

An Autonomous Autopilot Control System Design for Small-Scale UAVs

Corey Ippolito
QSS Group, Inc.
NASA Ames Research Center

Table of Contents

Introduction.....	1
Ground Station Monitoring and Control.....	2
Autopilot System Overview.....	3
Commands in the Flight Management System.....	4
Command Types.....	6
Command: Direct To Waypoint.....	6
Command: TrackToWaypoint.....	7
Command: Jump.....	8
Command: Circle.....	8
Command: TakeOff.....	8
Command: Landing.....	9
Command: SetFMSSMode.....	9
Flight Management System Modes.....	10
Autopilot Controller.....	11

Introduction

This paper describes the design and implementation of a fully autonomous and programmable autopilot system for small scale autonomous unmanned aerial vehicle (UAV) aircraft. This system was implemented in Reflection and has flown on the Exploration Aerial Vehicle (EAV) platform at NASA Ames Research Center, currently only as a safety backup for an experimental autopilot. The EAV and ground station are built on a component-based architecture called the Reflection Architecture. The Reflection Architecture is a prototype for a real-time embedded plug-and-play avionics system architecture which provides a transport layer for real-time communications between hardware and software components, allowing each component to focus solely on its implementation. The autopilot module described here, although developed in Reflection, contains no design elements dependent on this architecture.

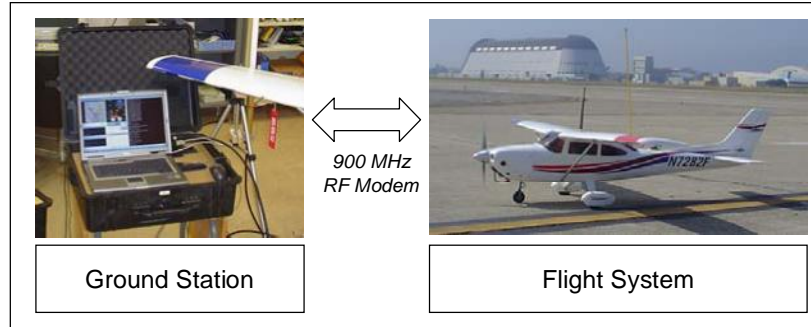


Figure 1. Components of the UAV Flight System

The UAV system is composed of two main components: a ground station component which provides telemetry feedback for the operator and allows for control of the aircraft, and a flight system component onboard the vehicle. The component details are shown in Table 1.

Table 1. System Hardware

	Ground Station	Flight System
Platform/Airframe	Dell Mobile Workstation	Hanger 9 Quarter Scale Cessna 172
CPU	DELL Precision M70 Workstation, Pentium M 770, 2.13GHz, 2GB RAM	Digital Logic MSMP3SEV, Pentium-III, 700MHz, 256MB RAM
Operating System	Windows XP Professional	Windows XP Embedded
Real-Time Infrastructure	Reflection Architecture	Reflection Architecture
Communications	Microhard Spectra 910, 1W 902-928MHz RF Modem	Microhard MHX-910, 1W 902- 928MHz RF Modem
Avionics		Athena Guidestar GS111m INS/GPS
		Pontech SV203 Servo Controller
		Custom Servo Switching Board

Ground Station Monitoring and Control

With the Reflection plug-and-play architecture, simulation and hardware can be mixed on the fly for in-situ simulation testing of hardware components at any level of granularity. When implementing the final ground station controller, several simulation components were reused to provide telemetry data back to the ground station. A screenshot of the ground station is shown in Figure 2 below.



Figure 2. Screenshot of Ground Station with 3D Visualization

The bottom left window provides a graphical interface for configuring and manipulating waypoint commands. Waypoints can be manipulated on a 2D representation of the terrain; waypoints are overlaid on GPS-localized 2D bitmap images. Once a mission is programmed, the user can 'upload' the commands to the aircraft. The ground station generates remote method invocations on the remote autopilot module, and passes these invocation requests to the Reflection distributed transport layer (RDTL). The RDTL converts the invocation requests into Reflection Virtual Machine byte-code commands and sends the commands over to the autopilot component via the 900 MHz communications link. On the flight computer, the RDTL receives and assembles the invocation requests, then executes the commands on the autopilot module.

Autopilot System Overview

The autopilot system is implemented as a stand alone plug-in component in the Reflection Architecture. Data flow through the system conceptually follows the diagram shown in Figure 3. Command objects generate FMS lateral and longitudinal mode instructions for the FMS. The FMS in turn generates controller mode instructions, which are used by the controller to command the actuators of the vehicle. The state information and the actuator commands are generated by external plug-and-play Reflection modules.

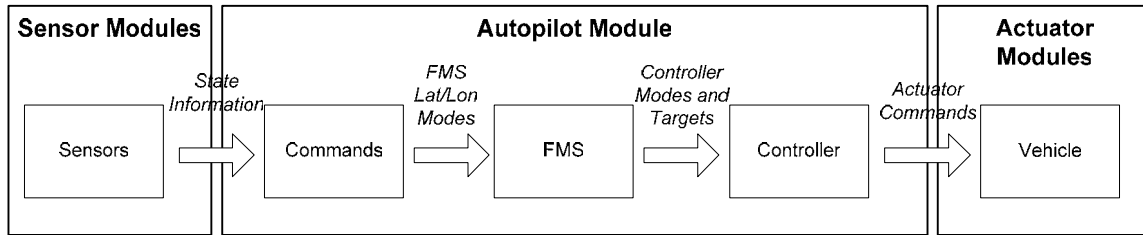


Figure 3. Conceptual Data Flow and Components

The top-most object in the control system is an instance of the *AutopilotSystem* class, as shown in Figure 4. The *AutopilotSystem* class is responsible for communicating with the rest of the Reflection system and maintaining the two main objects in the system: the *FMS* (flight management system) and the *Controller*. The *FMS* is responsible for maintaining the list of commands which specify FMS mode instructions. The mode instructions are used by the *FMS* to provide targets to the controller. The controller is responsible for implementing the control loops which control the aircraft through the vehicle's actuators.

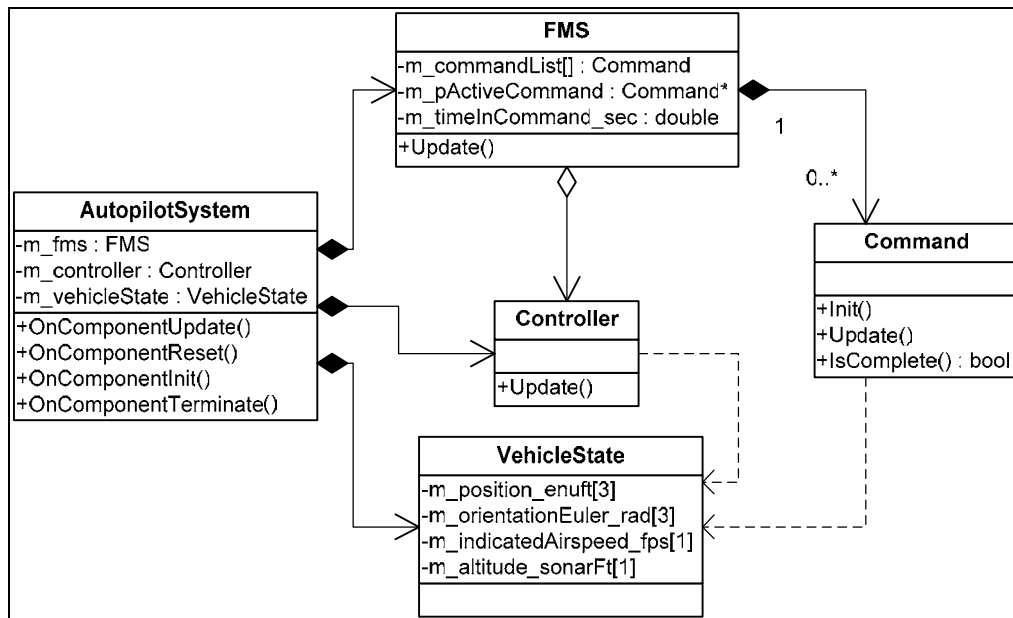


Figure 4. Top-Level Classes in the Autopilot System

Commands in the Flight Management System

The flight management system class is a state machine that controls the continuous feedback control system in the controller object. The *FMS* is responsible for maintaining a database of mission program, keeping track of the mission program execution, and setting the modes of the controller object's feedback loops.

The mission program is implemented *Command* objects, which are stored in a programmable list in the *FMS*. The command list is an ordered list of commands that can be programmed on the ground, or reprogrammed while the aircraft is in the air. In implementation, the *Command* class is a virtual interface class, from which specific types

of command classes inherit. All commands implement the interfaces defined by the base class, which includes methods to initialize their internal state, update their internal state at each OnComponentUpdate() callback, and to notify the FMS when the command objectives have been met.

Controller commands can be programmed through the ReflectionScript language, which interprets text-based commands into binary code which is passed to the Reflection Virtual Machine. The following scripts are examples of programmed routes for the Reflection component.

```
// Program the Autopilot System.
objAutopilot.ClearCommandList();
objAutopilot.AddCommand_Takeoff ( 4500, -2000, 500.0, 80.0, 1000.0, -0.34907, 50.0 );
objAutopilot.AddCommand_FlyToTrack ( 8070.9, -6009.7, 500.0, 80.0, 1000.0 );
// Following heading until specified altitude is reached.
objAutopilot.AddCommand_SetFMSMode ( HEADINGHOLD, 0.0, ALTICMD, 400.0, SPDCMD, 80,
    ALTITUDE, GT, 300 );
cmd=objAutopilot.AddCommand_FlyToTrack ( 10107.0, -1390.7, 500.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( 8904.5, -902.8, 500.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( 6919.8, -5568.9, 500.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( 3357.4, 1364.9, 500.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( -3319.9, 4457.2, 500.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( -5436.1, 3980.7, 500.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( -4371.0, 1696.2, 200.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( -741.1, 182.6, 50.0, 80.0, 1000.0 );
objAutopilot.AddCommand_FlyToTrack ( 2522.8, -1267.5, -10.0, 80.0, 1000.0 );
objAutopilot.AddCommand_Jump ( cmd );
```

Figure 5. Sample Control Program.

At the start of the mission, the controller activates the first command, which might be a *Takeoff* command for an autonomous takeoff. The update sequence, shown in Figure 6, is very simple sequence of steps; the FMS object is only responsible for maintaining the current state, transitioning between states at the appropriate time. At each component update callback, the FMS calls the Update() method of the active command, which updates its internal state. The FMS then checks the IsComplete() method of the command; when *TRUE* is returned, the FMS transitions to the next command, and signals a Reflection event in case any other Reflection component is monitoring autopilot state transitions.

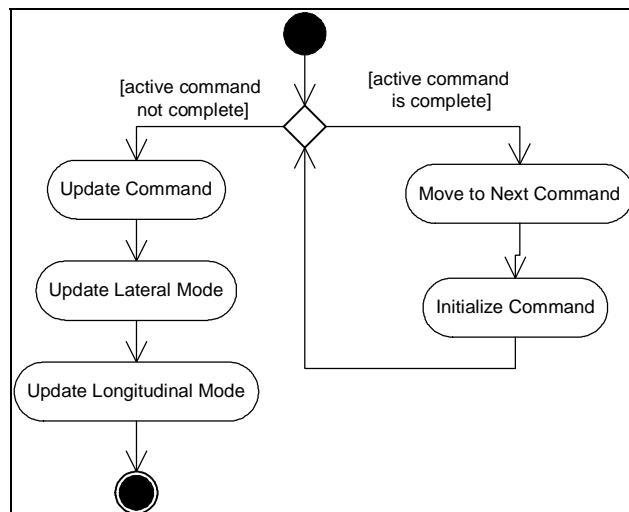


Figure 6. FMS Updates

The FMS internally maintains three different FMS control modes, an FMS lateral mode, FMS longitudinal mode, and FMS speed mode. The active command object internally maintains a state machine which outputs FMS mode commands and targets appropriately to meet the objectives of each command.

Command Types

Command: Direct To Waypoint

The *DirectToWaypoint* class controls the vehicle mostly through heading commands towards a specified 3D waypoint. When initiated, the *DirectToWaypoint* stores the initial position of the aircraft and the position of the next waypoint (if there is one). Depending on the transition mode, the command uses this information to determine the size of the waypoint radius based on current aircraft speed to allow a turn to the next waypoint without overshoot, and to determine if the aircraft has passed the waypoint based on a 2D axis perpendicular to the line from the initial position to the waypoint position, and passing through the waypoint.

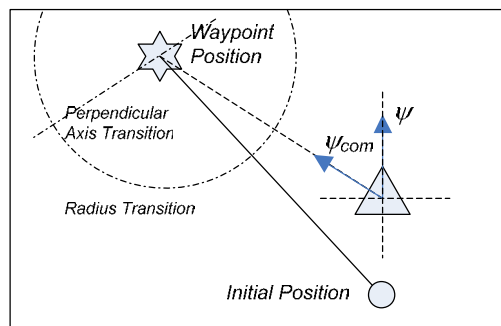


Figure 7. DirectToWaypoint Diagram

In implementation, the *DirectToWaypoint* class inherits from the *Command* base class, as shown in Figure 8.

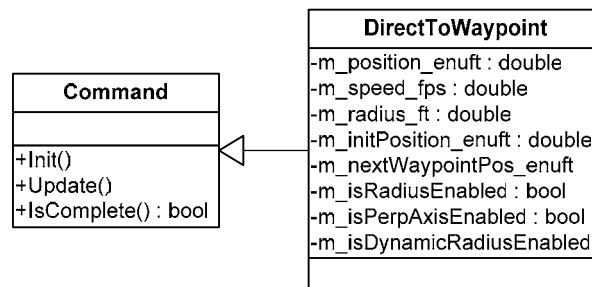


Figure 8. DirectToWaypoint Class

The *DirectToWaypoint* command sets the lateral FMS mode to 'FlyToWaypoint', and outputs a heading command. The longitudinal mode and speed mode are selected based on the waypoint command; the longitudinal controls or the speed controls can be used to

control either airspeed, altitude, or follow the vertical track between waypoints. Having throttle control altitude is the default, for safety if the engine fails.

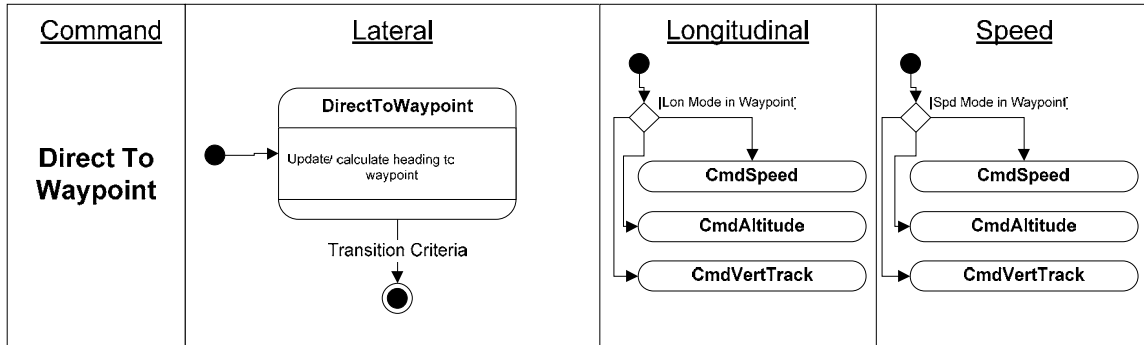


Figure 9. DirectToWaypoint Command FMS Modes

Command: TrackToWaypoint

The TrackToWaypoint command is similar to the DirectToWaypoint command, except that the aircraft is controlled to fly the track from the previous waypoint to the next waypoint, rather than heading straight to the waypoint. Similar to the DirectToWaypoint command, the transition to the next waypoint can be triggered by penetration of the perpendicular axis or the transition radius, which can be calculated based on the current aircraft state and the performance characteristics of the aircraft (turning radius).

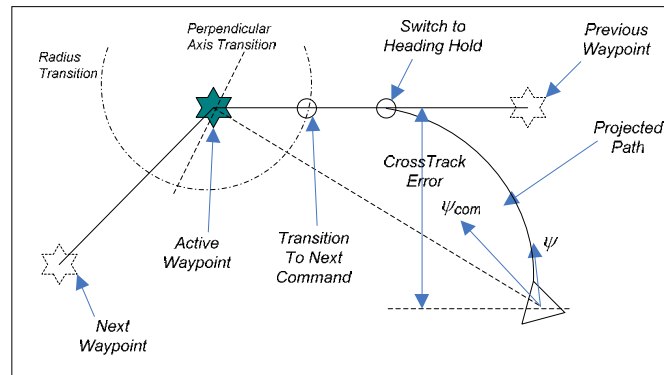


Figure 10. TrackToWaypoint Diagram

The TrackToWaypoint commands the TrackToWaypoint lateral FMS mode. The longitudinal mode is set to AltitudeCommand.

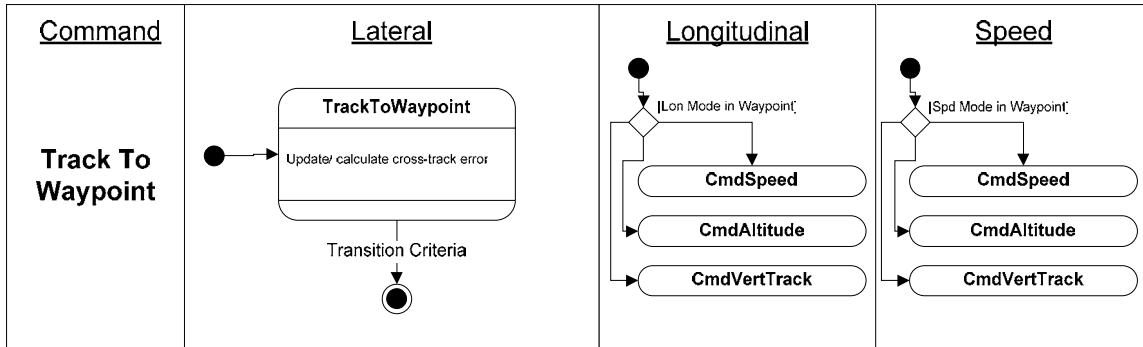


Figure 11. TrackToWaypoint FMS Mode State Diagram

The manner in which the altitude is captured can be set by the waypoint. Immediate altitude capture is the default, where the aircraft is controlled within the safety limits of the controller to attain the altitude as soon as possible. The glideslope mode will have the aircraft follow the slope between waypoints, and is used for instance in the final approach leg during a landing.

Command: Jump

The Jump command instigates an immediate transition to another command in the list. This control is used for instance to repeat a sequence of commands. There are no FMS modes associated with the Jump command.

Command: Circle

The Circle command controls the aircraft to fly a circle pattern of a given radius about a waypoint. The lateral PID mode is set to 'Circle', the longitudinal mode set to AltitudeCommand.

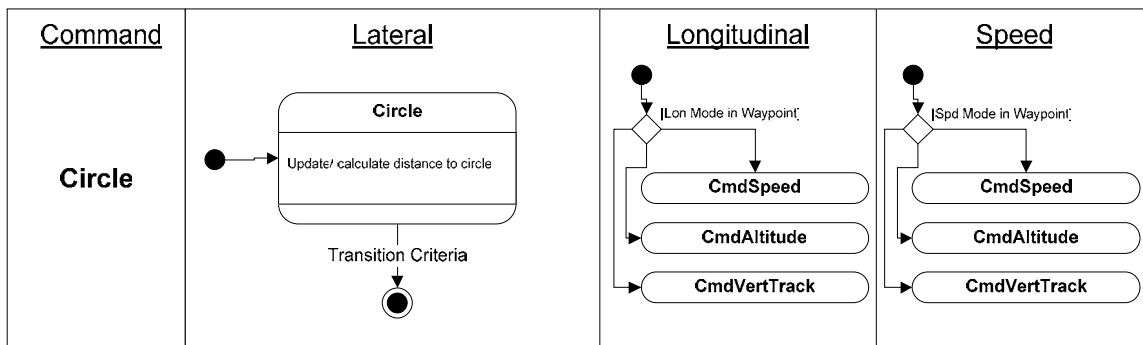


Figure 12. Circle Waypoint FMS Mode State Diagram

Command: TakeOff

The TakeOff command provides commands for an autonomous takeoff sequence. The FMS modes are set to a dedicated 'TakeOff' mode, which provide full throttle to the aircraft while maintaining heading on the runway based on steering and rudder inputs, as rudder and steering inputs are ganged together on the same servo command line. Once the rotation takeoff speed is reached, the aircraft performs a climb longitudinal maneuver while maintain lateral wings level, until a safe turning altitude is reached.

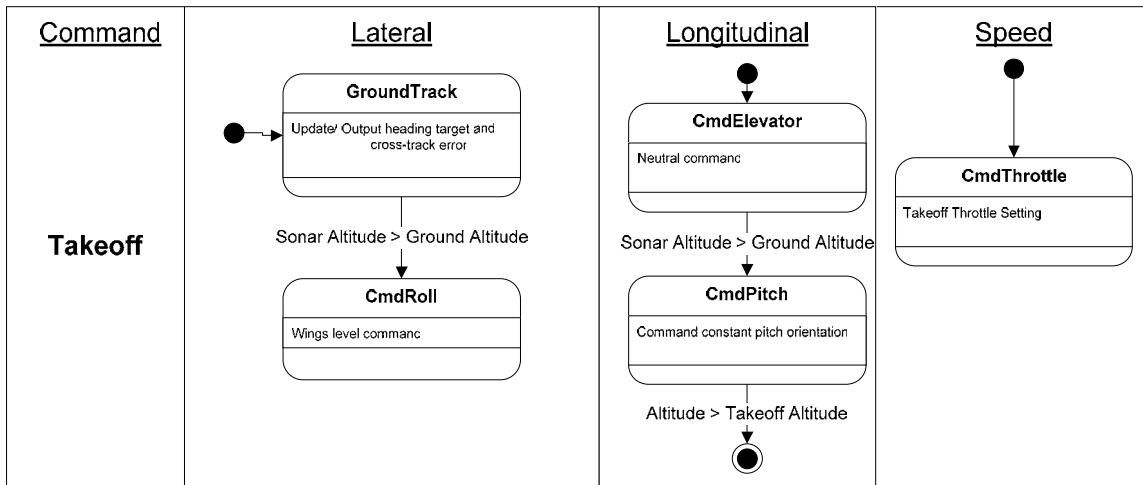


Figure 13. Takeoff FMS Mode State Diagram

Command: Landing

The Landing command is used for automated landings. The aircraft is commanded to maintain a track between waypoints and command a constant descent rate until the ground sonars pick up a reading from the ground, and the aircraft attains the specified distance to the ground. Note that position information based on pressure and GPS may not be accurate enough to determine the flare maneuver, so the flare command will institute this command based on an ultrasonic altimeter located on the bottom of the aircraft. When the flare mode is executed, the elevators are used to command a nose up attitude of the aircraft while power is reduced, until the altitude notifies that the rear wheels are on the ground. The power is cut while the elevator is used to drop the front wheel.

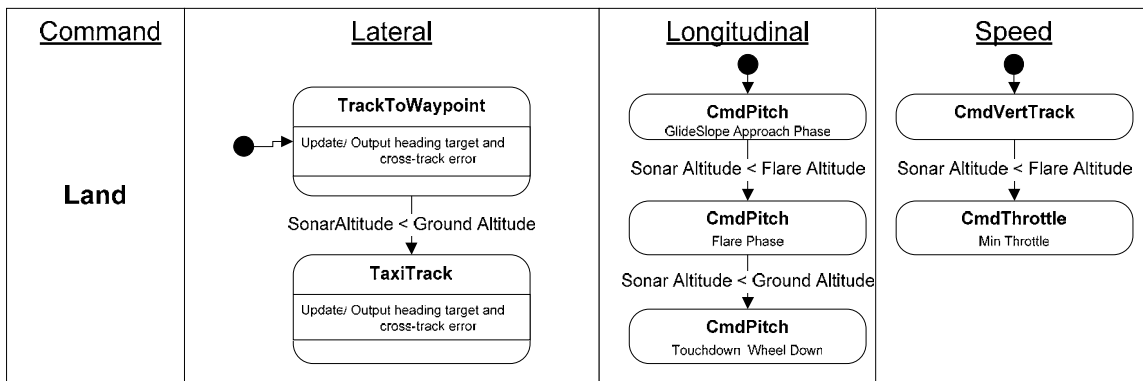


Figure 14. Land Command FMS Mode State Diagram

Command: SetFMSMode

The FMSModeTimed will command the specified FMS mode until a transition condition is realized. This command's class structure is shown in Figure 15.

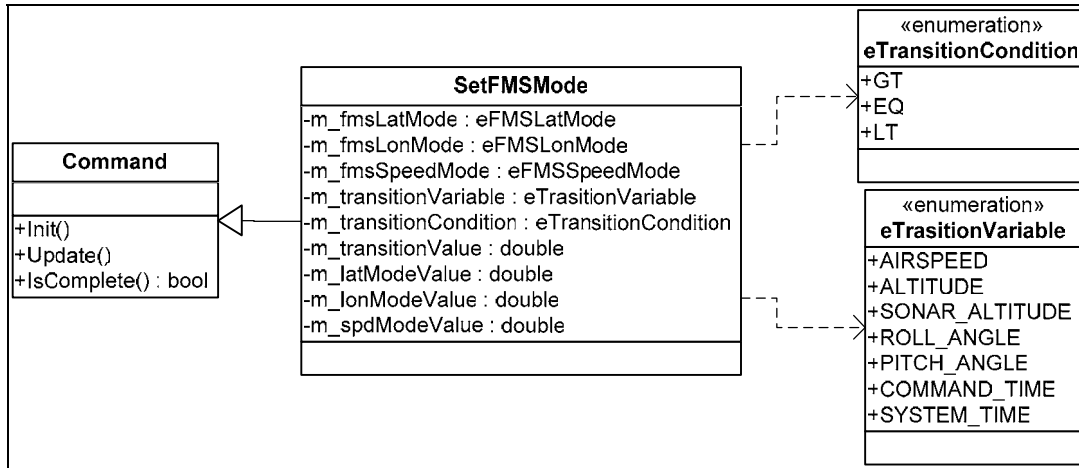


Figure 15. SetFMSMode Command Class Diagram

Flight Management System Modes

The FMS receives command modes and targets from the active command, and outputs command instructions to the autopilot. The list of FMS modes currently implemented is shown in Table 2.

Table 2. List of FMS Modes

<u>Lateral Mode</u>	<u>Longitudinal Mode</u>	<u>Speed Mode</u>
DirectToWaypoint	CmdSpeed	CmdSpeed
TrackToWaypoint	CmdAltitude	CmdAltitude
Circle	CmdVertTrack	CmdVertTrack
TaxiTrack	AltitudeAttain	AltitudeAttain
CmdRoll	AltitudeHold	AltitudeHold
CmdAileron	CmdElevator	CmdThrottle
Disabled (PilotCmd)	CmdPitch	Disabled (PilotCmd)
	Disabled (PilotCmd)	

Each mode is associated with one or more commands to the controller, and modes can be implemented as state machines. For instance, the CmdAltitude mode for both the lateral and longitudinal modes contains an internal state machine shown in Figure 16.

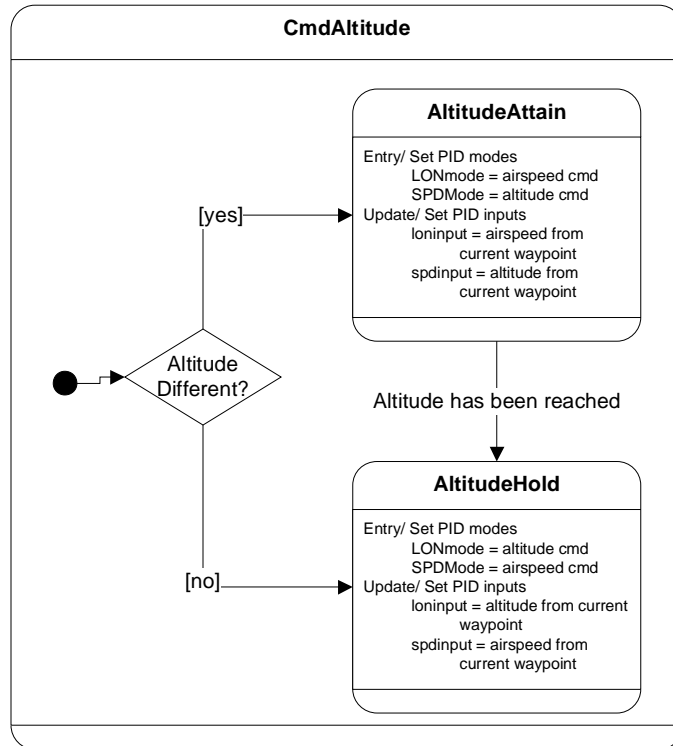


Figure 16. CmdAltitude Mode State Diagram

The two state machines implementing an FMS mode and a command object could be integrated into a single machine; for instance, the DirectToWaypoint command uses the CmdAltitude FMS state, but could implement the AltitudeAttain/AltitudeHold FMS states explicitly using the same state machine model. The reason to have two state machines is to help avoid redundant code. Since many different commands require the aircraft to attain then hold an altitude, this state machine was implemented as a common FMS control mode rather than duplicating the logic in several individual command machines.

Autopilot Controller

The controller object is responsible for implementing the feedback control loops that command the aircraft. The FMS object commands the controller by first setting high level modes, the providing actual target data. Similar to the FMS, there are three modes associated with the controller: lateral modes, longitudinal modes, and speed modes. Lateral modes control the ailerons and rudder, longitudinal modes control the elevator and also affect the rudder, and speed modes control the throttle. The controller modes are listed in Table 3.

Table 3. Controller Modes

<u>Lateral</u>	<u>Longitudinal</u>	<u>Speed</u>
Disengaged	Disengaged	Disengaged
Roll Cmd	Pitch Cmd	Fixed Throttle
Heading Cmd	Altitude Cmd	Airspeed Cmd

CrossTrack	Airspeed Cmd	Altitude Cmd
Circle	Vert Speed Cmd	Glide Slope
Takeoff		

The controller uses a cascaded control structure mainly composed of proportional-derivative-integral (PID) controller transforms. Several different paths can lead from sensor inputs to actuator command outputs, as shown in Figure 17. Specific paths are associated with enumerated controller modes. Lateral modes control the ailerons and rudder, longitudinal modes control the elevator and also affect the rudder, and speed modes control the throttle.

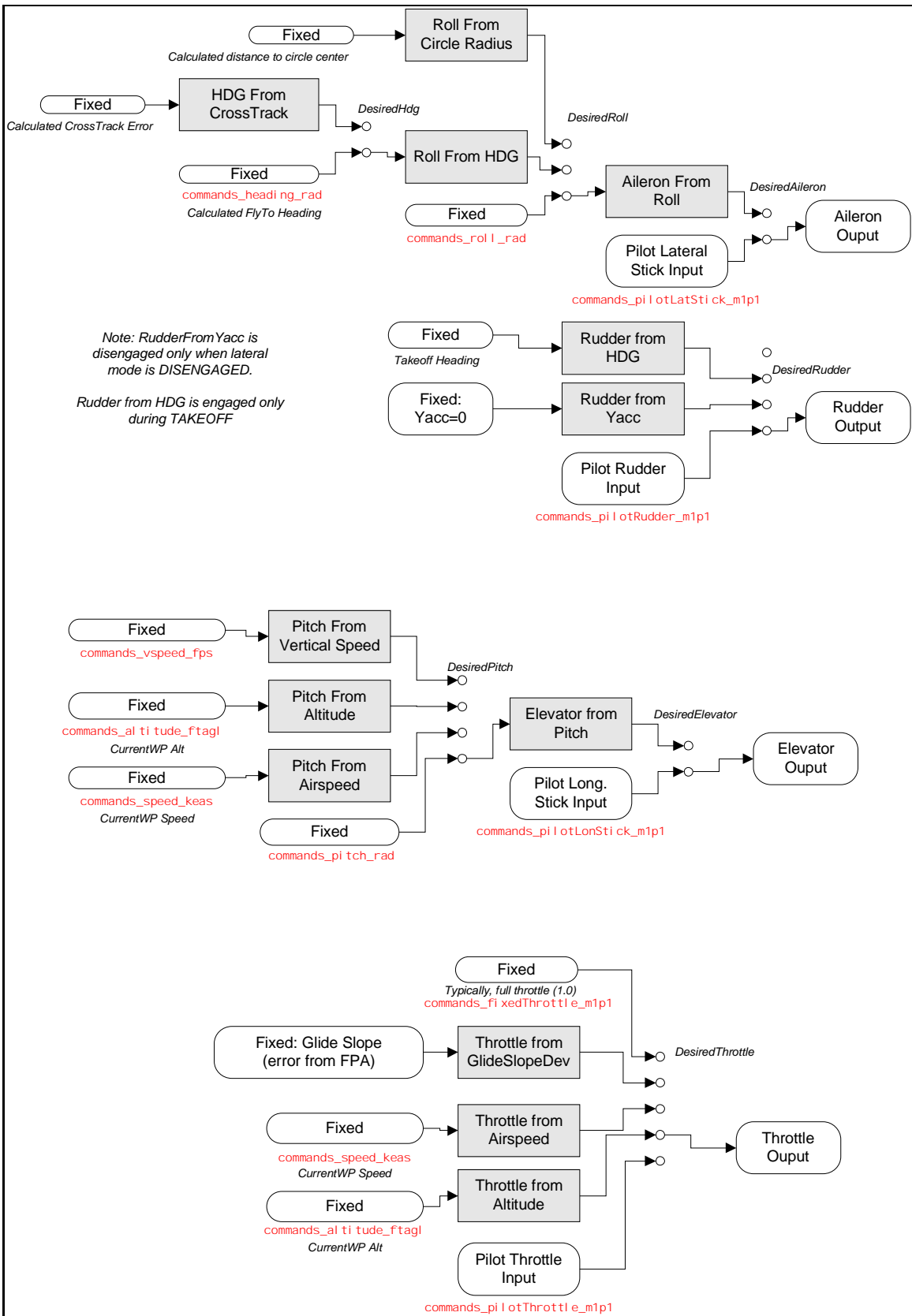


Figure 17. Cascading PID Control Structure in the Controller Class

Dependability Analysis Requirements For an Electric Unmanned Aerial Vehicle (UAV)

Corey Ippolito, Ganesh Pai, and Ewen Denney
NASA Ames Research Center

{corey.ippolito, ganesh.pai, ewen.denney}@nasa.gov

Summary

This document specifies the dependability analysis requirements for the Swift Unmanned Aircraft System (UAS). Section 1 gives a description of the system, a high-level concept of intended operations and an overview of the autopilot software to be deployed in the airborne part of the UAS. Section 2 specifies the analysis to be performed and the expected outcome.

1 System Description

1.1 The Swift UAS

The Swift UAS comprises a single airborne system: the electrically powered *Swift* unmanned aerial vehicle (UAV), a primary and secondary ground control station (GCS) and communication links for remote control and telemetry.

A pilot on the ground can control the UAV, or can it fly autonomously by following a *nominal flight plan*. The pilot might takeoff, land, or intercept at any time during flight. There is an *off-nominal flight plan* which describes the actions of a Contingency Management System (CMS), i.e., the failsafe trajectory, the procedures to be followed on the ground, and so on. The CMS is required is to keep the aircraft on range, so that if any crash occurs, it happens on range.

The protocol for the CMS is:

1. If there is a failure of the primary pilot system, switch to the redundant secondary system, on a different channel.
2. If the secondary also fails, engage the onboard autopilot.
3. If the autopilot fails, force a spiral descent to impact.

During semi-autonomous flight, the aircraft can operate in the primary pilot in control (PIC), or secondary pilot in control (SIC) modes, or in computer in control (CIC) modes. The GCS is used to upload commands to the UAV.

Figure 1 gives a physical breakdown of the complete Swift UAS. In this breakdown, we are interested in the contributions of the Avionics sub-system (shown as enclosed by the dotted box) to UAS hazards.

This will require reasoning about hazards from several perspectives. In particular:

1. From a mission perspective, i.e., specific environmental conditions during a specific phase of flight, in which there is an increased likelihood of unacceptable loss, e.g., sudden crosswind during takeoff (landing), obstacles in the flight path, etc.
2. From the perspective of the machine (the UAS), e.g., specific machine states (such as failure states) in which there is an increased likelihood of unacceptable loss, and
3. From the perspective of states of the avionics system, e.g., flight computer failure, which can contribute to an increased likelihood of unacceptable loss. These could take the form of interactions with other subsystems, or uncovered (failure) states, or interactions (which may or may not be legitimate) with environmental conditions that contribute to system hazards.

PA.1.0	Complete Swift UAS System
PA.1.1	Flight Vehicle System (FVS)
PA.1.1.1	Avionics System (AVS)
PA.1.1.1.1	Flight Critical Control System (FCS)
PA.1.1.1.2	Control and Data Handling System (CDHS)
PA.1.1.1.2.1	CDHU (Control and Data Handling Unit) Hardware
PA.1.1.1.2.2	FMS (Flight Management System)
PA.1.1.1.2.3	ESS (Embedded Software Systems)
PA.1.1.1.2.4	AP (Autopilot System)
PA.1.1.1.3	Communication Systems (COM)
PA.1.1.1.4	Flight Sensors (SENS)
PA.1.1.1.4.1	INS (ADHRS, IMU)
PA.1.1.1.4.2	GPS
PA.1.1.1.4.3	Air Data System (AIRDAT)
PA.1.1.1.4.4	DGPS
PA.1.1.2	Payload and Mission Data Systems (PYLD)
PA.1.1.2.1	Common Payload Data System (CPDS)
PA.1.1.3	Actuation System (ACT)
PA.1.1.3.1	Control Surface Actuation (CS)
PA.1.1.3.2	Steering and Brake Actuation
PA.1.1.4	Propulsion System (PRLP)
PA.1.1.4.1	Electric Motor Propulsion System
PA.1.1.5	Structures
PA.1.1.5.1	Fuselage Structure
PA.1.1.5.2	Fuselage Outer Mold Line
PA.1.1.5.3	Propulsion Support
PA.1.1.5.4	Avionics Mounting Structure
PA.1.1.5.5	Wing Structure
PA.1.1.5.6	Control Surfaces
PA.1.1.6	Electrical and Power System (EPS)
PA.1.1.6.1	Propulsion Power
PA.1.1.6.2	Avionics Power
PA.1.1.6.3	Actuation Power
PA.1.1.6.4	PV Regeneration System
PA.1.1.7	Contingency Management System (CMS)
PA.1.2	Ground Control System (GCS)
PA.1.2.1	Primary Ground Control Station
PA.1.2.2	Secondary Ground Control Station
PA.1.2.3	CPDS Payload Ground Station (PGS)
PA.1.3	Flight Operation and Procedures (OPS)
PA.1.3.1	Runway
PA.1.3.2	Airspace

Figure 1: Physical breakdown with Avionics subsystem highlighted

1.2 UAV Avionics

The avionics onboard the UAV is the computer-based control system that accepts pilot input (or pre-programmed / uploaded commands), processes the input, and regulates the flight control surfaces. Figure 2 shows a conceptual layout of the avionics system, in the context of the UAS.

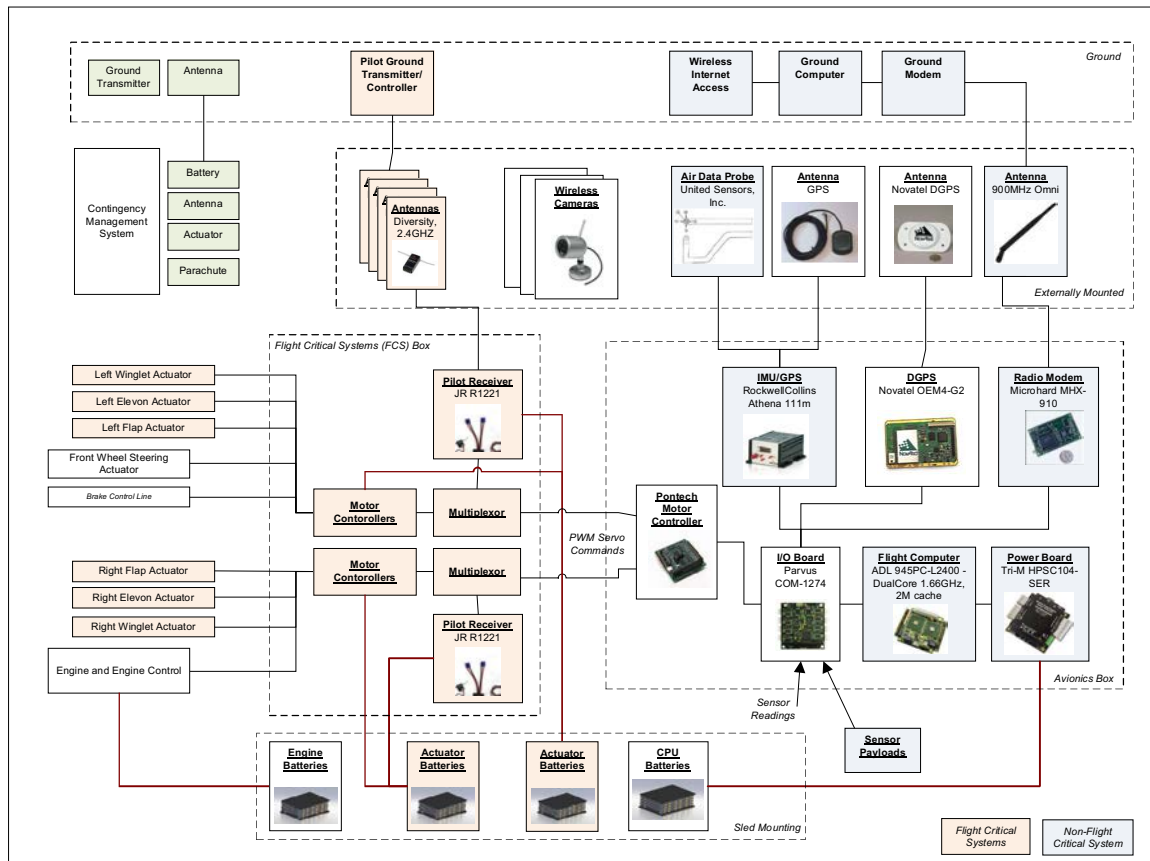


Figure 2: Avionics system conceptual layout

1.3 Concept of Operations

Figure 3 shows one of the concepts of operation for the Swift UAS. As seen in the figure, there are seven distinct phases of flight:

- (i) Takeoff
- (ii) Climb
- (iii) Cruise
- (iv) Survey
- (v) Return Cruise
- (vi) Descent
- (vii) Land

For this particular mission, the aircraft is operated out of line of sight, i.e., it will not always be within sight of the pilots or the GCS operator. However, there are safety observers stationed at pre-defined locations on range, so as to ascertain the aircraft continues to be on range, and to identify situations in the environment that are potentially hazardous.

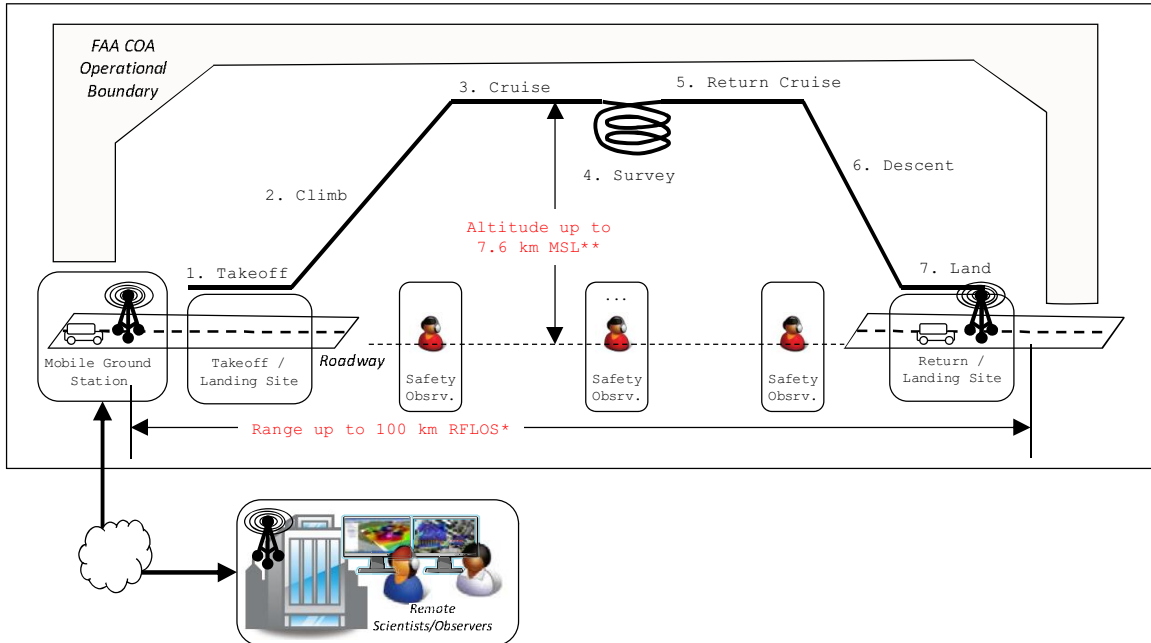


Figure 3: Concept of operations, showing different mission phases.

The dependability analysis of the Swift UAS will need to take into account the notion that during the different phases of flight, not all hazards pose the same level of safety risk, e.g., loss of power is a hazard at any point during flight, whereas loss of the autopilot need not be a hazard during landing, but could be a hazard during *cruise*, *survey*, *return cruise*, etc.

1.4 Autopilot Software

See attached document: Corey Ippolito, “An Autonomous Autopilot Control System Design for Small-Scale UAVs”.

2 Dependability Analysis Requirements

This section specifies the analyses potentially beneficial for safety assurance of the system described in Section 1.

1. Fault tree analysis and / or Event tree analysis
 - a. Generation of minimal cut sets
 - b. Probabilistic analysis of failure probabilities for hazardous top events
 - c. Event chains leading to system failure (or an undesirable outcome)
 - d. Probabilistic analysis o
2. Hazard analysis using a combination of fault trees and event trees
3. Phased-mission analysis.

Overview of the NASA Swift UAS



**Presentation to Professor Joanne Dugan
Dependable Computing Class
University of Virginia
April, 2012**

Corey Ippolito¹, Ganesh Pai², Ewen Denney³

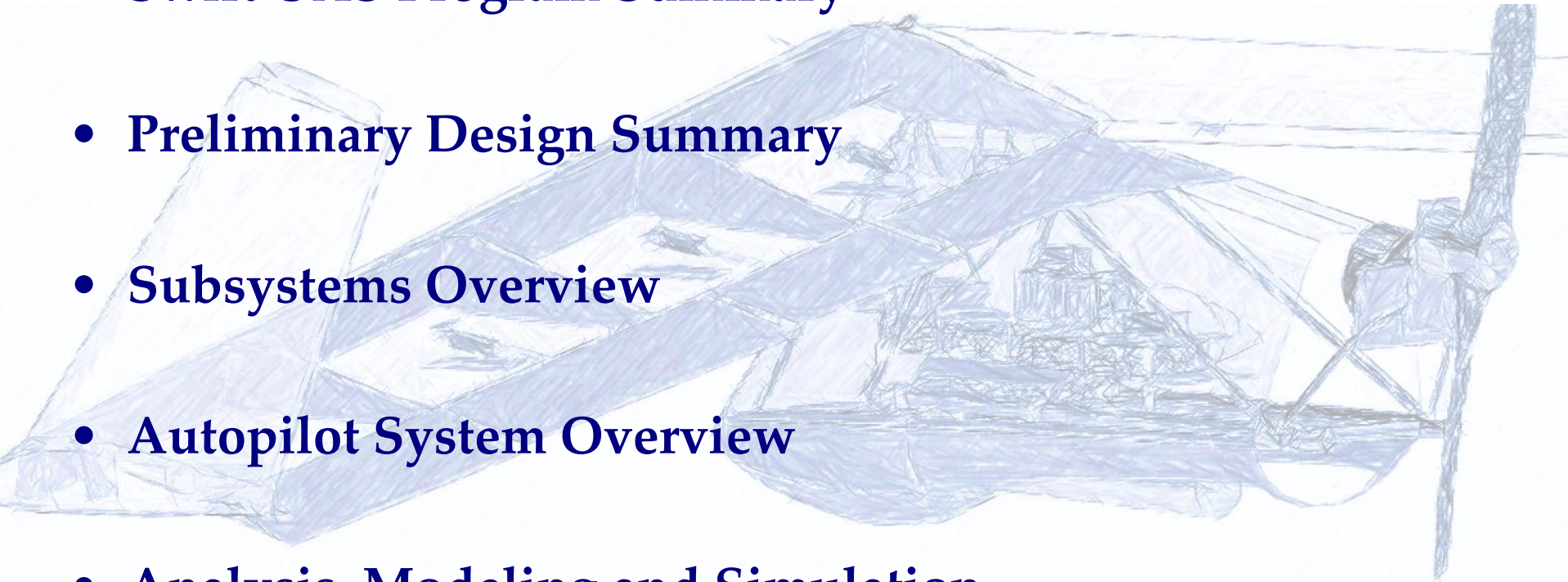
Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA
corey.a.ippolito@nasa.gov

- 1. Aerospace Scientist, NASA Ames Research Center*
- 2. Research Scientist, Stinger Ghaffarian Technologies, Inc.*
- 3. Senior Computer Scientist, Stinger Ghaffarian Technologies, Inc.*



Presentation Outline

- **Swift UAS Program Summary**
- **Preliminary Design Summary**
- **Subsystems Overview**
- **Autopilot System Overview**
- **Analysis, Modeling and Simulation**

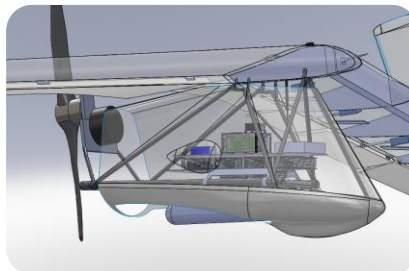
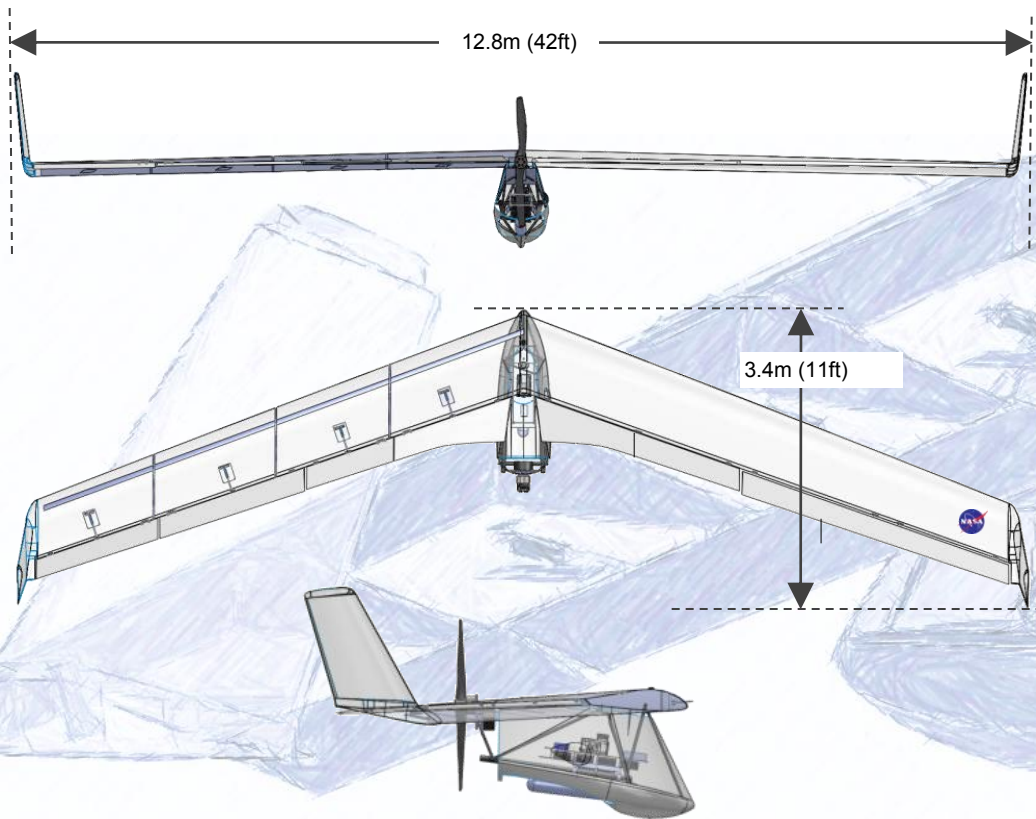


Goal of the Swift UAS

- **Develop a fully autonomous, fully electric, zero-emissions human flight capable research vehicle platform using today's technologies and applicable for next generation aircraft research**
- **Build on existing Bright Star Gliders SWIFT airframe, a high performance glider**
- **Utilizes GOTS/COTS control hardware, algorithms, and infrastructure available at NASA Ames Research Center.**



Swift UAS Specifications



Geometry

Wing Span:	12.8m (42ft)
Length:	3.4m (~11ft)
Wing area:	12.5 m ² (136 ft ²)
Aspect ratio:	12.9

Speeds

Speed, Cruise:	45 knots (23 m/s)
Speed, Stall:	20 knots (10 m/s)
Speed, V _{NE} :	68 knots (35 m/s)

Weight and Payload

Weight, Empty:	48 kg (106 lb)
Weight, Max Take Off :	145 kg (320 lbs)
Payload Capacity:	100kg (220lbs)

Performance

Maximum glide ratio:	27:1
Rate of sink:	36 m/min (118 ft/min)
Powered Rate of Climb:	2.5 m/s
Est Max Endurance:	~7 hours
Est Max Range:	~500 km
Takeoff Field Length:	50m
Service Ceiling:	>7.6km (>25,000 ft)

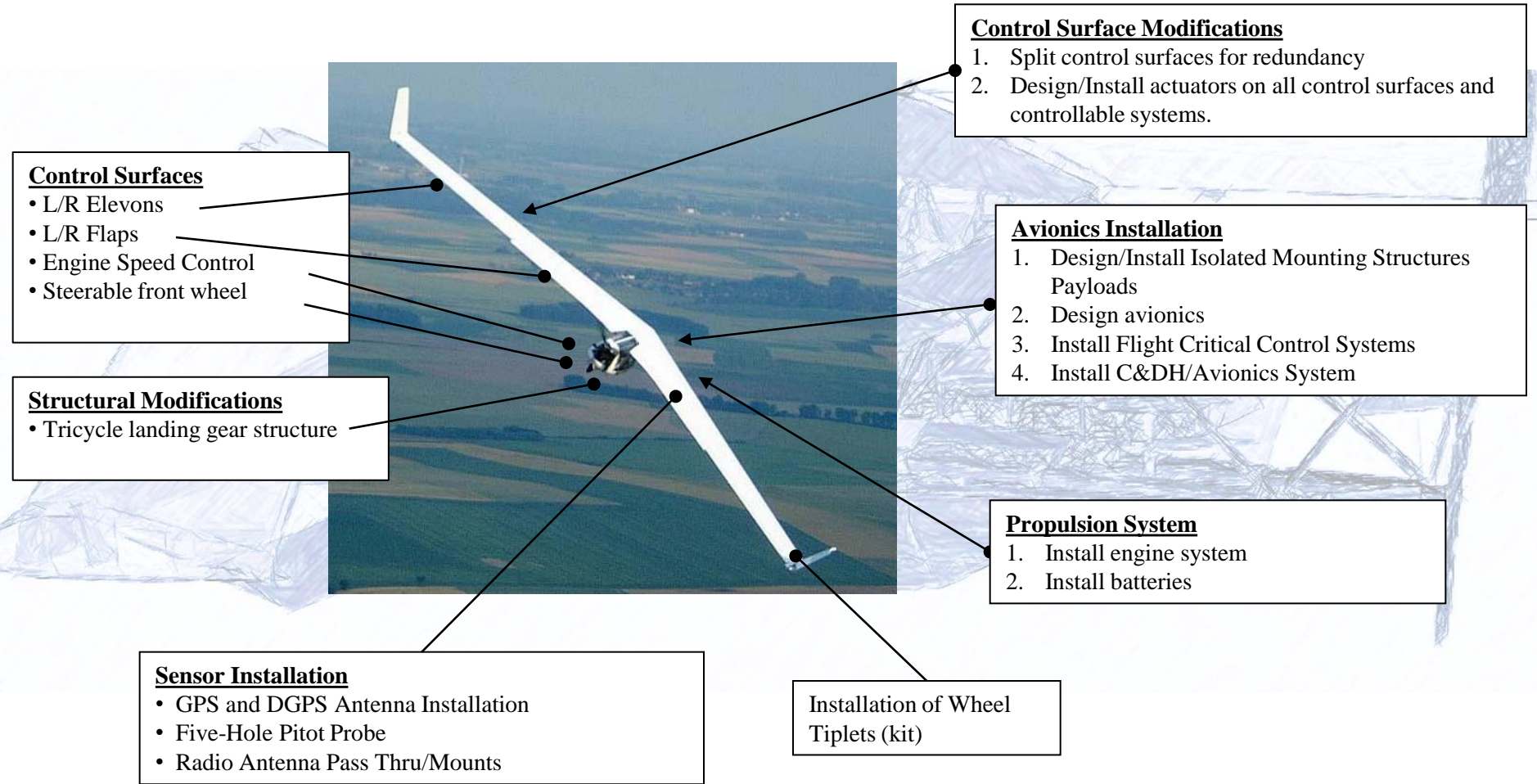
Control and Propulsion

Engine:	Electric Motor, Pusher Prop Config.
Propulsion:	Electraflyer 18hp/13.5kw DC Motor, 5.6KW-hr Li-Po
Propeller:	1.5m (60") carbon-fiber

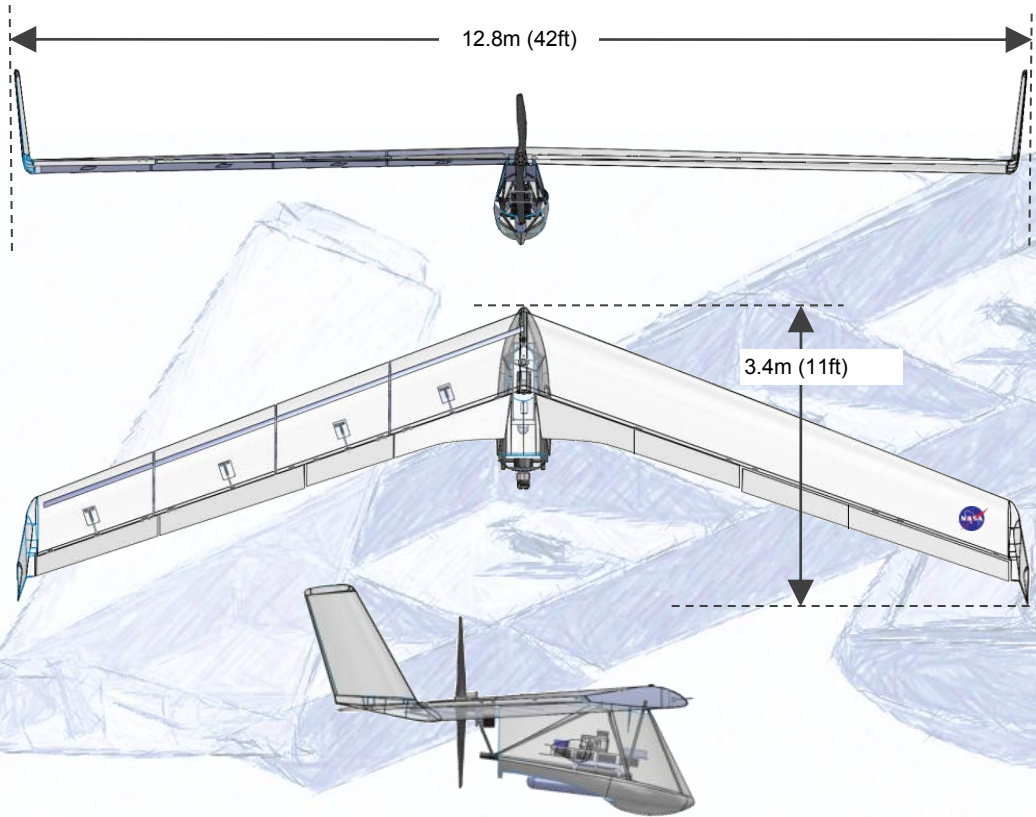
Materials

All Composite Wing Structure (Aramid/Carbon)
Steel/Aluminum Fuselage Frame

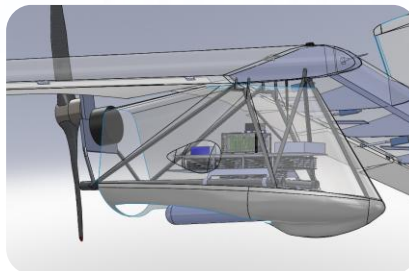
Flight Vehicle System : Modifications



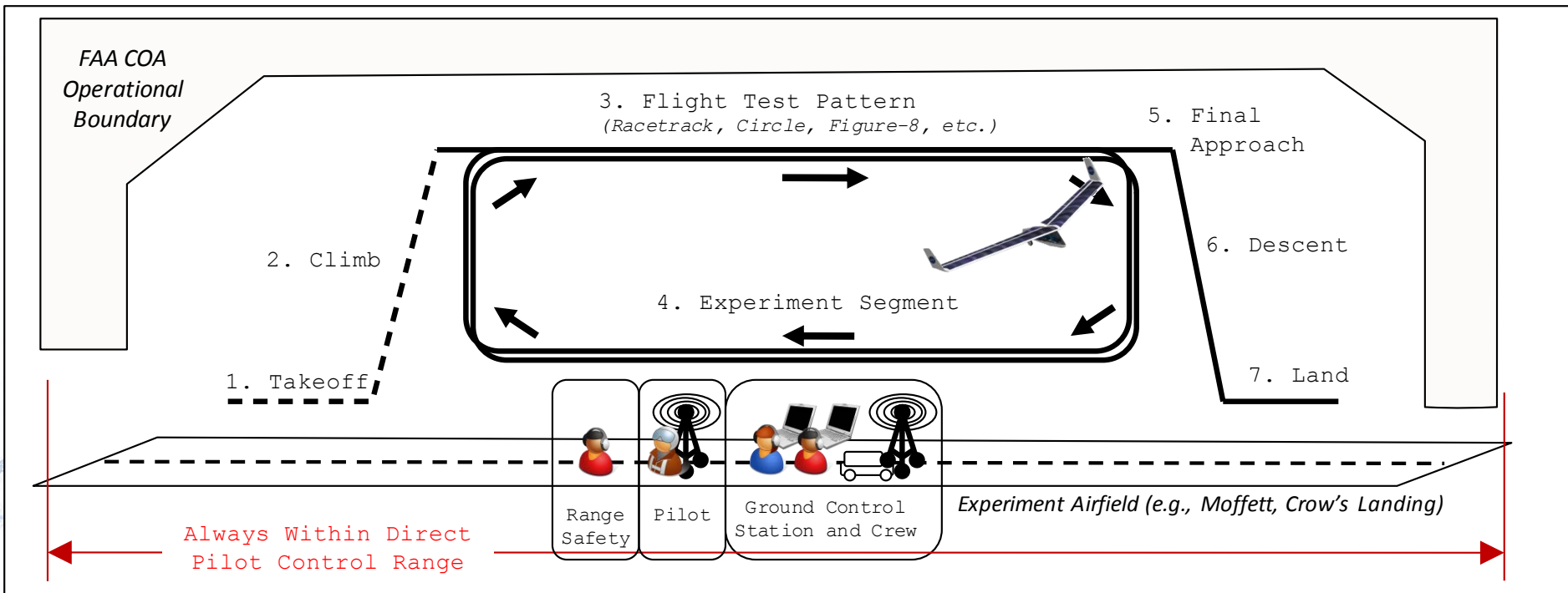
Swift's Unique Characteristics



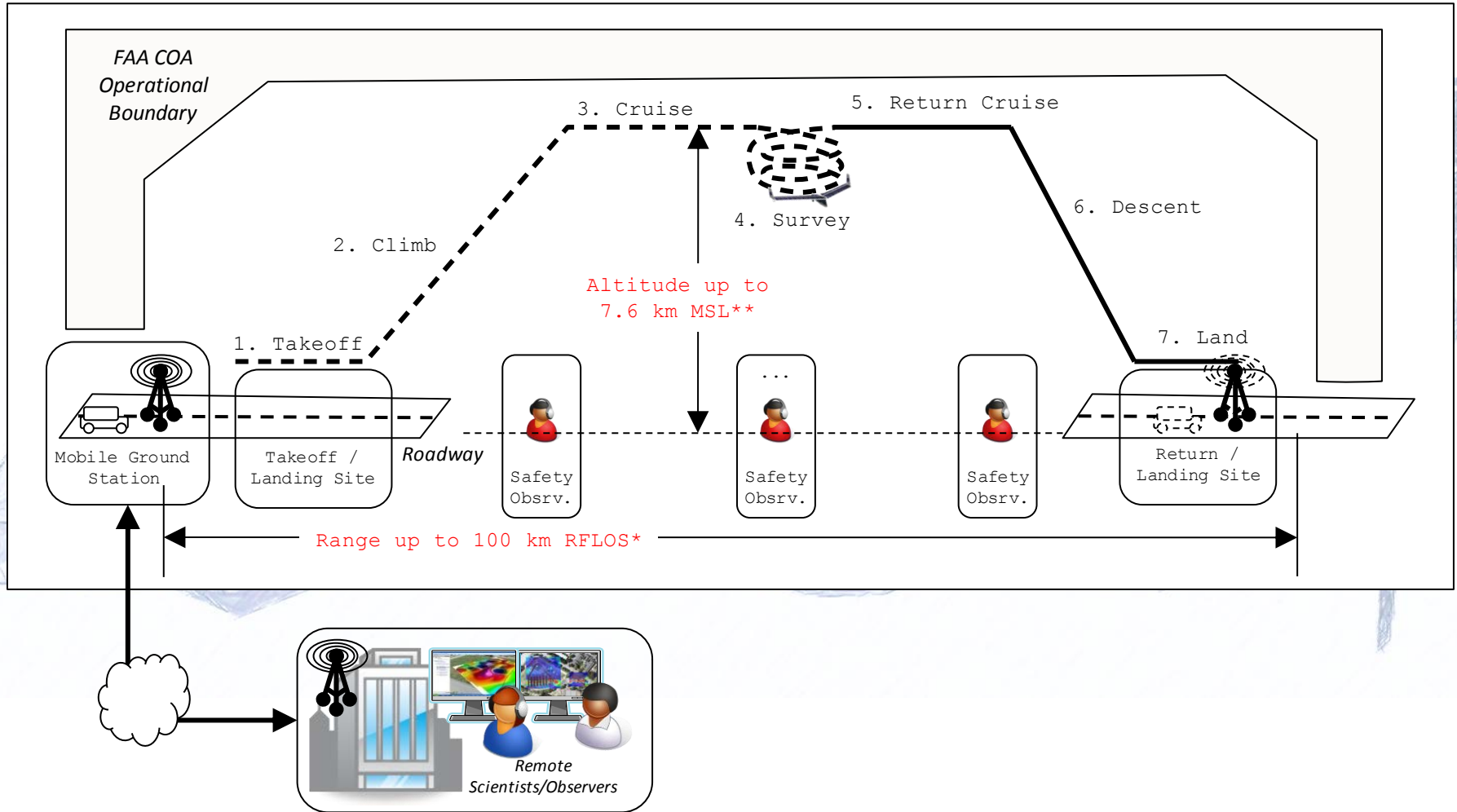
- Low cost reconfigurable platform with characteristics of larger aircraft
- Large flexible wing (13m/42ft span)
 - Wing structure is light-weight, flexible, simple, all-composite
- Redundant actuation
- Large payload capacity (~200lbs limited by COA)
- All-electric, zero emissions platform
- Slow cruise speeds ideal for long duration test segments
- Capable high altitude flight
- Quiet airframe structure (noise, emissions, EMI, etc.)
- Custom autopilot/navigation system
- Fully redundant and fully instrumented for local or remote autonomy
- Immediate pilot preemption over redundant links for safe/immediate termination of experiments



Control Experiment Mission Concept

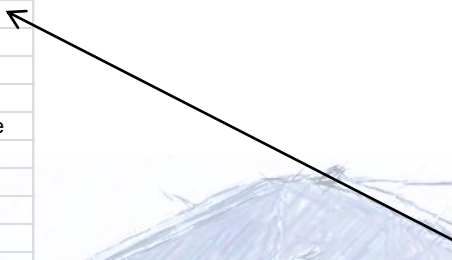


Remote Survey Mission Concept

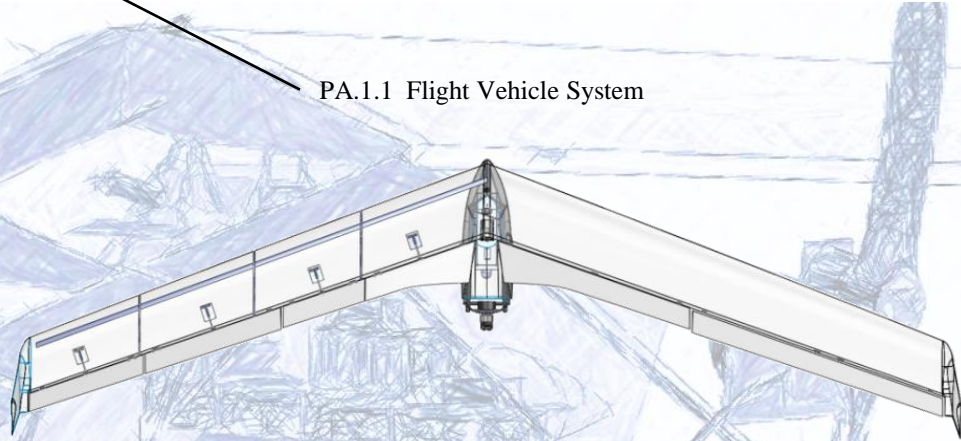


Swift UAS Systems Breakdown

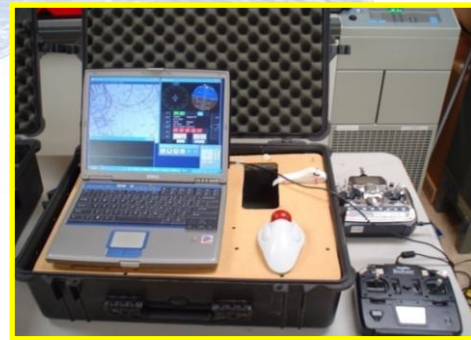
PA.1.0	Complete Swift UAS System
PA.1.1	Flight Vehicle System (FVS)
PA.1.1.1	Avionics System (AVS)
PA.1.1.1.1	Flight Critical Control System (FCS)
PA.1.1.1.2	Control and Data Handling System (CDHS)
PA.1.1.1.2.1	CDHU (Control and Data Handling Unit) Hardware
PA.1.1.1.2.2	FMS (Flight Management System)
PA.1.1.1.2.3	ESS (Embedded Software Systems)
PA.1.1.1.2.4	AP (Autopilot System)
PA.1.1.1.3	Communication Systems (COM)
PA.1.1.1.4	Flight Sensors (SENS)
PA.1.1.1.4.1	INS (ADHRS, IMU)
PA.1.1.1.4.2	GPS
PA.1.1.1.4.3	Air Data System (AIRDAT)
PA.1.1.1.4.4	DGPS
PA.1.1.2	Payload and Mission Data Systems (PYLD)
PA.1.1.2.1	Common Payload Data System (CPDS)
PA.1.1.3	Actuation System (ACT)
PA.1.1.3.1	Control Surface Actuation (CS)
PA.1.1.3.2	Steering and Brake Actuation
PA.1.1.4	Propulsion System (PRLP)
PA.1.1.4.1	Electric Motor Propulsion System
PA.1.1.5	Structures
PA.1.1.5.1	Fuselage Structure
PA.1.1.5.2	Fuselage Outer Mold Line
PA.1.1.5.3	Propulsion Support
PA.1.1.5.4	Avionics Mounting Structure
PA.1.1.5.5	Wing Structure
PA.1.1.5.6	Control Surfaces
PA.1.1.6	Electrical and Power System (EPS)
PA.1.1.6.1	Propulsion Power
PA.1.1.6.2	Avionics Power
PA.1.1.6.3	Actuation Power
PA.1.1.6.4	PV Regeneration System
PA.1.1.7	Contingency Management System (CMS)
PA.1.2	Ground Control System (GCS)
PA.1.2.1	Primary Ground Control Station
PA.1.2.2	Secondary Ground Control Station
PA.1.2.3	CPDS Payload Ground Station (PGS)
PA.1.3	Flight Operation and Procedures (OPS)
PA.1.3.1	Runway
PA.1.3.2	Airspace



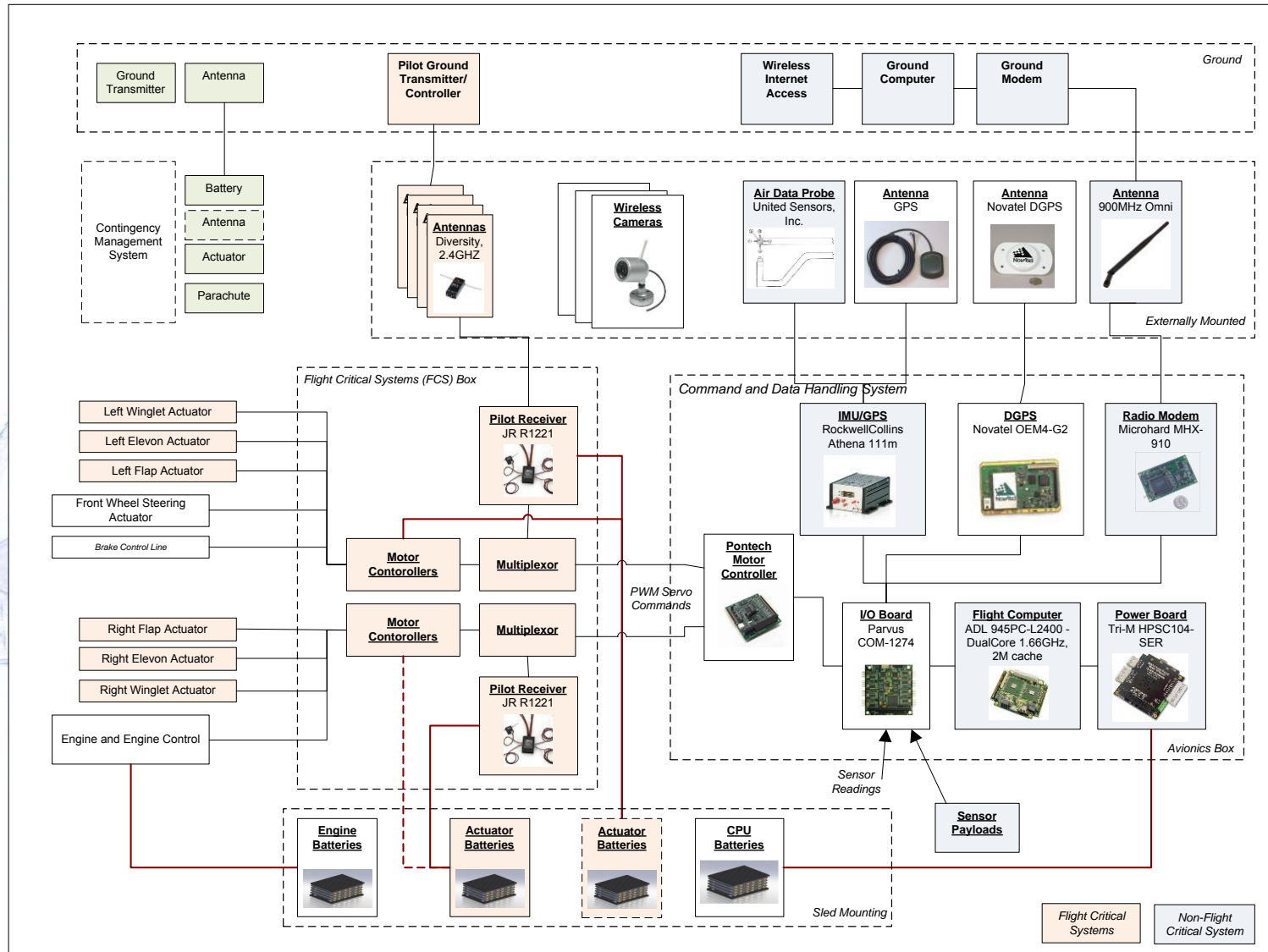
PA.1.1 Flight Vehicle System



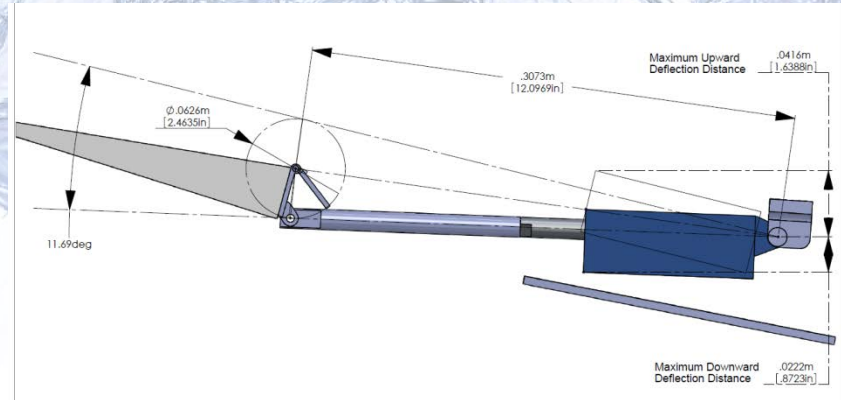
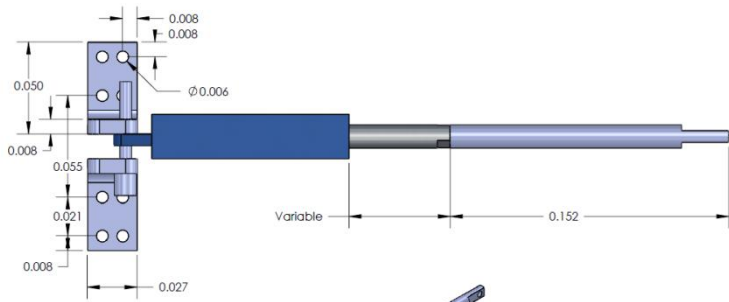
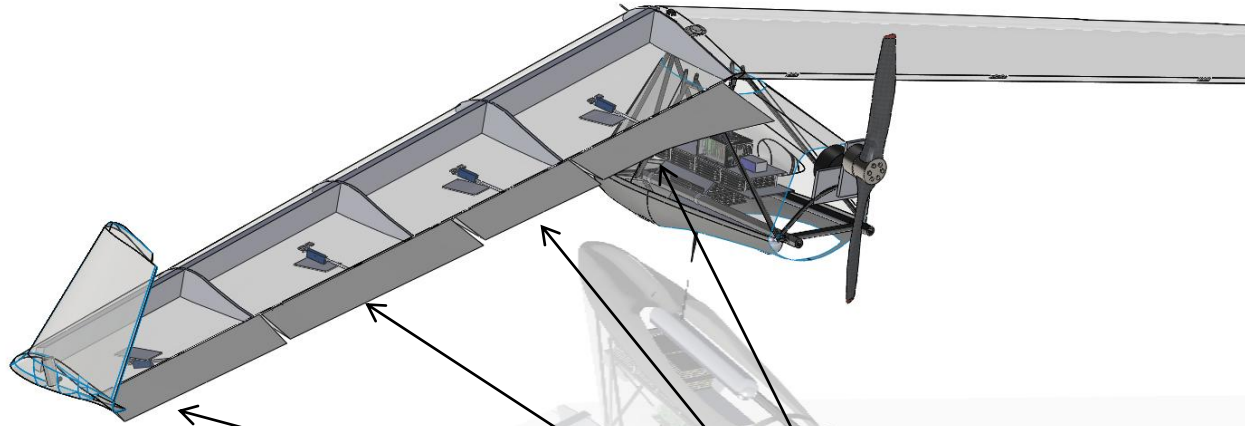
PA1.2 Ground Station



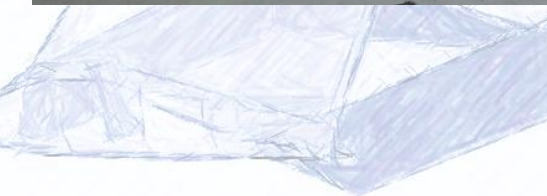
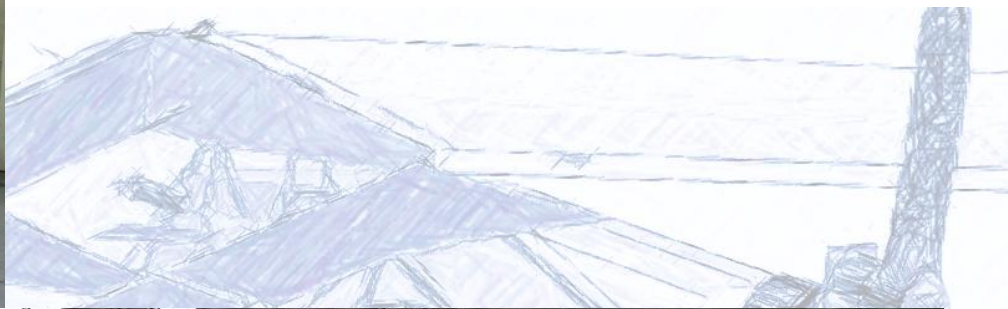
Conceptual Avionics Layout



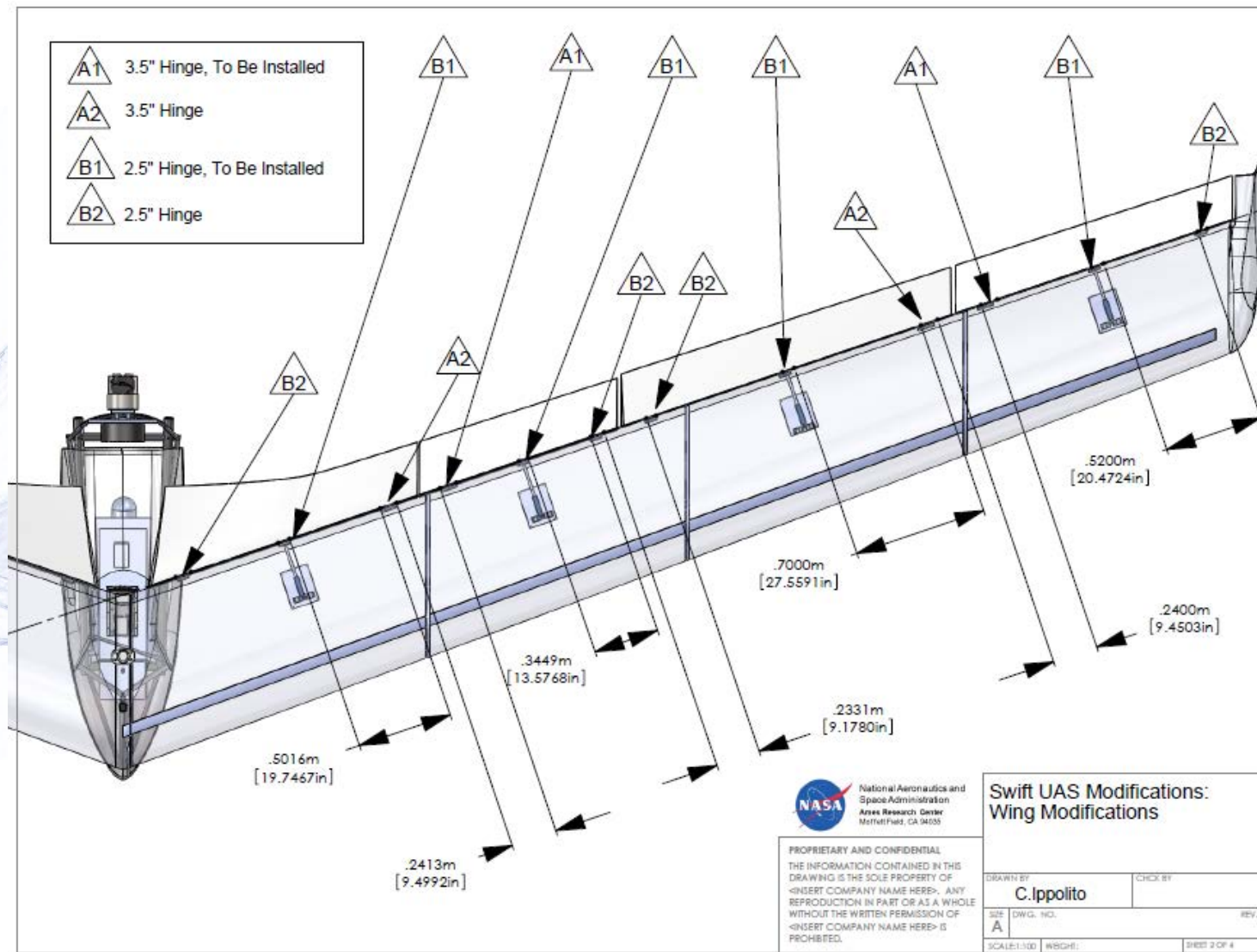
Swift UAS Actuation System



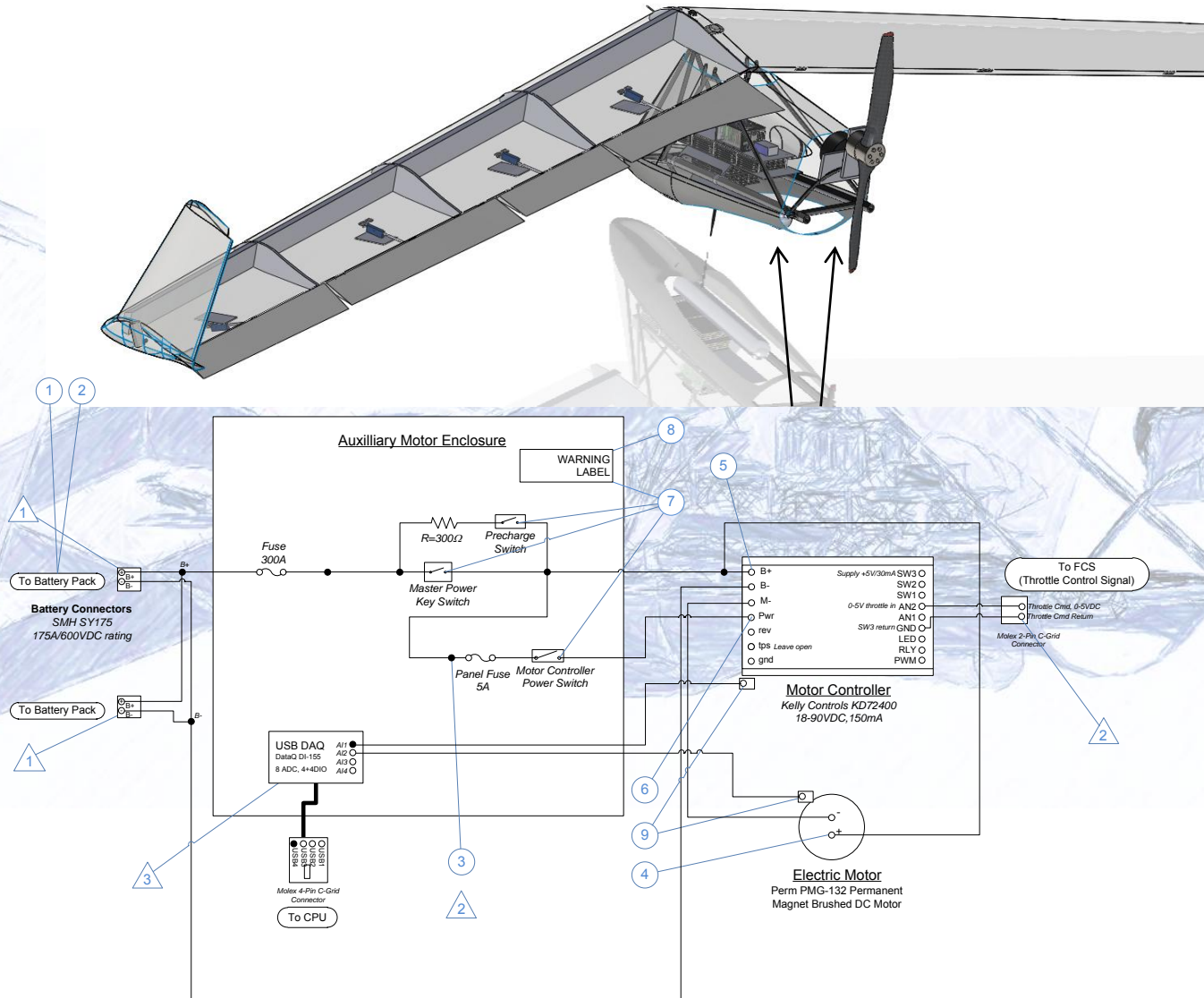
Swift UAS Landing Gear System



Actuation: Control Surface Specification



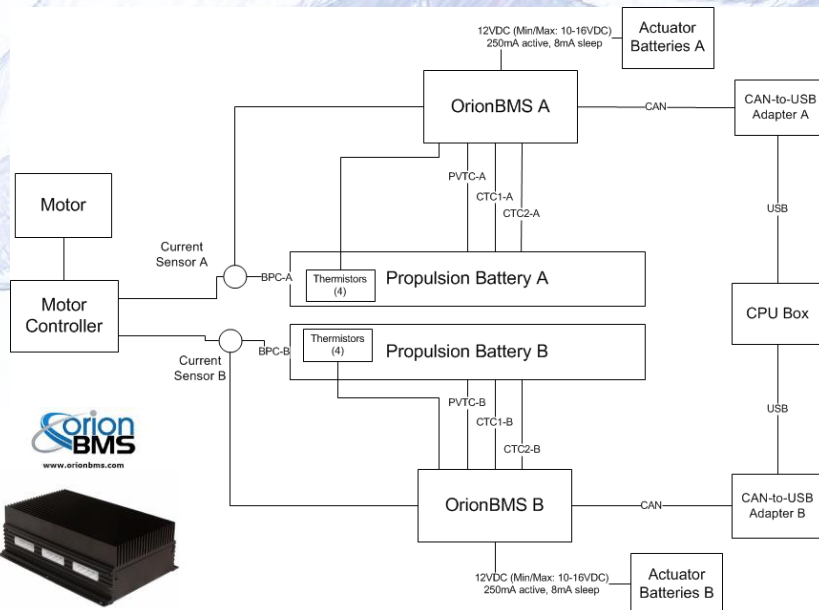
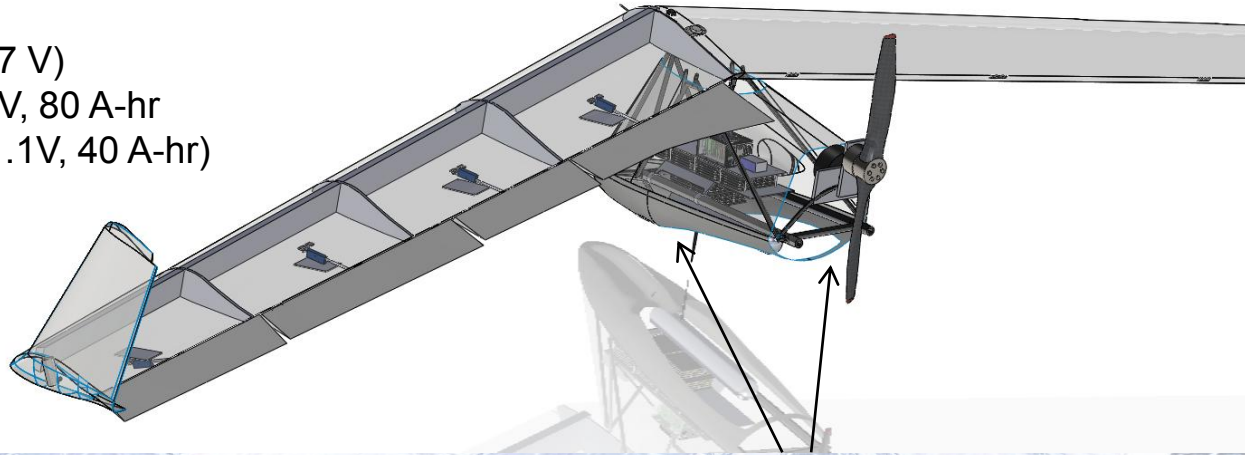
Swift UAS Propulsion System



Swift UAS Propulsion Power

Battery System

- 144 cells (10 A-hr, 3.7 V)
- 18S, 8P Pack = 66.6V, 80 A-hr
- Unit cell = 3S, 4P (11.1V, 40 A-hr)
- 80 lbs weight



Battery Packs

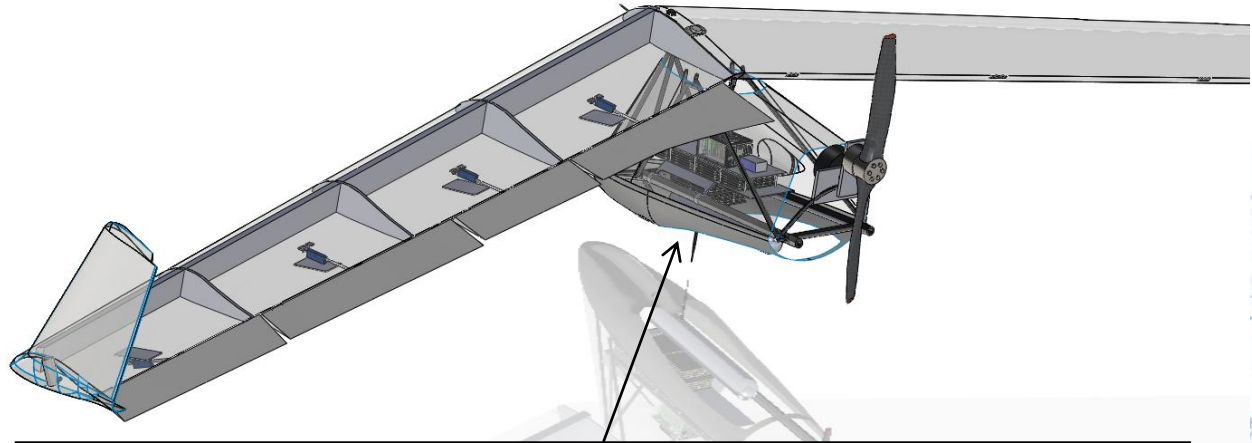


Installation

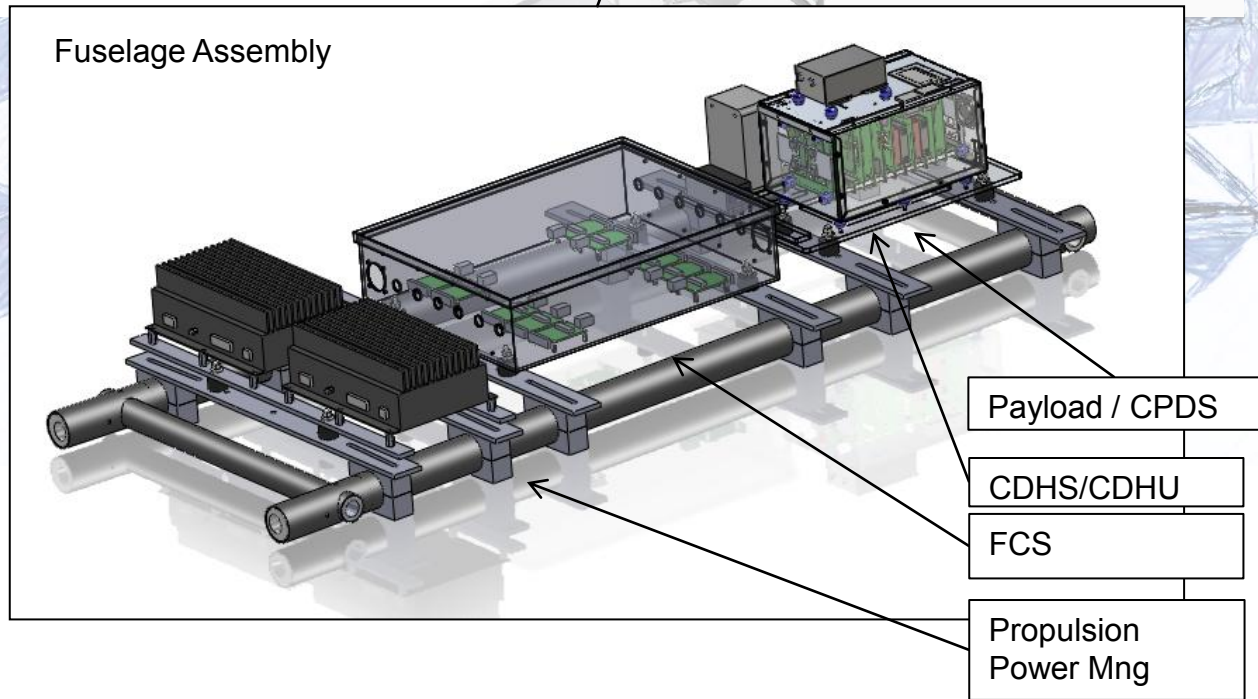


Battery Enclosure

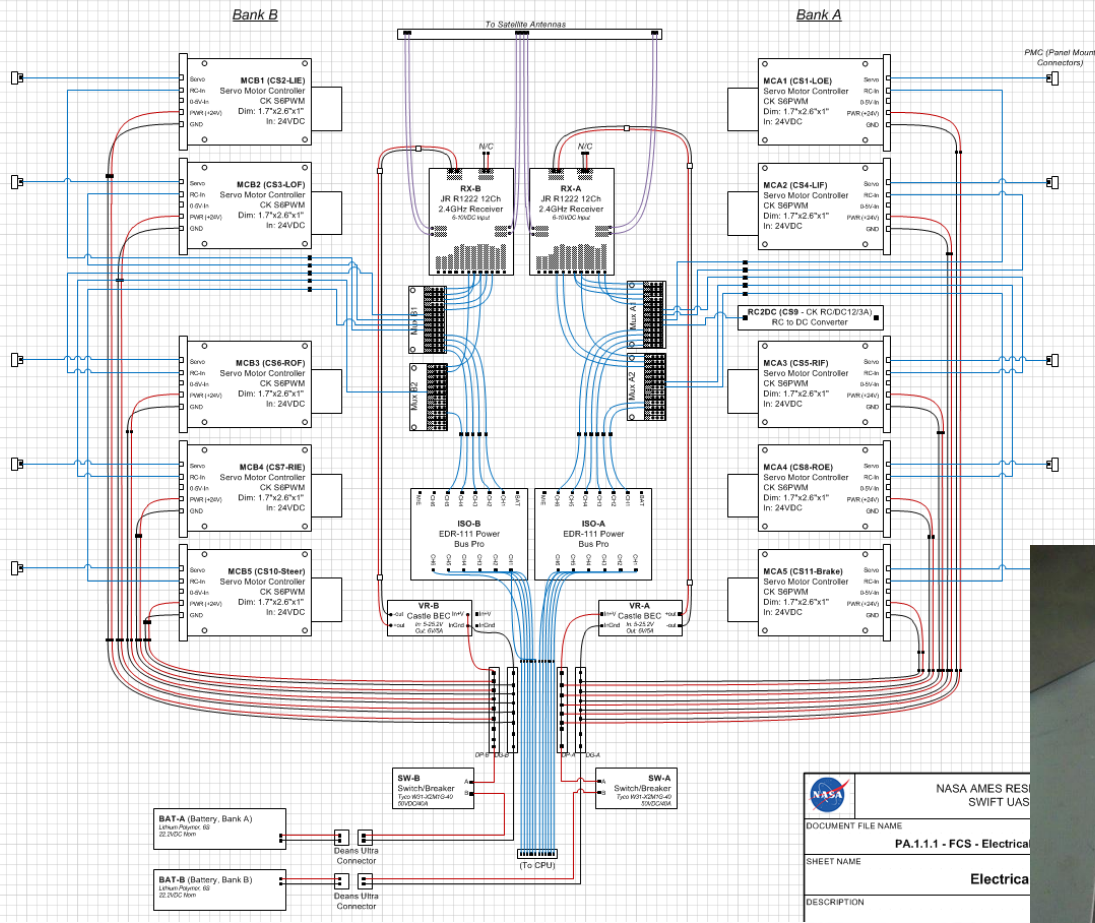
Swift UAS Avionics



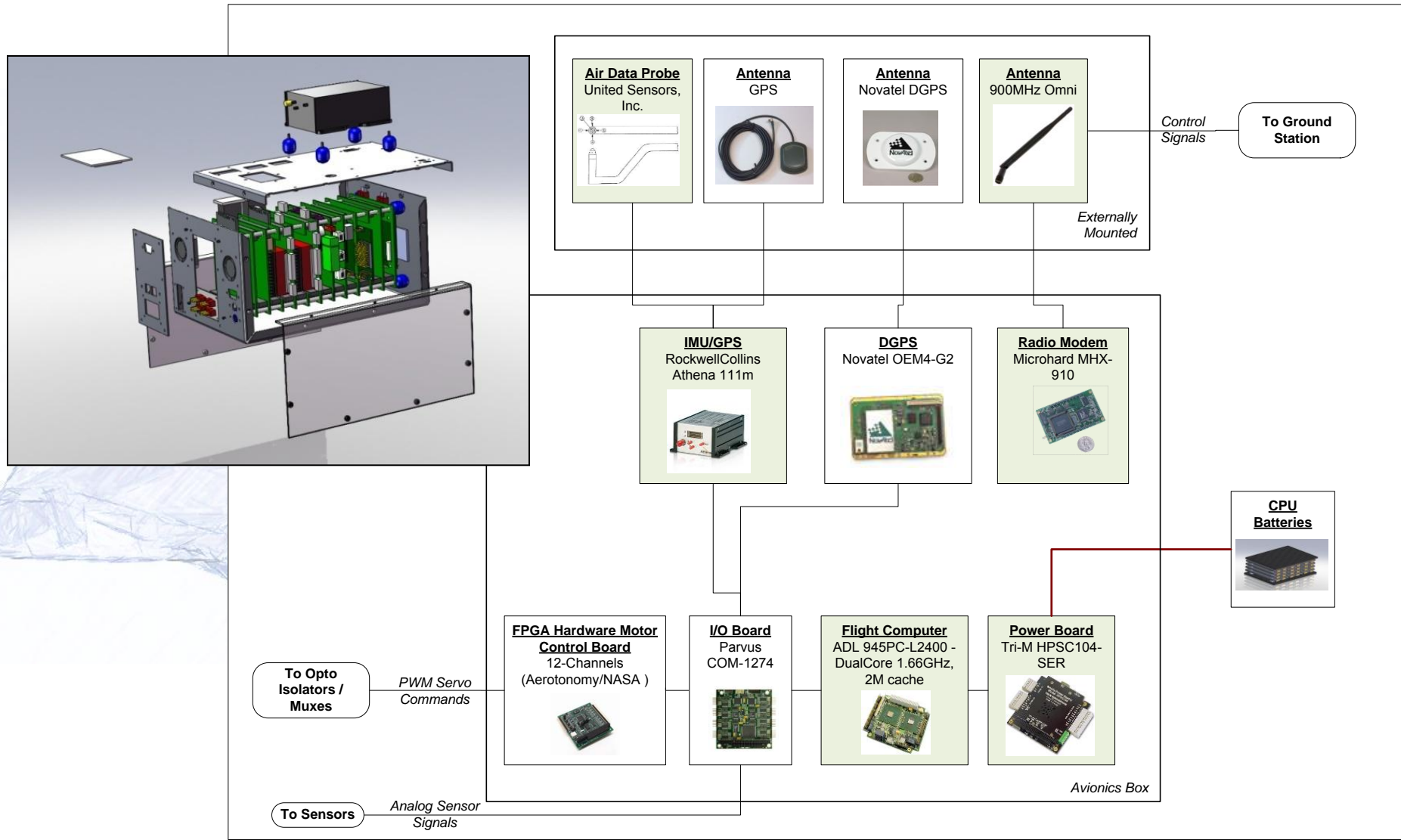
Fuselage Assembly





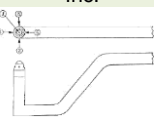

FCS (Flight Critical Systems)

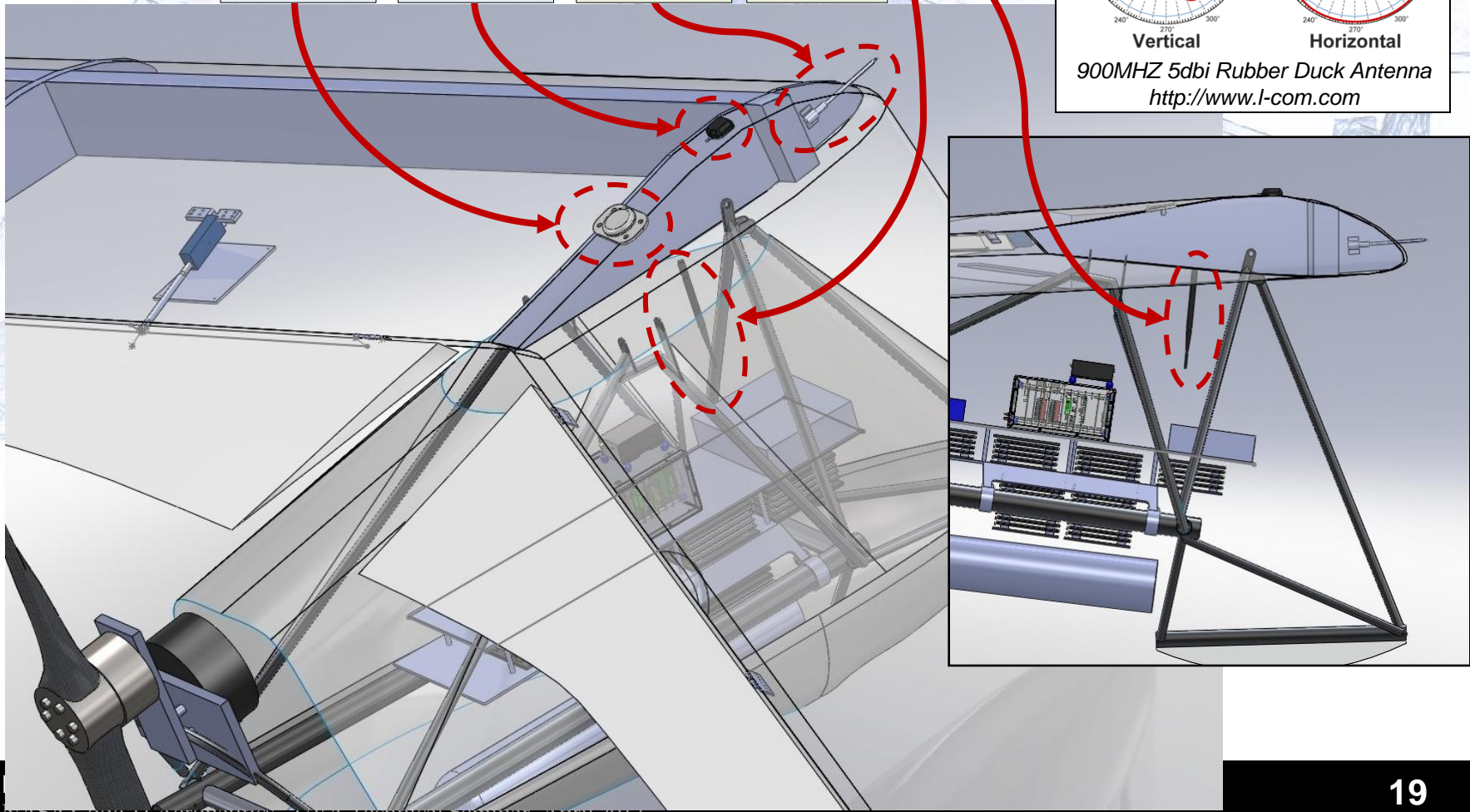
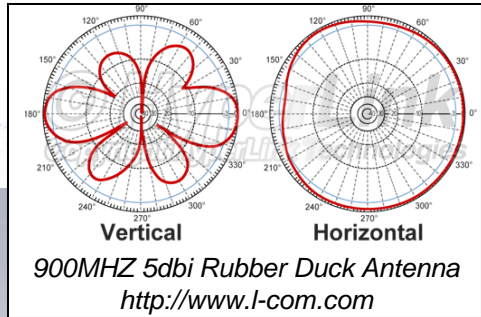


Instrumentation: CDHU



Instrumentation: Antennas and Probes

<p>Antenna Novatel DGPS</p> 	<p>Antenna GPS</p> 	<p>Air Data Probe United Sensors, Inc.</p> 	<p>Antenna 900MHz Omni</p> 
--	---	---	---



Ground Control System (GCS)

Reusing a well-tested redundant ground station configuration (EAV vehicle is shown).

Wireless Communications:

- Microhard Spectra 910 Wireless Modem (1 Watt Transmitter)

Wireless Communications:

- 6dB 900Mhz Omnidirectional Antenna, 5ft Fiberglass



CPU

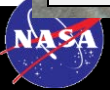
- Laptop Control Station with Reflection Architecture for UAS C&C through avionics CPU.

Controllers

- USB Controller through avionics CPU
- JR DSX9 2.4GHz DSM2 (bypass CPU)



Two completely separate, identical, and redundant ground stations.



Autopilot Controller Overview

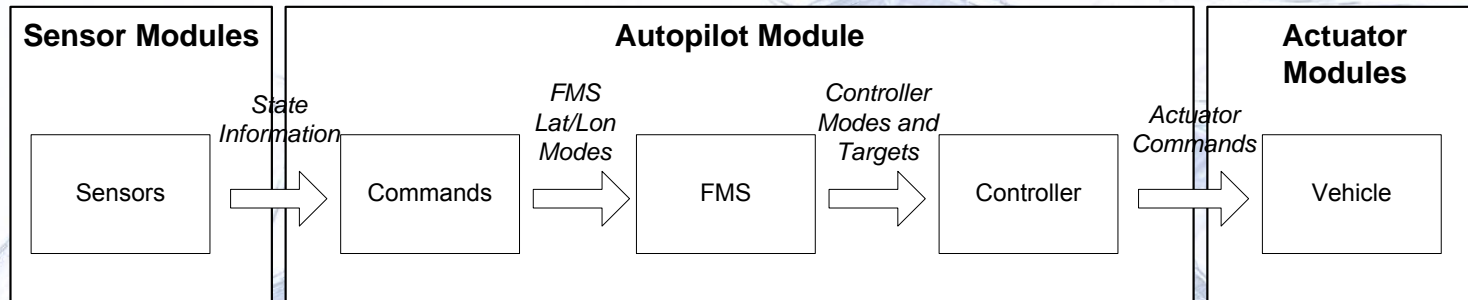


Figure. Conceptual Data Flow and Components

Sample Mission Script Commands

// Program the Autopilot System.

```
objAutopilot.ClearCommandList();  
objAutopilot.AddCommand_Takeoff ( 4500, -2000, 500.0, 80.0, 1000.0, -0.34907, 50.0 );  
objAutopilot.AddCommand_FlyToTrack ( 8070.9, -6009.7, 500.0, 80.0, 1000.0 );
```

// Following heading until specified altitude is reached.

```
objAutopilot.AddCommnd_SetFMSSMode ( HEADINGHOLD, 0.0, ALTCMD, 400.0, SPDCMD, 80,  
                                     ALTITUDE, GT, 300 );
```

// Follow a preprogrammed path.

```
cmd=objAutopilot.AddCommand_FlyToTrack ( 10107.0, -1390.7, 500.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( 8904.5, -902.8, 500.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( 6919.8, -5568.9, 500.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( 3357.4, 1364.9, 500.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( -3319.9, 4457.2, 500.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( -5436.1, 3980.7, 500.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( -4371.0, 1696.2, 200.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( -741.1, 182.6, 50.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_FlyToTrack ( 2522.8, -1267.5, -10.0, 80.0, 1000.0 );  
objAutopilot.AddCommand_Jump ( cmd );
```

Autopilot Controller

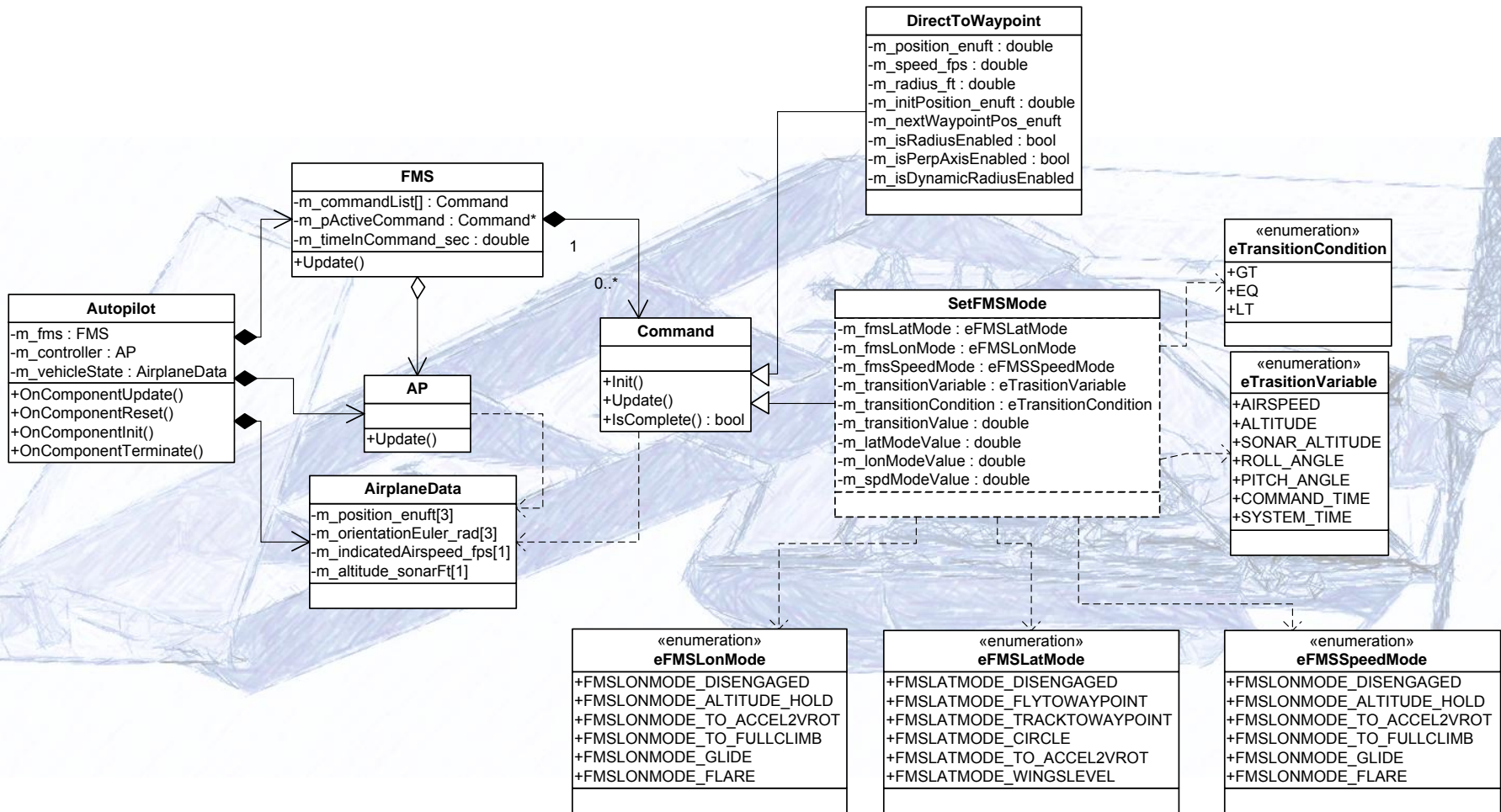
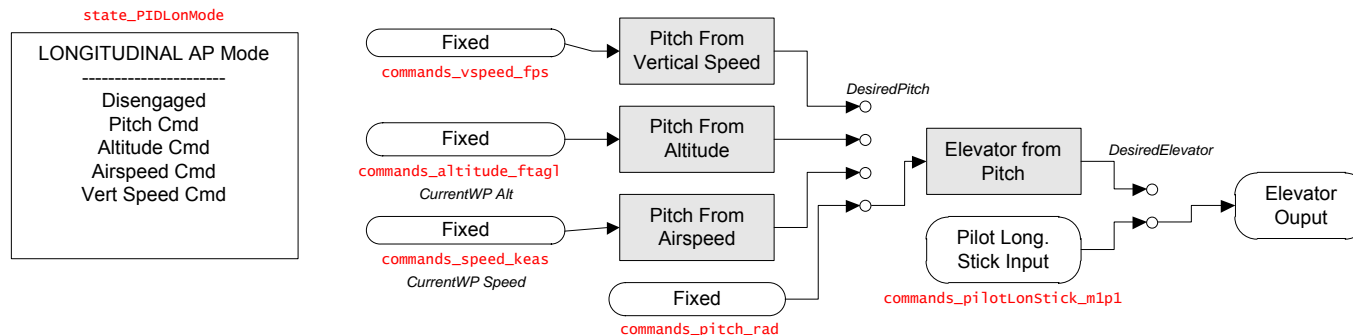
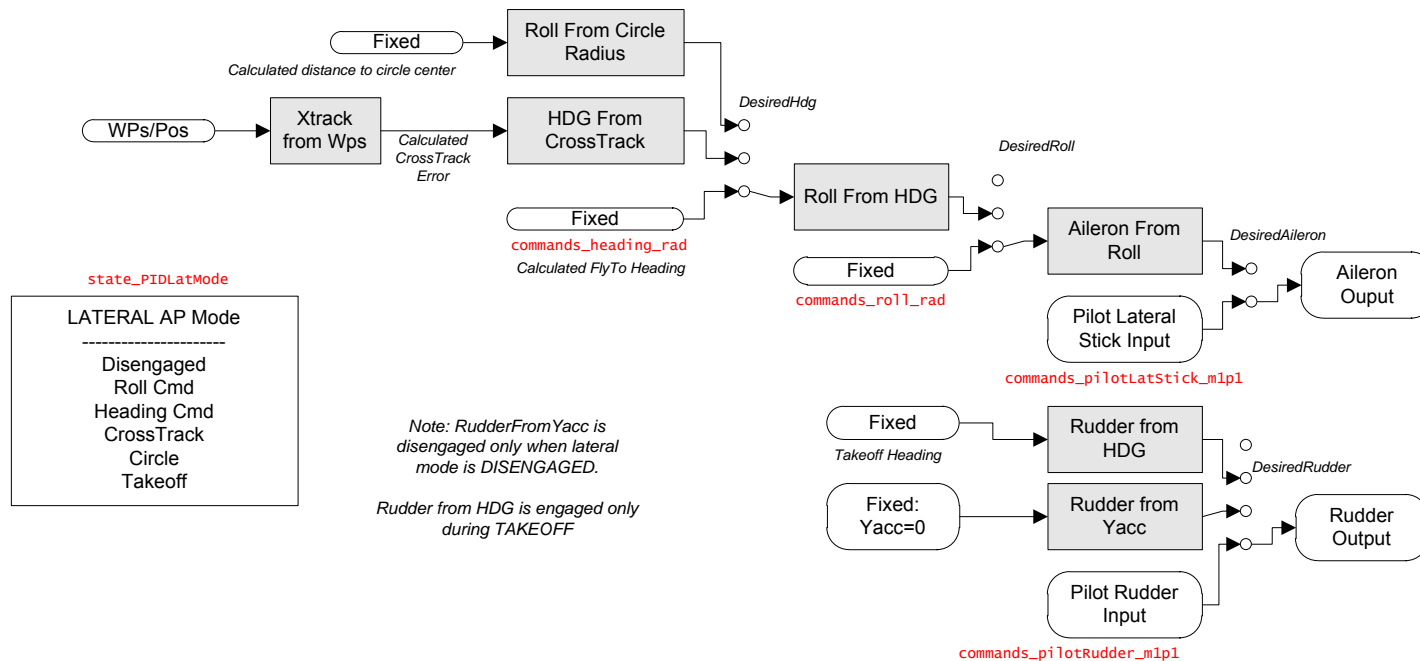


Figure. Top Level Classes

Autopilot Controller Architecture

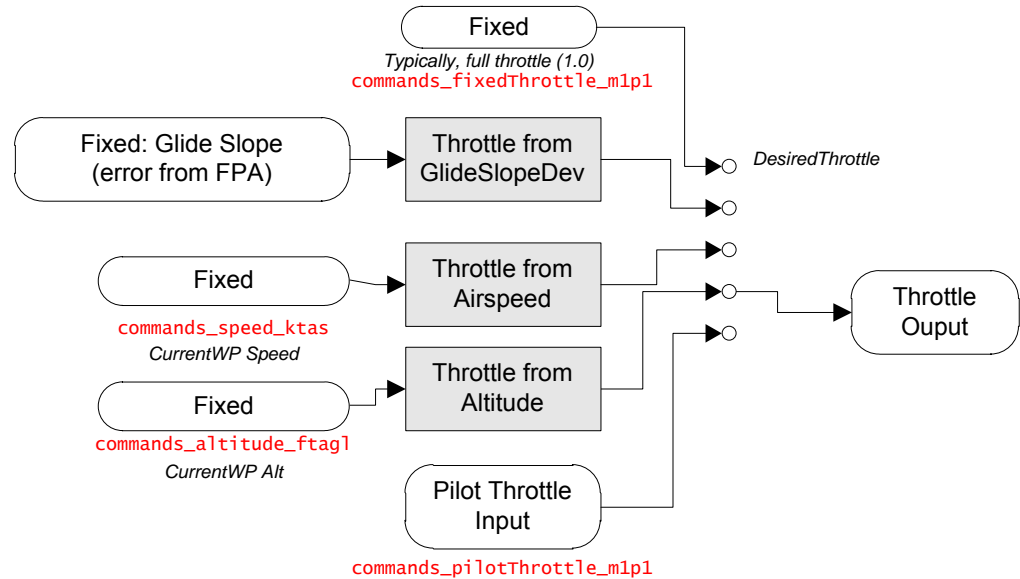


Autopilot Controller Architecture

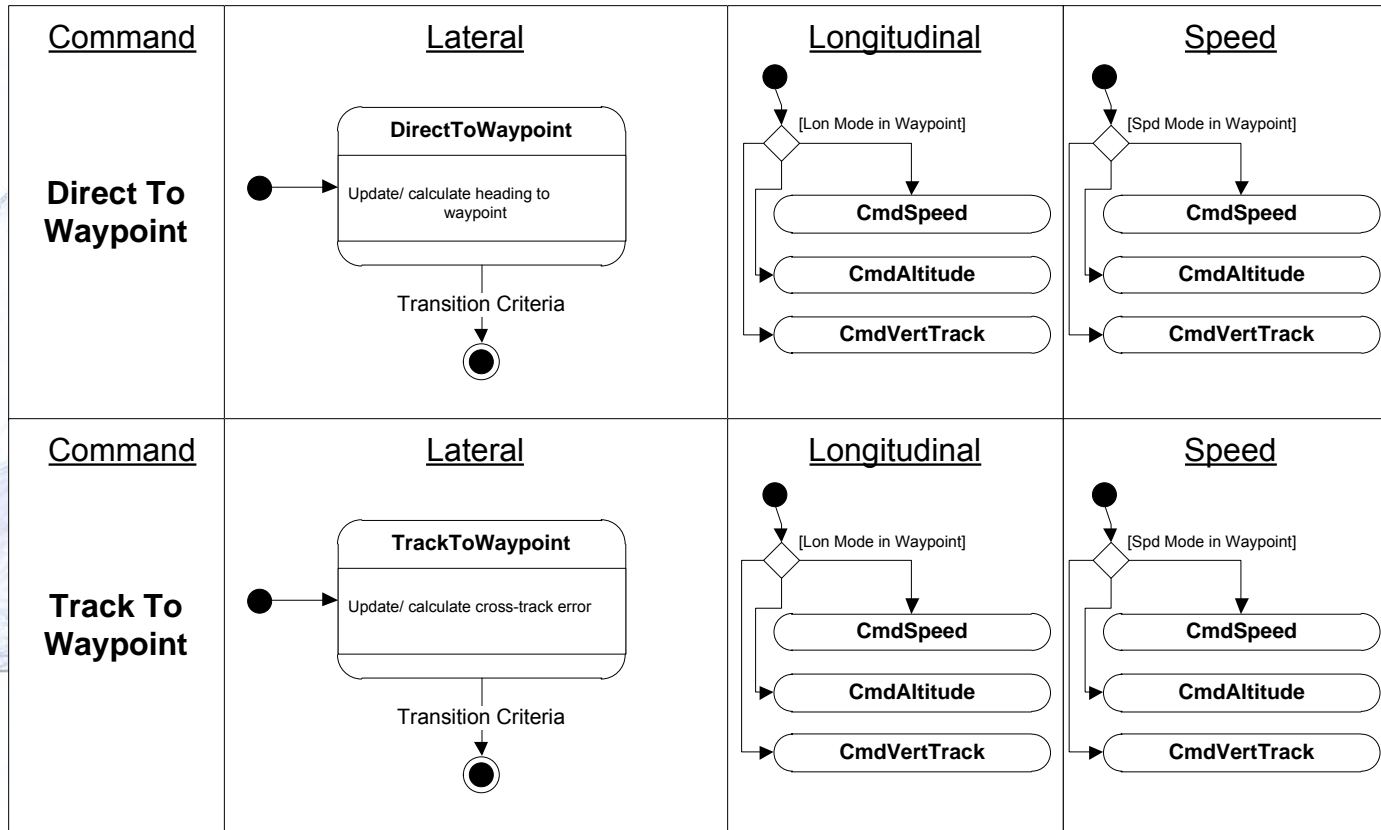
state_PIDSpdMode

SPEED AP Mode

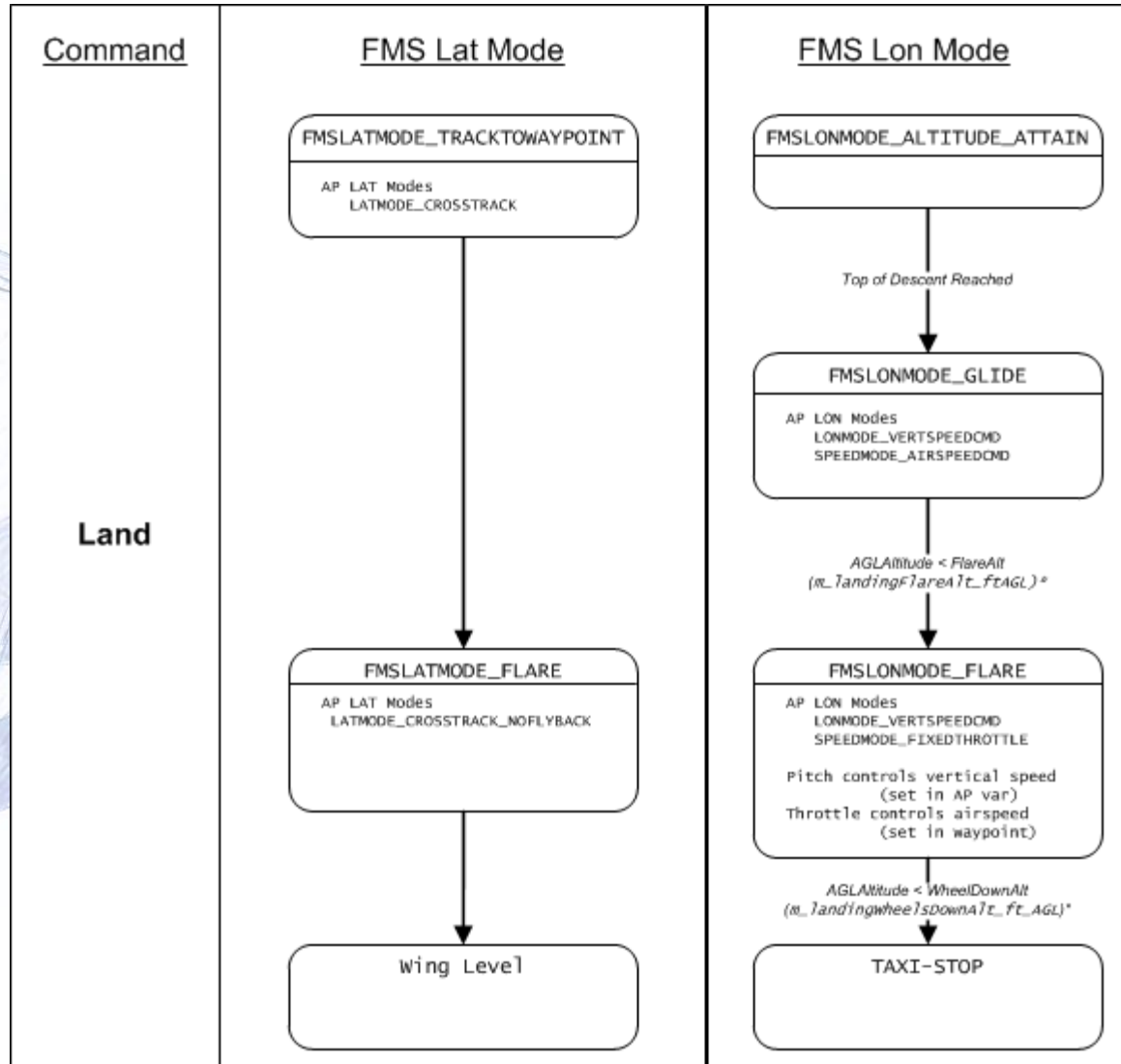
-
- 0 = Disengaged
- 1 = Fixed Throttle
- 2 = Airspeed Cmd
- 3 = Altitude Cmd
- 4 = Glide Slope



Example FMS State Diagrams

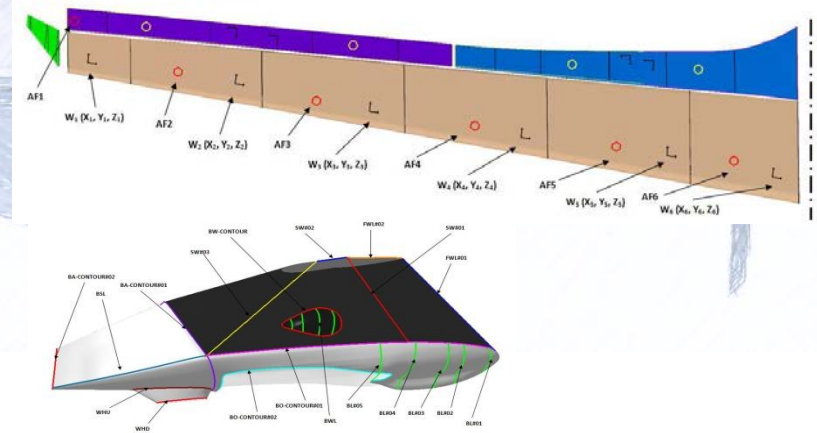


Example FMS State Diagrams



Geometric Modeling

- Geometric modeling utilized to derive computational fluid models, dynamics models
- Utilized robotic FaroArm for 3D geometric data acquisition and model generation
- Error Analysis
 - Position error maximum ± 9.00 mm
 - Variation of the drag cross section: -2.78% to 2.73%
 - Variation of the lift cross section: -0.56% to 0.57%

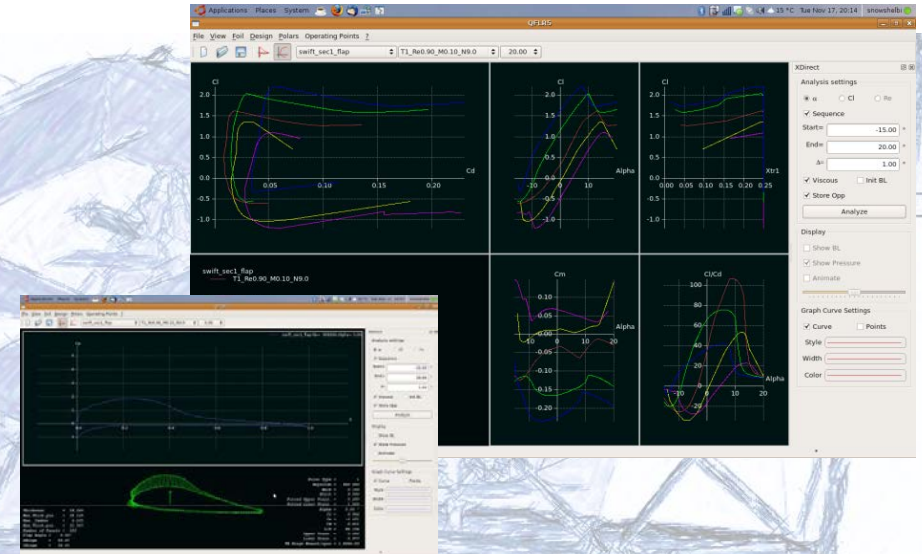


Sources	Low impact	Medium impact	Critical impact
Monitoring of the temperature	✓		
Monitoring of atmosphere composition	✓		
Flatness of the ground	✓		
Accuracy of the touch probe	✓		
Accuracy of the kinematic chain of the robotic arm	✓		
Accuracy of the processing hardware of the controller box	✓		
Imperfections on the surface of the foil during its fabrication		✓	
Play in the connection between the foil and the tip		✓	
Tiny damages at the extremity of the plane due to previous flights		✓	

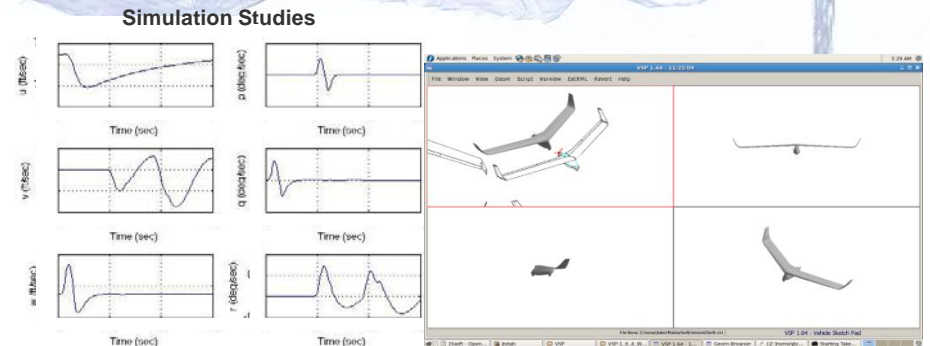
Sources	Low impact	Medium impact	Critical impact
Imperfection in the assembly of the different parts of the plane		✓	
Flexibility of the airfoil skin	✓		
Deformation of the stand supporting the foil		✓	
Accuracy of the operator	✓		

Dynamic Modeling and Simulation

- **Simulation studies used for dynamics and performance predictions, control law development**
- **Dynamic models generated from STAR Navier-Stokes analyses**
 - Time consuming process, provides estimates for all major dynamic coefficients
- **Lower fidelity method for comparison**
 - X-Foil model using estimated properties
 - Vehicle Sketchpad / Vorview models
- **Simulation Models**
 - Linear model (Matlab) containing stability derivatives, control law development
 - Non-linear model for simulation testing
- **Model assumptions and estimates will be continuously refined throughout development**
- **Flight test data will be used for parameter estimation**



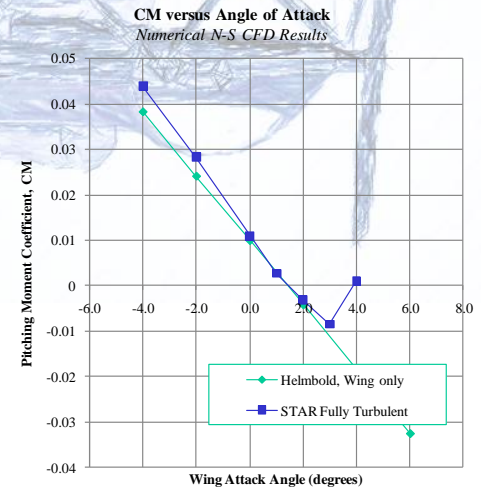
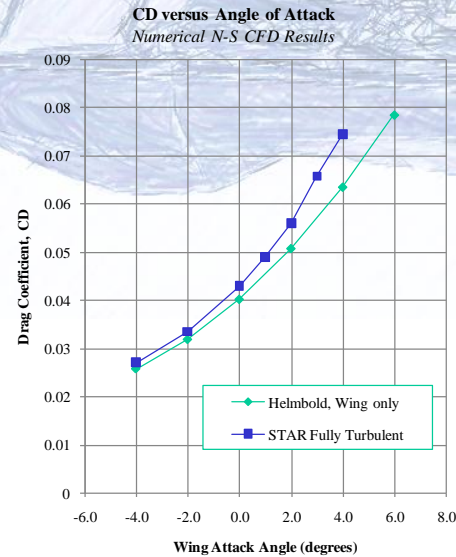
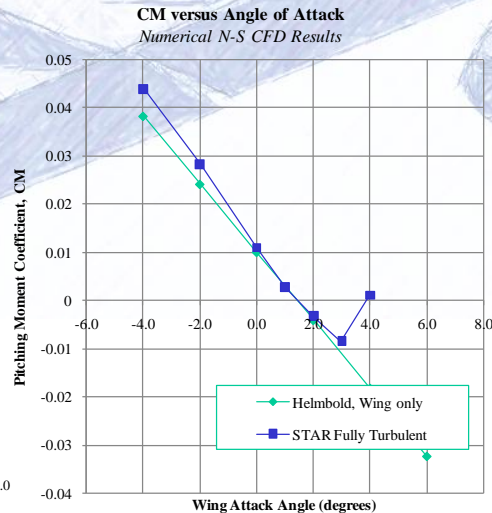
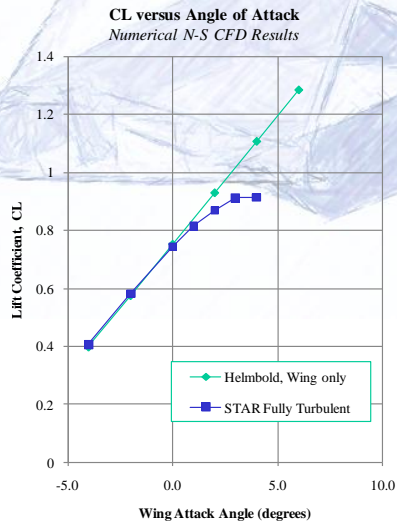
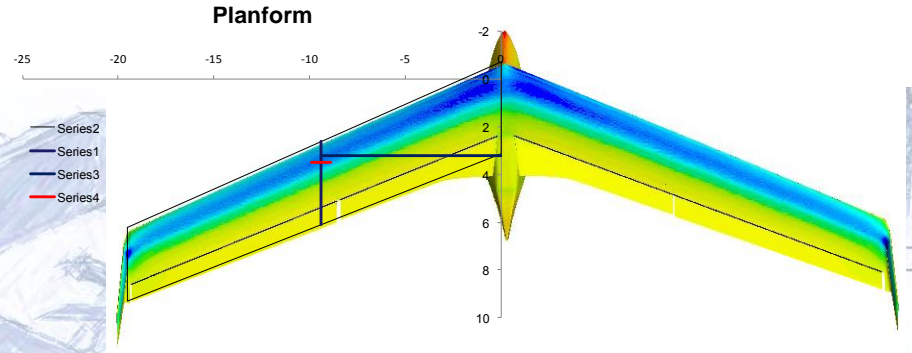
X-Foil courtesy of Z.Mahboubi (Stanford/AA)



SketchPad/Vorview courtesy of J. Totah (NASA/ARC/TI)

CFD Modeling

- CFD model and analysis performed by John Melton (ARC-AUS)
- STAR N-S analysis derived from geometry
 - Navier-Stokes drag coefficient estimates derived for small number of conditions
 - Assumed fully turbulent boundary layer
- Comparisons with lower order models
 - Airfoil derivatives from X-Foil for comparison



6-DOF Simulation Model

States:

$$P_e = [X_e \ Y_e \ Z_e]^T ; V_b = [u \ v \ w]^T ; q = [q_0 \ q_1 \ q_2 \ q_3]^T ; \omega_b = [p \ q \ r]^T$$

Kinematics:

$$\dot{P}_e = (\Omega_{E_e} P_e) + R_{b2e} V_b$$

$$\dot{q} = -0.5 \tilde{\omega}_b q$$

$$\dot{V}_b = -(\omega_b \times v_b) - (R_{e2b} \Omega_{E_e}^2 + R_{e2b} \Omega_{E_e} R_{b2e} \omega_b) + R_{e2b} g_e + m^{-1} (F_{PR_b} + [-Drag \ Y - Lift] R_{wi2b})$$

$$\dot{\omega}_b = -J_b^{-1} \tilde{\omega}_b J_b + J_b^{-1} (T_{PR_b} + [L \ M \ N])$$

Dynamics:

$$Lift = QSC_{l_\alpha}(\alpha, \delta_{flap}) + Q * S_{ht} * \frac{dC_l}{d\delta_{ele}} \delta_{ele} + \left(\frac{QS\bar{c}}{2V}\right) \frac{dC_l}{d\alpha} \dot{\alpha} + \left(\frac{QS\bar{c}}{2V}\right) * \frac{dC_l}{dq} q$$

$$Drag = QSC_{D_0} + QS * C_{D_\alpha}(\alpha, \delta_{flap}) + QSC_{D_{\delta_{flap}}}(\delta_{flap}) + QS_{HT} \left(\frac{dC_D}{d\delta_{ele}}\right) \delta_{ele} + QS \left(\frac{dC_D}{d\beta}\right) \beta$$

$$Y = QSC_{Y_\beta}(\beta) + QS \left(\frac{dC_Y}{d\delta_{ail}}\right) \delta_{ail} + QS \left(\frac{dC_Y}{d\delta_{rdr}}\right) \delta_{rdr} + QS \left(\frac{b}{2V}\right) \left(\frac{dC_Y}{dp}\right) p + QS \left(\frac{b}{2V}\right) \left(\frac{dC_Y}{dr}\right) r$$

$$L = QSC_{L_\beta}(\beta) + QS \frac{b^2}{2V} \left(\frac{dC_L}{dr}\right) r + QS \left(\frac{b^2}{2V}\right) \left(\frac{dC_L}{dp}\right) p + QSb \left(\frac{dC_L}{d\delta_{ail}}\right) \delta_{ail} + QS_{vt} \overline{MA}_{vt} \left(\frac{dC_L}{d\delta_{rdr}}\right) \delta_{rdr}$$

$$M = QS\bar{c} \left(\frac{dC_M}{d\delta_\alpha}\right) \alpha + QS\bar{c} C_{M_0} + QS\bar{c} C_{M_{flap}} + QS \left(\frac{\bar{c}}{2V}\right) \frac{dC_M}{dq} q + QS \frac{\bar{c}^2}{2V} * \dot{\alpha} * \frac{dC_M}{d\alpha} + QS_{ht} \overline{MA}_{ht} \left(\frac{dC_M}{d\delta_{ele}}\right) \delta_{ele}$$

$$N = QSb C_{N_\beta} \beta + QS \left(\frac{b^2}{2V}\right) \frac{dC_N}{dp} p + QS \left(\frac{b^2}{2V}\right) \frac{dC_N}{dr} r + QSb \left(\frac{dC_N}{d\delta_{ail}}\right) \delta_{ail} + QS_{vt} \overline{MA}_{vt} \left(\frac{dC_N}{d\delta_{rdr}}\right) \delta_{rdr}$$

- Table driven coefficients
- Data filled in from NASA EAV model

Simulation



Development Status and Progress

- **Preliminary Design**
 - **Engine Integration**
 - Preliminary Flight Test
 - **Modeling and Simulation**
 - **Airframe Modification**
 - Actuator installation, control surface splitting, landing gear installation
 - **Avionics Integration**
 - Actuation System Integration
 - CDHS/Autopilot Integration
 - Propulsion System Integration (completed April-2012)
-
- **Acceptance and Environmental Testing**
 - **Flight Worthiness Statement / AFSRB**
 - **Flight Testing and Autopilot Tuning**



Introduction to the Swift UAS

*Presentation to Prof Joanne Dugan
Dependable Computing Class
University of Virginia
April 16, 2012*

Corey Ippolito¹, Ganesh Pai², Ewen Denney³

Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA
corey.a.ippolito@nasa.gov

1. *Aerospace Scientist, NASA Ames Research Center*
2. *Research Scientist, Stinger Ghaffarian Technologies, Inc.*
3. *Research Scientist, Stinger Ghaffarian Technologies, Inc.*

