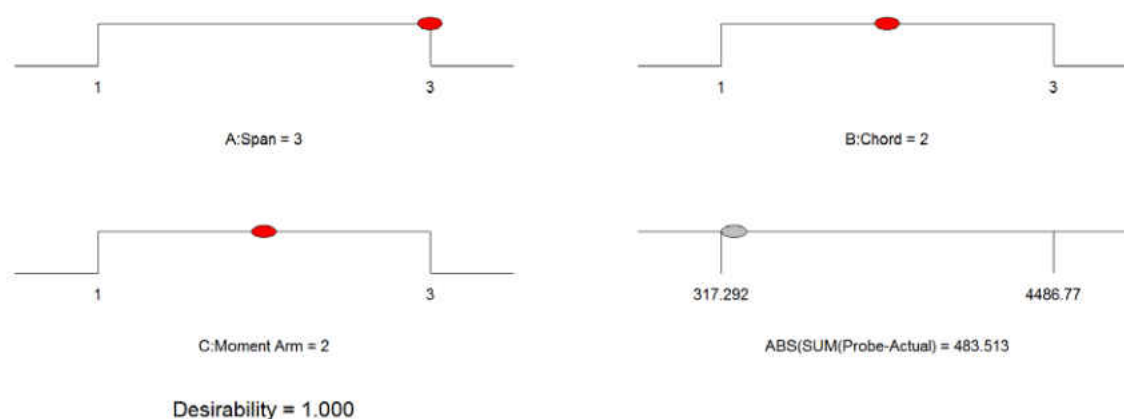


Figure 74: Fin Size Optimization

According to the model, the moment arm of three inches, provided the greatest performance of the vanes tested. Based on these results, an increase in span was decided to better improve the performance, and minimize error in testing.

5.2.3 Vane Damping

As stated in the testing methodology section, the damping ratio of the rig was tested to see if it matched industry standards as defined by Wieringa [21]. The vane was made to rest against a support beam whose angle is predetermined. When the wind tunnel reaches the target airspeed of 30 mph (13.4 m/s), a pulse is sent to a push pull solenoid, that releases the beam and releases the vane. Figures below show the moment of release, and the resulting damped motion of the vane from initial angles in five degree increments. Due to the nature of the release mechanism, negative values were not tested. Given the symmetrical nature of the vanes, differences in results at negative versus positive angles are assumed negligible.

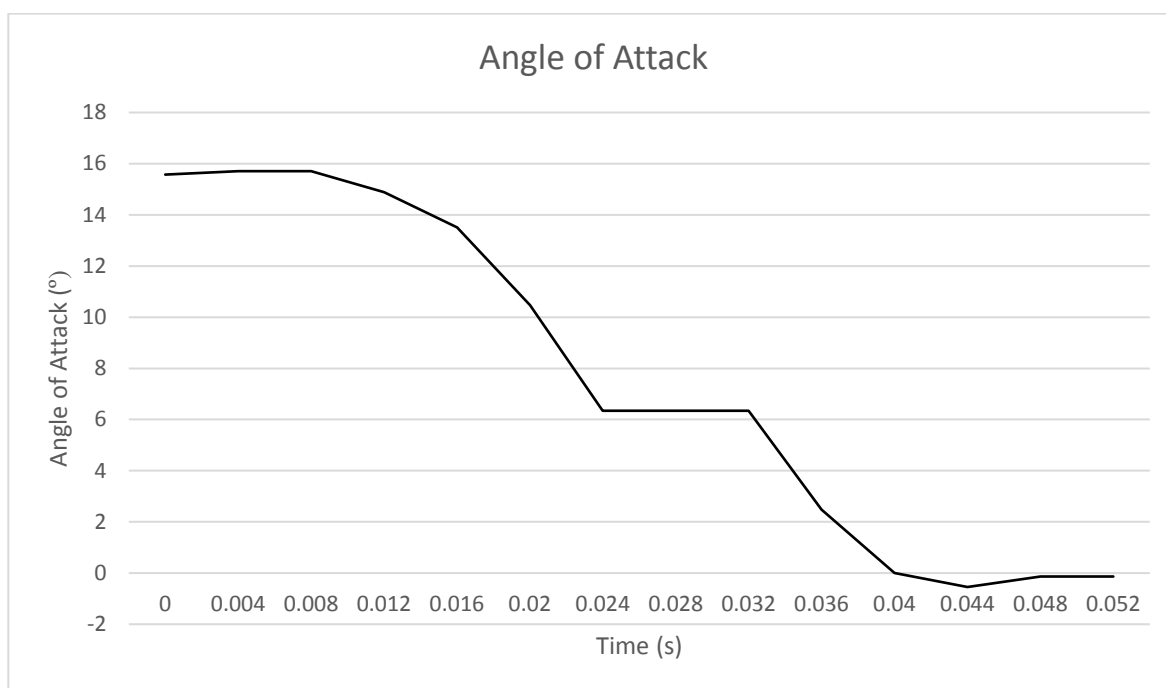


Figure 75: Damping Test 15 Degrees

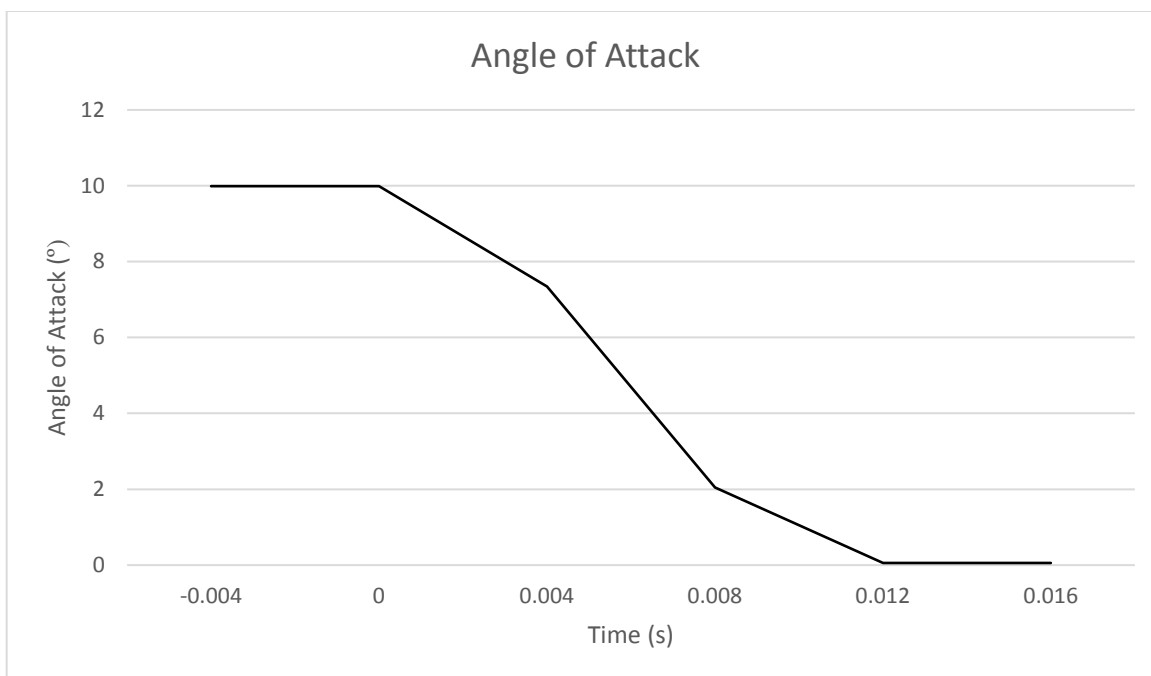


Figure 76: Damping Test 10 Degrees

Because stall effects are anticipated in the 12 to 14-degree range, an angle of 15 degrees is at the higher range of expected angles. The total time needed for full damping at the extreme angle was only 40 milliseconds. The ten-degree mark showed full damping in 16 milliseconds. The results from the 5-degree test were somewhat inconclusive, as the damping period could not be picked up by the Pixhawk's logging function. This proved true through three separate verification tests, each with an indistinguishable damping period. This is not entirely unexpected, as the Pixhawk's boosted sampling rate is still limited to 250 Hz. It is expected therefore, that the damping period falls between sampling points, meaning the overall damping period is less than 4 milliseconds.

The table below shows the anticipated damping ratio for the 5, 10, and 15 degree cases, and compares them against the collected test results. The term “ ζ ” represents the damping ratio and “ t_d ” is the dampened oscillation period of the vane derived by the equation below.

$$t_d = \frac{2\pi}{\left[\left(\frac{N}{J}\right) - \left(\frac{D}{2J}\right)^2\right]^{\frac{1}{2}}} \quad (\text{Eq. 53})$$

Angle	Theoretical Results		Test Results	
	ζ	t_d (s)	ζ	t_d (s)
15°	0.3784	0.0660	0.3857	0.060
10°	0.3174	0.0787	0.3235	0.016
5°	0.2513	0.0994	0.2561	>0.004

Table 9: Vane Damping Results

An odd point of note was the opposing trends of t_d between the results. The theoretical results display an increasing damping period with decreasing angle, which does not correlate with the test results. The source of this issue is likely because the wave was simulated as a harmonic damping wave and does not account for various real life effects such as mechanical friction from the potentiometers. These would serve to increase the overall damping motion.

The damping ratios between both cases are much closer and provide a strong metric for evaluating the performance. A reminder that the desired damping ratio is less than 0.33 to accomplish the standard. The five and ten-degree cases both fall below this guideline, indicating their damping success. The fifteen-degree case falls well above this guideline. The result is not desirable, but it is expected during testing that this is an extreme angle, and falls beyond the stall range of the aircraft.

5.2.4 Airspeed Calibration

Calibration of the airspeed sensor was a critical test for gauging its performance. The Ardupilot manual states that an oscillation between zero and small values (2-3 m/s) is normal when not measuring wind [30]. This pitot tube was mounted in the ODU low speed wind tunnel and data was collected across a series of known airspeeds. Data from the Pixhawk's pressure transducer are compared against the airspeed measurements available in the wind tunnel as seen in Figure 64. During the analysis phase, a ten second period of data at each airspeed was averaged to reduce the random noise.

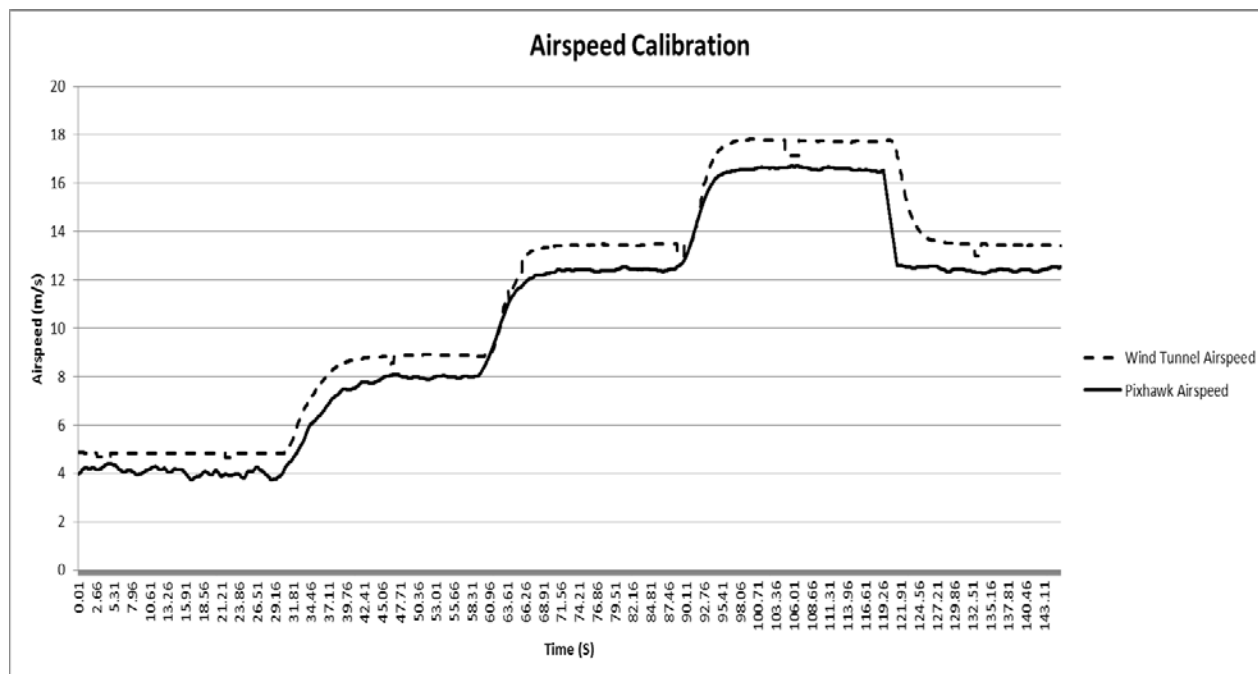


Figure 77: Airspeed Calibration

The overall performance was consistently below the expected benchmarks, yet the gap remains consistent. The airspeed sensor performs within 1m/s below the target across all measured airspeeds. Aside from a slight divergence at the highest airspeed, the overall performance of the sensor is quite accurate overall, especially given the sensors cost. The slight gaps in the rise and fall of these airspeeds is due to a slight difference in sampling rates between the two devices.

5.3 MANEUVERS

5.3.1 General Flight Data

A general overview of the test flights is provided in the data below. The pilot was instructed to reach a safe altitude between 70 and 100 m and fly several long straight sections across the runway. The parameters are set in Qgroundcontrol for a 2-1-1-2 maneuver and transmitted by telemetry to the Pixhawk. After these passes, control is given to the Pixhawk by toggling the multiplexer, and the plane is guided to trim conditions with the assistance of the

stabilize flight mode. When trim conditions are met, the pilot toggles the SYS_ID switch and the autopilot performs the maneuver. This process is repeated until the aircraft completes its pass. From there, the pilot returns to direct receiver control, and moves the plane for another pass, while another maneuver is selected.

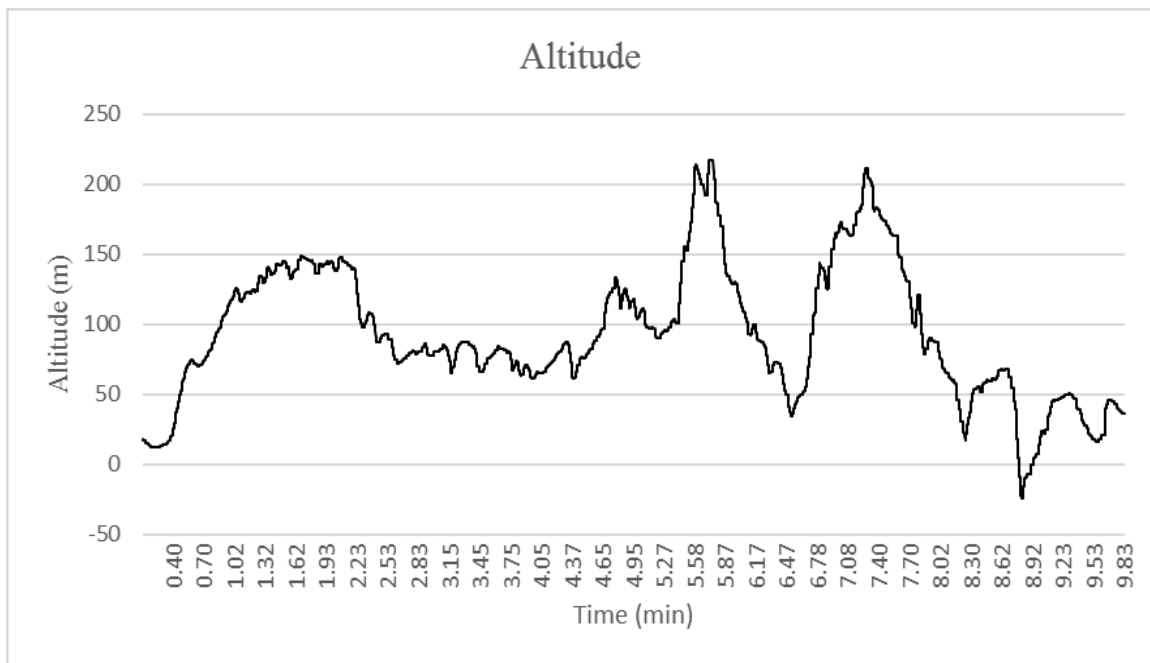


Figure 78: Altitude vs. Time

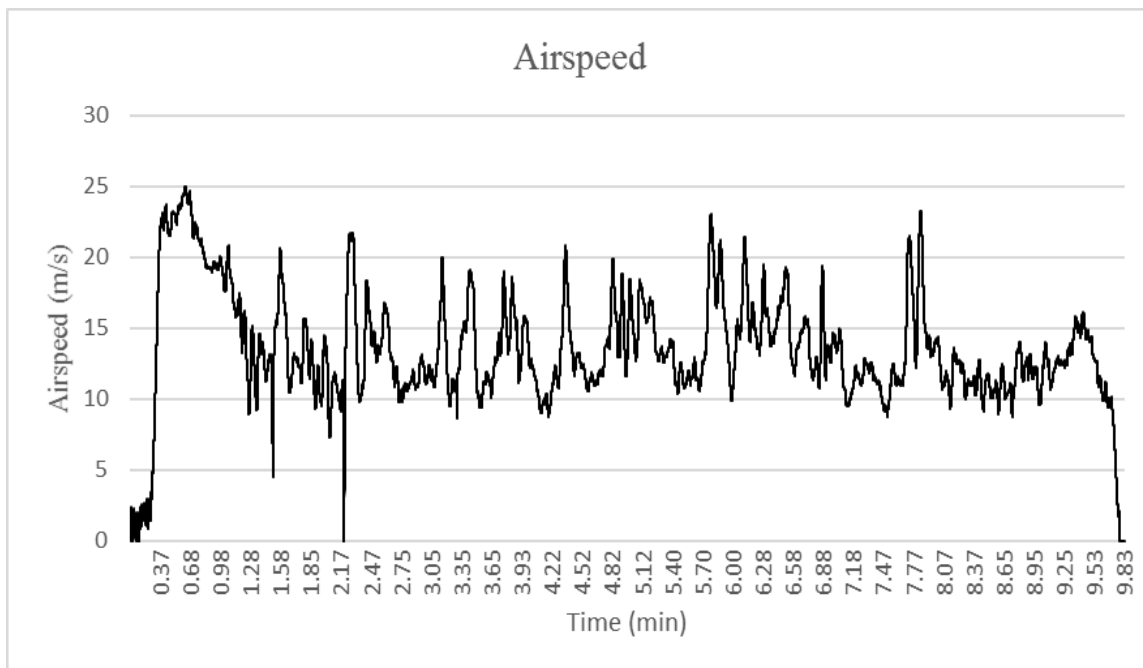


Figure 79: True Airspeed (Tas) vs. Time

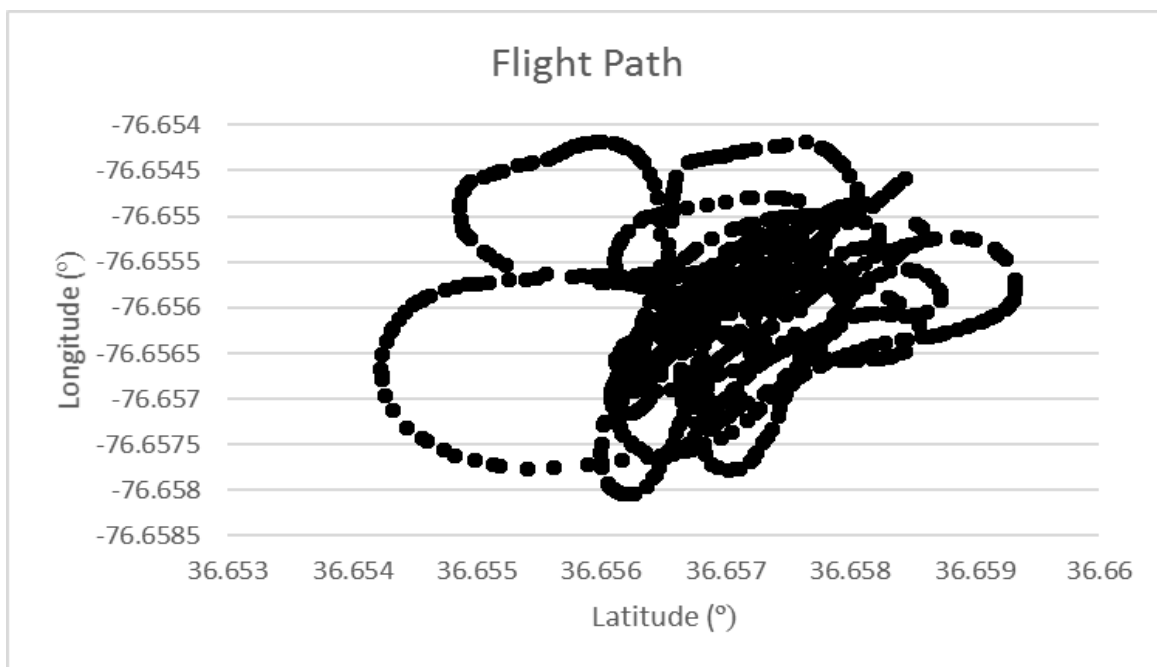


Figure 80: Air Titan Relative Flight Path

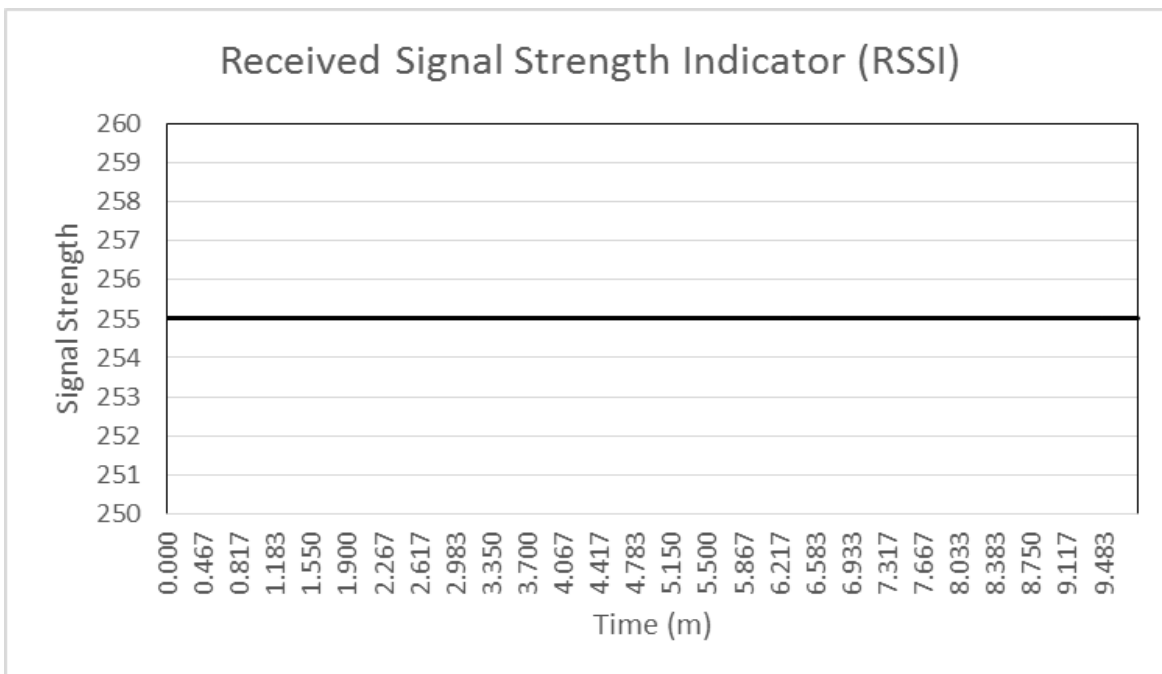


Figure 81: Received Signal Strength Indicator (RSSI)

The above plots serve to give an overall impression of the flight, and show parameters the Pixhawk is capable of tracking that are not elaborated on in the maneuver results.

5.3.2 Multistep Maneuvers

This section will cover information collected from the Pixhawk's multistep maneuvers. The 3-2-1-1 will be absent from these results, as early testing showed an excessive deflection in the aircraft's wings while executing the maneuver. The 2-1-1 makes for a good alternative as it too can generate the frequency sweep patterns associated with the 3-2-1-1. The 2-1-1-2 employed by Favaregh will also be tested here. Each multistep maneuver can be used for roll, pitch, and yaw motions, but due to the goals of this project, only the pitching maneuvers were tested.

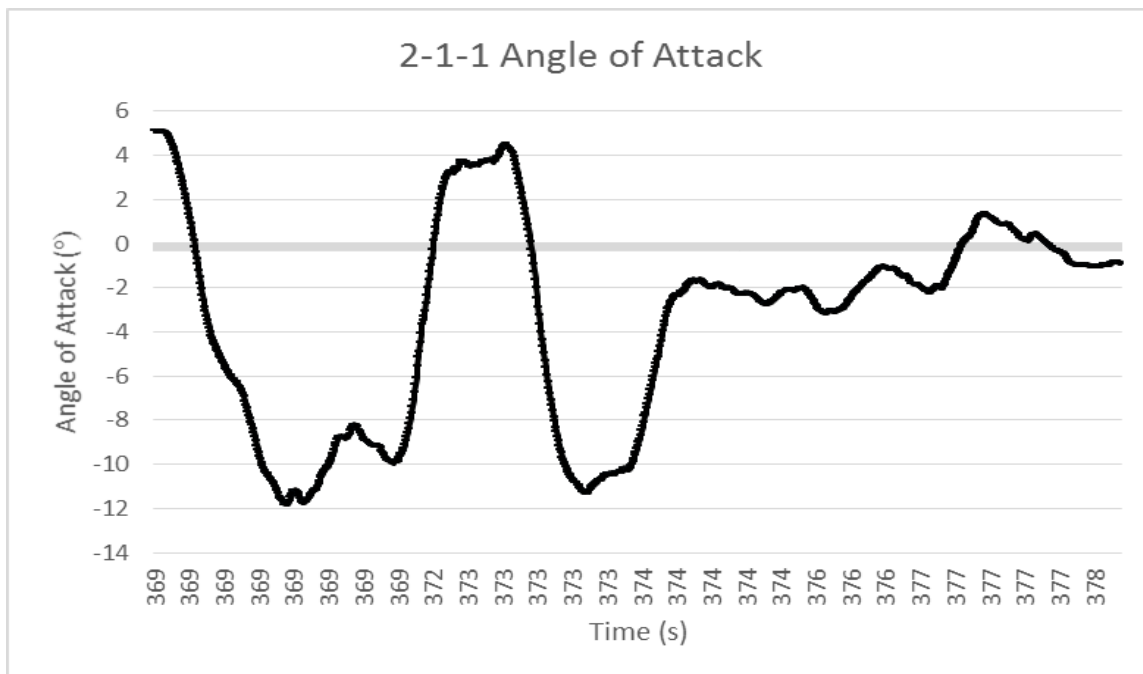


Figure 83: Angle of Attack vs. Time

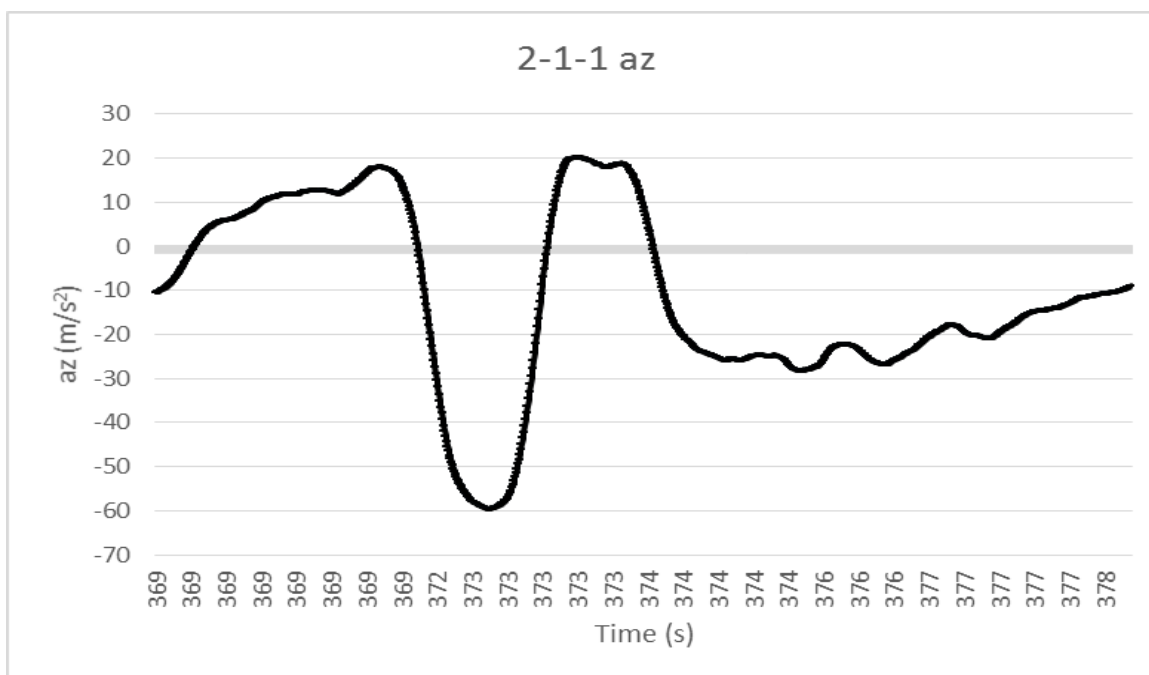


Figure 84: 2-1-1 Az vs. Time

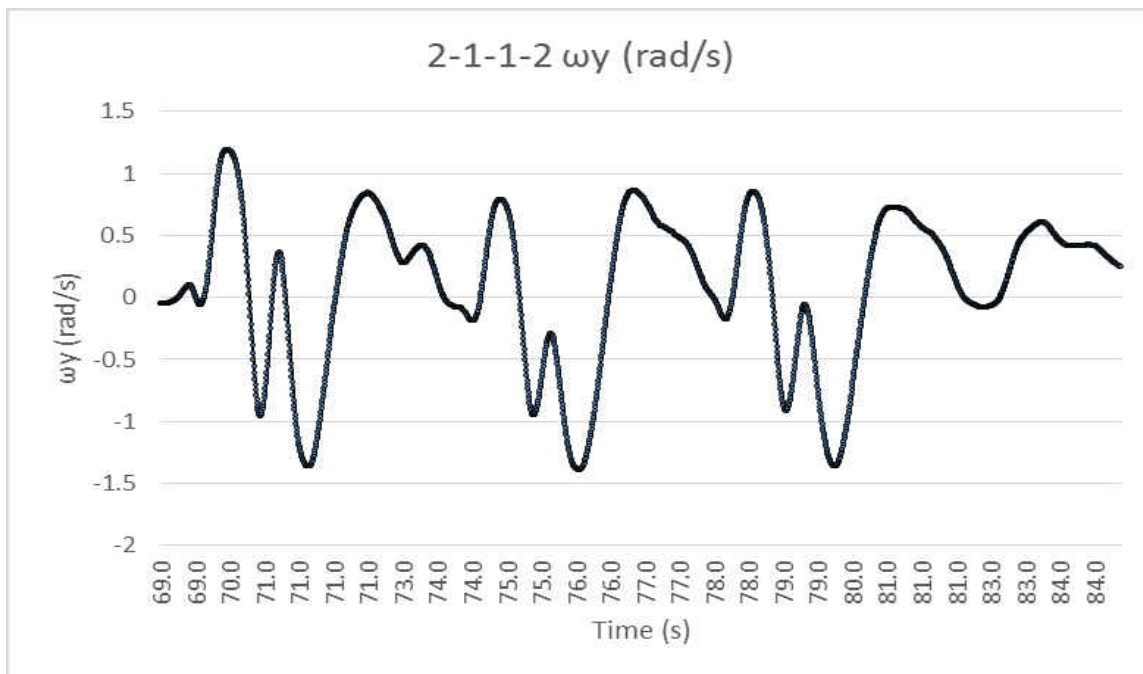


Figure 86: 2-1-1-2 Pitch Velocity

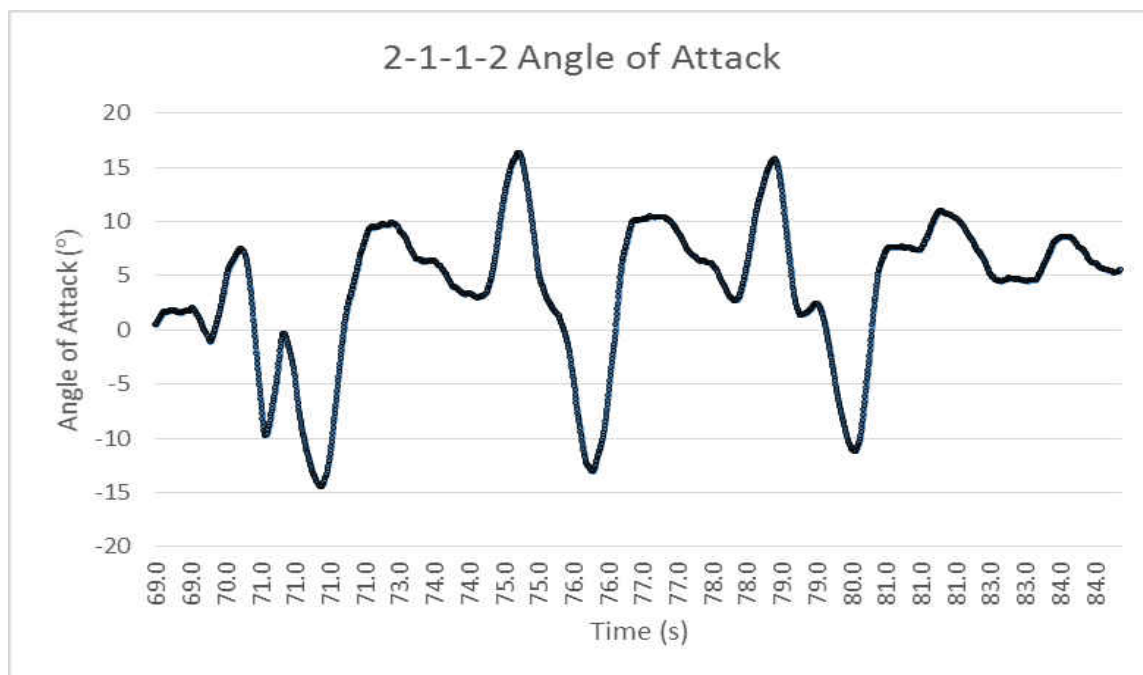


Figure 87: 2-1-1-2 Angle of Attack

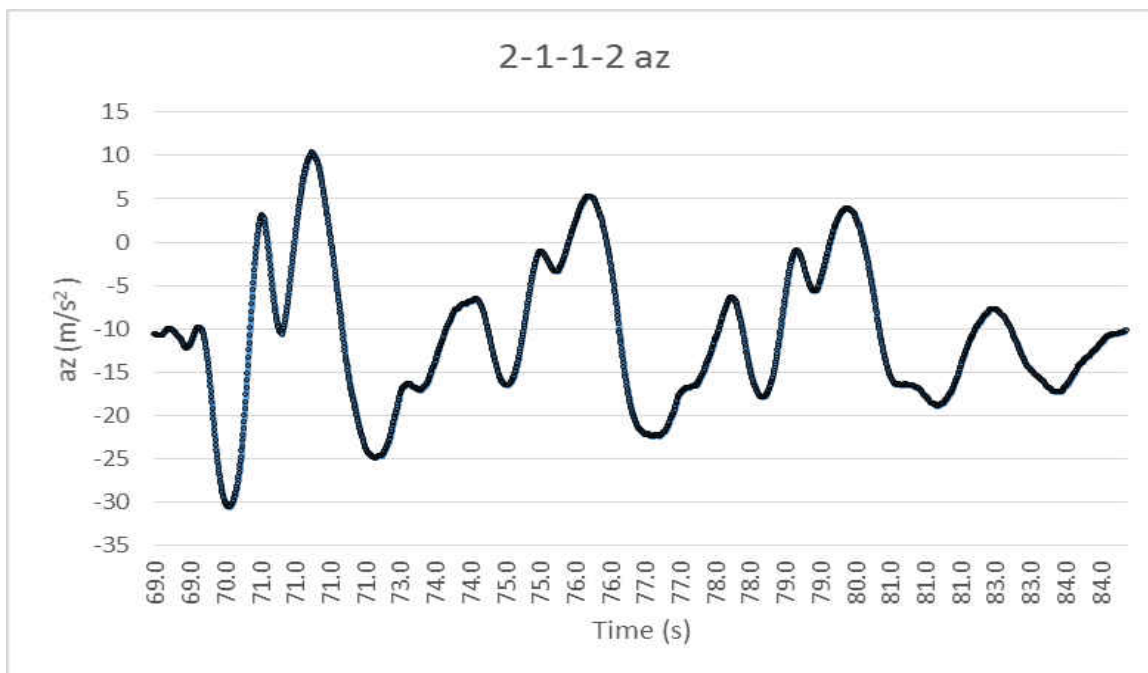


Figure 88: 2-1-1-2 Z-Axis Acceleration

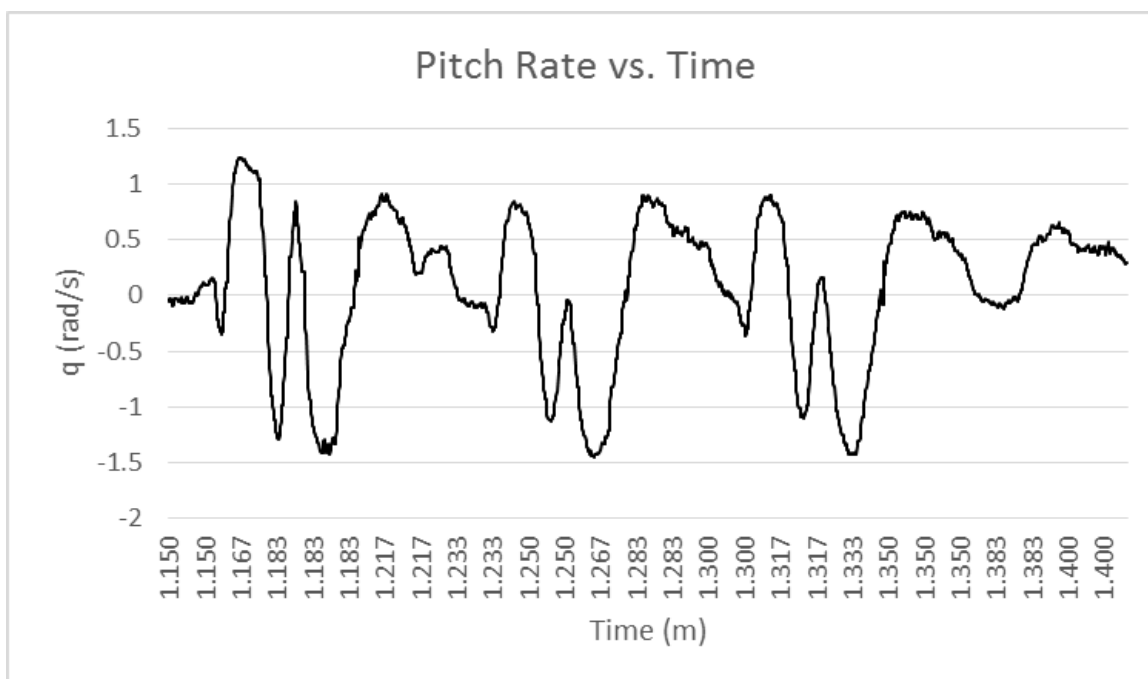


Figure 89: 2-1-1-2 Pitch Rate

Already the advantage of using a programmed multistep is made apparent. The patterns caused by the multistep maneuver show a consistency in performance across the three oscillations.

5.3.5 Chirp

The chirp is functionally a sine wave whose frequency increases and decreases at a fixed rate across a predetermined span of time. As with the multistep maneuvers, the chirp exists for each axis of motion, though for the purposes of this project, only the pitch maneuvers will be explored. As demonstrated in the vane tracking tests, the elevator is shifted at the desired frequencies, allowing for more consistent and accurate results than a pilot dictated maneuver.

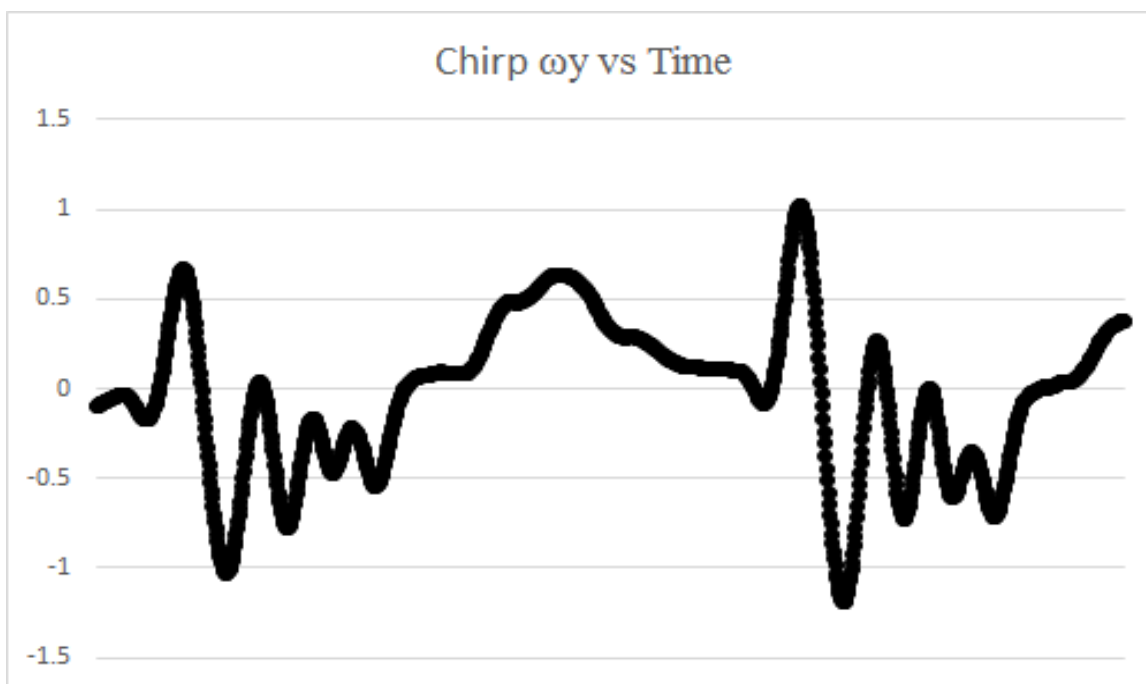


Figure 90: Chirp Pitching Velocity

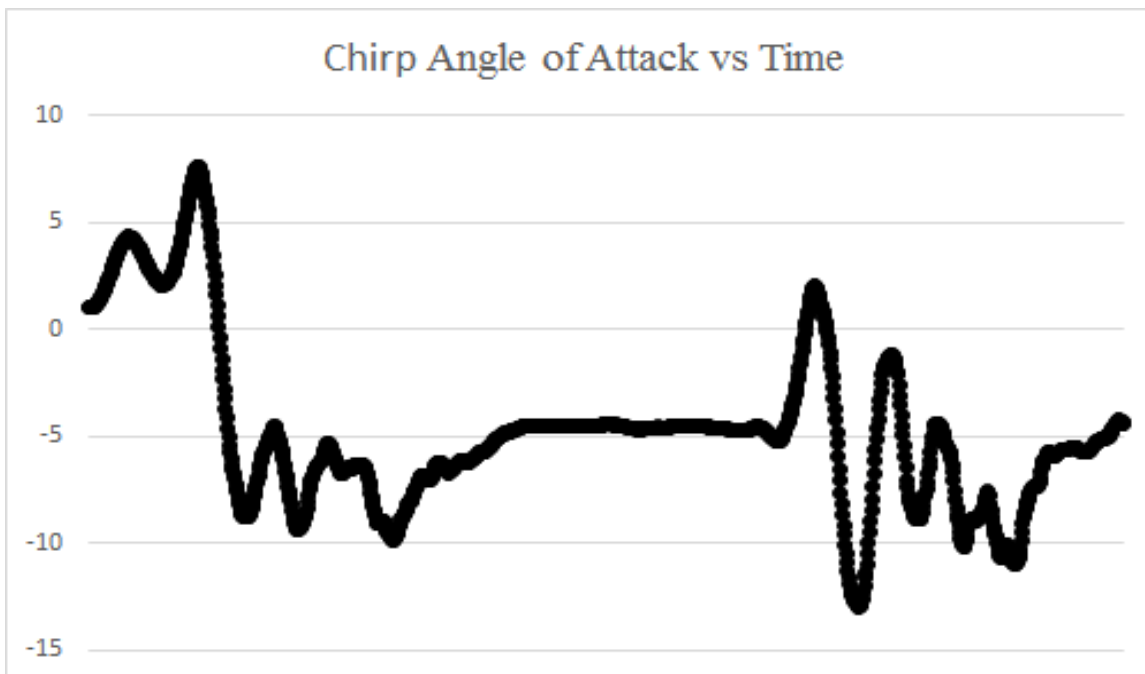


Figure 91: Chirp Angle of Attack

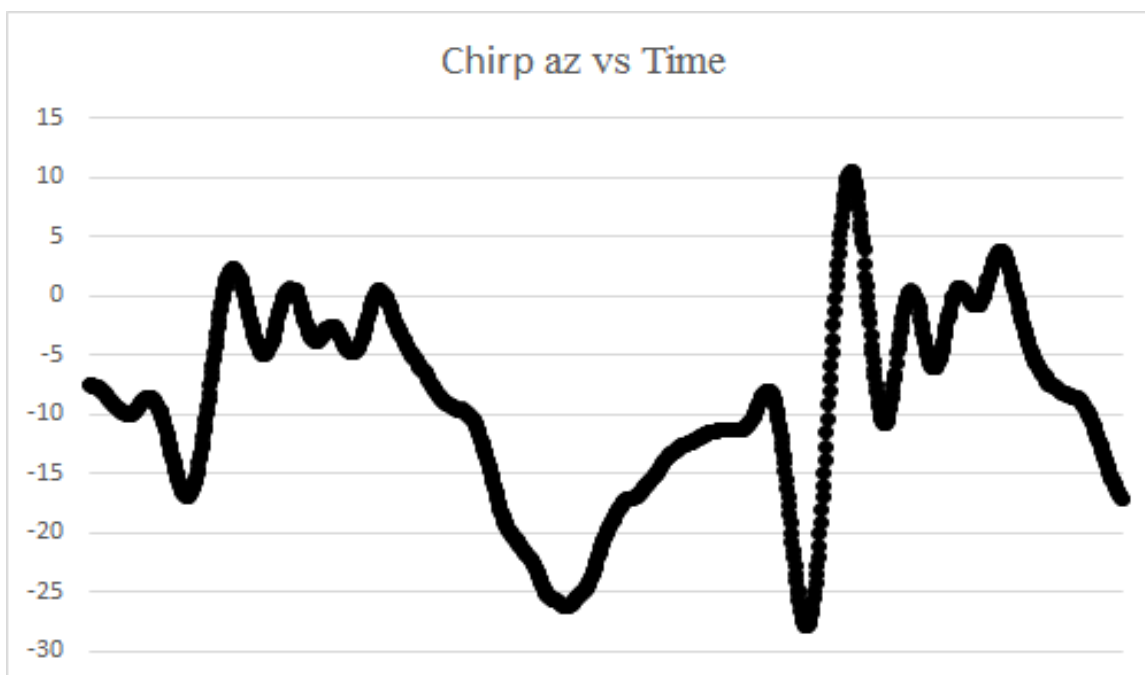


Figure 92: Chirp Z-Axis Acceleration

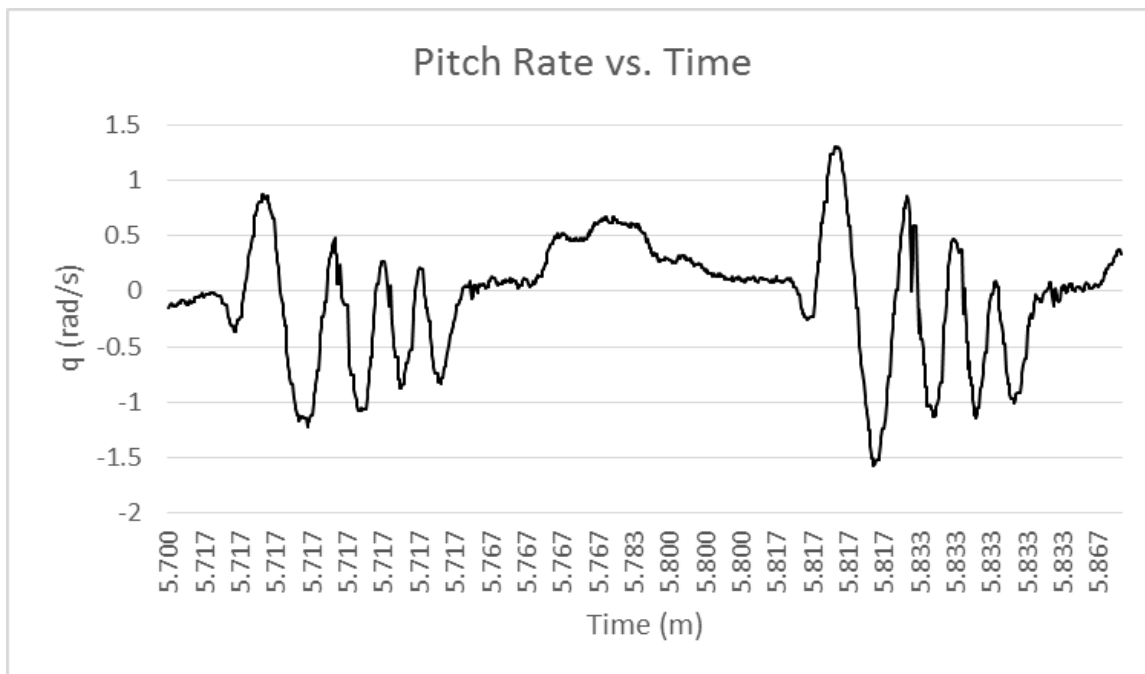


Figure 93: Chirp Pitch Rate

5.3.6 Sinusoid High Frequency

The sinusoid wave is similar to the chirp function, save that its frequency does not change for the duration of the maneuver. Also, as demonstrated by the vane tracking tests, the frequency of the control surface deflections well represented, and demonstrates a better fit than a traditional pilot executed maneuver. The figures below detail the results from a sinusoid motion at a frequency at $\frac{3}{4}$ Hz, and one at a frequency of $\frac{1}{4}$ Hz. Higher frequencies were avoided as the Air Titan's wings showed a dangerous degree of flexion at higher frequency maneuvers.

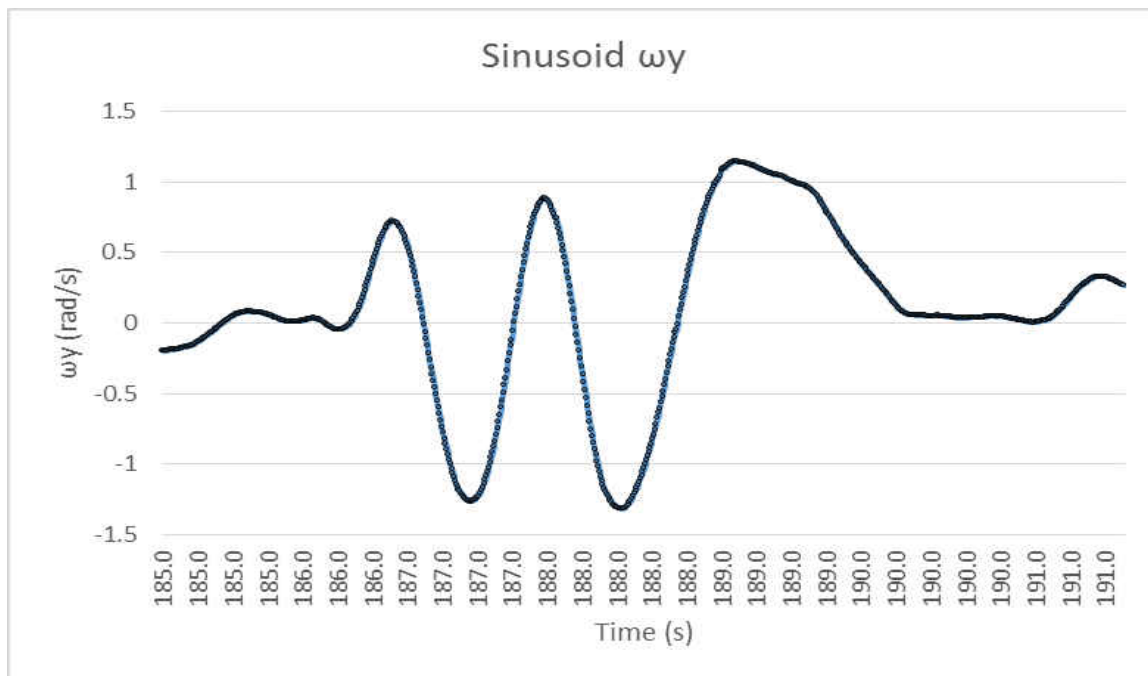


Figure 94: Sinusoid Pitching Velocity

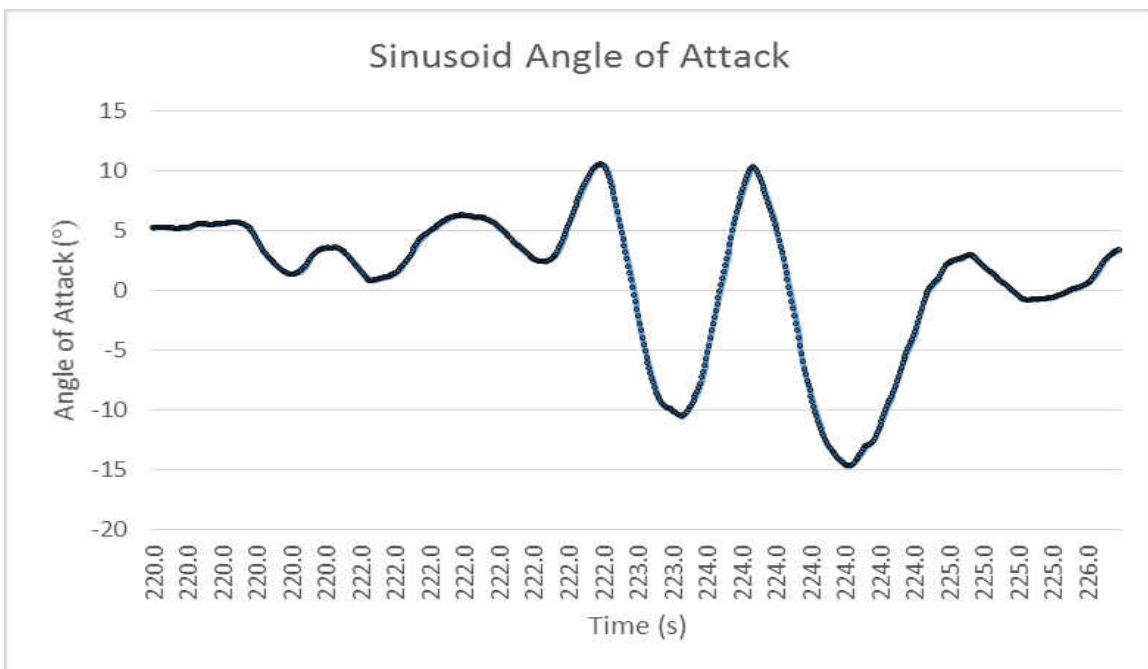


Figure 95: Sinusoid Angle of Attack

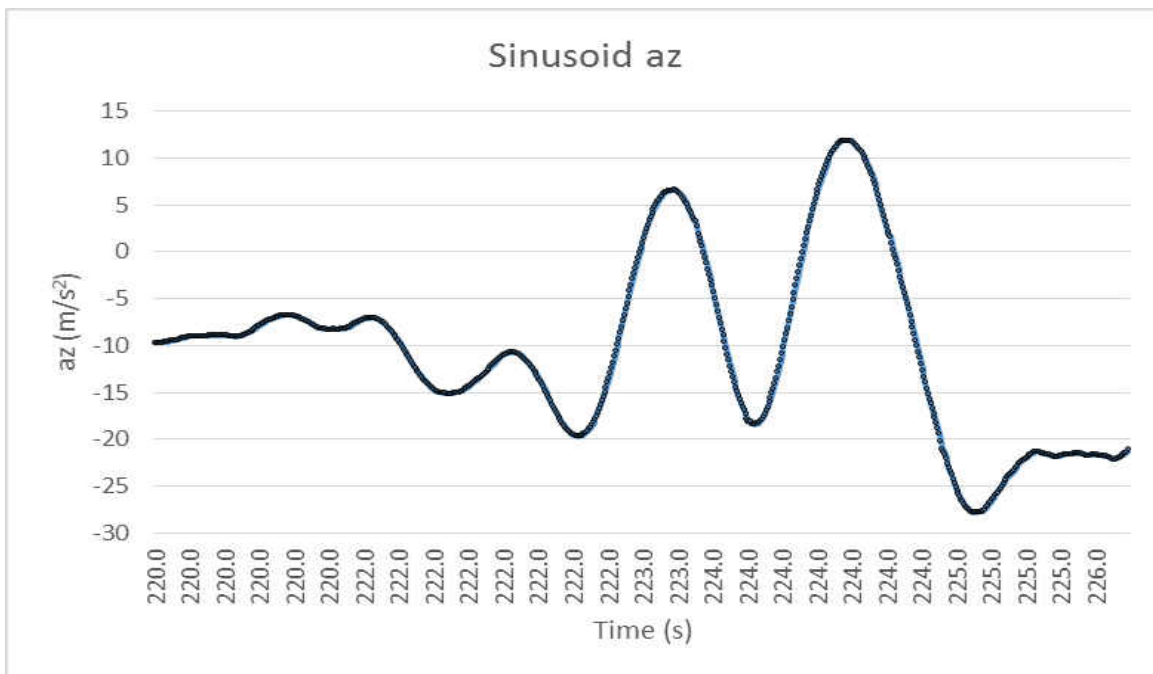


Figure 96: Sinusoid Z-Axis Acceleration

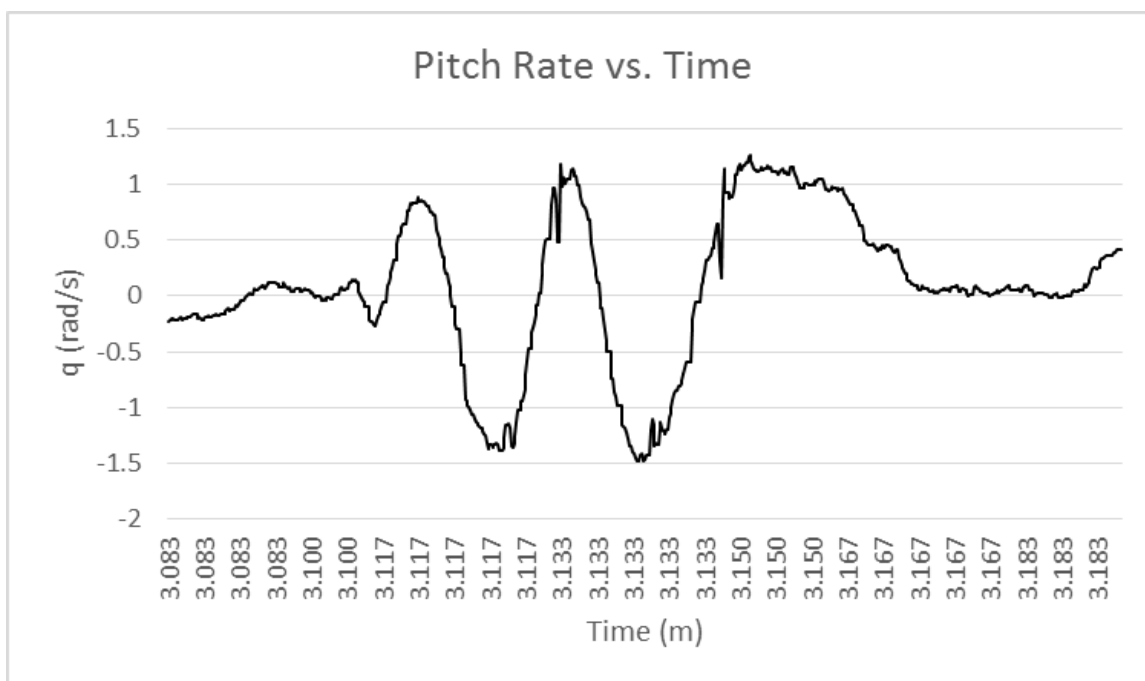


Figure 97: Sinusoid Pitch Rate

5.3.7 Sinusoid Short Frequency

A second test on the sinusoid maneuver was performed with a reduced frequency of oscillation. This tests also makes use of the Pixhawk's stabilize flight mode to assist the pilot in maintaining level flight conditions. This ensures that the pitching data will remain on the y-axis of rotation throughout the maneuvers duration, and not couple with the remaining two axes. As stated in the prior section, the oscillation is set to $\frac{1}{4}$ Hz for this test.

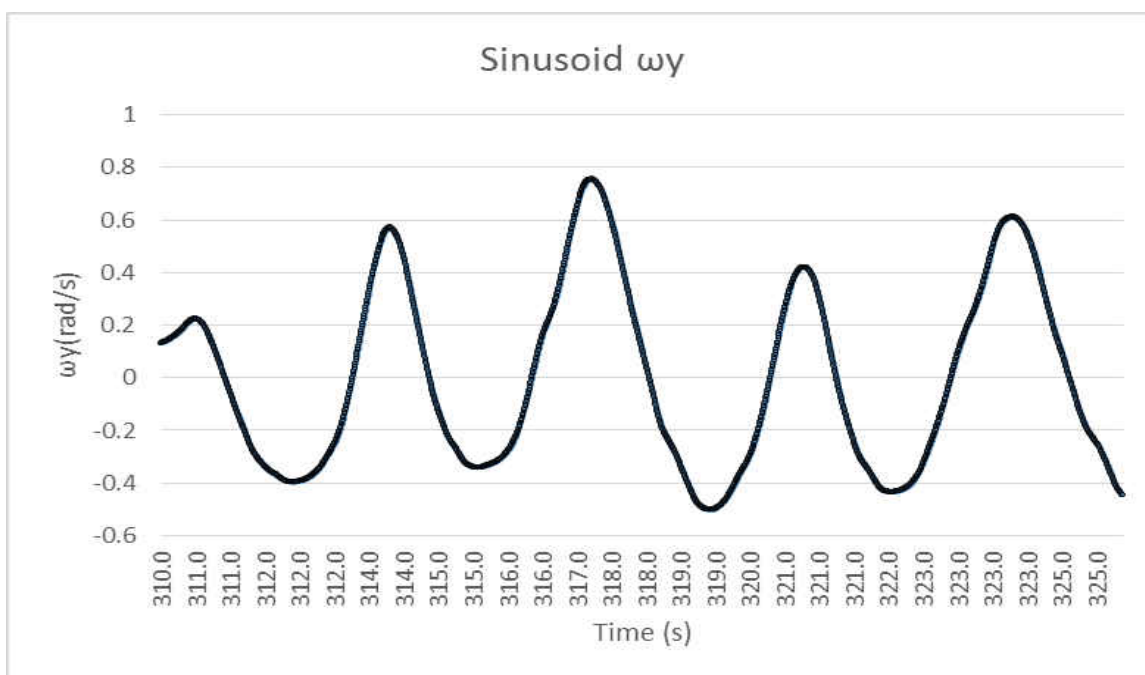


Figure 98: Sinusoid Pitching Velocity

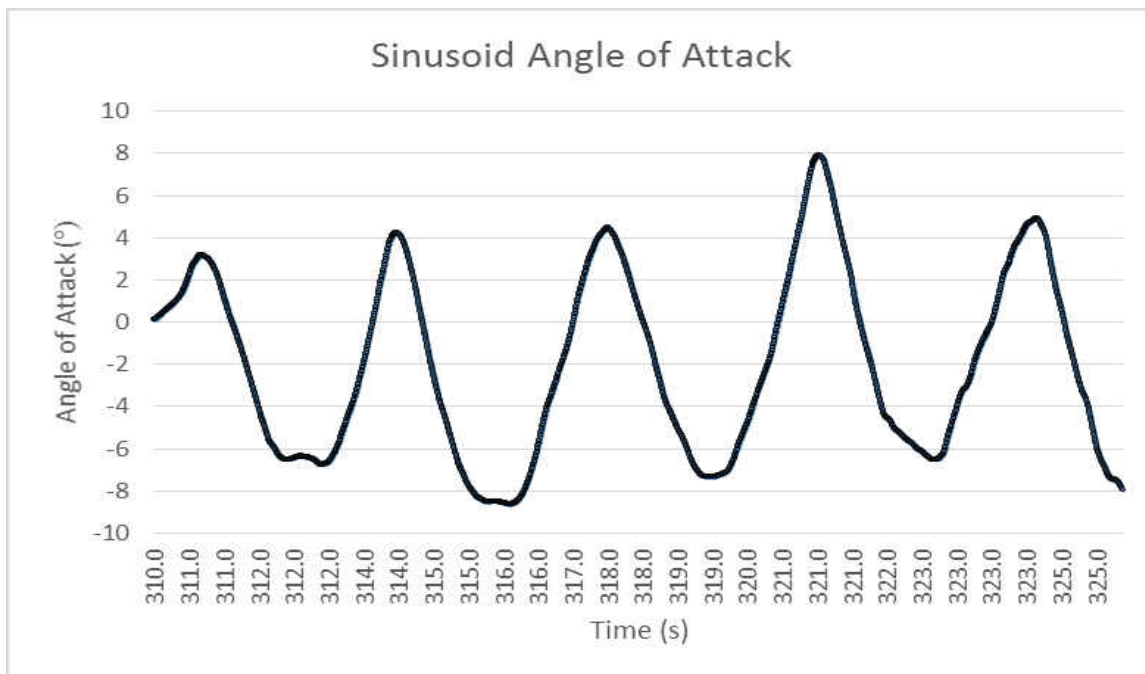


Figure 99: Sinusoid Angle of Attack

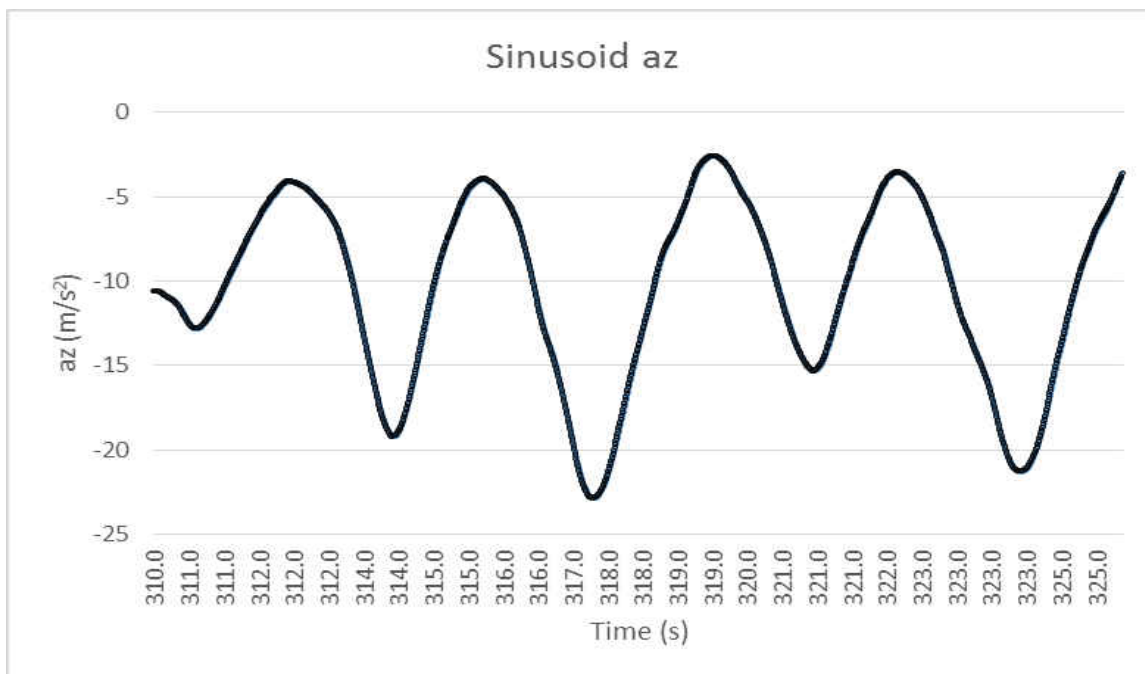


Figure 100: Sinusoid Z-Axis Acceleration

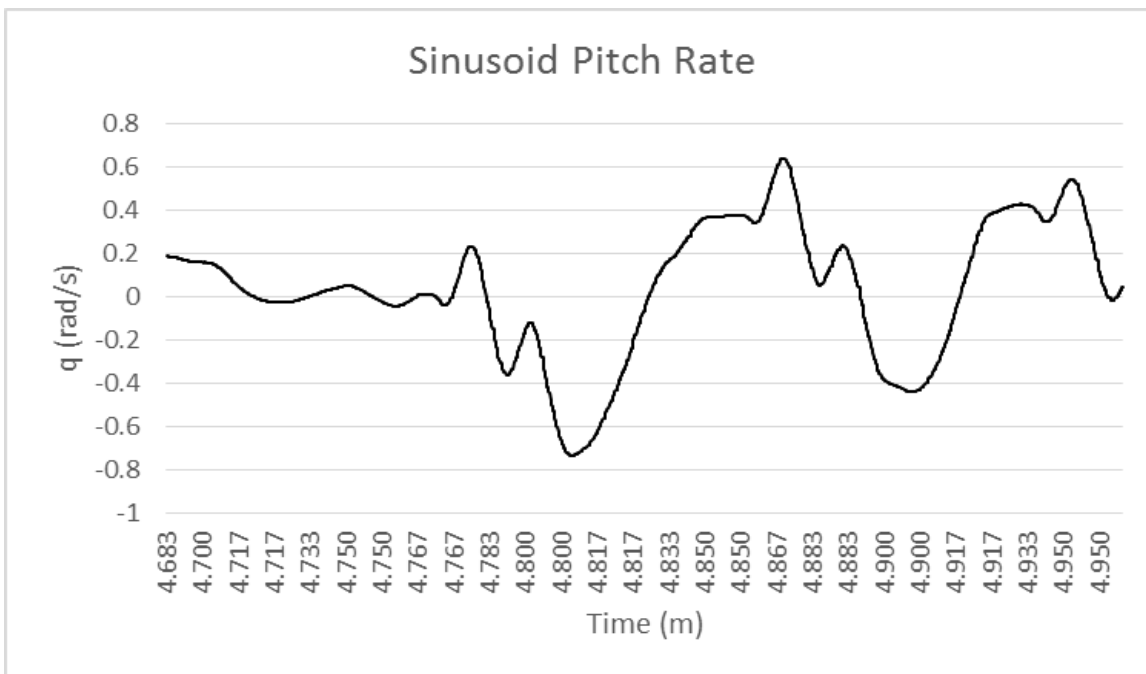


Figure 101: Sinusoid Pitch Rate

5.3.8 S-Turn

The S-Turn serves as the only pilot performed maneuver for this test. The pilot shifts the plane laterally while rotating on its z-axis to stimulate rotational effects in the x and z axes. This is to stimulate the sideslip angles, as the later pitching maneuvers should have little effect on the probe itself.

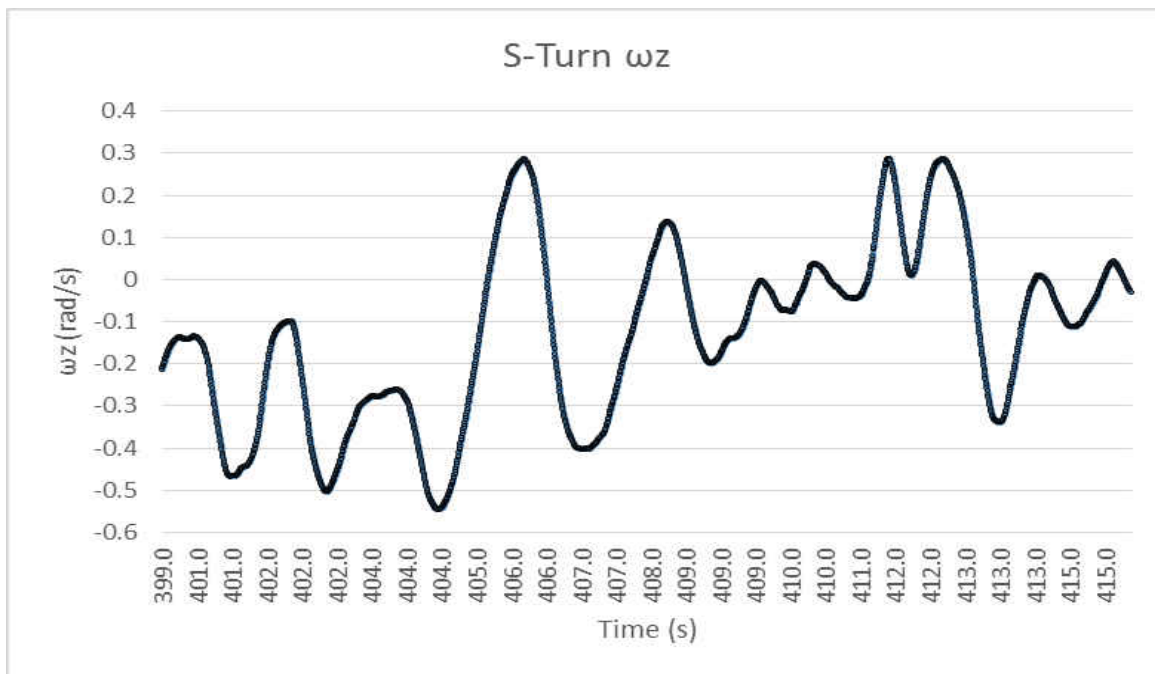


Figure 102: S-Turn Yaw Rotational Velocity



Figure 103: S-Turn Sideslip Angle

CHAPTER VI

CONCLUSIONS

The flexible coding available to the Pixhawk in conjunction with a custom air data vane proved capable for logging in-flight data for unmanned aircraft. The high logging rate, together with moderately accurate individual sensors, provides quality data from a mostly turnkey system. This, in conjunction with the Pixhawk's automated system identification maneuvers, allows for increased complexity in system ID maneuvers that would prove difficult through manual execution.

Using a formally designed experiment with this system, a model of pitch dampening as a function of power and angle of attack can be identified. While the final CCD was not performed due to time constraints, the system proved capable of performing the requested maneuvers and providing data to determine the stability and control derivatives for the pitching moment.

The following sections in this chapter show the key results and lessons learned across this project. This includes a brief overview of the tolerances for the Pixhawk, followed by a discussion on the vane dynamics of the custom air data vane. Next comes some lessons learned in aircraft system identification, the validity of DoE/RSM techniques, and ways to further improve the systems in future work.

6.1 PROBE VANE DYNAMICS

The vane dynamics were a primary focus because unlike most of the Pixhawk's systems, vanes were designed in house and were a major concern in the development process. The step input tests performed in the wind tunnel showed that even in a worst-case scenario, the vane motion would dampen out at around 40 milliseconds. In all tests, the vanes returned to the correct flow

direction, and had impeccable tracking for lower frequency oscillations. There exists a slight overshoot at the peaks of oscillation due to the inertia of the vane itself, but only in the most extreme cases, which is not indicative of typical flight behavior.

6.2 LESSONS LEARNED IN AIRCRAFT SYSTEM IDENTIFICATION

The automation of system identification in the Pixhawk has unlocked the potential for even more accurate in-flight data than was possible with the traditional multistep maneuver. Morelli states that the multi-sine is the “poor man’s sinusoid maneuver” [3]. A multistep maneuver is much simpler for a pilot to initiate, as it involves only holding the control surface at an extreme for a predetermined fraction of time. Sinusoid maneuvers are especially difficult to replicate precisely through manual control and chirp motions are even tougher still. The multistep maneuver serves as a manageable alternative for those restricted to manual control only.

By having the Pixhawk provide the inputs, a sinusoid and chirp maneuver can be performed with formulaic accuracy. The pilot only needs to flip the switch to begin the maneuver with settings dictated by telemetry on the ground. This means a multi-sine maneuver can be performed with a similar pattern to the multistep maneuver to generate even clearer results than would be possible for a manual operated pilot. Due to time constraints, this possibility could not be explored for this project, but future work would see even greater accuracy when collecting flight characteristics than is possible for the traditional multistep maneuver.

6.3 VALIDITY OF DESIGN OF EXPERIMENTS

While the employment of DoE/RSM methods is not without challenges, the FCD model proved effective for air data vane characterization. Statistical models from the design space provide a framework to validate the effectiveness of the test and create a more robust, defensible, regression model compared against the traditional OFAT methods. The optimization techniques

also proved useful in altering the design of the vanes for optimal performance. By randomizing the design points, the effect of any lurking factors is averaged over the entire experiment.

6.4 IMPROVEMENTS OVER PREVIOUS SYSTEMS

The concept for the Pixhawk autopilot system and the air data vane was created by Scott Hood. Given the similar hardware, similar performance results were expected, though the vanes used in this experiment had a lower resolution than the specifications provided in Hood's paper. This is not so much a fault in the vane design as it was the electrical noise inherent to the Pixhawk for this project that limited resolution. The sophistication of the PX4 development environment allows for a much more research friendly system than other competitors like the basic Arduino, and is far more user friendly than other candidates with greater processing power, like the Beaglebone Black.

6.5 FUTURE WORK

More work is planned for refining the sinusoid maneuvers as an improvement upon the traditional multistep maneuver. This would include implementing more complex sinusoid maneuvers into the PX4's code and testing their efficiency against the multistep maneuver. This development would then transition to the development of system identification maneuvers to collect in flight data for multiple flight characteristics simultaneously, potentially exciting motion across all three axes simultaneously to collect flight parameter data in minimal time.

Although much of the future work will revolve around a project's specific requirements, one of the overall goals is the creation of an ODU branch for the PX4 firmware. This would allow anyone working on ODU projects to track potential changes to the ODU code, reduce the duplication of effort, and pave the way for ODU produced features to be integrated into the PX4

firmware master branch. This also opens up new avenues for potential UAV automation for ODU's mechanical and aerospace engineering department.

While the short-period calculation was ultimately left unfinished in this project, the derivation of the Air Titan's moments of inertia is all that is needed to implement the procedure. The flexibility and complexity of the Air Titan's design makes finding these inertias difficult, but this is the last barrier to conducting a formal study using outdoor testing. Future testing should endeavor to use a more rigid airframe.

REFERENCES

1. Yechout, T.R., et al., *Introduction to aircraft flight mechanics : performance, static stability, dynamic stability, classical feedback control, and state-space foundations*. Second edition. ed. AIAA education series. 2014, Reston, Virginia: American Institute of Aeronautics and Astronautics, Inc. xvii, 700 pages.
2. Morelli, E.A. and R. DeLoach. *Response surface modeling using multivariate orthogonal functions*. 2001.
3. Klein, V. and E.A. Morelli, *Aircraft system identification : theory and practice*. AIAA education series. 2006, Reston, VA: American Institute of Aeronautics and Astronautics. xiv, 484 p.
4. Landman, D., et al., *A High Performance Aircraft Wind Tunnel Test Using Response Surface Methodologies*, in *2005 U.S. Air Force T&E Days*. 2005, American Institute of Aeronautics and Astronautics.
5. Montgomery, D.C., *Design and analysis of experiments*. Eighth edition. ed. 2013, Hoboken, NJ: John Wiley & Sons, Inc. xvii, 730 pages.
6. Myers, R.H., D.C. Montgomery, and C.M. Anderson-Cook, *Response surface methodology : process and product optimization using designed experiments*. 3rd ed. Wiley series in probability and statistics. 2009, Hoboken, N.J.: Wiley. xiii, 680 p.
7. Favaregh, N. and D. Landman. *Global Modeling of Pitch Damping from Flight Data*. 2006. Reston: American Institute of Aeronautics and Astronautics.
8. Klein, V., *Estimation of aircraft aerodynamic parameters from flight data*. *Progress in Aerospace Sciences*, 1989. **26**(1): p. 1-77.
9. Lensi, M., *Estimating open-loop stability and control derivatives for the Tu-144LL supersonic transport from flight data*, in *Atmospheric Flight Mechanics Conference*. 2000, American Institute of Aeronautics and Astronautics.
10. Kimberlin, R.D., *Flight testing of fixed-wing aircraft*. AIAA education series. 2003, Reston, VA: American Institute of Aeronautics and Astronautics. xviii, 441 p.
11. DeLoach, R. *MDOE perspectives on wind tunnel testing objectives (modern design of experiments)*. 2002. Reston, VA: American Institute of Aeronautics and Astronautics, Inc.
12. DeLoach, R., *Tactical Defenses Against Systemic Variation in Wind Tunnel Testing*. 2002, American Institute of Aeronautics and Astronautics: NASA Langley Research Center. p. 41.
13. DeLoach, R. and B.L. Berrier, *Productivity and quality enhancements in a configuration aerodynamics test using the modern design of experiments*. 2004.
14. Landman, D., et al., *Efficient Methods for Complex Aircraft Configuration Aerodynamic Characterization using Response Surface Methodologies*. 2006. p. 1-12.
15. *Pixhawk Autopilot*. 2016 [cited 2016 October 18]; Available from: <https://pixhawk.org/modules/pixhawk>.
16. Hood, S., *Development of a flight data acquisition system for small unmanned aircraft*. 2014, Oklahoma State University: Ann Arbor. p. 180.

17. Robotics, D. *How-to guide: Pixhawk with 6S batteries (>4S)*. 2013; Available from: https://pixhawk.org/users/tutorials/pixhawk_6s_mod.
18. Britcher, C., *Pressure Probes Lecture*. 2017, Old Dominion University: Norfolk, VA.
19. Gonzalez, J.C. and E.A. Arrington, *Five-Hole Flow Angle Probe Calibration for the NASA Glenn Icing Research Tunnel*. 1999.
20. Karam, J.T.J.R., *Dynamic behavior of angle-of-attack vane assemblies (model for aircraft thunderstorm penetration studies)*. *Journal of Aircraft*, 1975. **12**: p. 190-192.
21. Wieringa, J., *Evaluation and design of wind vanes*. *Journal of Applied Meteorology*, 1967. **6**(6): p. 1114-1122.
22. Bertin, J.J. and R.M. Cummings, *Aerodynamics for engineers*. Sixth edition. ed. 2014, Boston: Pearson. x, 822 pages.
23. *Github*. 2017 [cited 2016 09]; Available from: <https://github.com/Deafro/FirmwareAOASS>.
24. Meier, L. *PX4 Development Guide*. 2010 11/08/2016 [cited 2016 08/03]; Available from: <http://dev.px4.io/>.
25. Flyer, D. *QGroundControl*. 2009 [cited 2017 01]; Available from: <http://qgroundcontrol.org/>.
26. Nutt, G. *NuttX Real-Time Operating System*. 2017 [cited 2016 08]; Available from: <http://nuttx.org/>.
27. *Flying Unmanned Aircraft: A Pilot's Perspective*. 2011: Hampton.
28. *LSM303D Datasheet*. 2017 [cited 2015 08]; Available from: <https://www.pololu.com/file/0J703/LSM303D.pdf>.
29. *L3GD20H Datasheet*. 2017.
30. Ardupilot. *Using an Airspeed Sensor*. 2016 [cited 2016 05]; Available from: <http://ardupilot.org/plane/docs/airspeed.html>.
31. *Arduino*. 2016 [cited 2016 October 10]; Available from: <https://www.arduino.cc/>.
32. *Beaglebone Black*. 2016, October 20 [cited 2015 October 18]; Available from: <http://beagleboard.org/black>.
33. Bhandari, S., et al. *Avionics System for UAV Flight Controls Research*. 2013. Reston: American Institute of Aeronautics and Astronautics.

APPENDIX A

MICROCONTROLLER AND SOFTWARE OPTIONS

The microcontroller serves as the core of the data acquisition system. This tool collects over fifty different parameters from acceleration to angular rotation. An autopilot is useful for the validation of test runs because the data can be compared against the intended flight path for validation. To protect the collected data, the autopilot should be redundant in case of system failure. Given this project was on a strict budget, affordability was a serious consideration. Finally, a high priority is placed on the ease of use and the ability to expand on the system for various other projects for other parties who might use this data acquisition platform in the future.

A1. ARDUPILOT MEGA 2.8

The Ardupilot Mega 2.8 is produced by 3D Robotics. It is one of the older models on the market and users can use the Arduino libraries in its construction and shares a large open source community that provides software packages to resolve potential problems. This community makes adjusting the Ardupilot to other systems much simpler, since it is likely someone already developed a solution to a problem or finished most of the work. It must be efficient because it is referenced by researchers in several publications [31].



Figure 104: Ardupilot Mega 2.8

However, the hardware supporting the Ardupilot Mega 2.8 is not capable of providing the necessary power and functions compared to newer systems on the market. Many of its competitors boast newer software and more advanced functions. The lack of on-board memory and low CPU speeds are not powerful enough to handle this experiment. Also, there is no way to add additional memory to the Mega, which means that within a few years the system would be considered obsolete.

A2. CUSTOM ARDUINO

One way to solve the Ardupilot problem is to create a custom Arduino board. The hardware in this package would fit any needs that might arise without sacrificing the open source community and the Arduino libraries that make this system user friendly [16, 31].



Figure 105: Custom Arduino

The construction of a new system, however, means a lack of third party sensor support. Any sensors needed would have to be manually added and their code written. The effort of creating a fully independent system is a sizable one and would be difficult to complete given the time constraints of this project

A3. BEAGLEBONE BLACK

The Beaglebone Black boasts the strongest hardware among autopilots on the market. Unfortunately, this system lacks both the libraries and third-party sensor support. This system would not need to be constructed from the ground up, but much like the Custom Arduino, the integration of sensors would be a significant undertaking and may not yield ideal results [32].

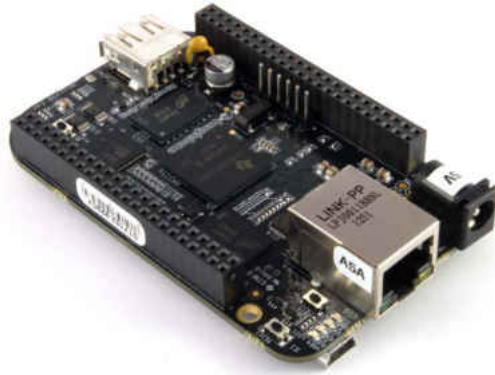


Figure 106: Beaglebone Black

A3.1 Field Programmable Gate Array (FPGA)

The FPGA is an integrated circuit designed to be configured after manufacturing. Researchers at Cal Poly Pomona used a FPGA board and a real-time operating system to gain full control over the hardware environment [33]. The freedom to customize the FPGA allows for the construction of an autopilot for extremely specific applications. Simply put, this option is building an autopilot from the ground up. This also makes it the most difficult of the options to construct, as the only features available are the one made by the designer themselves.



Figure 107: FPGA

A3.2 Data Analysis Options

A3.2.1. Flight Plot

Flight plot is a Java app that provides an alternative way of decoding and viewing sdlog2 files. This is not recommended for those looking to transition the data to a graphing program like Excel or MATLAB, yet it does provide the log data without the timely process of converting the logs to csv files. This makes it ideal for performing rapid iterations during testing to ensure instruments are working properly [24].

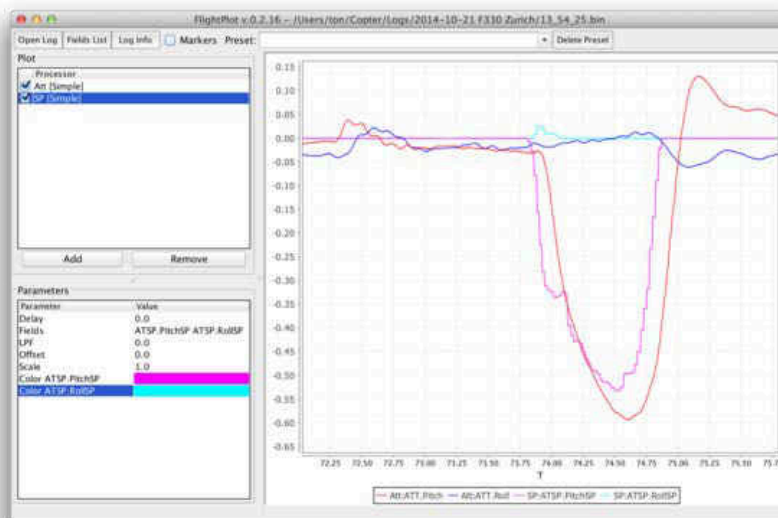


Figure 108: Flight Pilot GUI

A3.3 Log Muncher

Log Muncher is an online client that accepts log files directly and charts all parameters against the GPS timestamp given a few minutes to process. This is the software PX4's development team recommends to those using the Pixhawk, as it doesn't require converting to csv files to observe data. Also, being a web based client, it is easy to transfer data to colleagues [24].

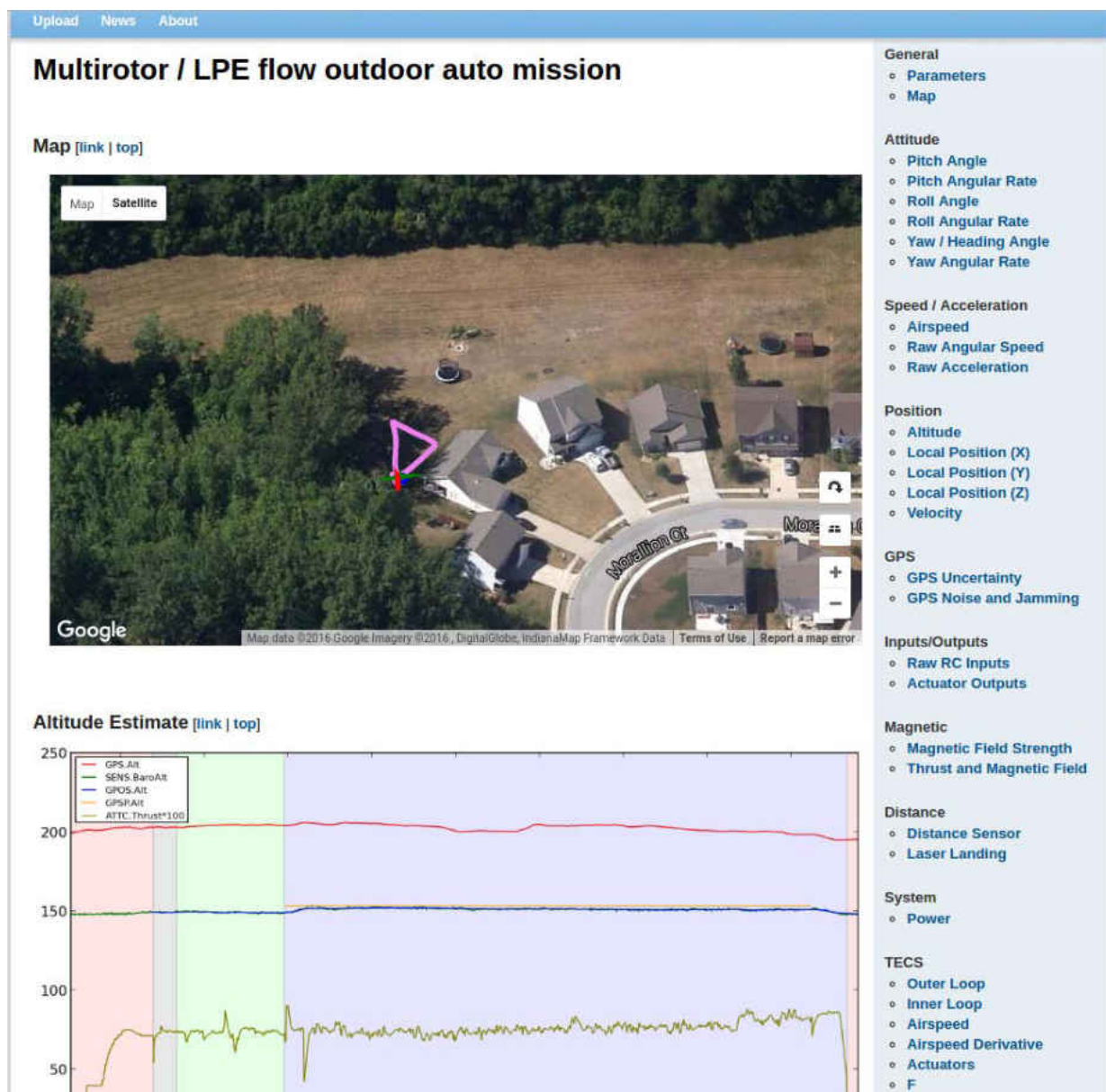
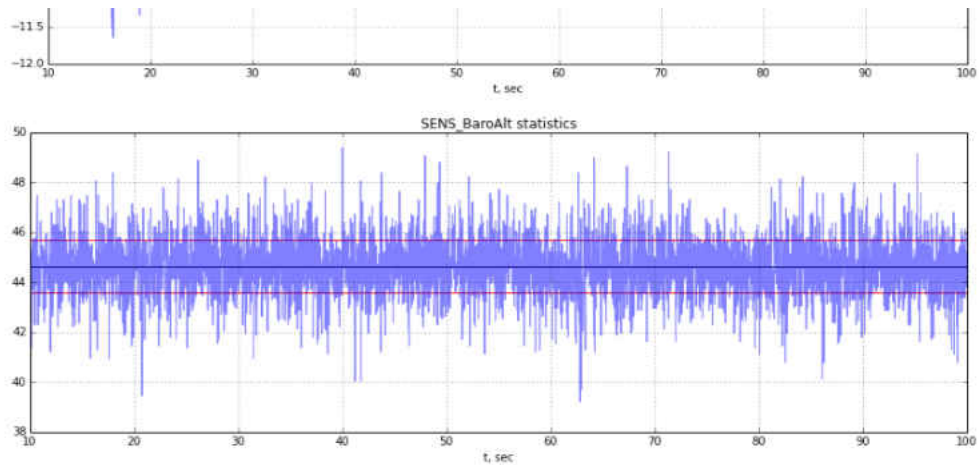


Figure 109: Log Muncher Web Page

The downside to this software is its lack of flexibility. Log Muncher does not chart graphs from custom channels, and it is difficult to collect specific information on any one part of a particular flight.

A3.4. Px4tools

Px4Tools is easy to share through the code repository github.com. This software has access to advanced plotting capabilities, heavy customization, and the tools needed for detailed analysis. It is a powerful tool for users familiar with python, though it does require the files be converted to csv before using PX4Tools [24].



```
In [11]: px4tools.find_lpe_gains(data[10:100])
```

```
Out[11]: {'LPE_ACC_XY': 0.000735915925487951,
'LPE_ACC_Z': 0.0013686317016863641,
'LPE_BAR_Z': 1.0520127982555052,
'LPE_GPS_VXY': 0.15151691018745062,
'LPE_GPS_VZ': 0.15677723863580798,
'LPE_GPS_XY': 0.53024927093676344,
'LPE_GPS_Z': 2.0419812493429053,
'LPE_LDR_Z': 0}
```

```
In [5]: px4tools.plot_velocity_loops(data1)
data1.STAT_MainState.plot()
```

```
Out[5]: <matplotlib.axes.AxesSubplot at 0x7f702c5ea450>
```

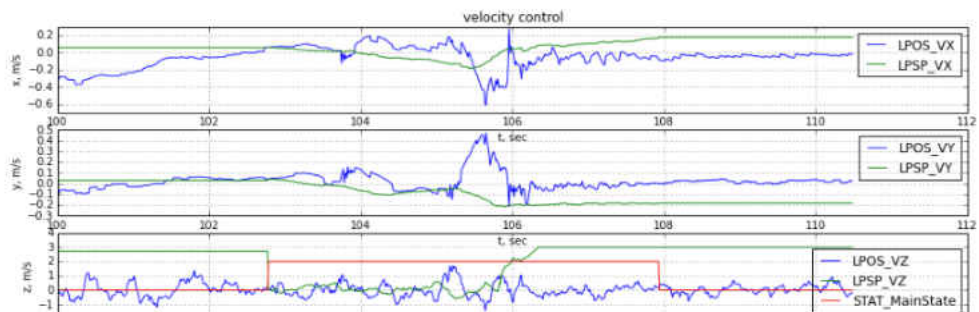


Figure 110: Px4 Tools

APPENDIX B

ADC LOGGING

The only change made to the firmware PX4 Flight Stack was to add an additional subscription to `sdlog2.c` that transmits data from the 3.3V ADC pins to the `sdlog2` function. Additional changes were implemented in `sdlog2_messages.h` to add a family tree and create names to house the collected data. The changes were minimal and did not interfere with the other sections of the code. Each code change is listed below, indicating the name and location of the file where changes were made [23]: The fully assembled code used in this project can be found at <https://github.com/Deafro/FirmwareAoASS>

A1. FIRMWARE/SRC/MODULES/SDLOG2/SDLOG.C

The PX4 developers include an additional module called `adc_report` that stores all voltage information across the eight ADC channels. In order to use this data in `sdlog2.c`, the function must subscribe to `adc_report`.

Line	Code Modified
114	<code>#include <uORB/topics/adc_report.h></code>

Table 10: ADC Report Subscription

The next change involved creating a structure variable for the function `adc_report` and `adc` source of information:

Line	Code Modified
1226	<code>struct adc_report_s adc;</code>

Table 11: AOAS Source Structure

The message structure for where the data are stored was then included. (The term “AOAS” originates from the modifications made to `sdlog2_messages.h` and will be detailed in this section.)

Line	Code Modified
1288	<code>struct log_AOAS_s log_AOAS</code>

Table 12: ADC Data Variable

The program is now subscribed to `adc_report` and the integer data from `adc_report` then needs to be collected:

Line	Code Modified
1338	<code>int adc_sub;</code>
1383	<code>subs.adc_sub = -1;</code>

Table 13: ADC Data Structure

The final change was the creation of the cells for data storage. Stored in `adc.channel_value[]` is the data for all 8 ADC pins and it was through trial and error that the last two digits corresponded with the 3.3V ADC pins (6 and 7 since the array starts with 0).

Line	Code Modified
2285	<code>if (copy_if_updated(ORB_ID(adc_report)), &subs.adc_sub, &buf.adc) {</code>
2286	<code>log_msg.msg_type = Log_AOAS_MSG;</code>
2287	<code>log_msg.body.log_AOAS.channel_value_aoa = buf.adc.channel_value[6];</code>
2288	<code>log_msg.body.log_AOAS.channel_value_ss = buf.adc.channel_value[7];</code>
2289	<code>LOGBUFFER_WRITE_AND_COUNT(AOAS); }</code>

Table 14: Sdlog Family Structure

A1.1. Firmware/src/modules/sdlog2_messages.h

`sdlog2_messages.h` creates the format for messages written by `sdlog2`. The changes were minor, which included the creation of the AOAS structure and the addition of two variable names to that structure: `channel_value_aoa` and `channel_value_ss`.

Line	Code Modified
651	#define LOG_AOAS_MSG 120
652	struct log_AOAS_s {
653	float channel_value_aoa;
654	float channel_value_ss;
655	};

Table 15: Message Names

The final change was to create the LOG_FORMAT function for AOAS. The phrase “ff” allows the program to know the message contains two floating numbers. The variable names were also added to the list:

Line	Code Modified
729	LOG_FORMAT(AOAS, “ff”, “AOA,SS”)

Table 16: Log Format

A2. PARAMETER ID MANEUVERS

In addition to ADC logging, the firmware has been modified to execute system identification maneuvers. The modified code creates a separate channel that when activated, will override manual control for a control surface with a preprogrammed system identification maneuver. The code can perform step, ramp, sinusoid, chip, and 2-1-1 parameter ID maneuvers for pitch, roll, and yaw. This will place the maneuver in control of the autopilot system and provide

more consistent results than a trained pilot. Credit for this modification go to Carl Olsson, one of the initial developers for the PX4 Firmware package. The remainder of this section will discuss the sections of this code that enable the 2-1-1 pitching maneuver. Further details on this modification are provided in the source.

A2.1. Msg/Manual_Control_Setpoint.Msg

This message file is responsible for the cataloging of the manual control RC channel options available to the user. These are unassigned parameters that can be assigned to specific channels to provide specific needs. Some of these include a kill switch and a channel for deploying landing gear. The modification here involves adding an additional channel option known as “sysid_switch.”

Line	Code Modified
57	uint8 sysid_switch # spring back switch to activate system identification maneuver.

Table 17: SysID Switch

A2.2. Msg/Rc_Channels.Msg

This file creates a function moniker for the names specified in the previous message file. This is needed as the sensors.cpp file will call upon these values. An additional channel is added for the system identification switch, and in addition, the functions mapping matrix must be expanded to make room for the added channel.

Line	Code Modified
25	uint8 RC_CHANNELS_FUNCTION_SYSIDSWITCH = 23
30	int8[24] function

Table 18: SysID Channel Creation

A2.3. Src/Modules/Sensors/Sensor_Params.C

Sensor parameters is responsible for defining parameters used by sensors.cpp. This modification defines in addition to the sysid_switch, but the switch to define the specified flight maneuver, and the variables needed to create the parameter maneuvers. RC_MAP_SYSID_SW is the parameter that tells the Pixhawk what channel to assign the sysID channel. This parameter appears in any ground station software, so this value can be changed to whatever is required. RC_SYSID_TH represents the total range of motion for the system. Both are required to define the channel.

Line	Code Modified
2575	/**
2576	* System identification switch channel mapping
2577	*
2578	* @min 0
2579	* @max 18
2580	* @group Radio Switches
2581	* @value 0 Unassigned
2582	* @value 1 Channel 1
2583	* @value 2 Channel 2
2584	* @value 3 Channel 3
2585	* @value 4 Channel 4
2586	* @value 5 Channel 5
2587	* @value 6 Channel 6
2588	* @value 7 Channel 7
2589	* @value 8 Channel 8
2590	* @value 9 Channel 9
2591	* @value 10 Channel 10

Table 19: SysID Channel Mapping Parameter

Table 19 (continued)

2592	* @value 11 Channel 11
2593	* @value 12 Channel 12
2594	* @value 13 Channel 13
2595	* @value 14 Channel 14
2596	* @value 15 Channel 15
2597	* @value 16 Channel 16
2598	* @value 17 Channel 17
2599	* @value 18 Channel 18
2600	*/
2601	PARAM_DEFINE_INT32(RC_MAP_SYSID_SW, 6);

Line	Code Modified
3854	* Threshold for the system identification transition switch
3855	*
3856	* 0-1 indicate where in the full channel range the threshold sits
3857	* 0 : min
3858	* 1 : max
3859	* sign indicates polarity of comparison
3860	* positive: true when channel>th
3861	* negative: true when channel<th
3862	*
3863	* @min -1
3864	* @max 1
3865	* @group Radio Switches
3866	*
3867	*
3868	*/
3869	PARAM_DEFINE_FLOAT(RC_SYSID_TH, 0.25f);

Table 20: SysID Channel Threshold Parameter

SID_MANOEUVRE is responsible for dictating to the modification what parameter ID maneuver will be dedicated to the specified channel. The value is set to nine to select the 2-1-1 pitching maneuver. To select the maneuver of choice, just select the corresponding integer and replace the standing value.

Line	Code Modified
3309	/**
3310	* Define the sysID manoeuvre
3311	*
3312	* @min 0
3313	* @max 16
3314	* @value 0 Disabled
3315	* @value 1 Step/Ramp in roll
3316	* @value 2 Step/Ramp in pitch
3317	* @value 3 Step/Ramp in yaw
3318	* @value 4 Step in throttle
3319	* @value 5 Chirp in roll
3320	* @value 6 Chirp in pitch
3321	* @value 7 Chirp in yaw
3322	* @value 8 2-1-1 in roll
3323	* @value 9 2-1-1 in pitch
3324	* @value 10 2-1-1 in yaw
3325	* @value 11 2-1-1-2 in roll
3326	* @value 12 2-1-1-2 in pitch
3327	@value 13 2-1-1-2 in yaw
3328	@value 14 3-2-1-1 in roll

Table 21: SysID Maneuver Parameter

Table 21 (continued).

3329	@value 15 3-2-1-1 in pitch
3330	@value 16 3-2-1-1 in yaw
3331	* @group SysID
3332	*/
3333	PARAM_DEFINE_INT32(SID_MANOEUVRE, 0);

SID_AMPLITUDE defines how far in both directions the control surface will extend. The value 1 represents maximum deflection and can be reduced in ten percent increments depending on a user's specific needs. A negative value will reverse the control surface motion.

Line	Code Modified
3329	/**
3330	* Define the amplitude of the sysID manoeuvre
3331	*
3332	* @min -1
3333	* @max 1
3334	* @decimal 1
3335	* @group SysID
3336	*/
3337	PARAM_DEFINE_FLOAT(SID_AMPLITUDE, 1.0f);

Table 22: SysID Altitude Parameter

SID_ON_TIME defines how long the autopilot has to perform the desired maneuver. This variable has a maximum peak of 15 seconds. The time can be expanded, though fifteen seconds should be enough for most maneuvers.

Line	Code Modified
3339	/**
3340	* Define the active of the sysID manoeuvre
3341	*
3342	* @min 0
3343	* @max 15
3344	* @unit seconds
3345	* @decimal 1
3346	* @group SysID
3347	*/
3348	PARAM_DEFINE_FLOAT(SID_ON_TIME, 4.0f)

Table 23: SysID Execution Time Parameter

SID_TRIM_TIME_B locks the desired control surface to its zero position for a specified time before the maneuver is executed. This gives the aircraft time to steady itself and minimize perturbations caused in flight. The maximum time for this trim flight is set to 60 seconds. Caution should be taken with this variable. Because the program overrides manual control, the pilot will have no control over the affected surface until the program runs its course.

Line	Code Modified
3350	/**
3351	* Define the trim time before the sysID manoeuvre
3352	*
3353	* The input signal will be zero before the sid manoeuvre
3354	* for this specified time
3355	*
3356	* @min 0
3357	* @max 60
3358	* @unit seconds
3359	* @decimal 1
3360	* @group SysID
3361	*/
3362	PARAM_DEFINE_FLOAT(SID_TRIM_TIME_B, 1.0f);

Table 24: SysID Trim Time before Parameter

SID_TRIM_TIME_A represents the duration the control surface is held in its zero position after the maneuver is executed. Other than its timing, this parameter is identical to SID_TRIM_TIME_B.

Line	Code Modified
3364	/**
3365	* Define the trim time after the sysID manoeuvre
3366	*
3367	* The input signal will be zero after the sid manoeuvre
3368	* for this specified time
3369	*
3370	* @min 0
3371	* @max 60
3372	* @unit seconds
3373	* @decimal 1
3374	* @group SysID
3375	*/
3376	PARAM_DEFINE_FLOAT(SID_TRIM_TIME_A, 1.0f);

Table 25: SysID Trim Time after Parameter

SID_START_FREQ and SID_STOP_FREQ are used in the sinusoid and chirp maneuver options to determine the frequency of the oscillation. For a sinusoid maneuver, these two values are equal, but a chirp maneuver will start its oscillation at the value specified by SID_START_FREQ and end at the frequency specified at SID_STOP_FREQ. The transition between these states is linear over time. The maximum frequency is set to five hertz, with a minimum of 0.1 hertz.

SID_RAMP_SLOPE indicates the time it takes to deflect the control surface to its extreme positions. For the 2-1-1 maneuver, this value is set to zero to maximize this speed, though the parameter can be set as high as 5 seconds between oscillations.

Line	Code Modified
3400	/**
3401	* Define the ramp slope. (1/rate)
3402	*
3403	* This corresponds to the time [s] it takes to go from 0 to 1 (max stick input)
3404	*
3405	* @min 0
3406	* @max 5
3407	* @decimal 1
3408	* @unit seconds
3409	* @group SysID
3410	*/
3411	PARAM_DEFINE_FLOAT(SID_RAMP_SLOPE, 0.0f);

Table 26: SysID Ramp Slope Parameter

A2.4. Src/Modules/Sensors/Sensors.Cpp

This file serves to map the PX4 drivers to the application layer of the PX4 flight core. While the other files served to define the sysID channel and its defining parameters, this file is responsible for properly organizing the data into trees and delegating needed resources for proper

application. Only one line of structure is removed from the source code of sensors.cpp as its contribution conflicts with the modifications.

Line	Code Modified
262	<code>struct vehicle_control_mode_s vcontrol_mode;</code>

Table 27: Remove Vehicle Control Mode Structure

The Pixhawk needs to know how much memory needs to be reserved for each new parameter. This section assigns each variable to a data structure, depending how depending on its requirements.

Line	Code Modified
311	int rc_map_sysid_sw;
337	Float rc_sysid_th;
349	Bool rc_sysid_inv;
362	int sid_manoeuvre;
363	float sid_amplitude;
364	float sid_on_time;
365	float sid_trim_time_b;
366	float sid_trim_time_a;
367	float sid_start_freq;
368	float sid_stop_freq;
369	float sid_ramp_slope;

Table 28: Data Structure Assignment

Parameters in C++ are stored in a branching group of functions called the parameter tree. If the firmware is to locate the newly defined parameters, they must be defined in this parameter tree.

Line	Code Modified
399	Param_t rc_map_sysid_sw;
429	Param_t rc_sysid_th;
446	Param_t sid_manoeuvre;
447	Param_t sid_amplitude;
448	Param_t sid_on_time;
449	Param_t sid_trim_time_b;
450	Param_t sid_start_freq;
451	Param_t sid_stop_freq;
452	Param_t sid_ramp_slope;

Table 29: Parameter Tree Assignment

It is at this point that the information from the previous files is properly defined in sensors.cpp. Due to the limited processing power of the Pixhawk, it is important that the modified script not be processed if no flight maneuvers are selected. This section of code checks the SID_MANOEUVRE parameter for a value of zero. If it sees zero, it considers the sysID tree disabled and will omit it from processing.

Line	Modified Code
585	/**
586	* Check if we shall perform sysID maneuvers
587	*/
588	void check_sysid_manoeuvre(manual_control_setpoint_s *manual);

Table 30: Abort Script Check

Sensors.cpp can now find the parameters in the parameter tree, and the parameter tree houses the values defined in sensor_params. This information is extracted from the parameter tree and assigned to a variable that can be used in functions.

Line	Modified Code
694	<code>_parameter_handles.rc_map_sysid_sw = param_find("RC_MAP_SYSID_SW");</code>
725	<code>_parameter_handles.rc_sysid_th = param_find("RC_SYSID_TH");</code>
752	<code>/* SysID Params */</code>
753	<code>_parameter_handles.sid_manoeuvre = param_find("SID_MANOEUVRE");</code>
754	<code>_parameter_handles.sid_amplitude = param_find("SID_AMPLITUDE");</code>
755	<code>_parameter_handles.sid_on_time = param_find("SID_ON_TIME");</code>
756	<code>_parameter_handles.sid_trim_time_b = param_find("SID_TRIM_TIME_B");</code>
757	<code>_parameter_handles.sid_trim_time_a = param_find("SID_TRIM_TIME_A");</code>
758	<code>_parameter_handles.sid_start_freq = param_find("SID_START_FREQ");</code>
759	<code>_parameter_handles.sid_stop_freq = param_find("SID_STOP_FREQ");</code>
760	<code>_parameter_handles.sid_ramp_slope = param_find("SID_RAMP_SLOPE");</code>

Table 31: Assigning Parameters to Variables

This section gives the user a warning message of the RC parameters show values outside their defined range.

Line	Modified Code
917	<code>if (param_get(_parameter_handles.rc_map_sysid_sw, &(_parameters.rc_map_sysid_sw)) !=OK) {</code>
918	<code>Warnx("%s", paramerr);</code>
919	<code>}</code>

Table 32: Warning Message

This section is where all the parameters are pulled together to fully define the SYSID channel.

Line	Code Modified
971	<code>param_get(_parameter_handles.rc_sysid_th, &(_parameters.rc_sysid_th));</code>
972	<code>_parameters.rc_sysid_inv = (_parameters.rc_sysid_th < 0);</code>
973	<code>_parameters.rc_sysid_th = fabs(_parameters.rc_sysid_th);</code>
991	<code>_rc.function[rc_channels_s::RC_CHANNELS_FUNCTION_SYSIDSWITCH] = _parameters.rc_map_sysid_sw -1;</code>
1096	<code>param_get(_parameter_handles.sid_manoeuvre, &(_parameters.sid_manoeuvre));</code>
1097	<code>param_get(_parameter_handles.sid_amplitude, &(_parameters.sid_amplitude));</code>
1098	<code>param_get(_parameter_handles.sid_on_time, &(_parameters.sid_amplitude));</code>
1099	<code>param_get(_parameter_handles.sid_trim_time_b, &(_parameters.sid_trim_time_b));</code>
1100	<code>param_get(_parameter_handles.sid_trim_time_a, &(_parameters.sid_trim_time_a));</code>
1101	<code>param_get(_parameter_handles.sid_start_freq, &(_parameters.sid_start_freq));</code>
1102	<code>param_get(_parameter_handles.sid_stop_freq, &(_parameters.sid_stop_freq));</code>
1103	<code>param_get(_parameter_handles.sid_ramp_slope, &(_parameters.sid_ramp_slope));</code>

Table 33: Fully Define SysID Channel

This next section stores the position of the SYSID switch. The firmware needs to know what position the switch must be in to begin the maneuver.

Line	Code Modified
2209	<pre> manual.sysid_switch get_rc_sw2pos_position(rc_channels_s::RC_CHANNELS_FUNCTION_SYSIDSWITCH, _parameters.rc_sysid_th, </pre>
2210	<pre> _parameters.rc_sysid_inv); </pre>

Table 34: Switch Position

The available modes take the form of a case structure. Each mode is designated a number by `SID_MANOEUVRE` which is referenced to find the proper case for that maneuver. This line checks for that value and passes it down to the case structure.

Line	Code Modified
2213	<pre> check_sysid_manoeuvre(&manual); </pre>

Table 35: Identify Desired Maneuver

Finally, this section governs the mechanics behind the various maneuvers. The first block establishes the Boolean variable “`is_doing_manoeuvre`” and the constants for when the switch is in the off position. When the switch is flipped on, the starting time is collected based on the Pixhawk’s internal clock, and “`is_doing_manoeuvre`” is flipped to true. This triggers the final block which defines the timeframe and chooses the case structure selected by “`SID_MANOEUVRE`”. In the name of conserving space, only case nine is given in this paper, though the remaining structures are present in the code.

Line	Code Modified
2550	Void
2551	Sensors::check_sysid_manoeuvre(manual_control_setpoint_s *manual)
2552	{
2553	static bool is_doing_manoeuvre = false;
2554	static uint64_t starting_time = 0;
2555	static int _prev_sysid_sw_pos = manual_control_setpoint_s::SWITCH_POS_OFF;
2556	static constant float tau = 6.2832f;
2557	
2558	if ((manual->sysid_switch == manual_control_setpoint_s::SWITCH_POS_ON)
2559	&& (manual->sysid_switch != _prev_sysid_sw_pos)) {
2560	is_doing_manoeuvre = !is_doing_manoeuvre;
2561	starting_time = hrt_absolute_time();
2562	//warnx("pressed");
2563	}
2564	
2565	if (is_doing_manoeuvre) {
2566	float dt = static_cast<float>(hrt_absolute_time() – starting_time) / 1e6f; //calculate dt in seconds
2567	float actual_ramp_time = fabsf(_parameters.sid_amplitude) * _parameters.sid_ramp_slope;

Table 36: System Identification Maneuvers

Table 36 (continued).

2568	
2569	if (dt > _parameters.sid_on_time + _parameters.sid_trim_time_b + _parameters.sid_trim_time_a + 2 * actual_ramp_time) {
2570	is_doing_manoeuvre = false;
2571	
2572	} else {
...	...
2698	// 2-1-1 in pitch
2699	case 9:
2700	if (dt < _parameters.sid_trim_time_b dt > _parameters.sid_on_time + _parameters.sid_trim_time_b) {
2701	manual->x = 0.0f;
2702	
2703	} else if (dt < _parameters.sid_trim_time_b + _parameters.sid_on_time * 0.5f) {
2704	manual->x = _parameters.sid_amplitude;
2705	
2706	} else if (dt < _parameters.sid_trim_time_b + _parameters.sid_on_time * 0.75f) {
2707	manual->x = (-1.0f)*_parameters.sid_amplitude;
2708	
2709	} else {
2710	manual->x = _parameters.sid_amplitude;
2711	}
2712	
2713	break;

APPENDIX C

SDLOG FILE CONTENTS

```

LOG_FORMAT(ATT, "qw,qx,qy,qz,Roll,Pitch,Yaw,RollRate,PitchRate,YawRate,GX,GY,GZ"), "ffffffffffff",
LOG_FORMAT(ATSP, "ffffffff", "RollSP,PitchSP,YawSP,ThrustSP,qw,qx,qy,qz"),
LOG_FORMAT_S(IMU, IMU, "ffffffffffff", "AccX,AccY,AccZ,GyroX,GyroY,GyroZ,MagX,MagY,MagZ,tA,tG,tM"),
LOG_FORMAT_S(IMU1, IMU, "ffffffffffff", "AccX,AccY,AccZ,GyroX,GyroY,GyroZ,MagX,MagY,MagZ,tA,tG,tM"),
LOG_FORMAT_S(IMU2, IMU, "ffffffffffff", "AccX,AccY,AccZ,GyroX,GyroY,GyroZ,MagX,MagY,MagZ,tA,tG,tM"),
LOG_FORMAT_S(SENS, SENS, "ffff", "BaroPres,BaroAlt,BaroTemp,DiffPres,DiffPresFilt"),
LOG_FORMAT_S(AIR1, SENS, "ffff", "BaroPa,BaroAlt,BaroTmp,DiffPres,DiffPresF"),
LOG_FORMAT(LPOS, "X,Y,Z,Dist,DistR,VX,VY,VZ,RLat,RLon,RAlt,PFlg,GFlg,EPH,EPV"), "ffffffffLLfBBff",
LOG_FORMAT(LPSP, "ffffffff", "X,Y,Z,Yaw,VX,VY,VZ,AX,AY,AZ"),
LOG_FORMAT(GPS, "GPSTime,Fix,EPH,EPV,Lat,Lon,Alt,VelN,VelE,VelD,Cog,nSat,SNR,N,J"), "QBffLLffffBHHH",
LOG_FORMAT_S(DGPS, GPS, "QBffLLffffBHHH", "GPSTime,Fix,EPH,EPV,Lat,Lon,Alt,VelN,VelE,VelD,Cog,nSat,SNR,N,J"),
LOG_FORMAT_S(ATTC, ATTC, "ffff", "Roll,Pitch,Yaw,Thrust"),
LOG_FORMAT_S(ATC1, ATTC, "ffff", "Roll,Pitch,Yaw,Thrust"),
LOG_FORMAT(STAT, "BBBBB", "MainState,NavState,ArmS,Failsafe,IsRotWing"),
LOG_FORMAT(VTOL, "fBBB", "Arsp,RwMode,TransMode,Failsafe"),
LOG_FORMAT(CTS, "ffffff", "Vx_b,Vy_b,Vz_b,Vinf,P,Q,R"),

```

```

LOG_FORMAT(RC, "fffffffffBBBL",
            "C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,RSSI,CNT,Lost,Drop"),
LOG_FORMAT_S(OUT0, OUT, "ffffff",
            "Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7"),
LOG_FORMAT_S(OUT1, OUT, "ffffff",
            "Out0,Out1,Out2,Out3,Out4,Out5,Out6,Out7"),
LOG_FORMAT(AIRS, "fff",                "IndSpeed,TrueSpeed,AirTemp"),
LOG_FORMAT(ARSP, "fff",                "RollRateSP,PitchRateSP,YawRateSP"),
LOG_FORMAT(FLOW,                                "BffffffLLHhB",
            "ID,RawX,RawY,RX,RY,RZ,Dist,TSpan,DtSonar,FrmCnt,GT,Qlty"),
LOG_FORMAT(GPOS, "LLffffff",            "Lat,Lon,Alt,VelN,VelE,VelD,EPH,EPV,TALT"),
LOG_FORMAT(GPSP, "BLLffBfbf",
            "NavState,Lat,Lon,Alt,Yaw,Type,LoitR,LoitDir,PitMin"),
LOG_FORMAT(ESC, "HBBBHffiffH",
            "count,nESC,Conn,N,Ver,Adr,Volt,Amp,RPM,Temp,SetP,SetPRAW"),
LOG_FORMAT(GVSP, "fff",                "VX,VY,VZ"),
LOG_FORMAT(BATT, "fffffB",            "V,VFilt,C,CFilt,Discharged,Remaining,Warning"),
LOG_FORMAT(DIST, "BBBff",            "Id,Type,Orientation,Distance,Covariance"),
LOG_FORMAT_S(TEL0, TEL, "BBBBHQB",
            "RSSI,RemRSSI,Noise,RemNoise,RXErr,Fixed,TXBuf,HbTime"),
LOG_FORMAT_S(TEL1, TEL, "BBBBHQB",
            "RSSI,RemRSSI,Noise,RemNoise,RXErr,Fixed,TXBuf,HbTime"),
LOG_FORMAT_S(TEL2, TEL, "BBBBHQB",
            "RSSI,RemRSSI,Noise,RemNoise,RXErr,Fixed,TXBuf,HbTime"),
LOG_FORMAT_S(TEL3, TEL, "BBBBHQB",
            "RSSI,RemRSSI,Noise,RemNoise,RXErr,Fixed,TXBuf,HbTime"),
LOG_FORMAT(EST0,                                "fffffffffBBHB",
            "s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,nStat,fNaN,fFault,fTOut"),
LOG_FORMAT(EST1,                                "ffffffffffffff",
            "s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23,s24,s25,s26,s27"),

```

```

LOG_FORMAT(EST2, "P0,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,GCHK,CTRL,fHealth"), "ffffffffffffHHB",
LOG_FORMAT(EST3, "P12,P13,P14,P15,P16,P17,P18,P19,P20,P21,P22,P23,P24,P25,P26,P27"), "ffffffffffff",
LOG_FORMAT(EST4, "VxI,VyI,VzI,PxI,PyI,PzI,VxIV,VyIV,VzIV,PxIV,PyIV,PzIV"), "ffffffff",
LOG_FORMAT(EST5, "MAGxI,MAGyI,MAGzI,MAGxIV,MAGyIV,MAGzIV,HeadI,HeadIV,AirI,AirIV"), "ffffffff",
LOG_FORMAT(EST6, "fffff", "FxI,FyI,FxIV,FyIV,HAGLI,HAGLIV"),
LOG_FORMAT(PWR, "fffBBBBB", "Periph5V,Servo5V,RSSI,UsbOk,BrickOk,ServoOk,PeriphOC,HipwrOC"),
LOG_FORMAT(MOCP, "ffffff", "QuatW,QuatX,QuatY,QuatZ,X,Y,Z"),
LOG_FORMAT(VISN, "ffffffff", "X,Y,Z,VX,VY,VZ,QuatW,QuatX,QuatY,QuatZ"),
LOG_FORMAT(GS0A, "s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15"), "BBBBBBBBBBBBBBBB",
LOG_FORMAT(GS0B, "s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15"), "BBBBBBBBBBBBBBBB",
LOG_FORMAT(GS1A, "s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15"), "BBBBBBBBBBBBBBBB",
LOG_FORMAT(GS1B, "s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15"), "BBBBBBBBBBBBBBBB",
LOG_FORMAT(TECS, "ASP,AF,FSP,F,AsSP,AsF,AsDSP,AsD,EE,ERE,EDE,EDRE,PtchI,ThrI,M"), "ffffffffffffB",
LOG_FORMAT(WIND, "fff", "X,Y,CovX,CovY"),
LOG_FORMAT(ENCD, "qfqf", "cnt0,vel0,cnt1,vel1"),
LOG_FORMAT(TSYN, "Q", "TimeOffset"),
LOG_FORMAT(MACS, "fff", "RRint,PRint,YRint"),
LOG_FORMAT(CAMT, "QI", "timestamp,seq"),
LOG_FORMAT(RPL1, "t,gIdt,aIdt,Tm,Tb,gx,gy,gz,ax,ay,az,magX,magY,magZ,b_alt"), "QffQQffffffff",

```

```
LOG_FORMAT(RPL2, "QQLLiMMffffffM",  
"Tpos,Tvel,lat,lon,alt,fix,nsats,eph,epv,sacc,v,vN,vE,vD,v_val"),  
LOG_FORMAT(RPL3, "QffffIB", "Tflow,fx,fy,gx,gy,delT,qual"),  
LOG_FORMAT(RPL4, "Qf", "Trng,rng"),  
LOG_FORMAT(RPL5, "Qffffff", "Tev,x,y,z,q0,q1,q2,q3,posErr,angErr"),  
LOG_FORMAT(RPL6, "Qff", "Tasp,inAsp,trAsp"),  
LOG_FORMAT(LAND, "B", "Landed"),  
LOG_FORMAT(LOAD, "f", "CPU"),  
LOG_FORMAT(TIME, "Q", "StartTime"),  
LOG_FORMAT(VER, "NZ", "Arch,FwGit"),  
LOG_FORMAT(PARM, "Nf", "Name,Value"),  
LOG_FORMAT(AOAS, "ff", "AOA,SS")
```

VITA

Donald Joseph Lear
Master of Science Candidate, 2017
Old Dominion University
Mechanical and Aerospace Engineering
238 Kaufman Hall
Norfolk, VA 23529

Education

Bachelor of Science, Physics, University of North Carolina in Asheville, December 2010

Undergraduate Research

Replicated Young's Double Slit experiments in Experimental Physics, detailing the significant implications of light acting as both a particle and a wave.

Internship

NASA Wallops Island Flight Facility, Chincoteague Island, VA June-August 2010
Assigned to use MATLAB to create a program that would allow two orbiting satellites to communicate with each other so both could collect data using specific wavelengths of light to determine varying gas densities in Earth's atmosphere.

Presentations and Lectures

"Introduction to MATLAB and Real Life Implementation of the Program," Old Dominion University, Norfolk, VA, 15 Sept. 2014, Oceanography and Physics Bldg. and 8 Mar. 2015, Health and Sciences Bldg., Academic Presentation.
"Mapping Concentration of Gases in a Planet's Atmosphere Using Spectroscopy," Wallops Island Flight Facility, Chincoteague Island, VA, 5 Aug. 2010. Academic Presentation.
"Introduction to Projectile Motion," University of North Carolina Asheville, Asheville, NC 10 Apr. 2010, Rhodes-Robinson Hall. Academic Presentation.
"Implications of Young's Double Slit Experiment," University of North Carolina Asheville, Asheville, NC, 14 Nov. 2010. Academic Presentation.