12-2021

Airframe and Systems Design, Analysis, and Testing of the Horizon Morphing-Wing Aircraft

Sabrina A. Snow

Utah State University

AIRFRAME AND SYSTEMS DESIGN, ANALYSIS, AND TESTING OF THE HORIZON MORPHING-WING AIRCRAFT

by

Sabrina A Snow

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Douglas F. Hunsaker, Ph.D. Major Professor	Stephen A. Whitmore, Ph.D. Committee Member

Zhongquan Zheng, Ph.D.
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah

2021

ABSTRACT

Airframe and Systems Design, Analysis, and Testing of the Horizon Morphing-Wing Aircraft

by

Sabrina A Snow, Master of Science

Utah State University, 2021

Major Professor: Douglas F. Hunsaker, Ph.D.
Department: Mechanical and Aerospace Engineering

Morphing trailing edge technology can provide the ability to dynamically alter the twist distribution, and therefore lift distribution, of an aircraft during flight. There are certain optimal lift distributions which can be chosen to create proverse yawing effects and eliminate the need for vertical control surfaces. The purpose of this project is to support the design and testing of a morphing, crescent flying wing airframe which will be used to evaluate yaw control in an aircraft without vertical control surfaces. There are three main objectives of this project, which are to perform static and dynamic analysis on the crescent wing design, develop electronics capable of mapping three pilot inputs to eleven control surface outputs, and ultimately build and flight test the aircraft. This report details the completion of these objectives, the final design of the aircraft and internal systems, and the results from each flight test completed.

(83 pages)

PUBLIC ABSTRACT

Airframe and Systems Design, Analysis, and Testing of the Horizon Morphing-Wing Aircraft

Sabrina A Snow

The lift distribution on an aircraft describes how much upwards force, or lift, is being produced at each point along the wings of the aircraft. The lift at a specific point on the wing can be increased, or decreased, by 'twisting' that part of the wing. This is most easily done with the use of a control surface, such as an aileron. The control surfaces are used to create inputs that cause the aircraft to respond in pitch, roll, or yaw. In many aircraft, there are dedicated control surfaces for each of these three directions of motion. Ailerons predominantly control roll, elevator controls pitch, and rudder controls yaw. However, it has been shown that pitch, roll, and yaw could simultaneously be controlled through direct manipulation of the lift distribution, instead of independently through each control surface. A 'morphing' wing, or one capable of altering its twist distribution in flight and therefore altering its lift distribution, would be capable of doing this. By implementing this technique, it would be possible to eliminate the need for a vertical control surface, like the rudder. It could also help reduce drag on the aircraft by using certain lift distributions known to minimize induced drag. The purpose of this project is to support the design and testing of a morphing, flying crescent wing airframe which will be used to evaluate whether an aircraft is capable of controlling yaw without vertical control surfaces. The objectives which will contribute to this larger goal are to perform static and dynamic analysis on the crescent wing design, develop electronics capable of mapping three pilot inputs to eleven control surfaces, and ultimately build and flight test the aircraft. This report details the completion of these objectives, the final design of the aircraft and internal systems, and the results from each flight test completed.

ACKNOWLEDGMENTS

This work was funded by the U.S. Office of Naval Research Sea-Based Aviation program (Grant No. N00014-18-1- 2502) with Brian Holm-Hansen as the program officer.

Sabrina A. Snow

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

NOTATION

C_D	=	coefficient of drag
C_{D_0}	=	zeroth order drag term
C_{D_1}	=	first order drag term
C_{D_2}	=	second order drag term
C_L	=	coefficient of lift
c	=	chord
W	=	total weight of aircraft
$C_{L_{\max}}$	=	max coefficient of lift for aircraft
ρ	=	air density
S_w	=	main wing area
V_{stall}	=	stall velocity of the aircraft
V_{MD}	=	minimum drag velocity of the aircraft
V_{MDV}	=	minimum power velocity of the aircraft
$V_{\max}$	=	maximum velocity of the aircraft
b	=	wingspan
V_∞	=	freesteam velocity
$\tilde{L}$	=	section lift
θ	=	spanwise location using cosine clustering
D_i	=	induced drag
$R_{n/l}$	=	roll-yaw control ratio
C_{n,δ_a}	=	nondimensional change in yawing moment with respect to aileron deflection
C_{l,δ_a}	=	nondimensionalized change in rolling moment with respect to aileron deflection
R_A	=	aspect ratio
κ	=	correction factor
ω	=	local twist or washout
$\tilde{C}_{L,a}$	=	local section lift slope

c_{root} = root chord

α_{L_0} = zero lift angle of attack

C_{L_α} = lift slope

$C_{m_{L0}}$ = zero lift pitching moment

C_{m_α} = pitching moment slope

CHAPTER 1

INTRODUCTION

The section lift distribution on an aircraft can be manipulated to minimize drag, maximize efficiency, and achieve cooperative flight control through yaw control. Using classical lifting line theory [1, 2] the section lift distribution on an un-swept wing with no dihedral can be written as a Fourier series in the form

$$\tilde{L}(\theta) = 2b\rho V_\infty^2 \sum_{n=1}^{\infty} A_n \sin(n\theta) \tag{1.1}$$

where $\theta \equiv \cos^{-1}(-2z/b)$, b is the wingspan, and A_n are Fourier coefficients. The total induced drag in steady level flight is a function of the Fourier coefficients and can be written as

$$D_i = \frac{2(W/b)^2}{\pi\rho V_\infty^2}(1 + \sum_{n=2}^{\infty} nB_n^2);\ B_n \equiv A_n/A_1 \tag{1.2}$$

where B_n are normalized Fourier coefficients used to define the lift distribution. In 1933, under the constraints of fixed total lift and moment of inertia of lift, and considering structural constraints, Prandtl minimized this drag equation and discovered what is commonly referred to as the Bell Shaped Lift Distribution (BSLD) [3,4]. However, in this same paper, Prandtl included a disclaimer that while this was an insightful solution, it may not be the most realistic formulation of the problem, and that it did not provide a global minimum of induced drag for all wings.

Phillips et al. [5] relaxed Prandtl's 1933 constraints and followed a problem formulation that more closely modeled physical wing conditions. They found that Prandtl's BSLD is one of a family of lift distributions that minimize drag on un-swept, rectangular wings. This family of optimal lift distributions is characterized by $B_n = 0$ for all $n \neq 3$, and $B_3 \leq 0$, depending on design parameters. The elliptic distribution corresponds to $B_3 = 0$, and Prandtl's 1933 bell-shaped lift distribution corresponds to $B_3 = -1/3$. Taylor and Hunsaker [6] showed, using similar methods, that this family of optimal lift distributions is nearly independent of taper.

In addition to minimizing induced drag, the lift distribution on a wing can be deliberately chosen to control adverse yaw. Adverse yaw is induced when higher drag on one wing induces a yawing moment that opposes the direction of the desired flight path. Historically, aircraft have required vertical control surfaces in order to counter the effects of adverse yaw. However, these vertical control surfaces add drag and inefficiencies to the aircraft. Removing the need for this vertical control surface would be beneficial to a range of applications.

In 2016, using Prandtl's 1933 BSLD, Bowers et al. [7] showed that this may be possible. They not only eliminated adverse yaw, they showed the opposite effect, proverse yaw, could be induced. The BSLD produces upwash spanwise along the wing before the wingtips. Because of this, during roll, the induced drag produced due to aileron deflection is negative, or effectively thrust. This thrust on the wingtip induces a yawing moment with the same sign as the rolling moment, known as proverse yaw. Through flight testing, Bowers experimentally demonstrated that the BSLD can create proverse yaw. To expand on that discovery, Hunsaker et al. [8] has shown analytically that within the optimal family of lift distributions found by Phillips et al. [5], any lift distribution, except the elliptic distribution, can create proverse yaw with proper aileron design.

Within this family of distributions capable of producing proverse yaw, there is a single distribution which also minimizes induced drag for a given flight condition. For example, Fig. 1.1 shows an additional lift distribution (dashed) that can be added to the BSLD (gray) to produce zero adverse yaw. The total resulting lift is shown in black. Hunsaker et al. [8] defined a roll-yaw control ratio, $R_{n/l}$, which defines the level of lateral cooperative control produced. When this ratio is negative, adverse yaw is produced, but when it is positive, proverse yaw is achieved. The analytic relationship between this ratio, the lift distribution, and the flight condition for straight wings is given by

$$R_{n/l} \equiv \frac{C_{n,\delta_a}/C_L}{C_{l,\delta_a}} = -\frac{3 + B_3(5 + 7\kappa)}{\pi R_A} \tag{1.3}$$

where κ is the ratio of the fourth and second terms of the Fourier series solved as purely a function of aileron deflection.

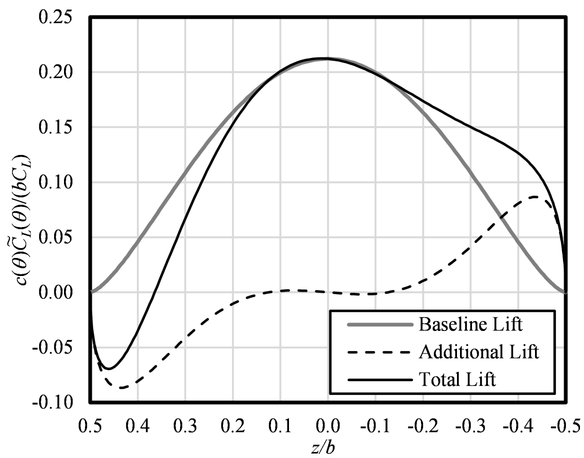


Fig. 1.1: Normalized local section lift along wing. Shows the additional lift distribution needed for the BSLD to produce zero adverse yaw.

Almost any lift distribution can be produced on a planform by manipulating the geometric or aerodynamic twist in the wing. [9] For the family of lift distributions described above, the relationship between chord, twist, and lift for straight wings is

$$\omega(\theta) = \frac{C_L}{\pi R_A}\left[\frac{4b(1-B_3)}{\tilde{C}_{L,\alpha}c_{\text{root}}} - 12B_3\right]\frac{\left(\frac{4b}{\tilde{C}_{L,\alpha}}\right)\left[\left(\frac{1-B_3}{c_{\text{root}}}\right) - \left(\frac{\sin(\theta)+B_3\sin(3\theta)}{c(\theta)}\right)\right] - 3B_3\left[1+\left(\frac{\sin(3\theta)}{\sin(\theta)}\right)\right]}{\left(\frac{4b(1-B_3)}{\tilde{C}_{L,\alpha}c_{\text{root}}}\right) - 12B_3} \tag{1.4}$$

As can be seen, the twist is a function of not only the chord distribution, $c(\theta)$, and desired lift distribution, B_3, but also depends on C_L. Therefore, twist is a function of flight condition.

This shows that for aircraft with a fixed amount of twist built into the wing, the desired lift distribution can only be achieved at a single flight condition. To expand this flight envelope, a wing must have variable twist capabilities. A morphing aircraft would have the unique capability of achieving a desired lift distribution over a large range of flight conditions by continuously altering the twist in the wing.

The Horizon Program is a collaborative effort to physically test and demonstrate this theory. The purpose of the Horizon Program is to evaluate whether an aircraft is capable of controlling yaw without vertical control surfaces. To accomplish this goal, a 3D printed, flying crescent wing with morphing trailing edge capabilities will be designed, built, and tested. The morphing technology implemented will be the final design presented in Moulton's paper [10]. The control algorithm for this aircraft is a complex algorithm and will follow a similar process to the one outlined in Montgomery's roll control paper [11].

The project discussed in this report contributes to the Horizon Program. The purpose of this project is to perform static and dynamic analysis on the crescent wing design, develop electronics capable of mapping three pilot inputs to eleven control surfaces, and ultimately build and flight test the Horizon aircraft.

CHAPTER 2

PROJECT OVERVIEW

There are three main objectives of this project. They are to

1. Design and analyze a morphing, crescent wing airframe,
2. Develop the internal systems of the aircraft including 3-to-11 channel mapping, and
3. Support flight testing of the aircraft and ensure successful data collection.

These objectives have been translated into three distinct stages. The first two are the design stages of the airframe and internal systems. The last is the physical testing stage. Each stage will be discussed in a dedicated chapter later in this document. These are broken down in the flowcharts shown in Fig. 2.1 and Fig. 2.2 below.

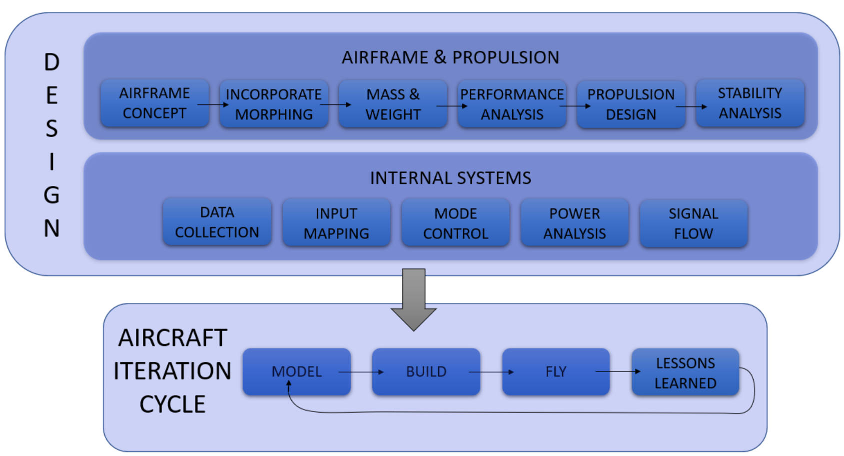


Fig. 2.1: Flow chart outlining the first two design stages as well as the Aircraft Iteration Cycle, which is repeated throughout the build and test stage.

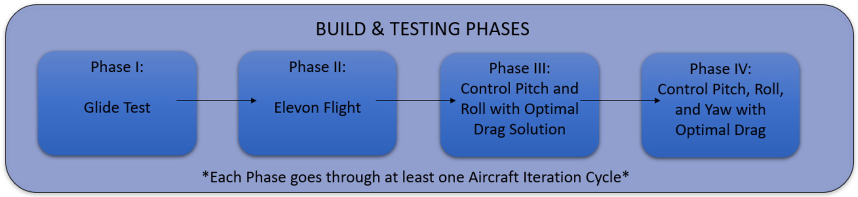


Fig. 2.2: Flow chart outlining the four different phases of the last stage of this project.

Fig. 2.1 outlines the two design stages and the main components involved with each. Fig. 2.2 outlines how the last stage is broken down into four phases. Each of those phases involves a complete Aircraft Iteration Cycle, shown in Fig. 2.1. The purpose of having four different phases is to sequentially test each level of complexity on the aircraft before building up to the final testing. For each of these phasese, there was a unique 'mode' that was developed to be the control algorithm for the aircraft. Montgomery developed each unique mode with a process similar to the one outlined in his roll control paper [11]. These phases and modes are discussed in greater detail below.

Phase I. The first phase of the physical testing of the Horizon aircraft is the glide test phase. The purpose of this phase is to validate the stability of the aircraft through pure gliding flight. This will be a piloted test with control via control surfaces, but it will not have any thrust or data collection onboard. The 'mode' implemented on this aircraft will be Mode 0, which treats each control surface as an elevon.

Phase II. The second phase of the physical testing of the Horizon aircraft is the elevon flight test phase. The purpose of this phase is to validate the aerodynamic stability and structural integrity of the aircraft through powered flight. This will be a piloted test with control via control surfaces and thrust via electric motors and ducted fans, but it will not have any data collection onboard. The 'mode' implemented on this aircraft will be Mode 1, which treats the control surfaces along each wing as one large elevon. Successful completion of this phase will boost confidence in progressing onto future phases which will have more data collection equipment onboard the aircraft.

Phase III. The third phase of the physical testing stage is the pitch and roll control phase. The purpose of this phase is to begin to implement control of the aircraft through modifying the lift distribution instead of traditional elevon deflection. The lift distribution chosen will be one that controls the desired pitch or roll command while minimizing drag based on the current flight condition. This will be a piloted test with data collection onboard. The 'mode' implemented on this aircraft will be Modes 2 and 3. Mode 2 controls pitch while implementing an optimal drag solution. Mode 3 controls both pitch and roll while still implementing the optimal drag lift distribution. This phase will be a critical step in the validation of the theory behind the control of this aircraft.

Phase IV. The final phase of the physical testing stage is the yaw control phase. The purpose of this phase is to control the aircraft in pitch, roll, and yaw through modifying the lift distribution while maintaining optimal drag solutions. This will involve piloted tests with data collection onboard. The 'mode' implemented on this aircraft will be Mode 4, which controls pitch, roll, and yaw while implementing an optimal drag solution. This final phase of testing will demonstrate the yaw authority of the aircraft when using the Mode 4 approach. Phase 4 may not be complete by the end of this project, however, the internal systems and data collection necessary for this phase will have been designed and tested in order for this to continue in future work.

CHAPTER 3

AIRFRAME AND PROPULSION DESIGN

This chapter outlines the design process and final results of the airframe and propulsion system design for the Horizon aircraft. This includes important reasoning and design decisions, the generated airframe design, and results from performance and stability analyses.

3.1 Airframe Design

The following section lays out the design of the airframe and key reasoning behind design choices. It also gives an overview of the morphing mechanism used and the important mass properties of the preliminary aircraft.

3.1.1 Geometry Definition

The geometry of the planform is important to multiple aspects of the design. The planform chosen has a high impact on the stability of the aircraft. The purpose of this aircraft is to provide a platform to test alternative ways of controlling yaw without the use of vertical control surfaces. A flying wing configuration was chosen, which removed the need for a traditional tail and the accompanying vertical surfaces. This aircraft was named Horizon, by which it will be referred to later in the text.

To maintain pitch stability, a flying wing aircraft must have swept wings. A traditional swept, flying wing aircraft with constant sweep, such as the B2, comes to a sharp point at the nose. This discontinuity in the geometry of the planform creates a decrease in the section lift at that point. This is known at the "middle effect". The Horten brothers [12] observed this effect and noted a loss of efficiency in the aircraft due to the decrease in lift at the center of the aircraft. Optimal lift distributions, such as the BSLD or the elliptic lift distribution, have a maximum in lift at the center. In order to minimize the middle effect and achieve closer to an optimal lift distribution, a crescent wing was chosen for Horizon. The crescent wing design implements linearly changing sweep. Most swept-wing aircraft implement constant sweep, where the wings are swept back at a constant angle from the root chord. Linear sweep instead has a linearly changing angle of sweep, where each wing section is at a different swept angle to the root chord, which results in a smooth curved geometry. This smooth geometry removes the step change in the geometry at the nose, and therefore also eliminates the typical dip in lift. There is also linearly changing dihedral in the main wing, which is defined similarly to linear sweep. The two points that define the sweep and dihedral angle definitions are outlined in Table 3.3 In addition to dihedral, winglets were added to Horizon to improve its baseline yaw stability. The winglets function as a vertical stabilizer and are analyzed as a separate 'wing' attached to the main wing. These winglets can be seen on the front view of Horizon in Fig. 3.1b.

The chord distribution of Horizon was created such that there is a longer (chord-wise) center section which smoothly transitions into the swept wings. The center section is a constant chord, which then transitions into a taper section, and ultimately flows into the wings. The chord distribution is outlined in Table 3.1. Values are linearly interpolated between except in the taper section of the aircraft. The equation defining the taper section of the chord distribution is

$$5061.67s^3 - 1800.34s^2 + 185.442s - 3.17 \tag{3.1}$$

where s is the percent span location along the y axis.

Table 3.1: Chord distribution of Horizon.

Section Name	**Percent Span**	**Chord** (ft)
Center	0.0	2.75
Center	7.561	2.75
Taper	7.561	Eq. 3.1
Taper	16.13	Eq. 3.1
Wing	16.13	1.1458
Wing	100	0.9167

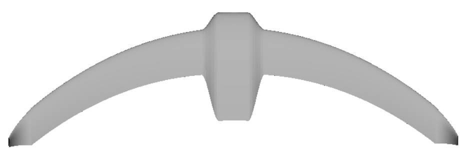
The center section of Horizon acts as a type of fuselage for the aircraft and provides a convenient location to carry the electronics and data collection payload. The only payload carried in the wings are the servos, which are used to deflect the control surfaces along the wing. There is a minimum chord length for which a servo can be placed in the wing and a morphing control surface can be feasibly manufactured. The dimensional chord scales with the overall size and span of the aircraft, so the wingspan was chosen such that the servos and associated control surfaces could be placed far enough outboard to ensure reasonable yaw control. The number of servos in each wing was determined by a control algorithm similar to the one discussed by Montgomery [11]. A top-view and front-view of the final Horizon planform design are shown in Fig. 3.1. The parameters defining most of the geometry are shown in Tables 3.2-3.3 below.

Table 3.2: Summary of important information describing the planform of Horizon.

Aspect Ratio, R_a	8
Wing Area, S_w (ft^2)	12.3
Wingspan, b (ft)	9.92
Average Chord, $\bar{c}$ (ft)	1.24

Table 3.3: Linearly changing sweep and dihedral definitions as a function of percentage along the span.

Percent Semi-Span	Sweep (deg)	Dihedral (deg)
0.0	0.0	0.0
1.0	40.5	10



(a) Top View of Horizon geometry showing linear sweep.

(b) Front View of Horizon geometry showing linear dihedral and winglets.

Fig. 3.1: Top and front views of the Horizon aircraft planform.

3.1.2 Morphing Mechanism and Structure

The morphing structure used to control the aircraft utilizes a fully 3D printed, plastic mechanism controlled through a push/pull servo mounted within the wing. Because the entire aircraft is 3D printed, the morphing mechanism is fully integrated into the aircraft design and is not a removable system. A CAD model of the cross section of the morphing mechanism is shown in Fig. 3.2. A printed wing section with the integrated morphing mechanism is shown in Fig. 3.3. For more information about the design, development, and internal mechanics of the morphing mechanism, see [10].

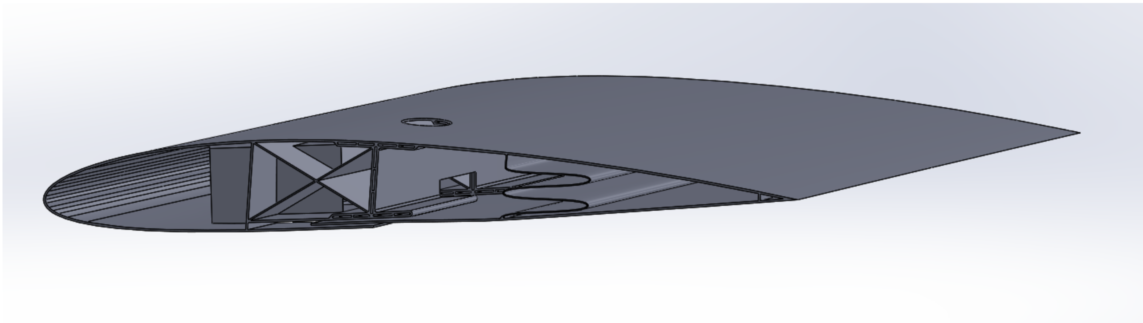


Fig. 3.2: Morphing wing section prototype created for the purpose of testing actuation. Shows how the morphing structure is integrated with the airframe.

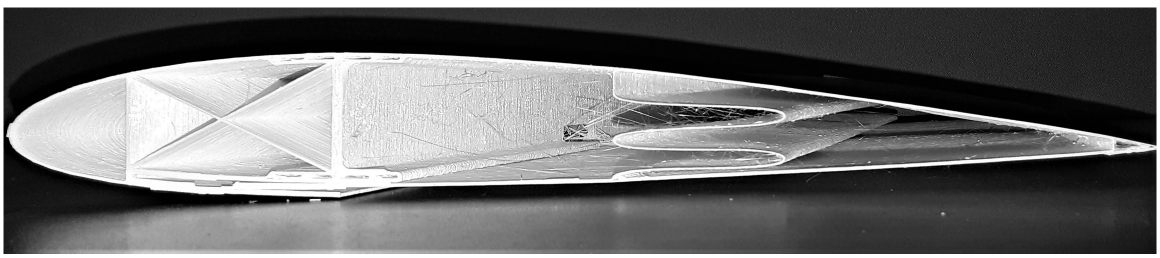


Fig. 3.3: Cross sectional view of the printed morphing wing section prototype which shows the internal workings of the morphing mechanism.

Because the morphing system is an integrated part of the aircraft, the mass properties of the morphing structure directly influence the mass properties of the aircraft. In order to approximate the mass properties of the entire airframe, two wing sections with morphing capabilities were printed and the resulting mass properties were measured. Specifically the weight, pseudo density, and location of the center of gravity (CG) were evaluated to create an estimate for these properties that could be interpolated and scaled by chord. The pseudo density is the weight of the wing section normalized by the equivalent area of the wing section, the spanwise length and the chord. This pseudo density allows the weight of any other wing section, with a known chord and spanwise length, to be approximated. The properties of the two morphing wing sections evaluated for these approximations are outlined in Table 3.4. These mass properties were later extrapolated to the entire aircraft, which will be explained in a later section.

Table 3.4: Mass properties of the morphing wing sections used for the airframe mass properties analysis.

	Section 1	**Section 2**
Airfoil	NACA 2412	NACA 2412
Chord (in)	12	33
Spanwise Length (in)	1	1
Weight (oz)	0.9347	2.6139
Equivalent Area (in^2)	12	33
Pseudo Density (oz/in^2)	0.07789	0.07921
CG location ($\%chord$)	45.38	45.43

3.1.3 Mass Properties with Morphing Structure

The mass properties of the morphing structure can be used to determine the overall mass properties of the entire airframe. This is done by approximating the aircraft as a series of wing sections each with a constant local chord. The weight of each wing section is obtained by linearly interpolating the 'pseudo density' relationship from the morphing prototype and scaling that by the chord and spanwise length of the section. The center of gravity of each wing section is approximated in y and z to be at the control points, and approximated in x as a certain percentage of the chord in the x direction. The percent chord x values for each section were also interpolated based on chord length. This resulted in a collection of discrete wing sections each with a respective weight and center of gravity location. The weight and CG estimate of the Horizon airframe without electronics is outlined in Table 3.5 below. A CAD model was also used to estimate the weight and CG of the airframe. The two approximations are compared in Table 3.5

Table 3.5: Discrete wing section approximation and CAD model estimates for the location of the center of gravity and weight for the Horizon airframe. All locations reported relative to the quarter chord point on the camber line of the center airfoil.

	$x(ft)$	$y(ft)$	$z(ft)$	Weight(lbs)
Wing Sections	-0.844	0.0	-0.129	9.0
CAD model	-0.828	0.0	-0.15	9.9

This airframe weight and CG approximation were then combined with the weights and CG locations of the electronics and payload Horizon would carry. With this information, an initial estimate for the center of gravity and the inertia tensor of Horizon with payload could be made. It was estimated that approximately 3 lbs of ballast would need to be added to the nose of the aircraft in order to maintain pitch stability. The following values were all done with this added 3 lbs of ballast.

The resulting approximations for the inertia tensor, location of the center of gravity, and weight are shown in Tables 3.6-3.7. A plot of the planform with the locations of important electronics and payload, as well as critical points such as the center of gravity and aerodynamic center, is shown in Fig. 3.4.

Table 3.6: The inertia tensor approximation for Horizon with payload. All units in $(slug * ft^2)$.

I_{xx}	0.322	I_{xy}	-0.0003
I_{yy}	0.283	I_{xz}	0.0342
I_{zz}	0.590	I_{yz}	-0.0001

Table 3.7: Approximated location of the center of gravity and weight for the Horizon aircraft. All locations reported relative to the quarter chord point on the camber line of the center airfoil.

$x(ft)$	$y(ft)$	$z(ft)$	Weight(lbs)
-0.348	0.003	-0.093	16.26

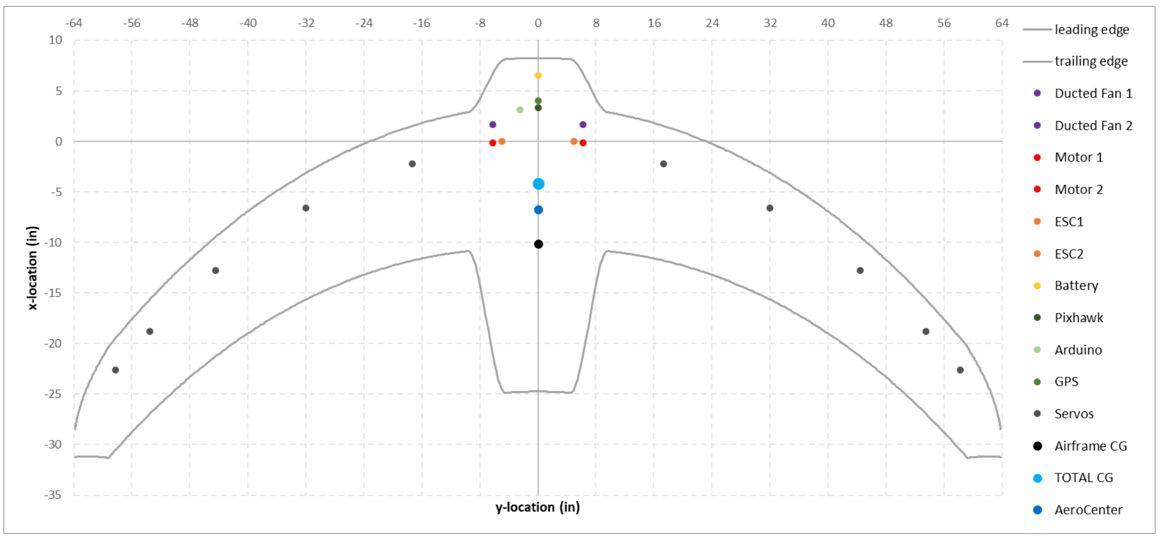


Fig. 3.4: Estimated layout of Horizon's electronics and payload. Important locations shown such as aerodynamic center and CG.

3.2 Propulsion System Design

The following section outlines the process involved in determining a reasonable propulsion system, as well as the final system chosen for the aircraft.

3.2.1 Preliminary Performance Analysis

It is necessary to understand the performance of the aircraft in order to predict the thrust and power required to maintain steady level flight. These are important parameters to understand when designing a propulsion system. In order to evaluate the performance of the aircraft, the aerodynamic analysis tool MachUpX[1] was used. MachUpX is an imple-

[1]https://github.com/usuaero/MachUpX

mentation of numerical lifting line theory with sweep corrections, which has been shown to be accurate for wings with sweep [13]. MUX requires a fully defined planform geometry, 2D airfoil properties, any control surfaces or control states, and the flight condition of the aircraft as inputs. It then returns nondimensional forces and moments acting on the aircraft in that flight condition. For all of the performance analysis, Horizon was approximated as a glider, so no thrust was considered in this preliminary estimation. All control surfaces and potential improvements from mode mapping were neglected and the pure glide performance of the airframe was analyzed.

The planform geometry for Horizon was defined as discussed earlier in this paper. For the initial performance and stability analysis of the aircraft, no control surfaces or control states were defined, as mentioned above. These were dealt with separately to determine the control mapping for the morphing aircraft. The flight condition for Horizon was set at a Reynold's number of 2e5. This Reynold's number was determined using the standard air properties at 5000 feet (approximately the elevation og Logan, UT, where Horizon will predominantly be tested at), the average chord defined above, and an arbitrarily chosen velocity of 50 ft/s. The airfoil selected for the Horizon aircraft was a NACA 2412 airfoil. The 2D airfoil properties for this airfoil were determined using data from airfoil tools [2]. This provides lift, drag, and pitching moment data over a range of angles of attack for the airfoil. A curve was then fit to this data to determine the 2D airfoil properties shown in Table 3.8 below. The curve fits for the lift and drag data are shown in Fig. 3.5.

[2] http://airfoiltools.com/polar/details?polar=xf-naca2412-il-200000-n5

Table 3.8: 2D Airfoil Properties for the NACA 2412 airfoil. All angles are in radians. These values were taken from curve fits to data from airfoil tools.

$\tilde{\alpha}_{L0}$	-0.0336
$\tilde{C}_{L_\alpha}$	6.0775
$\tilde{C}_{m_{L0}}$	-0.0473
$\tilde{C}_{m_\alpha}$	0.0506
$\tilde{C}_{D_0}$ (airfoil)	0.00974
$\tilde{C}_{D_1}$ (airfoil)	-0.00624
$\tilde{C}_{D_2}$ (airfoil)	0.01541
$\tilde{C}_{L_{\max}}$(airfoil)	1.2971

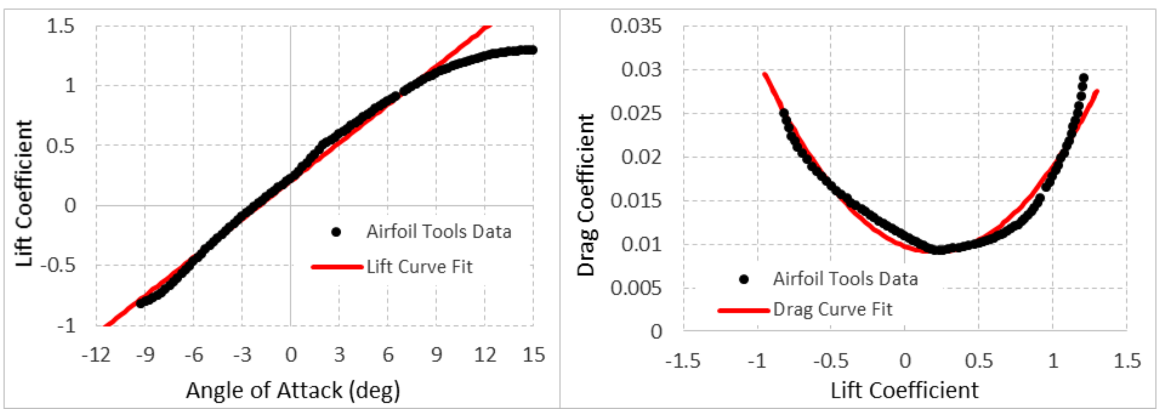


Fig. 3.5: Curve fits to the airfoil data from airfoil tools. The lift fit has an RMS value of 0.518, while the drag fit has an RMS value of 0.0112.

A useful measurement of the general efficiency and performance of an aircraft is the lift-to-drag ratio. Fig. 3.6 shows the lift-to-drag ratio of Horizon, as a function of the lift coefficient, resulting from this preliminary performance analysis. As can be seen in Fig. 3.6, there is an optimal lift coefficient at which the lift-to-drag ratio can be maximized. This maximum corresponds to the minimum in drag as a function of lift. Maximizing this ratio improves the overall efficiency of the aircraft. The results of this analysis show that there is an optimal lift coefficient, corresponding to an optimal speed, which Horizon can be designed for. As seen in the plot, Horizon is designed for a lift coefficient of 0.44.

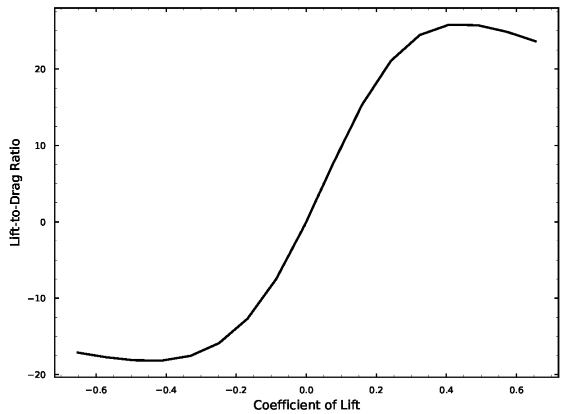


Fig. 3.6: Lift-to-Drag ratio as a function of coefficient of lift. A maximum can be seen at approximately $C_L = 0.44$.

It is also useful to examine the relationship between lift and drag an alternative way, where drag is a function of lift. To obtain this relationship, a quadratic curve fit was applied to the drag and lift data from MachUpX. The general form of the quadratic relationship between lift and drag is

$$C_D = C_{D_0} + C_1 C_L + C_{D_2} C_L^2 \tag{3.2}$$

Fig. 3.7 shows the raw data returned from MachUpX as well as the quadratic curve fit to the data. Table 3.9 shows the three drag coefficients required to define the curve fit shown in Fig. 3.7. The fit of the curve to the data has standard deviation in x of 3.05e−5 and in y of 1.09e−5. This uncertainty is well within the accuracy to which the data itself is known.

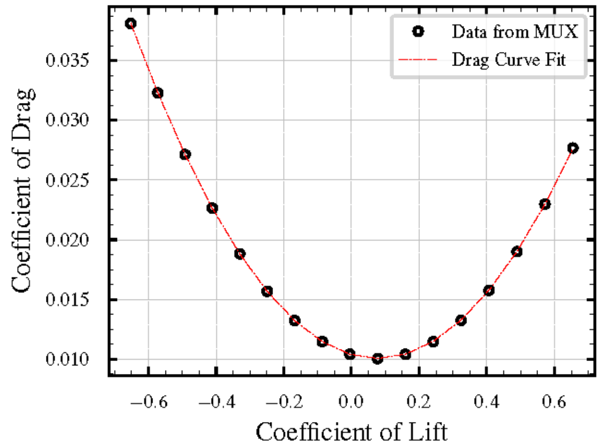


Fig. 3.7: C_D versus C_L plot to show the goodness of fit of the drag curve

Table 3.9: Drag coefficients which define Eq. 3.2 for the Horizon aircraft. Note that these are different than the drag coefficients for just the airfoil.

C_{D_0}	0.0104
C_{D_1}	-0.0081
C_{D_2}	0.0528

There are three important flight speeds that must be evaluated to help define the flight envelope. These include the stall velocity, the minimum drag velocity, the minimum power velocity, and the maximum velocity. These important flight speeds are calculated using the methods outlined in Mechanics of Flight by Warren Phillips [14]. The stall velocity is calculated as

$$V_{\text{stall}} = \sqrt{(2W/S_w)/(C_{L_{\max}}\rho)} \tag{3.3}$$

where $C_{L_{\max}}$ was estimated arbitrarily to be 1.2 for the entire aircraft. The minimum drag and power velocities are calculated as

$$V_{MD} = (\sqrt{2}/(C_{D_0}(1/C_{D_2})))^{1/4}\sqrt{(W/S_w)/\rho} \tag{3.4}$$

and

$$V_{MDV} = (2/\sqrt{(1/C_{D_2})C_{D_1} + \sqrt{((1/C_{D_2})C_{D_1})^2 + 12(1/C_{D_2})C_{D_0})})\sqrt{(W/S_w)/\rho} \tag{3.5}$$

It is important to note that some of these equations are dependent on certain air properties. All air properties were assumed to be the standard air properties at 5000 feet elevation, which is approximately the elevation Horizon will be flight tested at.

The results of these calculated velocities for Horizon, as well as the chosen maximum velocity, are outlined in Table 3.10. The maximum velocity was chosen arbitrarily to be approximately twice the minimum power velocity. As can be seen, the minimum drag velocity corresponds to the lift coefficient which also achieves the maximum lift-to-drag ratio in Fig. 3.6.

Table 3.10: Design velocities and corresponding lift coefficients for Horizon.

	Velocity (ft/s)	C_L
V_{stall}	32.80	1.2
V_{MD}	53.97	0.443
V_{MDV}	43.12	0.694
$V_{\max}$	80.0	0.202

It is worthy of note that these important velocities are also a function of the total weight of the aircraft. Fig. 3.8 below shows how much these may change as a function of weight. Most critical is the stall airspeed, which increases approximately 10 ft/s for an increase of

11 lbs in weight. This shows the importance of weighing the aircraft before every flight test and ensuring all calculations still hold.

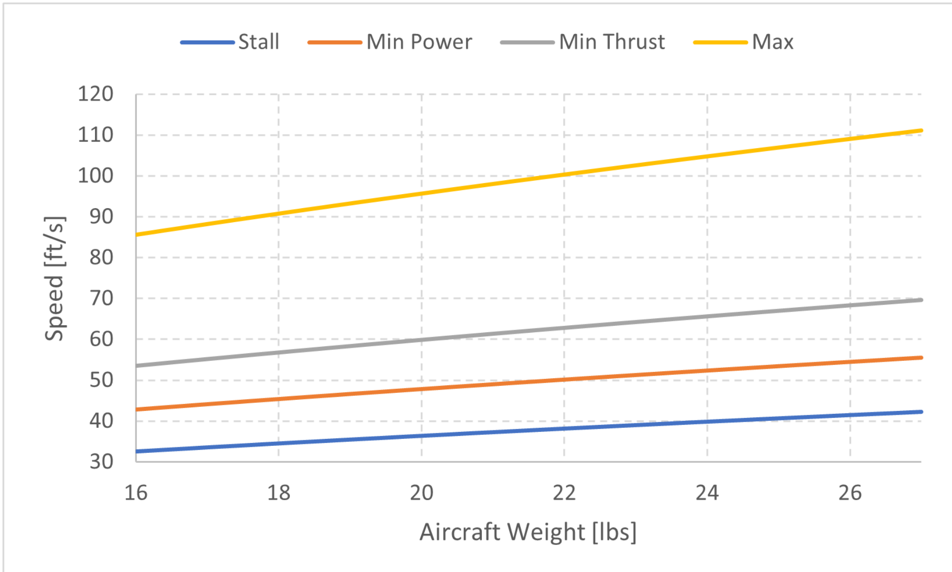


Fig. 3.8: Important velocities [ft/s] versus weight of the aircraft [lbs].

3.2.2 Propulsion System Specification

The propulsion system is a critical element to the performance of an aircraft. The purpose of Horizon is to demonstrate yaw control using a morphing trailing edge, as opposed to traditional vertical control surfaces. In order to fully test the yaw authority being exerted by the aircraft, it is helpful to be able to demonstrate an 'engine out' scenario, typical in aircraft testing. It is also desirable that the propulsion system has a minimal effect on the aerodynamics of the system, which would skew the yaw control approximations. To meet these requirements, the propulsion system configuration chosen was two ducted fans mounted on top of Horizon and mirrored across the centerline. Fig. 3.9 shows the section of Horizon with the ducted fan integrated into the airframe to minimize the aerodynamic impact of the fan.

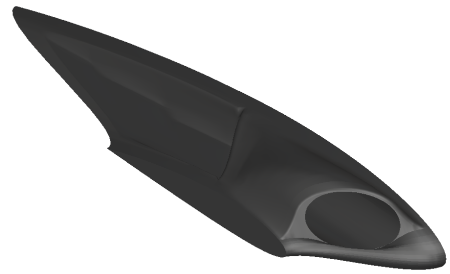


Fig. 3.9: Model of the ducted fan smoothly integrated with airframe. This section of the aircraft does not have a morphing control surface.

The propulsion system chosen, two ducted fans mirrored across the centerline, meets multiple requirements for the propulsion system. Primarily, this provides a way to test an ‘engine out’ scenario, by cutting power to one of the fans. The shroud for a ducted fan not only helps condition the air and improve propulsive efficiency, but it also provides a smooth transition from the airframe to the fan casing and back. The shroud helps minimize the extent to which the propulsion system disrupts the aerodynamics of the overall aircraft.

With the propulsion system configuration chosen, the specific system elements must be determined in order to obtain thrust and power available estimates. The thrust and power available must be greater than the thrust and power required at every flight speed within the flight envelope. To evaluate if this requirement is met, a dynamic thrust estimate must be determined. Most ducted fan manufacturers do not provide dynamic thrust data as part of the specifications of the systems. To estimate the dynamic thrust, the fan and motor combination was tested in a wind tunnel at different speeds to determine a dynamic thrust curve. The dynamic thrust was approximated to be linear with respect to velocity, and the linear fit to the dynamic thrust curve data was used to approximate the maximum thrust available. While there is a lot of uncertainty in this approximation, it provides a realistic preliminary estimate for what the available thrust of a propulsion system may be.

Using this dynamic thrust approximation, a propulsion system was chosen that met the requirements outlined above. The propulsion system chosen is outlined in Table 3.11. The approximation for thrust and power available, as well as the thrust and power required, are shown in Figs. 3.10-3.11 below. This approximation is for a single fan, showing that Horizon could be powered solely from one engine to demonstrate yaw control.

Table 3.11: Propulsion System chosen for the aircraft.

Fan	**Motor**	**Battery**	**ESC**
Hacker Motor 70mm, 9 blade	E40-S 2.5D	5S 6500 mAh LiPo	85 A

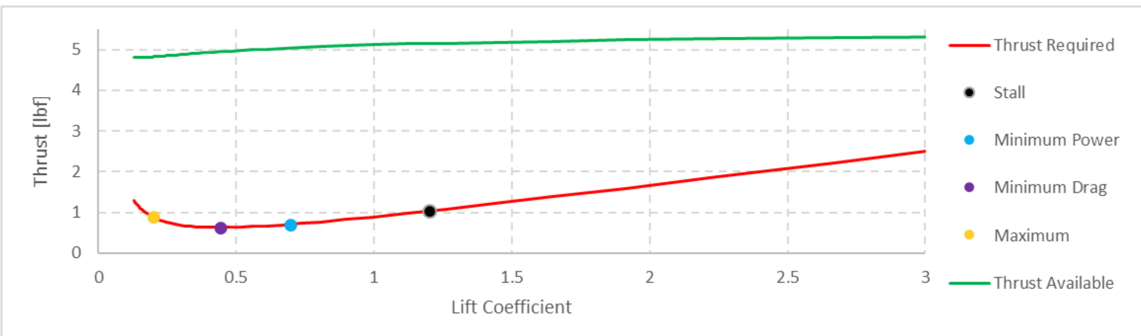


Fig. 3.10: Thrust required versus predicted thrust available for the chosen propulsion system with only a single fan and motor combo.

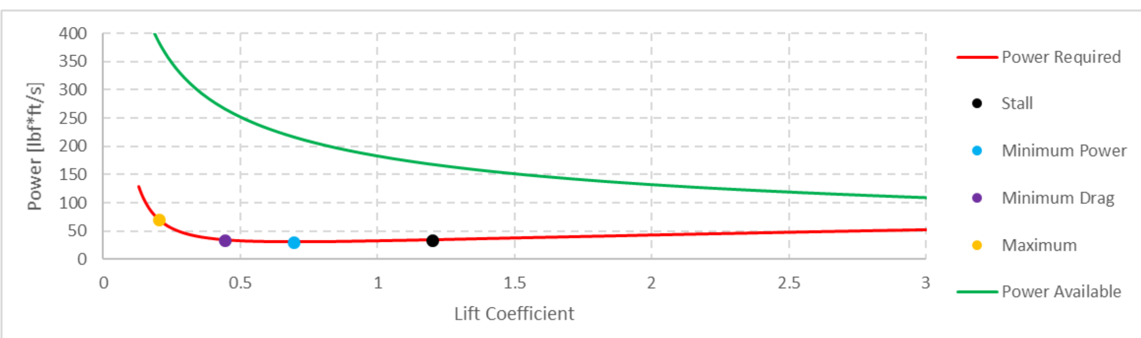


Fig. 3.11: Power required versus predicted power available for the propulsion system with a single fan and motor.

As can be seen in Figs. 3.10-3.11 the thrust and power available exceed the thrust and power required at every point within the flight envelope.

With the battery chosen for the system, if Horizon was flown at full throttle it would be expected to last about 5 minutes. However, a more realistic estimation for the available flight time is 15 minutes, which would be the approximate time if Horizon was flown at the minimum drag velocity.

3.3 Stability Analysis

This final section in the first design stage lays out how the preliminary stability analysis was performed for the Horizon aircraft. This stability analysis was later validated with a glide test in Phase 1 of the Horizon project.

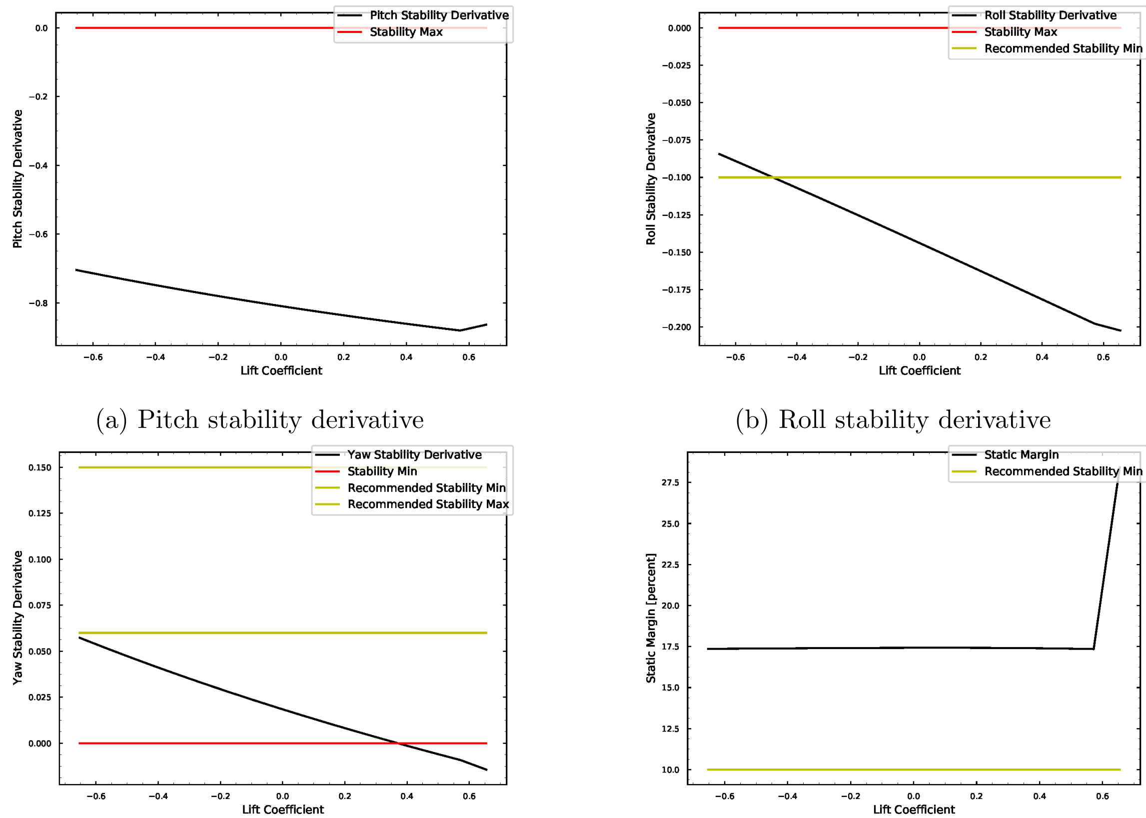
3.3.1 Static Stability Analysis

The first analysis done on the aerodynamic stability of the aircraft was static lateral and longitudinal stability. The stability derivatives, $C_{m,\alpha}$, $C_{l,\beta}$, and $C_{n,\beta}$ were examined over a range of lift coefficients to examine static stability over a range of flight conditions. The aerodynamic center and static margin were also examined over this range to determine if the payload loading of the aircraft needed to be altered to improve longitudinal stability. A summary of the requirements for static stability for each stability derivative as well as the static margin is shown in Table 3.12. These requirements were taken from Mechanics of Flight by Phillips [15, 16].

Table 3.12: Static Stability requirements and recommended ranges for each stability derivative and static margin.

Derivative	Static Stability Requirement	Recommended Range
Pitch Stability $C_{m,\alpha}$	$C_{m,\alpha} < 0$	< 0
Roll Stability $C_{l,\beta}$	$C_{l,\beta} < 0$	(-0.1, 0)
Yaw Stability $C_{n,\beta}$	$C_{n,\beta} > 0$	(0.06, 0.15)
Static Margin	approx. 5 percent	5-25 percent

The stability derivative values over the range of angles of attack for Horizon are shown in Figs. 3.12a-3.12d. As can be seen in these figures, Horizon is statically stable in pitch and roll over the entire flight envelope. The yaw stability derivative dips below the recommended minimum value, but it should not have a significant effect on the performance of the aircraft. It is just barely below the minimum requirement at the lift coefficient Horizon will be flown at. The static margin for Horizon seems to be within a reasonable range based on other flying wings.



(a) Pitch stability derivative

(b) Roll stability derivative

(c) Yaw stability derivative

(d) Static margin

Fig. 3.12: Static Stability results for the aircraft over a range of flight conditions.

It is not surprising that a flying wing with minimal vertical surfaces has some stability issues in yaw. Many of these could be dealt with through morphing control and additional yaw damping. However, it is desirable to try and mitigate as many problems as possible so that the control demonstration is truly focused on yaw authority, not maintaining yaw stability. For this reason, examining the stability is an important aspect of the design, but minor problems, like the ones seen here, can be dealt with in the trimming and testing phase instead of the design phase.

It has been observed on other flying wing aircraft that a static margin around 17 percent is necessary to keep the aircraft stable in pitch. Horizon has an average static margin of 17.5 percent in order to meet that previously found stability requirement.

3.3.2 Dynamic Stability Analysis

In addition to how it statically behaves, it is important to understand how the aircraft will dynamically react over the expected flight envelope. For this dynamic analysis, the traditional handling qualities were used to evaluate the aircraft. This may not be the best approximation because this is an RC aircraft controlled by a pilot outside of the aircraft itself, but it is a good first pass analysis. The method of analyzing each dynamic mode was taken from Phillip's Flight Mechanics Book [17] and the aircraft was analyzed as a Class B aircraft performing Flight Phase IV maneuvers [18].

In order to capture the entire flight envelope, and the important aspects of the flight, each dynamic mode was evaluated at four different flight conditions. The first flight condition was the stall speed of the aircraft in steady level flight. This helped capture roughly the configuration it will be in as it glides in to land as well as the condition it will be launched in. The aircraft was not designed with landing gear, so it is bungee launched. The next flight condition was the minimum drag velocity in steady level flight. This reflected what will most likely be the cruise condition of Horizon. The third flight condition was the minimum power in steady level flight, which provided another important point within the flight envelope. The last flight condition was the maximum velocity in steady level flight which provided a data point at the other extreme end of the flight envelope.

At each of these four flight conditions, the five dynamic modes of the aircraft were evaluated based on traditional handling qualities. The method of dynamic analysis used follows the method outlined in Phillips' textbook, Mechanics of Flight [17, 18]. This method uses a linearized system of equations based on Newton's second law. The linear equations are nondimensionalized and the longitudinal and lateral modes are solved for simultaneously in order to capture some of the effects of longitudinal and lateral coupling. The derivatives used to perform this analysis are outlined in Appendix A, with the specific values for Horizon included. The results of this dynamic analysis the important flight conditions discussed are summarized in Table 3.13 below.

Table 3.13: Summary of the handling quality levels for each dynamic mode over various flight conditions. Each flight condition is evaluated in steady level flight.

Mode	Stall	Min Drag	Min Power	Max
Short Period	1	1	1	1
Phugoid	3	2	2	2
Dutch Roll	1	4	4	4
Roll	1	1	1	1
Spiral	4	4	4	4

Horizon was designed for the purpose of being a test-bed to investigate yaw authority without the use of traditional vertical control surfaces. It is expected to perform well for that purpose, and it may be able to provide a platform for testing further applications of morphing wing technology. It has been shown that the flying wing configuration chosen for Horizon is statically stable over most of the flight envelope. The dynamic stability of the aircraft shows some potential issues, however, the parameters used to evaluate dynamic mode stability were developed using piloted aircraft. There is a large amount of research that must be done before that data can be accurately applied to a sub-scale, remote controlled aircraft such as Horizon. Because this aircraft is remote-controlled, the extreme care and detail that typically goes into the safety and stability of an airplane can be reduced and more of that time and effort can be directed towards physical testing. Testing will be a better indicator of the stability and performance of Horizon, since there is not much data on the handling qualities of sub-scale, remote controlled aircraft.

CHAPTER 4

ELECTRONIC SYSTEMS DESIGN

The second stage of this project was to design the internal systems of the Horizon aircraft. The internal systems of Horizon must be able to fulfill the following requirements including,

- to map 3-channel pilot inputs in pitch, roll, and yaw to the 11 servos powering the control surfaces on the aircraft and to make this programmable and adjustable depending on the Mode chosen,
- to provide thrust for the aircraft through an electric motor and ducted fan combination,
- to collect and store data during flight about the position, attitude, and flight condition of the aircraft,
- to collect video from on-board the aircraft of the wings and control surfaces during flight, and
- to transmit position, attitude, and flight condition data real-time back to a ground station.

The final design of the electronic systems on-board Horizon meets all of the requirements above. In order to better understand the internal workings of this system, a review of the control surfaces and electronics on-board Horizon is given below. The system is broken down into three main sub-systems.

1) The Control System. This includes the receiver, transmitter, and Teensy controller, as well as the servos which move each control surface. Horizon has 11 control surfaces, each powered by a dedicated servo. A top-view of Horizon is shown in 4.1 below, with each

control surface labeled. This control surface naming convention was used throughout the project. A Teensy controller (a derivative board of the Arduino) is also on-board the aircraft and is used to hold the mapping which controls the servos.

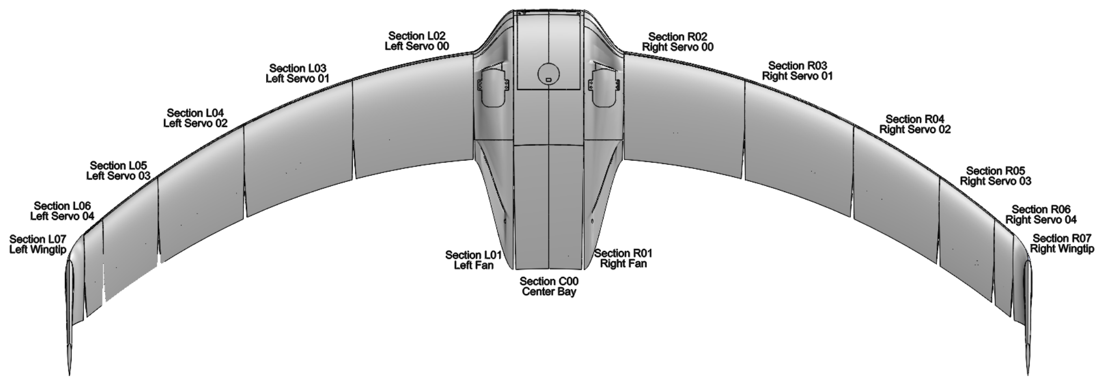


Fig. 4.1: Naming convention for each control surface of Horizon and the associated servo name.

The receiver used on Horizon is a DragonLink 433 MHZ micro receiver. This can function as solely a receiver or can double as a transmitter to send data back to a ground station. This dual purpose is why it was chosen. This communicates with the DragonLink transmitter on the ground, which is connected in dual-flight mode to a Spektrum 8e remote control, which is what the pilot uses to send inputs to the aircraft.

2) The Power System. This includes the built-in ducted fans, the motors, ESC's and BEC's, and the battery. Horizon utilizes a two engine setup, with two, 70mm ducted fans placed symmetrically across the x-axis of the aircraft. These are each connected to an electric motor which is controlled and powered through an 85A ESC. Both ESCs are wired in parallel to the battery. Each ESC also has an internal battery eliminator circuit (BEC), which is used to power the servos and other electronics on-board the aircraft, which require less power than the motors.

3) The Data Collection System. The other electronics on Horizon include the three cameras, another BEC to help increase power available to the servos, and a Pixhawk, which is used for data collection. Connected to the Pixhawk are various sensors including an IMU, a GPS, a pitot tube, and a battery harness connected in series with the battery to monitor voltage and current draw.

As was discussed in Chapter 2, there are four phases of the testing stage of the project. These correspond to three different internal systems setups which are

1. the Glide Setup,
2. the Traditional RC Setup, and
3. the Mode Testing Setup.

Each of these setups have a unique set of electronic components required to be on the aircraft and are therefore all wired differently. Each setup will be discussed in detail in the following sections with schematics showing how power is distributed throughout as well as how each component communicates with others.

4.1 Glide Setup

The Glide Setup involves the electronics used for Phase 1, the glide testing phase. The components included in this setup are 10 servos, a receiver and transmitter, and the battery and BEC to power the circuit.

The physical wiring of these components is shown in Fig. 4.2 below.

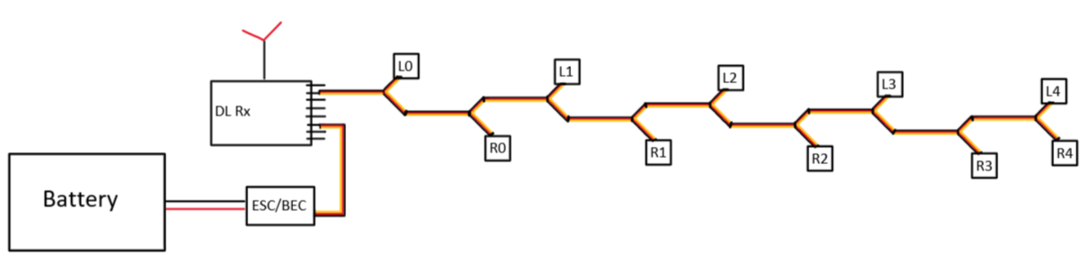


Fig. 4.2: Wiring diagram for the glide setup circuit.

As can be seen in Fig. 4.2, the servos are connected along water-falling y-cables, which results in them sharing a common voltage, ground, and signal. For this initial test, which was not planned to be more than a few seconds with minimal pilot control, this was a sufficient setup.

The power is distributed as shown in Fig. 4.3

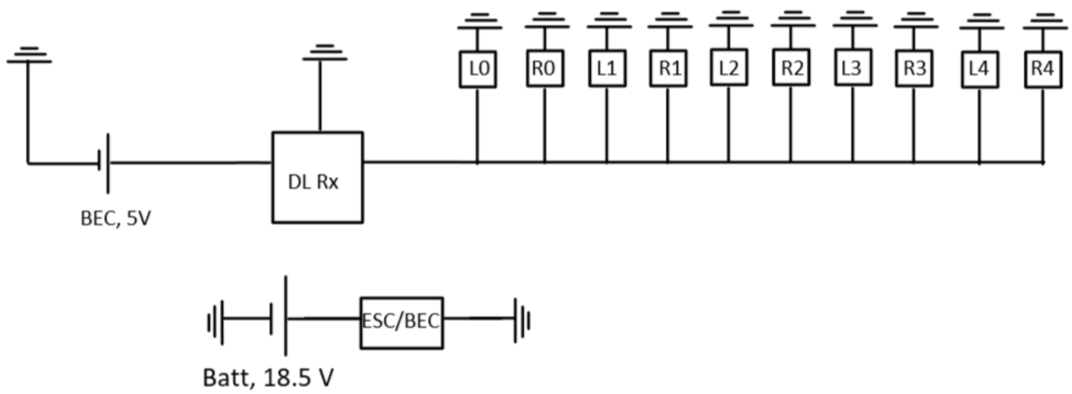


Fig. 4.3: Power distribution diagram for the glide setup circuit.

As can be seen, all power from the circuit flows through the Battery Eliminator Circuit (BEC), which drops the battery voltage down from 18.5V to 5V. The BEC used is rated for a current of 5A. Each servo used draws on average 400 mA when deflecting a control surface for this version. If all servos were simultaneously deflected it would draw approximately 4000 mA, or 4A, through the circuit. The BEC should be more than capable of handling this load. Each servo runs off of 5V, so the BEC wired in parallel with each servo also provides plenty of voltage to the crcuit.

The signal flow to control each element in this circuit and communicate between them is shown in Fig. 4.4

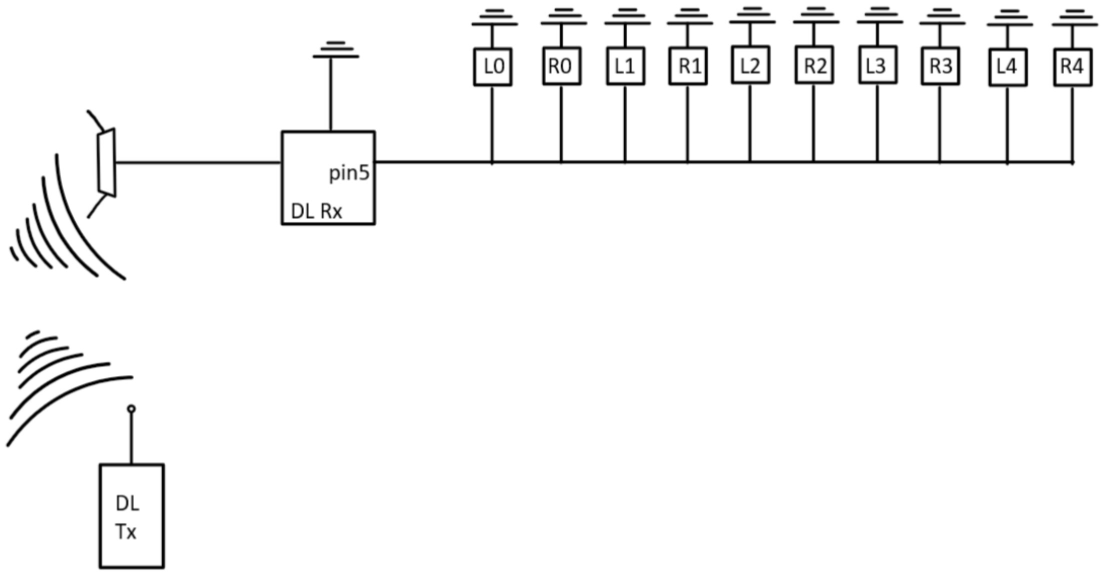


Fig. 4.4: Signal distribution diagram for the glide setup circuit.

Again, we can see that each servo is connected via a y-cable, so every servo sees the same input from the pilot. The transmitter was set up to control the aircraft in an elevon configuration, so each control surface was simultaneously controlled as an elevon.

4.2 RC Setup

The RC Setup involves the electronics used for Phase 2, the elevon flight testing phase. The components included in this setup are 11 servos, a receiver and transmitter, the Teensy controller, two motors and ducted fans, two ESCs, a camera, and the battery.

The physical wiring of these components is shown in Fig. 4.5 below.

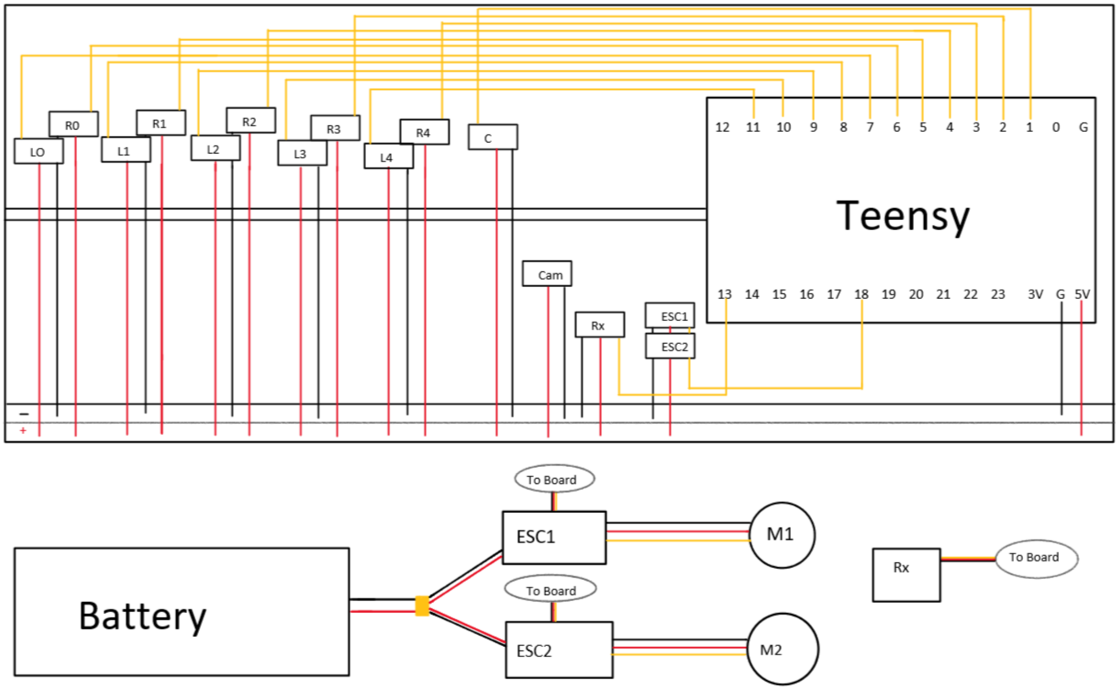


Fig. 4.5: Wiring diagram for the rc setup circuit.

This wiring setup is considerably more complicated than the previous y-cable configuration. This is also the first time that the Teensy was used on-board to control each servo individually. This was done because each control surface has a unique mode of deflection. Thus, a linear deflection mapping was created to map the degrees of deflection commanded to the actual degrees deflected. Because this increased level of difficulty was added in this stage, the wiring also became more complex and a breadboard was utilized to help send each Teensy command to the correct servo.

The power is distributed as shown in Fig. 4.6

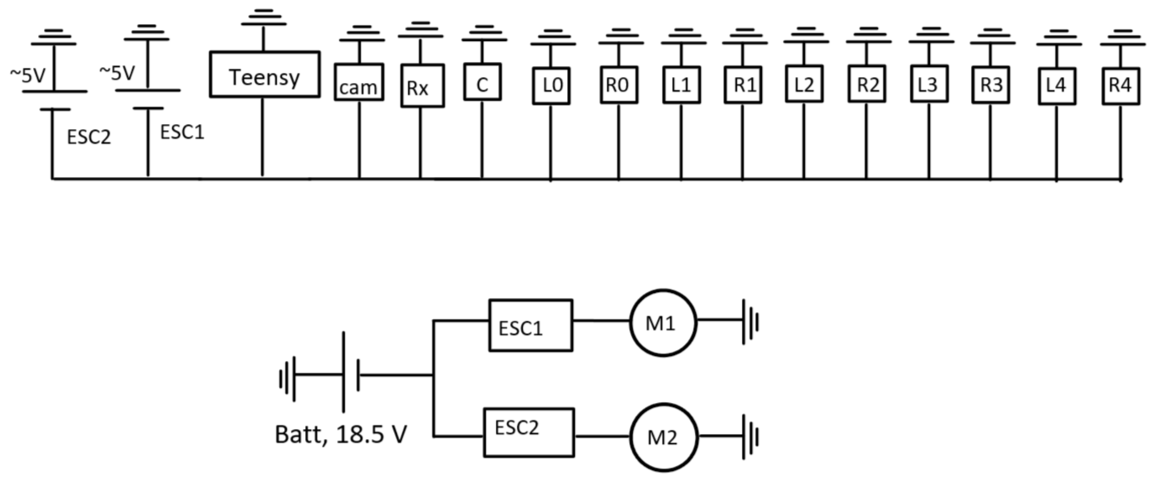


Fig. 4.6: Power distribution diagram for the rc setup circuit.

Similar to the glide setup, each component still shares a common voltage and ground. It is worth noting, that in this configuration the BEC's (from the ESC's) are wired together in parallel. This is actually not the best setup, because if one BEC's voltage is slightly different from the other, there may be some issues with the competing voltages. This did not cause issues during the flight test, but it was remedied in later setups.

In this setup, the motors add a significant load to the overall circuit. The max current an individual motor is expected to draw is 50A. Each ESC is rated for 85A, so the ESC's should be more than enough to handle the draw of the motor. Multiplied by 2, this would be a max current draw of 100A from just the motors alone. The other electronics also contribute a, somewhat smaller, current load. There are three different kinds of servos on Horizon. The center bay servo is a Hitec HS-645MG servo. This servo has a stall current draw of 2500mA, which would be the worst case. On average, this servo is expected to draw around 1200mA. The 4 inboard servos on each side, R3 through L3, are all Hitec HS-5245MG servos. These have a stall current of 1200mA, but are expected to draw roughly 500mA on average. Finally, the two outboard servos, R4 and L4, are Hitec HS-5065MG servos. These have a max stall current of 1200mA as well, but are expected to draw roughly 450mA on average. All servos combined are expected to draw roughly 6100mA on average, or 6.1A. With both ESC's wired in parallel, they can provide up to 10A, which should be plenty for the circuit including the load of the receiver, camera, and Teensy. The motors were expected to be very overpowered, so it was estimated that they would draw roughly 25 percent of their max load. Thus, the total average current draw during flight was expected to be 31.1A. With a 6500mAh battery, this gave an expected flight time of 12.54 minutes.

The signal flow to control each element in this circuit and communicate between them is shown in Fig. 4.7

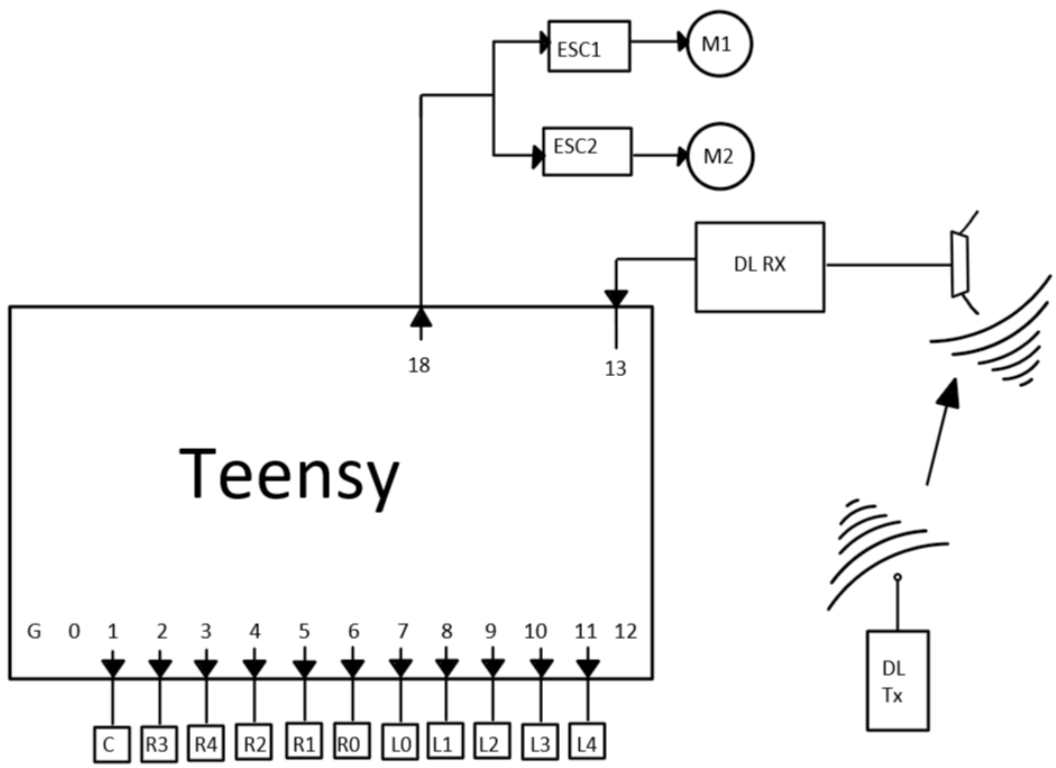


Fig. 4.7: Signal distribution diagram for the rc setup circuit.

As was mentioned above, each servo, and therefore control surface, has its own unique signal output from the Teensy. The Teensy mapped these control values using the code attached in Appendix B using the V3RC code. This was developed using both the degree mapping specific to that airframe as well as Mode 1 control mapping, which essentially treats the control surfaces along the wing as one giant elevon.

4.3 Modes Setup

The Modes Setup involves the electronics used for Phases 3 and 4, the mode testing flights phase. The components included in this setup are 11 servos, a receiver and transmitter, the Teensy controller, two motors and ducted fans, two ESCs, a Pixhawk data collection unit with imu, a pitot tube, a GPS, added telemetry wiring, 3 cameras, and the battery.

The physical wiring of these components is shown in Fig. 4.8 below.

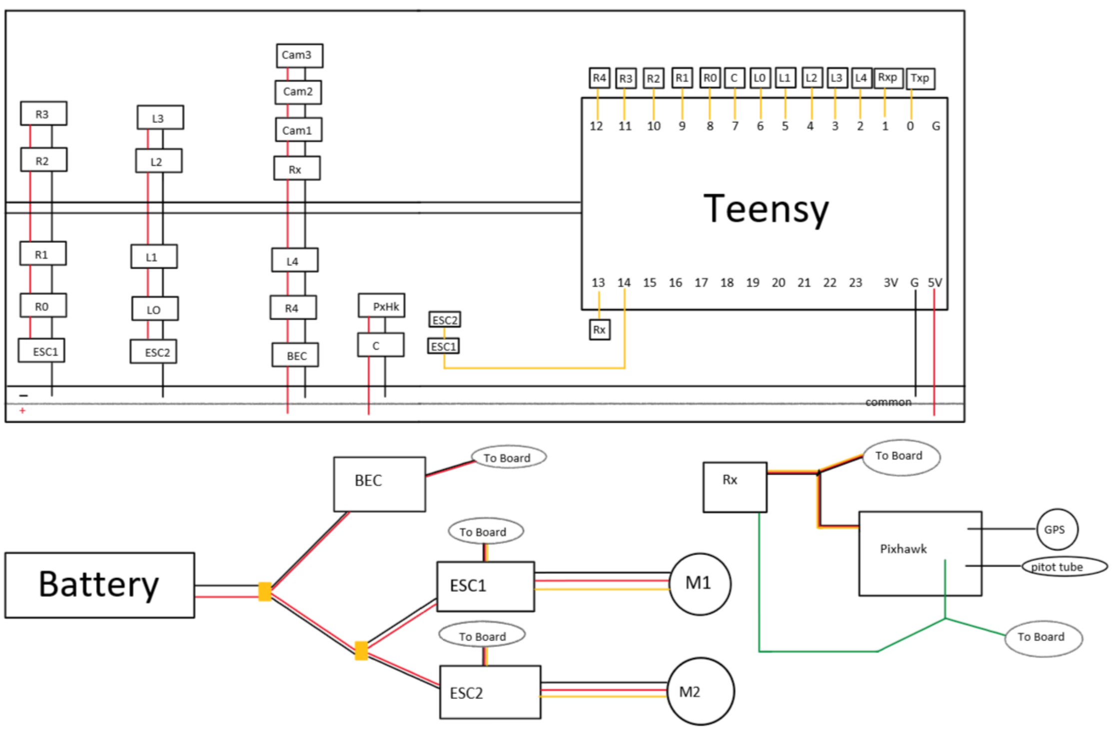


Fig. 4.8: Wiring diagram for the mode setup circuit.

This wiring setup is very similar to the one used for the RC setup. Some important differences are the addition of the Pixhawk and data collection equipment, the addition of the extra BEC to power the circuit, and the separation of the ESC voltages. The PPM signal from the receiver is also split out to both the Teensy and the Pixhawk in order to have a way of recording the pilot inputs during the flight.

The power is distributed as shown in Fig. 4.9

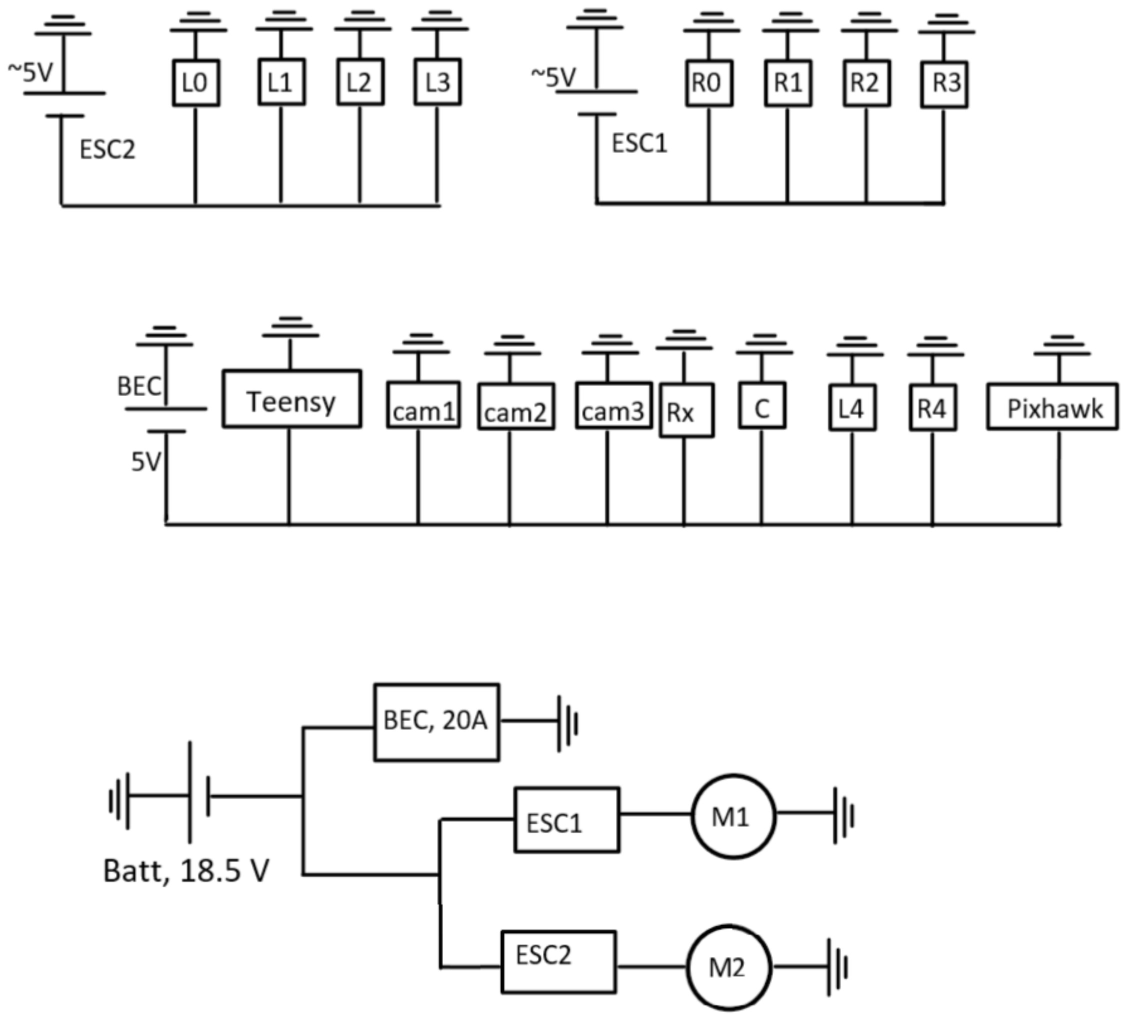


Fig. 4.9: Power distribution diagram for the mode setup circuit.

As mentioned above, one important change made to this circuit is that each BEC has its own dedicated circuit. This decreased the chances of any potential errors due to misaligned voltages. Both ESC circuits powered 4 HS-5245MG servos. The stall current of these servos is 1200mA, which means that if every servo was stalled it would put a 4.8A load on the circuit. The BEC in each ESC is rated for 5A, so both circuits were equipped to handle a worst-case scenario. Similarly, the third BEC circuit had an HS-645MG servo and two HS-5065MG servos. The stall currents for these are 2500mA and 1200mA respectively. If all three servos were simultaneously stalled it would put a load of 4.9A on the circuit. This would be in addition to the load of the three cameras, receiver, Teensy, Pixhawk, and other data collection equipment. The BEC powering this circuit is rated for 20A, so it should have more than enough capability of handling everything on the circuit.

Lastly, the motors were the same as used in previous flights. They had a max expected current draw of 100A. Combined with the maximum possible draw from each circuit, there was a maximum current draw of approximately 115A. The battery has a 150C rating with 6500mAh, so it can discharge up to 950A continuously. Thus, the battery is well equipped to power this load. With a 6500mAh capacity, at 115A this would give an approximate flight time of 3.3 minutes. However, it is unlikely that the aircraft would be at max current draw for the entire flight. A more relaisitic approximation would be a draw of approximately 60A, with the motors at about 50 percent, which would give a flight time of closer to 6.5 minutes.

Originally, it was thought that the motors would be very over-powered in terms of thrust capabilities, so it was expected that they would be run on average at about 25 percent throttle. This would have resulted in a current draw of roughly 30A, and a resulting flight time of about 13 minutes. Those predictions are what the battery was originally chosen for. However, previous flight tests demonstrated that there was not as much excess thrust as predicted, so a flight time of 3-6 minutes is a more accurate prediction for the life of the battery.

The signal flow to control each element in this circuit and communicate between them is shown in Fig. 4.10

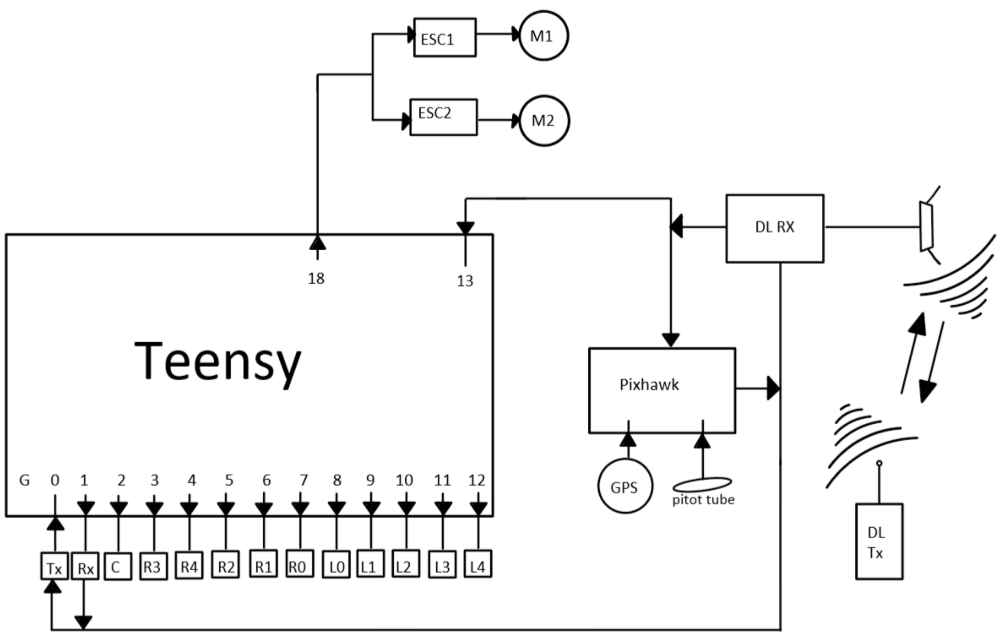


Fig. 4.10: Signal distribution diagram for the mode setup circuit.

Similar to the RC setup, this setup uniquely controls each servo and control surface, however, it now employs a modal mapping to do this. Each mode will be progressively tested in a series of flight tests. The algorithm for each mode was developed by Montgomery, similar to what was laid out in his roll control paper [11]. The modal mapping relies on the inputs from the sensors on-board, which is why there is a signal wire from the Pixhawk to the Teensy controller. This sends current flight condition information collected on the Pixhawk back to the Teensy to be used in the modal calculations. This telemetry signal is also split and sent back to the receiver, which transmits it down to a ground station as well.

This setup not only allowed for testing of the mode control, but also provided significantly more data for troubleshooting flight tests, in the event that something went wrong. This final Mode Setup meets all the requirements discussed earlier. It maps 3 pilot inputs to 11 control surface outputs, it has motors and ducted fans, which provide propulsion for the aircraft, it collects, stores, and transmits current flight condition data, and it has three on-board cameras which video the wings during flight.

CHAPTER 5

PHYSICAL TESTING PHASES

The final stage of this aircraft was the building and flight testing of the Horizon aircraft. Flight testing is critical to the validation of this project. One of the main goals of this project was to produce an airframe to be flight tested. The Horizon Program has completed multiple flight tests and will likely continue to test after this project has been completed. As of the most recent few flight tests, all objectives of this project have been met. A summary of each flight test is shown in Table 5.1. Each test is discussed in greater detail in this chapter.

Table 5.1: Summary of the Flight Tests of Horizon.

Test Designation	**Date**	**Phase of Aircraft**	**Goal of Testing**
V2G	15 April 2021	Phase I	Airframe Stability
V3RC	3 May 2021	Phase II	Traditional RC Control
V4RC	29 June 2021	Phase II	Structural Integrity
V6M2	7 October 2021	Phase III	Modes 2 and 3
V7M2	4 November 2021	Phase III	Launching

5.1 V2G: Version 2 Glide Phase

The very first iteration of the Horizon aircraft was dedicated to developing the build process. The second iteration was built for the glide test in Phase I. The goal of this test was to demonstrate the airframe was designed such that it was stable in a trimmed glide without any propulsion. The aircraft was launched with a bungee mechanism, and flew relatively stable for the duration of the flight. This was seen as a successful test and gave the team confidence that the aiframe design was stable.

5.2 V3RC: Version 3 RC Phase

The third iteration of Horizon was the first powered flight of the aircraft. This was the first flight in Phase II, the RC powered flight test phase. The goal of this test was to demonstrate that the aircraft could be controlled like a traditional flying wing aircraft with elevons. This test had a structural failure mid-flight, so changes had to be made to strengthen the airframe.

5.3 V4RC: Version 4 RC Phase

The version 4 iteration made structural changes to the aircraft in order to strengthen it and prevent the failure seen in the pervious test. This change was successful and the Phase II flight was repeated. The aircraft flew well, demonstrating controllability in a traditional RC configuration, but some connection issues at the end of the flight led to it ultimately crashing. However, this was seen as a success for the RC Phase, and progress continued onto Phase III.

5.4 V6M2: Version 6 Mode 2 Phase

The fifth iteration of Horizon used the Mode Electronics Setup and had a few structural changes to improve the structure and parabolic defelction of the control surfaces. However, the aircraft was broken in transit, so this iteration was not flown. For the sixth iteration, some structural improvements were made to help prevent accidents like that occurring in the future. The V6M2 flight was the first to have full data collection capability on board the aircraft, as well as a ground station. This flight was extremely short, as the aircraft stalled immediately off launch, so not many conclusions could be drawn about the success of the Mode 2 mapping. The biggest takeaway from this test was the need to improve the repeatability and reliability of the bungee launching mechanism.

5.5 V7M2: Version 7 Mode 2 Phase

The most recent version of Horizon was similar to the previous, with just a few adjustments made to the structure to improve the deflection curve of the control surfaces, and to improve the load capability of the launching hook. This flight test still had all data collection equipment on-board during the flight, and Mode 2 was put on the Teensy as an option during flight. The launching mechanism was changed from the last flight in order to increase the force it could launch the aircraft and improve the repeatbility with which it was launched. The aircraft launched well and flew for approximately 2.5 minutes in Mode 1 until coming down in a field near the landing zone. The reason for the crash is still being investigated. While this was a successful flight test demonstrating that the systems all worked well, this flight never validated the Mode 2 mapping as intended.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This project has successfully completed all of the original objectives. These were to perform static and dynamic analysis on a crescent wing design, develop electronic systems capable of mapping three pilot inputs to eleven control surface outputs, and to build and flight test the aircraft. All three of these were discussed in detail in the previous chapters.

The project discussed in this report is just a portion of the larger Horizon Program. The theory which provides a basis for this project has already been analytically derived. This project developed a stable airframe and designed the internal systems needed to create a test bed to validate this theory. Multiple flight tests have already been completed, and a standard process has been established to help future flight tests go smoothly.

This project is a starting point for a great amount of potential future research. The next steps on this project would be to continue to iterate and improve the Horizon aircraft and successfully flight test it in all modes. From these flights, data could be collected and then interpreted in order to ultimately be mapped back to fundamental aerodynamic parameters. These parameters could then be compared to the theory presented in the beginning of this report. From the comparison of this physical testing data to analytic derivations, a method could be developed for quantifying the yaw authority of the aircraft in each mode. This could then ultimately lead to the validation of the theory, as well as a platform to develop future aircraft with similar capabilities.

The Horizon platform holds more than just the ability to control yaw without vertical control surfaces. Because the minimization of drag is implicitly included in each mode, it also presents a potential way to increase the efficiency of aircraft on a large scale. This could lead to major improvements in many levels of the aerospace industry. It has been a pleasure to be a part of such a project.

REFERENCES

[1] Prandtl, L., "Tragflugel Theorie," *Nachricten von der Gesellschaft der Wissenschaften zu Gottingen*, 1918, pp. 451–477.

[2] Prandtl, L., "Applications of Modern Hydrodynamics to Aeronautics," *NACA TR-116*, June 1921.

[3] Prandtl, L., "Uber Tragflugel kleinstein induzierten Widerstandes," *Zeitschrift fur Flugtechnik und Motorluftschiffahrt*, Vol. 24, No. 11, 1933, pp. 305,306.

[4] Hunsaker, D. F. and Phillips, W. F., "Ludwig Prandtl's 1933 Paper Concerning Wings for Minimum Induced Drag, Translation and Commentary," *AIAA Scitech 2020 Forum*, Jan 2020, p. 1.

[5] Phillips, W. F., Hunsaker, D. F., and Joo, J. J., "Minimizing Induced Drag with Lift Distribution and Wingspan," *Journal of Aircraft*, Vol. 56, March 2019, pp. 431–441.

[6] Taylor, J. D. and Hunsaker, D. F., "Minimum Induced Drag for Tapered Wings Including Structural Constraints," *AIAA Scitech 2020 Forum*, Jan 2020, p. 1.

[7] Bowers, A. H., Jensen, R., Eslinger, B., Murillo, O. J., and Gelzer, C., "On Wings of the Minimum Induced Drag: Spanload Implications for Aircraft and Birds," *NASA Technical Publication*, Mar 2016, p. 1.

[8] Hunsaker, D. F., Montgomery, Z. S., and Joo, J. J., "Control of Adverse Yaw During Roll for a Class of Optimal Lift Distributions," *AIAA Scitech 2020 Forum*, Jan 2020, p. 6.

[9] Phillips, W. F. and Hunsaker, D. F., "Designing Wing Twist or Planform Distributions for Specified Lift Distributions," *Journal of Aircraft*, Vol. 56, Mar 2019, pp. 847–849.

[10] Moulton, B., "3D-Printed Wings with Morphing Trailing-Edge Technology," *AIAA SciTech 2021 Forum*, AIAA, 2021, p. all.

[11] Montgomery, Z. S., Hunsaker, D. F., and Joo, J. J., "A Methodology for Roll Control of Morphing Aircraft," *AIAA Scitech 2019 Forum*, Jan 2019, p. all.

[12] Horten, R., Selinger, P., and (translator), J. S., *Nurflugel: The story of Horten flying wings*, Weishapt Verlag, Graz, Austria, 1985.

[13] Goates, C., "A Practical Implementation of General Numerical Lifting-Line Theory," *AIAA SciTech 2021 Forum*, AIAA, 2021, p. all.

[14] Phillips, W. F., "Aircraft Performance," *Mechanics of Flight*, chap. 3, John Wiley and Sons, Inc., 2nd ed., 2010, pp. 264,272,317–318.

[15] Phillips, W. F., "Longitudinal Static Stability and Trim," *Mechanics of Flight*, chap. 4, John Wiley and Sons, Inc., 2nd ed., 2010, pp. 383,402.

[16] Phillips, W. F., "Lateral Static Stability and Trim," *Mechanics of Flight*, chap. 5, John Wiley and Sons, Inc., 2nd ed., 2010, pp. 500,549.

[17] Phillips, W. F., "Longitudinal-Lateral Coupling," *Mechanics of Flight*, chap. 9, John Wiley and Sons, Inc., 2nd ed., 2010, pp. 925–939.

[18] Phillips, W. F., "Aircraft Handling Qualities and Control Response," *Mechanics of Flight*, chap. 10, John Wiley and Sons, Inc., 2nd ed., 2010, pp. 956–957,962–966.

APPENDICES

APPENDIX A

DERIVATIVES USED FOR DYNAMIC STABILITY

NOTE: In this nomenclature, a comma between subscripts indicates a derivative of the first subscript with respect to the second.

Derivative of	**With Respect To**	**Symbol**	**Value**
Δ Lift	Mach Number	$C_{L,M}$	0.0
Δ Drag	Mach Number	$C_{D,M}$	0.0
Δ Pitching Moment	Mach Number	$C_{m,M}$	0.0
Δ Lift	Angle of Attack	$C_{L,\alpha}$	4.7076
Δ Rolling Moment	Angle of Attack	$C_{l,\alpha}$	0.0012
Δ Pitching Moment	Angle of Attack	$C_{m,\alpha}$	-0.8312
Δ Yawing Moment	Angle of Attack	$C_{n,\alpha}$	3.0807e-05
Δ Lift	Dimensionless $\partial/\partial t$ of α	$C_{L,\hat{\alpha}}$	0.0
Δ Drag	Dimensionless $\partial/\partial t$ of α	$C_{D,\hat{\alpha}}$	0.0
Δ Pitching Moment	Dimensionless $\partial/\partial t$ of α	$C_{m,\hat{\alpha}}$	0.0
Δ Lift	Forward (x) Velocity	$C_{L,\hat{u}}$	0.0
Δ Drag	Forward (x) Velocity	$C_{D,\hat{u}}$	0.0
Δ Pitching Moment	Forward (x) Velocity	$C_{m,\hat{u}}$	0.0
Δ Side Force	Sideslip Angle	$C_{Y,\beta}$	-0.1065
Δ Rolling Moment	Sideslip Angle	$C_{l,\beta}$	-0.1589
Δ Pitching Moment	Sideslip Angle	$C_{m,\beta}$	-9.5415e-15
Δ Yawing Moment	Sideslip Angle	$C_{n,\beta}$	0.0102
Δ Side Force	Rolling Rate	$C_{Y,p}$	-0.2554
Δ Rolling Moment	Rolling Rate	$C_{l,p}$	-0.5929
Δ Yawing Moment	Rolling Rate	$C_{n,p}$	0.0109
Δ Lift	Pitching Rate	$C_{L,q}$	1.1745
Δ Drag	Pitching Rate	$C_{D,q}$	-0.0518
Δ Pitching Moment	Pitching Rate	$C_{m,q}$	-1.7774
Δ Side Force	Yawing Rate	$C_{Y,r}$	0.0498
Δ Rolling Moment	Yawing Rate	$C_{l,r}$	0.0747
Δ Yawing Moment	Yawing Rate	$C_{n,r}$	-0.0099

APPENDIX B

TEENSY CODE USED FOR FLIGHT TESTS

V3 RC FLIGHT TEST ON MAY 3RD

```
/*
Code Outline
1) Read in PPM signal from reciever
2) Interpret PPM signal as separate PWM components
  - Throttle [Thr] --> PASS THROUGH
  - Aileron  [Ail]
  - Elevator [Elev]
  - Rudder   [Rud] --> NOT USED
3) Map Ail & Elev inputs into the corresponding 11 servo outputs
  - see mode1 function
  - [SL0] [SL1] [SL2] [SL3] [SL4] [SC] [SR0] [SR1] [SR2] [SR3] [SR4]
5) Convert servo deflections (degrees) back into PWM signals
6) output the throttle & 11 servo commands as individual PWM signals
*/

/*
 * NOTES:
 * PWM values treated as integers
 * degree values treated as doubles
 *
 * CHECK THE DEGREE TO PWM MAPPING FUNCTION
 */

// Servo Library -- to control each servo/ send out individual PWM signals
#include "Servo.h"
// PulsePosition Library -- to read in / interpret the PPM signal
#include <PulsePosition.h>
// Math library may not be needed for Mode 1??
//#include <math.h>

/*
 * FUNCTIONS USED
 */
void ReadInPPM(int PPMinput[]);
void mode1(int dl_pwm, int dm_pwm, double *lr);
double left(double s, double a);
double right(double s, double a);
double bounds(double x);
double pwm2frac(int pwm);
double DegtoPWM(double deg,int i);
void SendPPM(int output[]);

/*
 * CONSTANTS USED
 */
// transmitter values
#define TRANS_PWM_MIN 900.0
#define TRANS_PWM_MAX 2096.0
#define TRANS_PWM_NOM 1495.0
#define TRANS_PWM_NOISE 150.0
```

```
#define D 20.0  // max deflection of any given control surface (deg)

/*
 * GLOBAL VARIABLES
 */

// Create the PPM input variable - read on FALLING edge
PulsePositionInput DLRXinput(FALLING);

// Define 12 Servo instances & their corresponding pin on the Teensy
// 11 servos & 1 motor (signal split to both motors)
Servo Thrust;
int pThr = 18;
Servo L0;
int pL0 = 7;
Servo L1S;
int pL1 = 8;
Servo L2S;
int pL2 = 9;
Servo L3S;
int pL3 = 10;
Servo L4S;
int pL4 = 11;
Servo Center;
int pC = 1;
Servo R4;
int pR4 = 3;
Servo R3;
int pR3 = 2;
Servo R2;
int pR2 = 4;
Servo R1;
int pR1 = 5;
Servo R0;
int pR0 = 6;

// Launch the serial port in setup
void setup() {

  DLRXinput.begin(13); //signal from RX must be connected to pin 13

  Thrust.attach(pThr);
  L0.attach(pL0);
  L1S.attach(pL1);
  L2S.attach(pL2);
  L3S.attach(pL3);
  L4S.attach(pL4);
  Center.attach(pC);
  R4.attach(pR4);
```

```
  R3.attach(pR3);
  R2.attach(pR2);
  R1.attach(pR1);
  R0.attach(pR0);

  Serial.print("Setup Complete\n");
}

void ReadInPPM(int PPMinput[]){
  // read in the PWM channels from the RX PPM signal
  int Ail = DLRXinput.read(1);  // roll - delta l
  int Elev = DLRXinput.read(2); // pitch - delta m
  int Thr = DLRXinput.read(3);  // motor - pass through directly
  //int Rud = DLRXinput.read(4);  // yaw - ignore Mode 1

  // Write out the Thrust command immediately
  Thrust.write(Thr);

  // Write the Ail & Elev values to the PPMinput array
  PPMinput[0] = Ail;
  PPMinput[1] = Elev;

}

void mode1(int dl_pwm, int dm_pwm, double *lr) {
    double s[5], a[5], dl, dm;
    int i;

    dl = pwm2frac(dl_pwm);
    dm = pwm2frac(dm_pwm);

    for (i=0; i<5; i++) {
        s[i] = D * dm;
        a[i] = D * dl;
    }

    // set center control surface
    lr[5] = bounds(D * dm);
    // loop thru inboard to outboard control surfaces
    for (i=0; i<5; i++) {
        // set left control surface
        lr[4-i] = bounds( left(s[i], a[i]));
        // set right control surface
        lr[6+i] = bounds(right(s[i], a[i]));
    }
}

double left(double s, double a) {
    /*
     * s = symmetric value (deg)
```

```
     * a = asymmetric value (deg)
    */
    return s - a;
}

double right(double s, double a) {
    /*
     * s = symmetric value (deg)
     * a = asymmetric value (deg)
    */
    return s + a;
}

double bounds(double x) {
    /*
     * x = value in degrees
    */
    if (x < -D) x = -D;
    if (x >  D) x = D;
    return x;
}

double pwm2frac(int pwm) {
    if (pwm > int(TRANS_PWM_MAX + TRANS_PWM_NOISE) || pwm < int(TRANS_PWM_MIN -
TRANS_PWM_NOISE)) return 0.0;
    if (pwm > int(TRANS_PWM_MAX)) return 1.0;
    if (pwm < int(TRANS_PWM_MIN)) return -1.0;
    return 2.0 * (double(pwm) - TRANS_PWM_MIN) / (TRANS_PWM_MAX - TRANS_PWM_MIN) - 1.0;
}

// NOTE: This function needs to be updated with actual degree mapping
double DegtoPWM(double deg, int i){
  double C1 = 1.0;
  double C2 = 90.0;
  // degree mapping for different servo sections
  switch(i){
    // default returns the original degree value
    default: break;

    case 0: // L4
    {
      C1 = 1.6483;
    }
    break;

    case 1: // L3
    {
      C1 = 1.8466;
    }
    break;
```

```
case 2: // L2
{
  C1 = 1.746;
}
break;

case 3: // L1
{
  C1 = 2.193;
}
break;

case 4: // L0
{
  C1 = 2.2315;
}
break;

case 5: // Center
{
  C1 = -3.7146;
}
break;

case 6: // R0
{
  C1 = -2.1149;
}
break;

case 7: // R1
{
  C1 = -2.246;
}
break;

case 8: // R2
{
  C1 = -2.1162;
}
break;

case 9: // R3
{
  C1 = -1.6606;
}
break;

case 10: // R4
```

```
    {
      C1 = -1.5492;
    }
    break;
  }
  double pwm = C1*deg + C2;
  //pwm = int(pwm);

  return pwm;
}

void SendPPM(double output[]){
  // individually sends out servo commands - PWM signal

  L4S.write(output[0]);
  L3S.write(output[1]);
  L2S.write(output[2]);
  L1S.write(output[3]);
  L0.write(output[4]);
  Center.write(output[5]);
  R0.write(output[6]);
  R1.write(output[7]);
  R2.write(output[8]);
  R3.write(output[9]);
  R4.write(output[10]);
}

void loop() {
  int PPMinput[2];

  // 1) Read in the PPM signal & convert to 4 PWM signals
  ReadInPPM(PPMinput);

  double degOutput[11];
  // map the traditional degree inputs to the 11 servo inputs
  mode1(PPMinput[0],PPMinput[1],degOutput);

  // initialize the array to hold the PWM outputs
  double pwmOutput[11] = {0,0,0,0,0,0,0,0,0,0,0};

  int i;
  // convert the 11 servo deflections to pwm signals (skip throttle)
  for (i = 0; i < 11; i++){
    pwmOutput[i] = DegtoPWM(degOutput[i],i);
  }

  // send out the individual PWM signals
  SendPPM(pwmOutput);

}
```

V4 RC FLIGHT TEST ON JUNE 29TH

```
/*
Code Outline
1) Read in PPM signal from reciever
2) Interpret PPM signal as separate PWM components
  - Throttle [Thr] --> PASS THROUGH
  - Aileron  [Ail]
  - Elevator [Elev]
  - Rudder   [Rud] --> NOT USED
3) Map Ail & Elev inputs into the corresponding 11 servo outputs
  - see mode1 function
  - [SL0] [SL1] [SL2] [SL3] [SL4] [SC] [SR0] [SR1] [SR2] [SR3] [SR4]
5) Convert servo deflections (degrees) back into PWM signals
6) output the throttle & 11 servo commands as individual PWM signals
*/

/*
 * NOTES:
 * PWM values treated as integers
 * degree values treated as doubles
 *
 * CHECK THE DEGREE TO PWM MAPPING FUNCTION
 */

// Servo Library -- to control each servo/ send out individual PWM signals
#include "Servo.h"
// PulsePosition Library -- to read in / interpret the PPM signal
#include <PulsePosition.h>
// Math library may not be needed for Mode 1??
//#include <math.h>

/*
 * FUNCTIONS USED
 */
void ReadInPPM(int PPMinput[]);
void mode1(int dl_pwm, int dm_pwm, double *lr);
double left(double s, double a);
double right(double s, double a);
double bounds(double x);
double pwm2frac(int pwm);
double DegtoPWM(double deg,int i);
void SendPPM(int output[]);

/*
 * CONSTANTS USED
 */
// transmitter values
#define TRANS_PWM_MIN 900.0
#define TRANS_PWM_MAX 2096.0
#define TRANS_PWM_NOM 1495.0
#define TRANS_PWM_NOISE 150.0
```

```
#define D 20.0  // max deflection of any given control surface (deg)

/*
 * GLOBAL VARIABLES
 */

// Create the PPM input variable - read on FALLING edge
PulsePositionInput DLRXinput(FALLING);

// Define 12 Servo instances & their corresponding pin on the Teensy
// 11 servos & 1 motor (signal split to both motors)
Servo Thrust;
int pThr = 18;
Servo L0;
int pL0 = 7;
Servo L1S;
int pL1 = 8;
Servo L2S;
int pL2 = 9;
Servo L3S;
int pL3 = 10;
Servo L4S;
int pL4 = 11;
Servo Center;
int pC = 1;
Servo R4;
int pR4 = 3;
Servo R3;
int pR3 = 2;
Servo R2;
int pR2 = 4;
Servo R1;
int pR1 = 5;
Servo R0;
int pR0 = 6;

// Launch the serial port in setup
void setup() {

  DLRXinput.begin(13); //signal from RX must be connected to pin 13

  Thrust.attach(pThr);
  L0.attach(pL0);
  L1S.attach(pL1);
  L2S.attach(pL2);
  L3S.attach(pL3);
  L4S.attach(pL4);
  Center.attach(pC);
  R4.attach(pR4);
```

```
  R3.attach(pR3);
  R2.attach(pR2);
  R1.attach(pR1);
  R0.attach(pR0);

  Serial.print("Setup Complete\n");
}

void ReadInPPM(int PPMinput[]){
  // read in the PWM channels from the RX PPM signal
  int Ail = DLRXinput.read(1);  // roll - delta l
  int Elev = DLRXinput.read(2); // pitch - delta m
  int Thr = DLRXinput.read(3);  // motor - pass through directly
  //int Rud = DLRXinput.read(4);  // yaw - ignore Mode 1

  // Write out the Thrust command immediately
  Thrust.write(Thr);

  // Write the Ail & Elev values to the PPMinput array
  PPMinput[0] = Ail;
  PPMinput[1] = Elev;

}

void mode1(int dl_pwm, int dm_pwm, double *lr) {
    double s[5], a[5], dl, dm;
    int i;

    dl = pwm2frac(dl_pwm);
    dm = pwm2frac(dm_pwm);

    for (i=0; i<5; i++) {
        s[i] = D * dm;
        a[i] = D * dl;
    }

    // set center control surface
    lr[5] = bounds(D * dm);
    // loop thru inboard to outboard control surfaces
    for (i=0; i<5; i++) {
        // set left control surface
        lr[4-i] = bounds( left(s[i], a[i]));
        // set right control surface
        lr[6+i] = bounds(right(s[i], a[i]));
    }
}

double left(double s, double a) {
    /*
     * s = symmetric value (deg)
```

```
     * a = asymmetric value (deg)
    */
    return s - a;
}

double right(double s, double a) {
    /*
     * s = symmetric value (deg)
     * a = asymmetric value (deg)
    */
    return s + a;
}

double bounds(double x) {
    /*
     * x = value in degrees
    */
    if (x < -D) x = -D;
    if (x >  D) x = D;
    return x;
}

double pwm2frac(int pwm) {
    if (pwm > int(TRANS_PWM_MAX + TRANS_PWM_NOISE) || pwm < int(TRANS_PWM_MIN -
TRANS_PWM_NOISE)) return 0.0;
    if (pwm > int(TRANS_PWM_MAX)) return 1.0;
    if (pwm < int(TRANS_PWM_MIN)) return -1.0;
    return 2.0 * (double(pwm) - TRANS_PWM_MIN) / (TRANS_PWM_MAX - TRANS_PWM_MIN) - 1.0;
}

// NOTE: This function needs to be updated with actual degree mapping
double DegtoPWM(double deg, int i){
  double C1 = 1.0;
  double C2 = 90.0;
  // degree mapping for different servo sections
  switch(i){
    // default returns the original degree value
    default: break;

    case 0: // L4
    {
      C1 = 1.5922;
    }
    break;

    case 1: // L3
    {
      C1 = 1.7089;
    }
    break;
```

```
case 2: // L2
{
  C1 = 1.7762;
}
break;

case 3: // L1
{
  C1 = 1.9802;
}
break;

case 4: // L0
{
  C1 = 1.9902;
}
break;

case 5: // Center
{
  C1 = -4.7937;
}
break;

case 6: // R0
{
  C1 = -2.1036;
}
break;

case 7: // R1
{
  C1 = -1.6441;
}
break;

case 8: // R2
{
  C1 = -1.712;
}
break;

case 9: // R3
{
  C1 = -1.9267;
}
break;

case 10: // R4
```

```
    {
      C1 = -1.6738;
    }
    break;
  }
  double pwm = C1*deg + C2;
  //pwm = int(pwm);

  return pwm;
}

void SendPPM(double output[]){
  // individually sends out servo commands - PWM signal

  L4S.write(output[0]);
  L3S.write(output[1]);
  L2S.write(output[2]);
  L1S.write(output[3]);
  L0.write(output[4]);
  Center.write(output[5]);
  R0.write(output[6]);
  R1.write(output[7]);
  R2.write(output[8]);
  R3.write(output[9]);
  R4.write(output[10]);
}

void loop() {
  int PPMinput[2];

  // 1) Read in the PPM signal & convert to 4 PWM signals
  ReadInPPM(PPMinput);

  double degOutput[11];
  // map the traditional degree inputs to the 11 servo inputs
  mode1(PPMinput[0],PPMinput[1],degOutput);

  // initialize the array to hold the PWM outputs
  double pwmOutput[11] = {0,0,0,0,0,0,0,0,0,0,0};

  int i;
  // convert the 11 servo deflections to pwm signals (skip throttle)
  for (i = 0; i < 11; i++){
    pwmOutput[i] = DegtoPWM(degOutput[i],i);
  }

  // send out the individual PWM signals
  SendPPM(pwmOutput);

}
```

V6 M2 FLIGHT TEST ON OCT 7TH

```
/* Code Outline
1) Read in PPM signal from reciever
2) Read in telemetry data
3) Check for the mode and evaluate the control mapping functions
5) Convert control surface deflections to servo arm deflections
6) output the throttle & 11 servo commands
*/

/* NOTES:
 * PWM values treated as integers and are packaged into the custom struct
 *      "pilot" which is a global variable
 * degree values treated as doubles and there are two arrays, one for the
 *      control surface deflections "deg" and another for the servo arm
 *      deflections "servoDeg". Both of these are global variables
 * dL is a instance of the running average class that will help smooth out
 *      the sensor data for lift control. This is also a global variable.
 * all functions and variables associated with the control mapping are
 *      included in the header file "controlMapping.h".
 */

// included the needed libraries
//###########################################################################
// Servo Library -- to control each servo/ send out individual PWM signals
#include "Servo.h"
// PulsePosition Library -- to read in / interpret the PPM signal
#include <PulsePosition.h>
// library to read in pixhawk data
#include "ardupilotmega/mavlink.h"
// include control mapping items
#include "libraries\controlMapping\controlMapping.h"
// include running average
#include "libraries\runningAverage\runningAverage.cpp"

// FUNCTIONS USED
//###########################################################################
// Sabrina's functions
void ReadInPPM();
void deg2servoDeg();
void Send2Servo();
void debug();
void comm_receive();
void printVal(char *name, double val);

// define the global constants (they are all in uppercase for ease of identifying)
//###########################################################################
// deg2servoDeg mapping values
#define INTERCEPT 90.
#define SLOPE_L4 1.7093
#define SLOPE_L3 1.5384
#define SLOPE_L2 1.7961
```

```
#define SLOPE_L1 1.7799
#define SLOPE_L0 1.9429
#define SLOPE_CE -3.5752
#define SLOPE_R0 -2.0551
#define SLOPE_R1 -1.7414
#define SLOPE_R2 -1.6435
#define SLOPE_R3 -1.5585
#define SLOPE_R4 -1.5637

// GLOBAL VARIABLES
//###########################################################################
// Create the PPM input variable - read on FALLING edge
PulsePositionInput DLRXinput(FALLING);
// Define 12 Servo instances & their corresponding pin on the Teensy
// 11 servos & 1 motor (signal split to both motors)
Servo Thrust;
int pThr = 14;
Servo L0;
int pL0 = 6;
Servo L1;
int pL1 = 5;
Servo L2;
int pL2 = 4;
Servo L3;
int pL3 = 3;
Servo L4;
int pL4 = 2;
Servo Ce;
int pC = 7;
Servo R4;
int pR4 = 12;
Servo R3;
int pR3 = 11;
Servo R2;
int pR2 = 10;
Servo R1;
int pR1 = 9;
Servo R0;
int pR0 = 8;

// create arrays for degrees deflection and servo arm degrees
double deg[11], servoDeg[11];
// create instance of telemetry data and pilot commands
struct telemetryData pix;
struct pilotCommands pilot;
// initialize the running average
runAvg* dL = new runAvg(200, 0.);

//###########################################################################
//###########################################################################
```

```
//###########################################################################

void loop() {
  //int i;

  // 1) Read in the PPM signal & convert to 4 PWM signals
  ReadInPPM();

  // read in pixhawk data
  comm_receive();

  // update dL
  dL->update(calc_dL(pix));

  // check for the mode
  if (pilot.modeSwitch < TRANS_PWM_NOM) {
    mode1(pilot, deg);
  }
  //else if (pilot.modeSwitch < ?) {
     //implement a third mode
  //}
  else {
    mode2(pilot, dL->getAverage(), deg);
  }

  // convert degrees control surface deflection to degrees servo arm deflection
  deg2servoDeg();

  // send out the individual PWM signals
  Send2Servo();

  // displays values to the serial monitor
  debug();
}

// Launch the serial port in setup
void setup() {

  DLRXinput.begin(13); //signal from RX must be connected to pin 13

  Thrust.attach(pThr);
  L0.attach(pL0);
  L1.attach(pL1);
  L2.attach(pL2);
  L3.attach(pL3);
  L4.attach(pL4);
  Ce.attach(pC);
  R4.attach(pR4);
  R3.attach(pR3);
  R2.attach(pR2);
```

```
  R1.attach(pR1);
  R0.attach(pR0);

  Serial1.begin(57600);    // TELEM2 from Pixhawk

  // Serial.begin(9600);

  //Serial.print("Setup Complete\n");

  pix.airspeed = 16.5;
  pix.climbRate = 0.;
  pix.bankAngle = 0.;
  pix.elevationAngle = 0.;
  pix.rollRate = 0.;
}

void ReadInPPM(){
  // read in the PWM channels from the RX PPM signal
  pilot.ail = DLRXinput.read(1);  // roll - delta l
  pilot.ele = DLRXinput.read(2);  // pitch - delta m
  Thrust.write( DLRXinput.read(3) );  // motor - pass through directly
  pilot.rud = DLRXinput.read(4);  // yaw - ignore Mode 1
  // we will need to read in one more channel, likely a 2 or 3 way switch so that the pilot
can change modes as desired
  pilot.modeSwitch = DLRXinput.read(5);
}

void deg2servoDeg(){
  servoDeg[0]  = deg[0]  * SLOPE_L4 + INTERCEPT;
  servoDeg[1]  = deg[1]  * SLOPE_L3 + INTERCEPT;
  servoDeg[2]  = deg[2]  * SLOPE_L2 + INTERCEPT;
  servoDeg[3]  = deg[3]  * SLOPE_L1 + INTERCEPT;
  servoDeg[4]  = deg[4]  * SLOPE_L0 + INTERCEPT;
  servoDeg[5]  = deg[5]  * SLOPE_CE + INTERCEPT;
  servoDeg[6]  = deg[6]  * SLOPE_R0 + INTERCEPT;
  servoDeg[7]  = deg[7]  * SLOPE_R1 + INTERCEPT;
  servoDeg[8]  = deg[8]  * SLOPE_R2 + INTERCEPT;
  servoDeg[9]  = deg[9]  * SLOPE_R3 + INTERCEPT;
  servoDeg[10] = deg[10] * SLOPE_R4 + INTERCEPT;
}

void Send2Servo(){
  // individually sends out servo commands - PWM signal
  L4.write(servoDeg[0]);
  L3.write(servoDeg[1]);
  L2.write(servoDeg[2]);
  L1.write(servoDeg[3]);
  L0.write(servoDeg[4]);
  Ce.write(servoDeg[5]);
  R0.write(servoDeg[6]);
```

```
  R1.write(servoDeg[7]);
  R2.write(servoDeg[8]);
  R3.write(servoDeg[9]);
  R4.write(servoDeg[10]);
}

void comm_receive() {
  uint8_t c;
  mavlink_message_t msg;
  mavlink_status_t status;

  while(Serial1.available() > 0) { // as long as buffer size bigger than 0 -- pull out serial
data

    c = Serial1.read(); // reads in the serial message

    // Try to get a new message
    if(mavlink_parse_char(MAVLINK_COMM_0, c, &msg, &status)) {

        switch(msg.msgid) {
        default: break;
        // IMU data -- roll & yaw rates
        case MAVLINK_MSG_ID_RAW_IMU:  // #105 highres IMU / Check scaled IMU for acutal value
readings
          {
            mavlink_raw_imu_t raw_imu;
            mavlink_msg_raw_imu_decode(&msg, &raw_imu);
            pix.rollRate = raw_imu.xgyro;
            // printVal("roll rate raw: ", raw_imu.xgyro);
          }
          break;
        // airspeed
        case MAVLINK_MSG_ID_VFR_HUD:  // #74 VFR_HUD
          {
            mavlink_vfr_hud_t vfr_hud;
            mavlink_msg_vfr_hud_decode(&msg, &vfr_hud);
            pix.airspeed = vfr_hud.groundspeed;  // m/s
            pix.climbRate = vfr_hud.climb;  // m/s
            // printVal("airspeed: ", vfr_hud.airspeed);
            // printVal("climb rate: ", vfr_hud.climb);

          }
          break;
        // attitude data -- roll angles (bank angle)
        case MAVLINK_MSG_ID_ATTITUDE:  // #30 ATTITUDE
          {
            mavlink_attitude_t attitude;
            mavlink_msg_attitude_decode(&msg, &attitude);
            pix.bankAngle = attitude.roll;  // rad (-pi..+pi)
            pix.elevationAngle = attitude.pitch;
```

```
            // printVal("bank: ", attitude.roll);
            // printVal("elevation: ", attitude.pitch);

          }
          break;
      }
    }
  }
}

// void printVal(char *name, double val) {
    // Serial.print(name);
    // Serial.printf("%20.12f", val);
    // Serial.println();
    // delay(1);
// }

void debug(){
    int i;

    Serial.print("pilotCommands: ");
    Serial.printf("%4u", pilot.ail);
    Serial.print(", ");
    Serial.printf("%4u",pilot.ele);
    Serial.print(", ");
    Serial.printf("%4u",pilot.rud);
    Serial.print(", ");
    Serial.printf("%4u",pilot.modeSwitch);
    Serial.print(", ");
    Serial.print(" pixhawk: ");
    Serial.printf("%6.2f", pix.airspeed);
    Serial.print(", ");
    Serial.printf("%6.2f", pix.climbRate);
    Serial.print(", ");
    Serial.printf("%6.2f", pix.bankAngle);
    Serial.print(", ");
    Serial.printf("%6.2f", pix.elevationAngle);
    Serial.print(", ");
    Serial.printf("%6.2f", pix.rollRate);
    Serial.print(", ");
    Serial.printf("%5.3f", dL->getAverage());
    Serial.print(", ");

    Serial.print(" degrees: ");
    for (i=0; i<11; i++){
        Serial.printf("%6.2f", deg[i]);
        Serial.print(", ");
    }
    Serial.println();
}
```