DESIGN OF AN AUTOPILOT FOR SMALL UNMANNED AERIAL VEHICLES

by

Reed Siefert Christiansen

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Brigham Young University

August 2004

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

Of a thesis submitted by

Reed Siefert Christiansen

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Randal W. Beard, Chair
Date	Timothy W. McLain
Date	D. J. Lee

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Reed Siefert Christiansen in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Randal W. Beard Chair, Graduate Committee

Accepted for the Department

Mark L. Manwaring
Graduate Coordinator

Accepted for the College

Douglas M. Chabries
Dean, College of Engineering and Technology

ABSTRACT

DESIGN OF AN AUTOPILOT FOR SMALL UNMANNED AERIAL VEHICLES

Reed Siefert Christiansen

Department of Electrical and Computer Engineering

Master of Science

This thesis presents the design of an autopilot capable of flying small unmanned aerial vehicles with wingspans less then 21 inches. The autopilot is extremely small and lightweight allowing it to fit in aircraft of this size. The autopilot features an advanced, highly autonomous flight control system with auto-launch and auto-landing algorithms. These features allow the autopilot to be operated by a wide spectrum of skilled and unskilled users. Innovative control techniques implemented in software, coupled with light weight, robust, and inexpensive hardware components were used in the design of the autopilot.

ACKNOWLEDGMENTS

I would like to acknowledge the contributions made by several people in the design of the autopilot and supporting systems. First, I extend my gratitude to Dr. Randy Beard and Dr. Timothy McLain for providing the vision, guidance and the endless supply of money for parts and airplanes. I would like to thank Walt Johnson for countless hours spent laying out the incredibly small traces that made the autopilot board as small and light as it is. I also want to acknowledge Derek Kingston for his work in tackling the elusive attitude estimation problem. Josh Hintz and Dave Hubbard deserve credit on three accounts: First, for sticking with the project in the "early days" when working on the autopilot was not fun; Second, for spending long hours in the sun testing doomed autopilot algorithms while everyone in the lab thought we were out sunbathing at the park; Third, for developing the venerable Virtual Cockpit ground station. In addition, I would like to thank Kendall Fowkes for the hours spent designing and building airplanes only to have them broken into a million pieces. I would also like to thank Carolyn Cornaby for having the courage to edit and proofread a 240 page thesis. Finally, I would like to thank the following people for their many contributions: Steve Griffiths, Josh Redding, Morgan Quigley, Deryl Snyder, Mark Denny, and Jen Wilds.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have proven their usefulness in military applications in recent years. Large UAVs such as the General Dynamics Predator (27 foot wingspan) [1] have become an integral part of the U.S. arsenal. Since 1994, the Predator has logged more than 20,000 flight hours, executing surveillance and tactical missions in virtually every part of the world. The military is also making use of smaller UAVs, known as mini UAVs. The AeroVironment Pointer Mini-UAV was one of the first Mini UAVs deployed [2]. It has been used all over the world by small military units as a short range video reconnaissance platform. It has a cost of $88,000 [3], relatively low in comparison with the Predator's price tag of $50 million. The Pointer has proven itself robust and reliable, but it does have disadvantages. This UAV is designed to be "man-packable" but the UAV is heavy and cumbersome for one person to carry. Its 9-foot wingspan also makes it easily visible to the enemy at the low altitudes it flies. Hence, the armed forces have made a significant push to develop a replacement for Pointer. This effort has lead to the Dragon Eye Mini UAV. The Dragon Eye is smaller and lighter than Pointer. However, Dragon Eye is still large enough (4 foot wing span) [4] to be visible to the enemy and requires a 2-man crew to transport and operate. There is room for improvement in the areas of UAV size and portability.

Recently, interest has increased in the relatively undeveloped area of UAVs with wingspans of less than 2 feet. Military personnel, encouraged by the successes of Pointer and Dragon Eye, desire to get the same close area surveillance capabilities into smaller, lighter packages. These Micro UAVs have several advantages over Mini UAVs. Their small size makes them easier to carry. It is possible for a Micro UAV to be transported and operated by one individual, as opposed to the two- or three- man crews that are currently required to operate the Mini UAVS. The size of Micro UAVs also allows them to be packed in and deployed from small containers. For one application, this would allow several Micro UAVs to be deployed from a Mini or Large UAV to gather close-range surveillance data. Micro UAVs are also less expensive than their larger counterparts. They are smaller and require less material to build. This cost saving gives them potential to be a disposable asset; Micro UAVs could be flown in unfriendly areas where the risk of damage is high. It also frees the operator from the task of ensuring the airplane is recovered, allowing more effort to be spent on primary mission goals.

In addition to military applications, there are many potential civilian applications for Micro UAVs [5]. The size and cost advantages that make them appealing to the military also make them attractive for use in the private sector. However, the use of UAVs in non-military applications is currently limited for reasons of cost, safety, and special requirements. The prohibitive cost of Large and Mini UAVs makes it difficult for individuals and private organizations to purchase and own. The current generation of Mini and Large UAVs also pose a potential danger to the civilian population because of their weight. Micro UAVs are much safer in this aspect, as they can weigh less than 1 kilogram. In addition to high cost and physical danger associated with currently available UAVs, the challenges of using these aircraft in the private sector include significant training of the operator. Large UAVs also require large runways to operate.

Micro UAVs have the potential to overcome many of the limitations of Mini and Large UAVs. For this reason there is great interest in adopting Micro UAVs for a variety of applications. These include government applications, such as fire fighting and law enforcement. The U.S. spends millions of dollars each year patrolling borders with conventional aircraft [6]. If border agents were equipped with low-cost Micro UAVs, thousands of dollars could be saved, as UAVs cost much less then similarly equipped full-size aircraft. In the private sector, there also exists a range of surveillance applications for UAVS. One example is use by the media. Newscasters spend millions of dollars on helicopters to cover breaking news stories. Micro UAVs could be carried by news crews and launched over a breaking story to gather aerial footage immediately. The Micro UAVs would also pose less risk to the civilian population than low-flying full-sized helicopters. It is likely that as Micro UAVs become available, they will be employed in many more applications to cut cost and improve coverage.

For Micro UAVs to be successful militarily and commercially, they must be low-cost, easy to operate, and small. Currently, there are no fully autonomous operational UAVs available with wingspans of less than 2 feet. One of the main roadblocks in developing Micro UAVs is the size and weight of the autopilot system. For an aircraft with a wingspan of less than 2 feet to carry an autopilot and useful payload, the autopilot cannot weigh more than a few ounces. It must also be small enough to fit in the airframe. There are several autopilots on the market, but none of them small enough and inexpensive enough to fill this application.

1.1 THESIS CONTRIBUTIONS

The purpose of this thesis is to outline the development of an autopilot that can be used in small UAVs with a wingspan of less than 2 feet. The discussion will focus on the features and capabilities of the autopilot that make it useful for military and commercial applications. The following are examples of contributions made by this thesis to the area of flight control for small UAVs: First, the autopilot is very small and lightweight. Second, the ground station and autopilot hardware allow the operator to seamlessly switch between manual control and autopilot control. Third, the PID gains can be changed during flight through the ground station. Fourth, the autopilot uses inexpensive, commercially available hardware and innovative software filtering to estimate aircraft attitude. Fifth, a novel approach to auto take-off and landing are introduced. Sixth, a comprehensive set of waypoint commands are implemented on the autopilot which provide the user with a variety of navigation commands. Finally, a communication protocol is used which permits access to all internal autopilot variables via the ground station.

1.2 DESIGN METRICS

To help guide the development of the autopilot, a set of specifications were established. The following list is a set of design metrics that were used during autopilot development:

1. The autopilot weight should be less than 3 ounces.
2. The autopilot physical dimensions should be less than 2” x 4” x 3/4 inches square.
3. The autopilot should have a robust, damage-resistant physical structure.
4. The autopilot should be highly autonomous with auto-launch and auto-land abilities.
5. It should be possible to manufacture the autopilot for less than $1000.

This thesis will show the development of an autopilot system that satisfies these design metrics. An off-the-shelf airframe was used in the autopilot development to maximize the effort and resources put toward the development of the autopilot itself.

1.3 RELATED WORK

There are several autopilots on the market that could be purchased for use in UAVs. The following is a brief list of the most popular units:

1. Micropilot 2028g [7]:
    a. Full waypoint navigation
    b. Auto-launch, auto-land with height above ground sensor
    c. Weight: 1 ounce
    d. Size: 3.9” x 1.6” x .6”
    e. Cost: $6800
2. Piccolo [8]:
    a. Full waypoint navigation

   b. Weight: 7.5 ounces
   c. Size: 4.8” x 2.4” x 1.5”
   d. Cost: $7500

3. UAV Flight Systems AP-50 Flight Control System [9]:
   a. Full waypoint navigation
   b. Weight: 1.8 ounces
   c. Size: 5.5” x 1.85” x 1.6”
   d. Cost: $3400

While these three products are capable autopilots they do not meet the requirements set forth in Section 1.2: all three exceed the cost and size requirements. In addition, only the Micropilot has the auto-launch and auto-land abilities.

1.4 TECHNICAL CHALLENGES

Several significant technical challenges were overcome during the autopilot’s development. The first was designing the autopilot to stay within the weight and size specifications, while remaining low-cost. Careful selection of components and innovative use of inexpensive off-the-shelf Microelectromechanical Systems (MEMS) devices were employed to meet these requirements. This solution will be covered in Chapters 3 and 4.

Micro UAVs are difficult to control because their size pushes the envelope of aircraft technology. Current research focuses on developing stable airframes at the Reynolds numbers at which Micro UAVs operate. Currently, most airframes in this size and weight category are inherently unstable. Thus the autopilot must be able to control an unstable airframe. The control methods used to overcome this second challenge are developed in Chapter 2.

The third challenge lies in testing and developing an autopilot on an inexpensive airframe without crashing and damaging hardware. This was accomplished by modifying an off-the-shelf combat-type flying wing (Zagi) for use as a UAV and constructing a Bypass circuit that allowed a human pilot to take control of the autopilot in the event of an autopilot malfunction. The use of the Zagi aircraft and Bypass circuit allowed much of the flight testing to occur at a local park. This made development quick and affordable. The Bypass circuit and airframe are discussed in Chapter 3.

Finally, for a Micro UAV to be used in the commercial sector it must be easy to use. For ease of use, high-level algorithms were developed that allow the aircraft to take off and land itself. The abilities of the autopilot will be discussed in Chapter 4.

1.5 THESIS ORGANIZATION

This thesis will outline the development of an autopilot for Micro UAVS. The technical challenges and solutions to these challenges will be addressed in the subsequent chapters. Chapter 2 will cover the control algorithms used to stabilize and navigate the UAV. Chapter 3 will discuss the hardware. Chapter 4 will discuss the autopilot and ground station software. Chapter 5 is a User's Manual, written to allow those unfamiliar with the system to install, configure, and operate the autopilot.

CHAPTER 2

FLIGHT CONTROL

The purpose of this thesis is to design an autopilot for small unmanned air vehicles (UAVs). As the size of the UAV approaches the micro category, its control and stabilization become more difficult. This is due to several factors, including the low mass of the vehicle, lower Reynolds numbers, and light wing loading. These factors make it more difficult to design a flight control system. This chapter will detail the design of a flight control system to adequately stabilize and control small UAVS. This will be done in several parts. First, the flight characteristics of the airframe used in autopilot development will be analyzed using a 6 degree-of-freedom MATLAB simulation. Next, the flight control PID loop structure will be discussed. The chapter will finish with a closed-loop MATLAB simulation of the autopilot and aircraft.

2.1 ZAGI DYNAMICS

The first step in designing a controller for any physical system is to characterize the dynamics of that system. The dynamics of the Zagi flying wing aircraft used in this thesis can be decoupled into lateral dynamics and longitudinal dynamics [10]. The lateral dynamics are the aircraft’s response along the roll and yaw axes. The lateral modes are generally excited with aileron and rudder inputs. The longitudinal dynamics are the response of the aircraft along the pitch axis. The aircraft’s response in velocity to thrust is also included in the longitudinal dynamics. The longitudinal modes are excited with the elevator control surface and throttle. The equations for the longitudinal motion is given by

$$\begin{Bmatrix} \dot{U} \\ \dot{W} \\ \dot{Q} \\ \dot{\theta} \end{Bmatrix} = \begin{bmatrix} X_u & X_w & 0 & -g \\ Z_u & Z_w & u_0 & 0 \\ M_u + M_{\dot{w}} Z_u & M_w + M_{\dot{w}} Z_w & M_q + M_{\dot{w}} u_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} U \\ W \\ Q \\ \theta \end{bmatrix} + \begin{bmatrix} X_\delta & X_{\delta_T} \\ Z_\delta & Z_{\delta_T} \\ M_\delta + M_{\dot{w}} Z_\delta & M_{\delta_T} + M_{\dot{w}} Z_{\delta_T} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta \\ \delta_T \end{bmatrix}. \qquad (2.1)$$

The equation for lateral motion is given by

$$\begin{Bmatrix} \dot{V} \\ \dot{P} \\ \dot{R} \\ \dot{\phi} \end{Bmatrix} = \begin{bmatrix} Y_v & Y_p & -(u_0 - Y_r) & g \cdot \cos(\theta_0) \\ L_v & L_p & L_r & 0 \\ N_v & N_p & N_r & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V \\ P \\ R \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & Y_{\delta_R} \\ L_{\delta_A} & L_{\delta_R} \\ N_{\delta_A} & N_{\delta_R} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_A \\ \delta_R \end{bmatrix}. \qquad (2.2)$$

The coefficients for these equations are provided in Appendix A.

To build a controller for the aircraft, the aircraft's dynamics must be modeled. The dynamic characteristics of interest are the aircraft's response to step inputs in the lateral and longitudinal modes. To understand the response of the Zagi, a model of the Zagi is imported into a simulation environment constructed in MATLAB/Simulink [11].

2.1.1 SIMULATION ENVIRONMENT

The longitudinal and lateral aircraft dynamic equations (2.1, 2.2) were imported into MATLAB/Simulink. The aircraft coefficients for the Zagi were obtained from the Slope Soaring Simulator [11]. The Slope Soaring Simulator is an open-source flight simulator that has many different aircraft dynamic models. The MATLAB/Simulink diagram of the aircraft dynamic model is shown in Figure 2.1. The inputs to the model are the control surface deflections (δ). The model Zagi has the following control surfaces: aileron, elevator, and throttle. A rudder is added to the model to simulate the response of the Zagi as if a rudder were installed. The outputs of the model are the 12 aircraft states. The following states are used:

1. X: inertial X coordinate (meters north)
2. Y: inertial Y coordinate (meters east)
3. H: altitude in inertial coordinates (meters)
4. U: body frame X-velocity (meters / second)
5. V: body frame Y-velocity (meters / second)
6. W: body frame Z-velocity (meters / second)
7. ϕ: roll angle (radians)
8. θ: pitch angle (radians)
9. ψ: heading angle (radians)
10. P: roll rate (radians / second)
11. Q: pitch rate (radians / second)
12. R: yaw rate (radians / second)

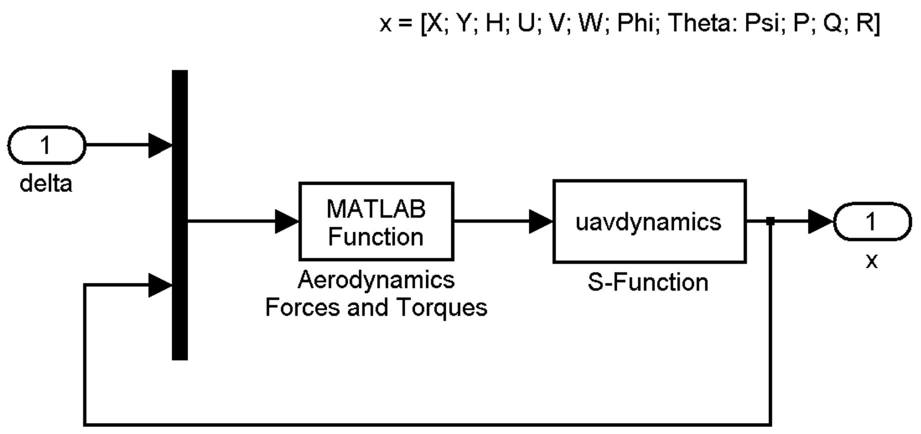

x = [X; Y; H; U; V; W; Phi; Theta: Psi; P; Q; R]
1
delta
MATLAB Function
Aerodynamics Forces and Torques
uavdynamics
S-Function
1
x

Figure 2.1: 6 DOF MATLAB Aircraft Simulation.

2.1.2 SETTING UP THE SIMULATION

The simulation is set up to calculate the trim conditions for level, un-accelerated flight of the aircraft model at the beginning of each simulation. The control deflections are centered around these trim conditions. The trim function calculates the required pitch angle, and corresponding elevator and throttle deflections, for the trim velocity of the simulation. The trim velocity used in most simulations is 12 m/s, which is the nominal velocity of the Zagi airframe.

2.1.3 LATERAL DYNAMICS SIMULATION

The lateral modes of interest are the Dutch roll, spiral, and roll. The roll and spiral modes are of special interest as they represent the aircraft's natural ability to level itself in the roll axis in response to changing control input or wind gust. These modes are easily excited by applying a step input to the aileron. To excite the lateral modes, the aircraft model is trimmed for a level flight at a speed of 12 m/s. A step input is applied to the aileron. The aileron deflection is neutralized after 1 second. The aircraft's response in ϕ, ψ, and P are noted in Figure 2.2. The aileron step induces a roll rate, which dampens out over the 40 second simulation. The roll angle (ϕ) oscillates, then dampens out at a value of about 35 degrees. The simulation shows that the Zagi develops a semi-divergent spiral to small steps in aileron. The spiral diverges until the roll angle reaches approximately 35 degrees. The roll angle stabilizes at this point. This is consistent with flight data from the Zagi.

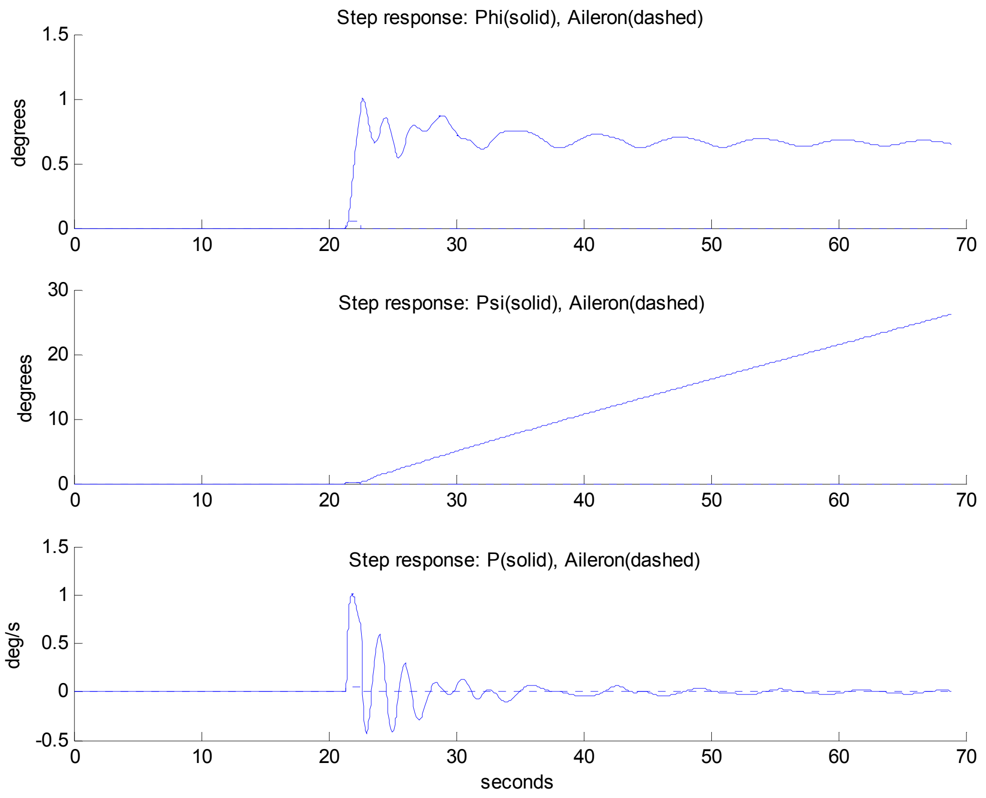


Figure 2.2: Lateral response in the Zagi to step in aileron.

The next simulation is a test of the lateral response to a step in rudder control. The actual Zagi airframe does not have a rudder, but a set of rudders could easily be mounted in the winglets. The purpose of this simulation is to show the response of the aircraft to a disturbance force acting in the same manner as the force caused by a rudder deflection. Figure 2.3 shows the aircraft response. The rudder input induces a side slip motion, which dampens out within 20 seconds. The rudder input also induces a roll angle.

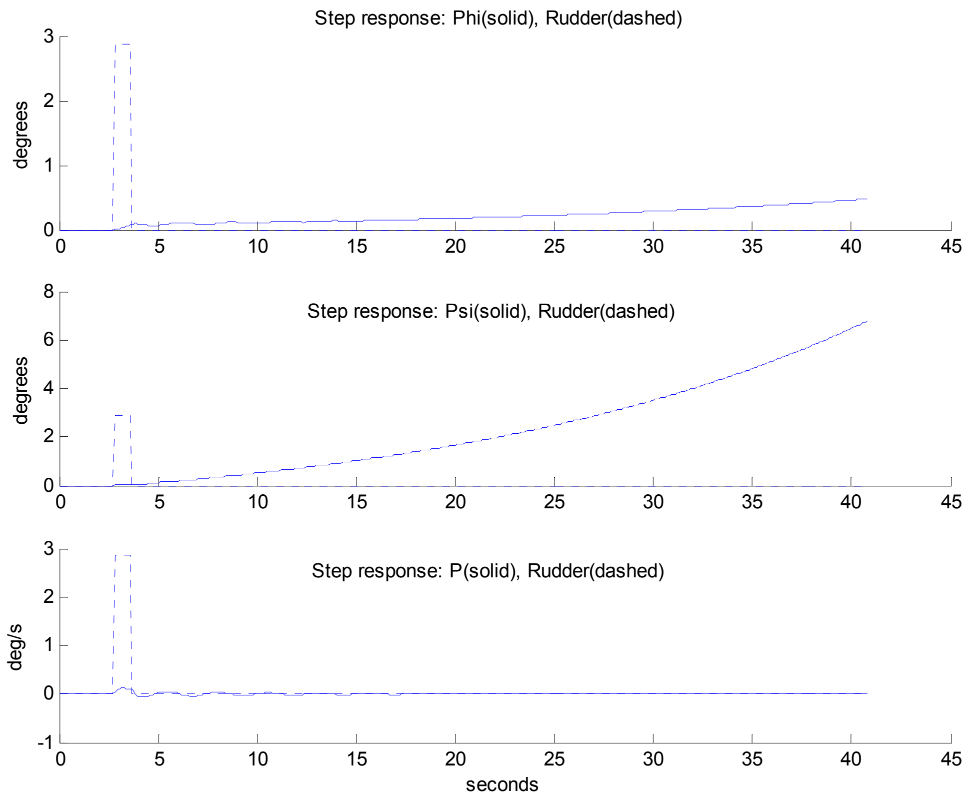


Figure 2.3: Lateral response in the Zagi to step in rudder.

2.1.4 LONGITUDINAL DYNAMICS SIMULATION

The longitudinal mode of interest is the short period mode. The longitudinal dynamics are excited in the same manner as the lateral dynamics. With the aircraft in a level flight trim condition, the elevator is deflected for 1 second and then returned to neutral. The response of the aircraft is shown in Figure 2.4. The step in elevator induces a pitch rate which does not dampen out within the 40 second simulation. This behavior is consistent with an unstable short period mode. It is clear that the aircraft will require active stabilization in the pitch axis.

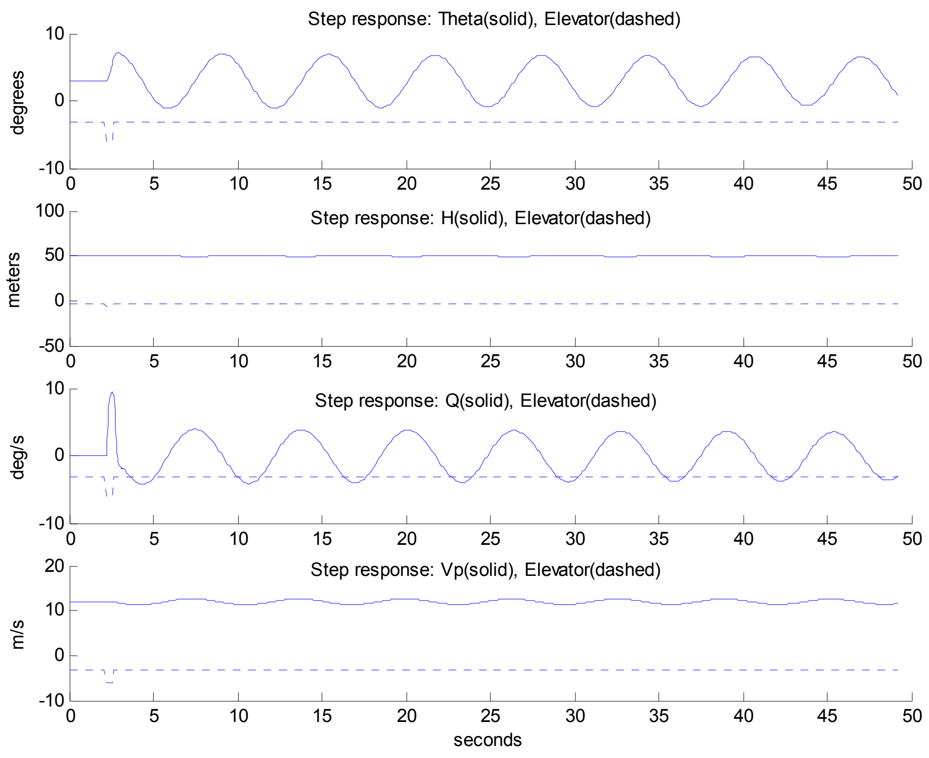


Figure 2.4: Longitudinal response in the Zagi to step in elevator.

2.1.5 SIMULATION ANALYSIS

The simulations show that the aircraft exhibits instabilities in the longitudinal and lateral modes. Instabilities in the lateral modes show that the aircraft is unstable in roll and tends to diverge to a roll angle of 35 degrees in response to small lateral disturbances. The longitudinal simulations show the aircraft to be unstable in the short period mode. To address these instabilities, the aircraft will require stabilization in the pitch and roll axes.

2.2 AIRCRAFT CONTROL DESIGN

In the previous section it was determined that the Zagi has longitudinal and lateral instabilities. In order to achieve stable autonomous flight, these instabilities must be addressed. This section discusses the PID control structure developed to stabilize and control the aircraft. To fly the aircraft autonomously, the autopilot must be capable of navigating waypoints. This requires that the autopilot be able to control the heading, altitude, and airspeed of the aircraft. For manual control, it is desirable that the autopilot also accept pitch and roll angle commands. To accomplish this, a controller constructed of nested PID loops has been developed. The aileron, elevator, and throttle commands are controlled via inner PID loops that stabilize the roll, pitch and throttle. The altitude and heading are controlled with outer loops, which produce commanded values for the inner loops. The autopilot control is divided into two controllers: the lateral controller and the longitudinal controller. This section will detail each of these controllers.

2.2.1 LATERAL CONTROL

The lateral controller is responsible for controlling the yaw rate, roll angle, and heading. This is accomplished with three inner servo loops and one outer loop. The inner loops produce efforts that drive the aileron and rudder. The outer loops produce commanded values for the inner loops. The inner lateral loops are as follows:

1. Aileron from Roll: This loop generates an aileron deflection from the roll error. This loop is responsible for holding the roll attitude of the aircraft. This loop is shown in Figure 2.5.
2. Aileron from Roll Rate: This loop generates an aileron deflection from the roll rate. It is responsible for damping the roll rate of the aircraft. The control effort for this loop is summed with the effort from the Aileron from Roll loop and sent to the aileron servo actuator. See Figure 2.5.
3. Rudder from Yaw Rate: The purpose of this loop is to control yaw rate of the aircraft. This loop drives the rudder servo. This loop is shown in Figure 2.6

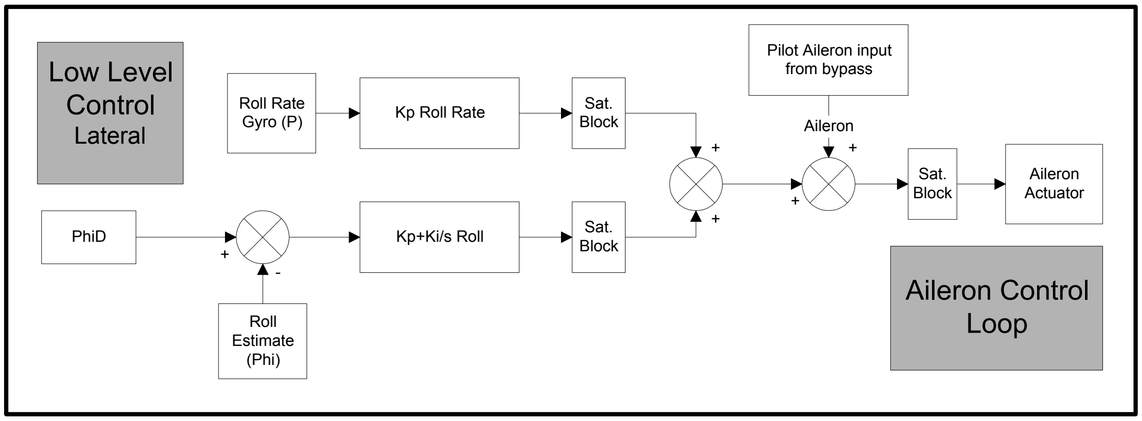


Figure 2.5: Inner lateral roll and roll rate controller.

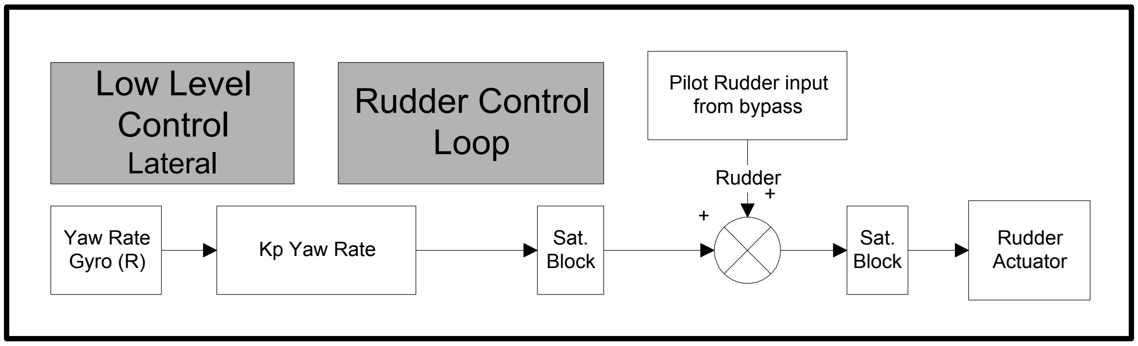


Figure 2.6: Inner lateral yaw rate controller.

The outer lateral control loop is the following:

1. Roll from Heading: This is the loop responsible for controlling the heading of the aircraft. It generates a roll angle from the heading error. This roll angle serves as the commanded roll angle for the Aileron from Roll loop. This loop is shown in Figure 2.7

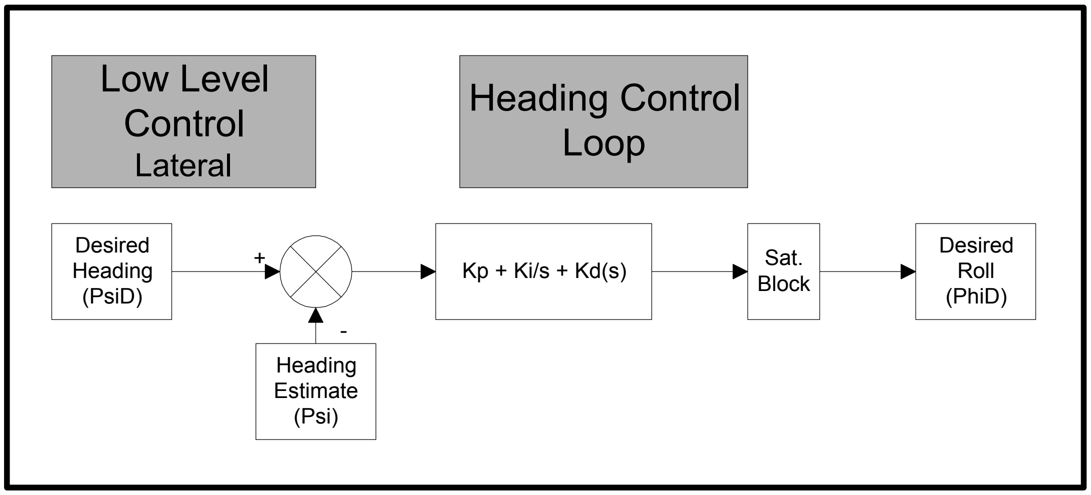


Figure 2.7: Outer lateral heading angle controller.

2.2.2 LONGITUDINAL CONTROL

The longitudinal controller is responsible for controlling the velocity, pitch angle, and altitude. This is accomplished with 3 inner servo loops and 2 outer loops. The inner loops produce efforts that drive the elevator and throttle. The outer loops produce commanded values for the inner loops. The inner lateral loops are as follows:

1. Elevator from Pitch: This loop generates an elevator deflection from the pitch error. This loop is responsible for holding the pitch attitude of the aircraft. This loop is shown in Figure 2.8.
2. Elevator from Pitch Rate: This loop generates an elevator deflection from the pitch rate. It is responsible for damping the pitch rate of the aircraft. This loop's control effort is summed with the Elevator from Pitch loop and sent to the elevator servo actuator. See Figure 2.8.
3. Throttle from Airspeed: The purpose of this loop is to control the aircraft's airspeed by adjusting the throttle. This loop drives the throttle servo. This loop is shown in Figure 2.9

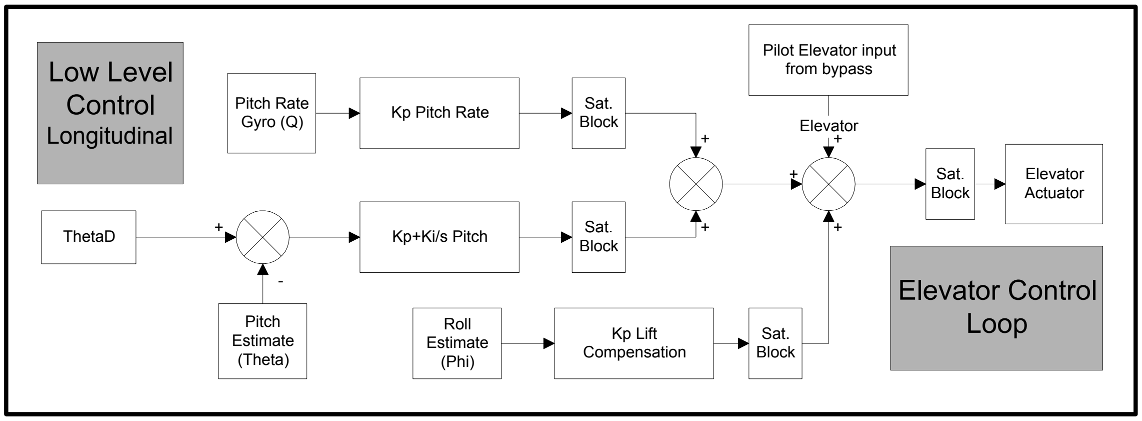


Figure 2.8: Inner longitudinal pitch and pitch rate controller.

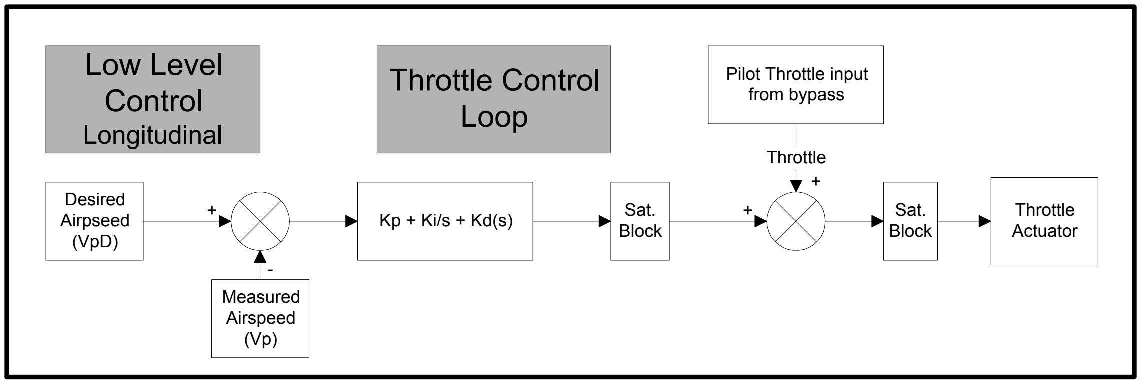


Figure 2.9: Inner longitudinal airspeed controller.

The outer lateral control loops are as follows:

1. Pitch from Altitude: This loop generates a commanded pitch angle from the altitude error. The output of this loop connects directly to the Elevator from Pitch loop. This loop is ideal for controlling the aircraft's altitude when the altitude error is small. For large altitude errors, the Pitch from Airspeed loop should be used. This loop is shown in Figure 2.10
2. Pitch from Airspeed: This loop controls the aircraft's airspeed by adjusting the pitch angle. The output of this loop connects directly to the Elevator from Pitch loop. This loop is used to regulate the aircraft's airspeed during climb and descent. This loop is shown in Figure 2.11

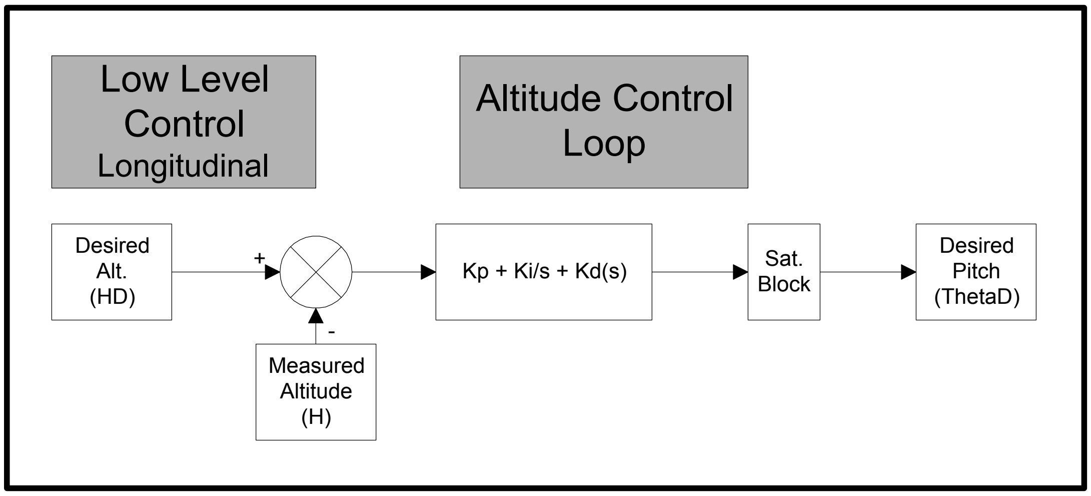


Figure 2.10: Outer longitudinal altitude controller.

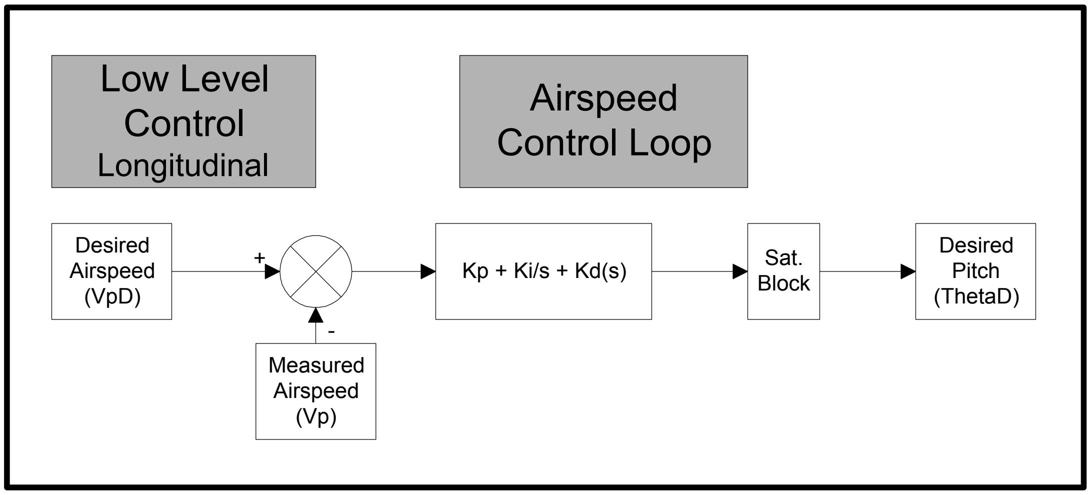


Figure 2.11: Outer longitudinal airspeed controller.

2.3 CLOSED-LOOP AUTOPILOT SIMULATION

The autopilot control structure is implemented in MATLAB/Simulink for simulation. The autopilot block is connected to the Zagi aircraft model discussed in

Section 2.1. The simulation is set up such that individual control loops can be simulated separately. The input to the simulation is the desired velocity, altitude, and heading. Figure 2.12 shows the autopilot and aircraft dynamics Simulink blocks. Figure 2.13 shows the autopilot block in detail. The autopilot performance in the lateral and longitudinal modes is simulated by applying step inputs. The simulation of the closed-loop autopilot performance in the lateral modes will be discussed in Section 2.3.1. The longitudinal mode performance will be discussed in Section 2.3.2.

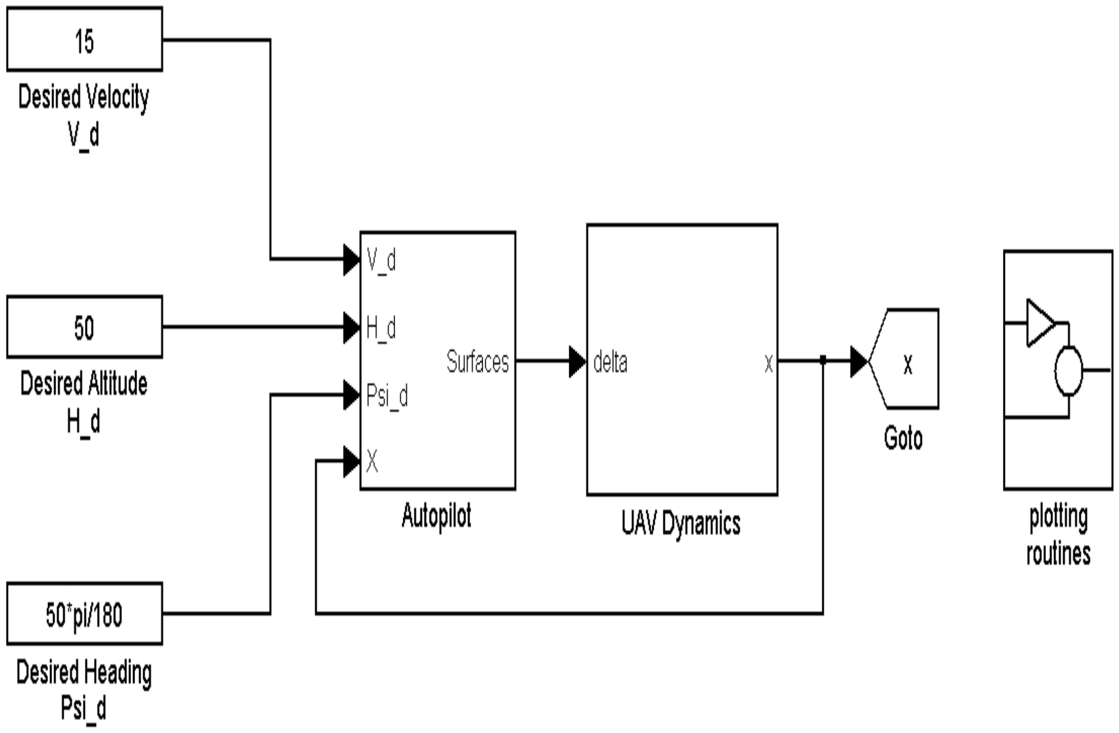


Figure 2.12: Simulink diagram of the closed loop autopilot simulation.

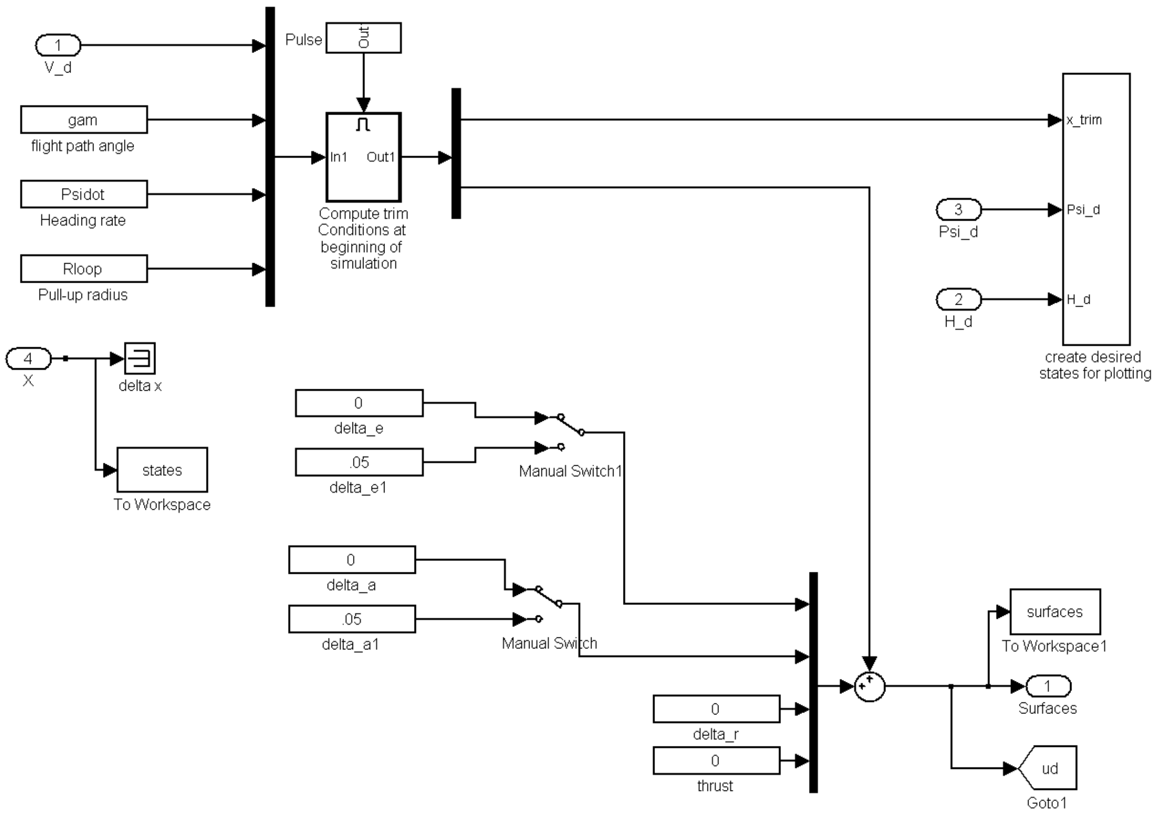


Figure 2.13: Simulink diagram of autopilot block.

2.3.1 LATERAL AUTOPILOT SIMULATION

The closed loop lateral performance of the autopilot and aircraft is simulated in two stages. First, the inner roll PID loop is simulated by giving a step input in the desired roll angle (ϕ). The results can be seen in Figure 2.14. The outer heading loop is simulated by giving a step input in the desired heading (ψ). These results are visible in Figure 2.15. The simulation shows that adequate autopilot performance in the lateral modes can adequately be achieved with PID control.

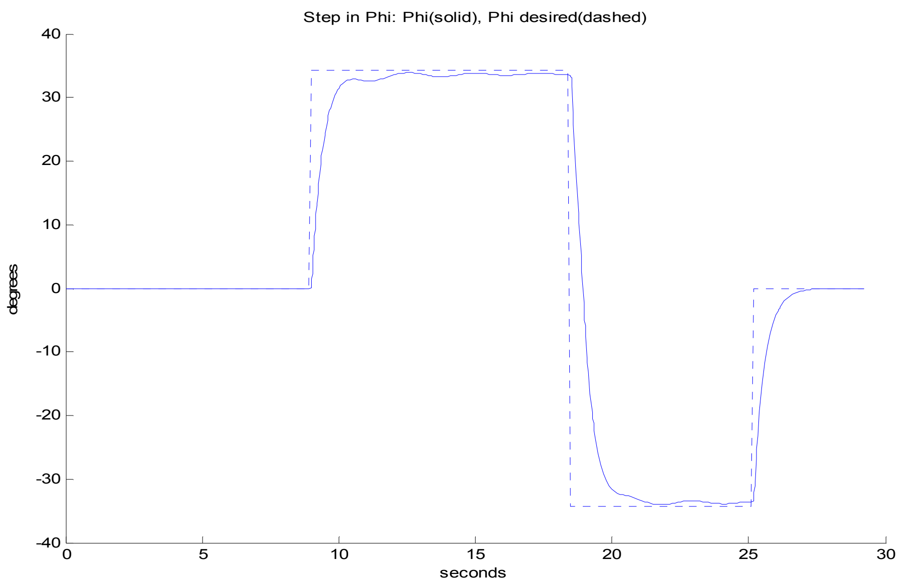


Figure 2.14: Closed loop lateral response to step in desired roll angle (ϕ).

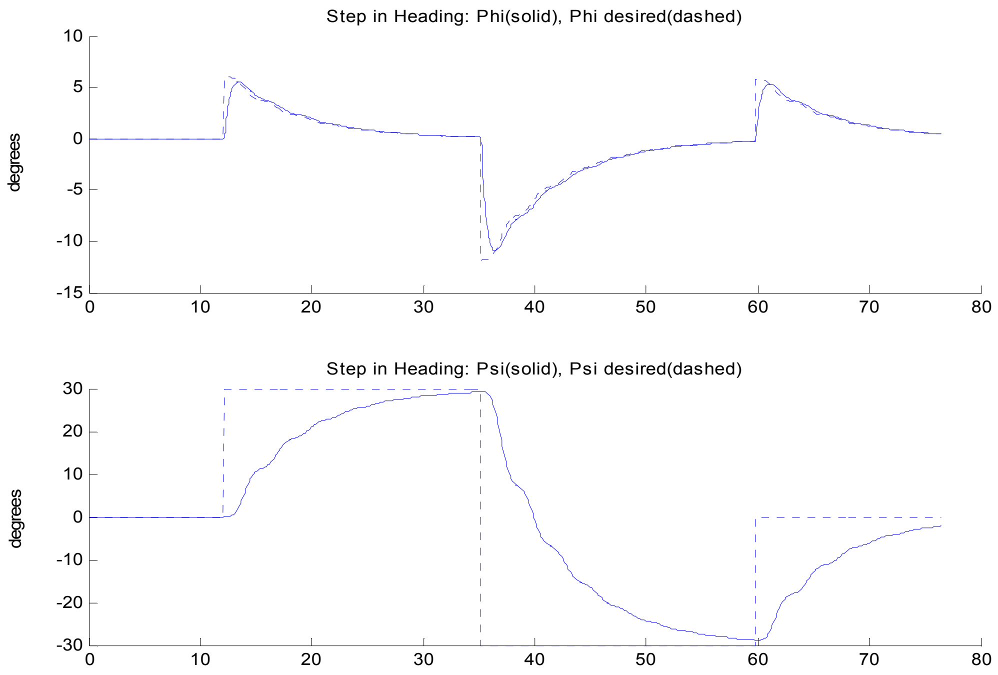


Figure 2.15: Closed loop lateral response to step in desired heading (ψ).

2.3.2 LONGITUDINAL AUTOPILOT SIMULATION

The closed loop longitudinal performance of the autopilot and aircraft is simulated in three stages. First, the inner pitch PID loop is simulated by giving a step input in the desired pitch angle (θ). The results can be seen in Figure 2.16. The response of the altitude loop is simulated next by giving a step in the desired altitude (H). These results are contained in Figure 2.17. The response of the velocity loop is simulated in the same manner. The results are visible in Figure 2.18.

The longitudinal simulation shows that the PID structure outlined in this chapter can adequately control the altitude of the aircraft, velocity, and pitch angle in response to step inputs.

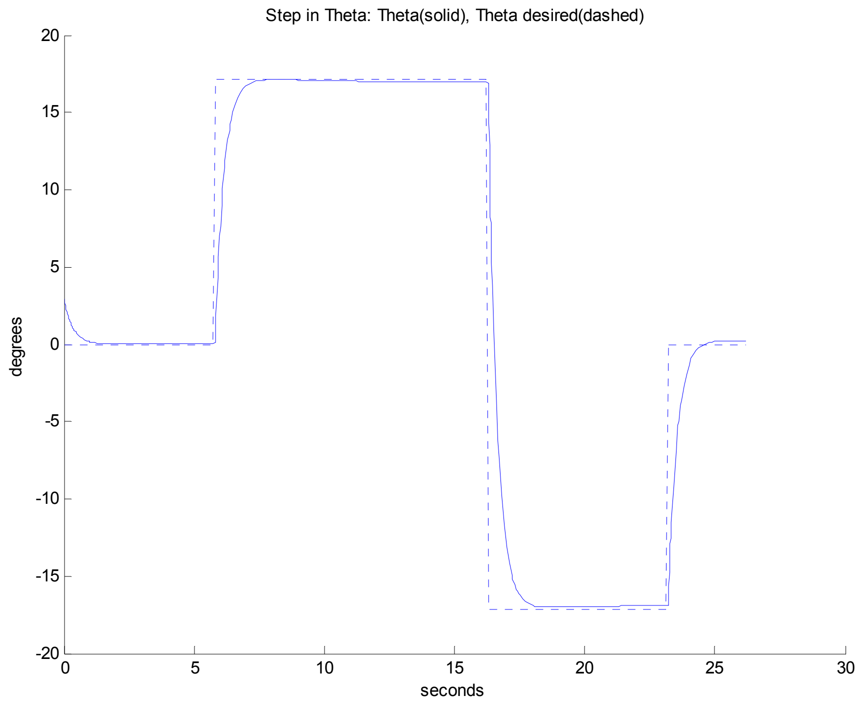


Figure 2.16: Closed loop longitudinal response to step in desired pitch angle (θ).

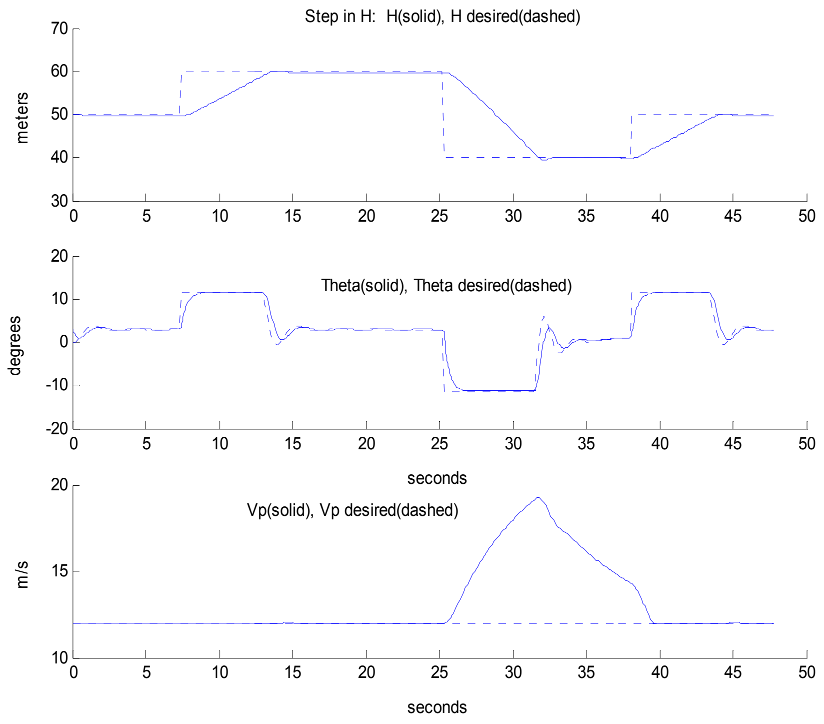


Figure 2.17: Closed loop longitudinal response to step in desired altitude (H).

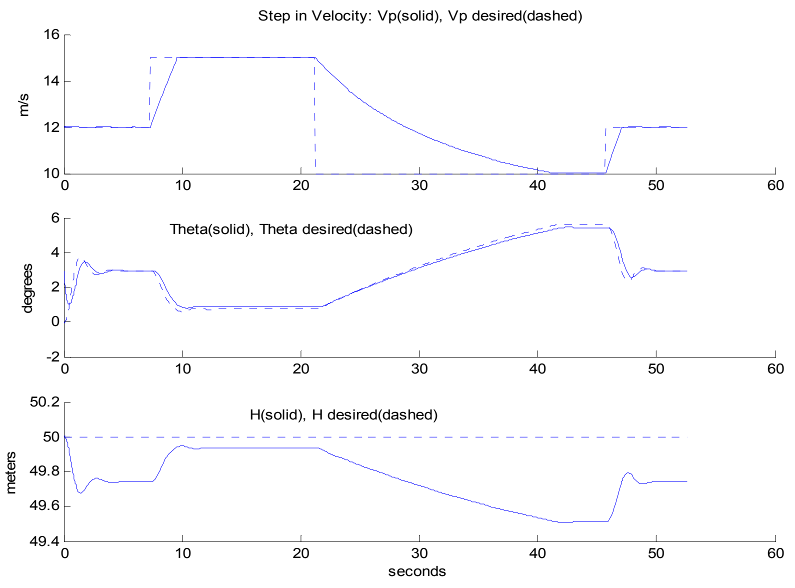


Figure 2.18: Closed loop longitudinal response to step in desired velocity (Vp).

CONCLUSION

This chapter has explained the design of autopilot algorithms used to control the Zagi flying wing aircraft. It began by analyzing the dynamics of the flying wing by applying step inputs to a 6-DoF MATLAB model of the Zagi. This model had lateral and longitudinal instabilities that have to be corrected with the autopilot control loops. From this knowledge, a controller was constructed using nested PID loops. Through simulation, it was found that controller could adequately control the pitch, roll, velocity, altitude, and heading of the aircraft. The chapter concluded by simulating the closed-loop performance of the autopilot and aircraft by applying step inputs.

CHAPTER 3

AUTOPILOT HARDWARE

Chapter 2 outlines the control algorithms and structure used in the autopilot. In this chapter, the autopilot hardware is discussed. This discussion will begin by outlining the autopilot on a system level, and follow with an explanation of the choice of airframe, batteries, actuators, and propulsion system. In addition, the avionics will be discussed, which include the autopilot and supporting hardware, such as the communications links and RC Bypass circuit. The next section will cover the hardware used in the video surveillance payload. We will conclude by discussing the ground station hardware.

3.1 AUTOPILOT SYSTEM OVERVIEW

A simple approach to understanding the UAV system is to separate it into four self-contained subsystems. The four subsystems are shown in Figure 3.1 as (1) flight platform, (2) avionics, (3) payload, and (4a, 4b) ground station.

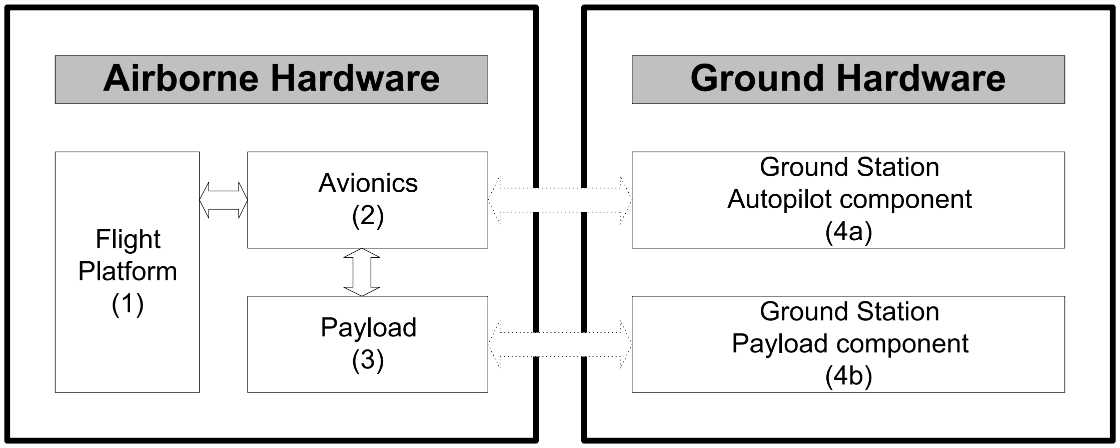

Airborne Hardware
Ground Hardware
Flight Platform (1)
Avionics (2)
Payload (3)
Ground Station Autopilot component (4a)
Ground Station Payload component (4b)

Figure 3.1: Autopilot Hardware.

The purpose of the flight platform is to carry the avionics and the payload. The flight platform consists of the airframe, the actuators that move the control surfaces, the batteries that power the autopilot, and the propulsion system. The flight platform will be discussed in detail in Section 3.2. The avionics consist of the autopilot, GPS receiver, and digital modem. Their purpose is to control the aircraft and communicate with the ground station. The avionics will be discussed in Section 3.3. The payload subsystem consists of the hardware not related to the flight platform or avionics. The payload is placed on the aircraft to accomplish user objectives. One of the purposes of this thesis is to demonstrate the feasibility of small UAVs for surveillance purposes. Therefore, a video surveillance payload is used on our flight platform. This payload will be discussed in Section 3.3. The ground station includes hardware and software needed to support the avionics and payload in flight. The ground station includes a laptop computer, a 900 MHz digital modem, a Futaba RC controller to accept pilot inputs, and a 2.4 GHz analog video receiver and display for the video surveillance payload. A graphical interface was written for the autopilot. The graphical interface runs on the ground station computer. The ground station GUI will be discussed in Chapter 5. The ground station hardware will be discussed in Section 3.4.

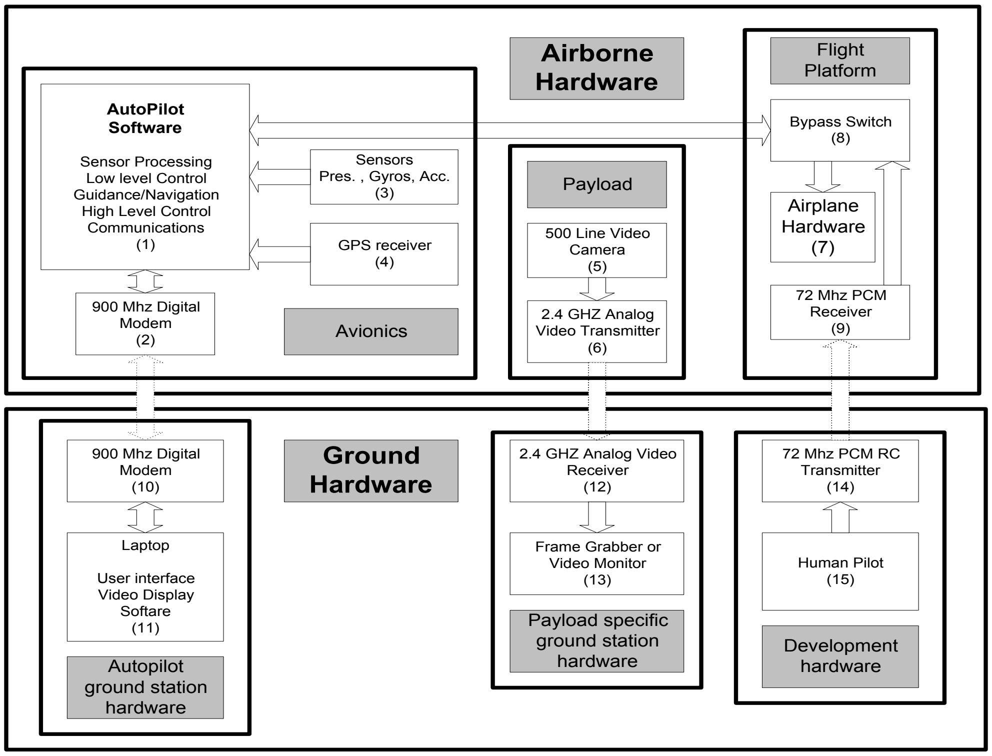


Figure 3.2: Detailed UAV system diagram.

Figure 3.2 shows the autopilot system in greater detail. Each of these subsystems will be discussed in this chapter. The following list aids in the explanation of Figure 3.2.

1. Figure 3.2(1) Autopilot software.
2. Figure 3.2(2) 900 MHz digital modem and airborne antenna.
3. Figure 3.2(3) Primary sensors on the autopilot board.
4. Figure 3.2(4) GPS receiver.
5. Figure 3.2(5) 500 line NTSC video camera.
6. Figure 3.2(6) 2.4 GHz analog video transmitter.

7. Figure 3.2(7) Primary flight surface actuators.
8. Figure 3.2(8) Bypass circuit to switch to bypass the autopilot for development.
9. Figure 3.2(9) 72 MHz RC receiver and antenna.
10. Figure 3.2(10) 900 MHz spread spectrum digital modem and ground antenna.
11. Figure 3.2(11) Laptop running user interface.
12. Figure 3.2(12) 2.4 GHz analog video receiver.
13. Figure 3.2(13) Frame grabber to digitize analog video from the airplane.
14. Figure 3.2(14) 72 MHz RC transmitter and antenna.
15. Figure 3.2(15) Human pilot.

3.2 UAV FLIGHT PLATFORM

As stated above, the flight platform includes the airframe and all the hardware needed to keep the airframe in the air. Specifically, the flight platform includes the airframe, actuators, electronic speed control, flight batteries, propulsion system, and 72 MHz receiver. Because the scope of this thesis is the development of the autopilot, not UAV airframes, the flight platform description and discussion will be limited. Requirements of the flight platform will be presented, followed by a discussion of the selection of the airframe, actuators, propulsion system, and flight batteries. The final section will detail the modification of the airframe from a glider to powered electric flight.

3.2.1 FLIGHT PLATFORM REQUIREMENTS

The purpose of the flight platform is to carry and protect the avionics and payload. The performance of the UAV system is heavily dependent on the flight platform. The objective of this thesis is to develop an autopilot for small UAVs. In order to better demonstrate the autopilot's ability to fly small aircraft, the chosen flight platform exhibits many of the desired characteristics for small UAVs. A good UAV flight platform must stay airborne long enough to complete a useful mission, be small enough to be portable, be capable of carrying the autopilot and payload, and be resistant to damage caused by crashes and hard landings. The battery, motor and component selections also heavily influence the performance of the flight platform. Table 3.1 outlines the design goals for the flight platform.

Table 3.1: Flight platform requirements.

Specification	**Thesis Goal**
UAV size	48 inch
UAV weight	< 32 oz
Payload weight	> 6 oz
Avionics weight	< 3 oz
Flight time	> 30 min

The Zagi THL airframe (shown in Figure 3.3) by Trick RC fulfills the size and payload requirements. The Zagi THL has many advantages as the autopilot's airframe. First, the THL is a simple flying wing shape with very few appendages to break or suffer damage in a crash. Secondly, the airplane has a soft leading edge, which absorbs impact energy protecting the aircraft and autopilot. Third, the THL is extremely lightweight, weighing only 11 ounces. Fourth, the flying wing design has plenty of room for installation of the autopilot avionics and antennas. By choosing the THL instead of an airframe with a pre-installed electric motor, an efficient choice of motor and battery with higher performance and runtime could be installed. Table 3.2 lists the airframe specifications before and after the conversion. This conversion will be discussed in detail in Section 3.3.3

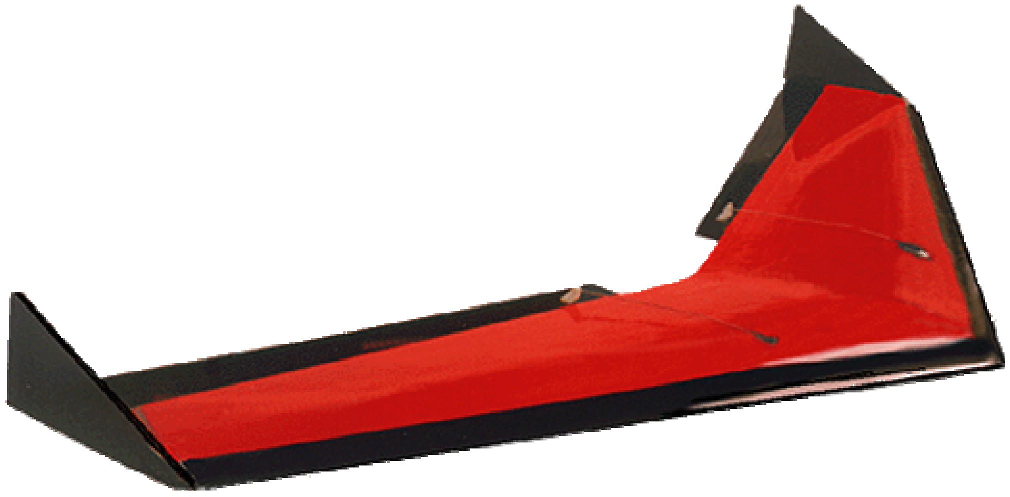


Figure 3.3: Stock Zagi THL combat flying wing glider from Trick RC.

Table 3.2: Airframe specifications.

Specification	Stock THL	Ready to Fly
Wing Span	48 inch	48 inch
Wing Area	2.83 ft^2	2.83 ft^2
Airfoil	Zagi999	Zagi999
Weight	11.5 oz	30 oz
Wing Loading	4 oz/ft^2	10 oz/ft^2

3.2.2 FLIGHT PLATFORM COMPONENTS

Flight platform component selection is critical in meeting the runtime and payload specifications. Hitec HS-55 servos are the primary flight control actuators, due to their quick response time, robustness, and light weight. The FMA direct FS-5 receiver is used because of its lightweight, failsafe circuitry. The Hacker b12-15l brushless motor with 4:1 gear reduction is efficient with a high power-to-weight ratio. A 9x6 GWS soft plastic propeller was used because it is a good match for the Hacker and is soft enough to flex (instead of breaking) during ground strike on landing.

Battery selection is one of the most critical factors in meeting flight time and UAV range requirements. It is a balance between weight and energy required. The battery must contain enough energy to power the motor for the duration of the cruise flight, while being capable of delivering the high current needed by the motor during climb. To achieve the necessary voltage for maximum power out of the motor/propeller combination, Kokam lithium polymer cells were wired in a 3-cell series configuration. The necessary battery capacity to achieve 30 minutes of flight time was determined through experimentation. Three Kokam 3270 MAh cells wired in series provide approximately 35 minutes of flight time.

Table 3.3 contains the list of components used in the flight platform, their approximate price, and the source.

Table 3.3: Flight platform component list.

Component	Price	Source
Zagi THL	$50	www.zagi.com
HS-50 servos	$40	www.hobby-lobby.com
Hacker B15-12L Brushless motor	$150	www.hackerbrushless.com
Jeti 18-3p Sensor less ESC	$80	www.hobby-lobby.com
FMA Direct FS-5 Receiver	$100	www.fmadirect.com
Kokam 3270 LiPo Battery	$100	www.fmadirect.com

3.2.3 ELECTRIC CONVERSION

The Zagi THL airframe is designed as a thermal hand launch glider. To achieve powered flight, a propulsion system is installed in the glider. A slot for the motor is cut in the trailing edge on the centerline of the Zagi. The elevons are trimmed for propeller clearance. Holes are cut in the foam to accommodate the autopilot and flight pack. Figure 3.4 shows the completed UAV flight platform.

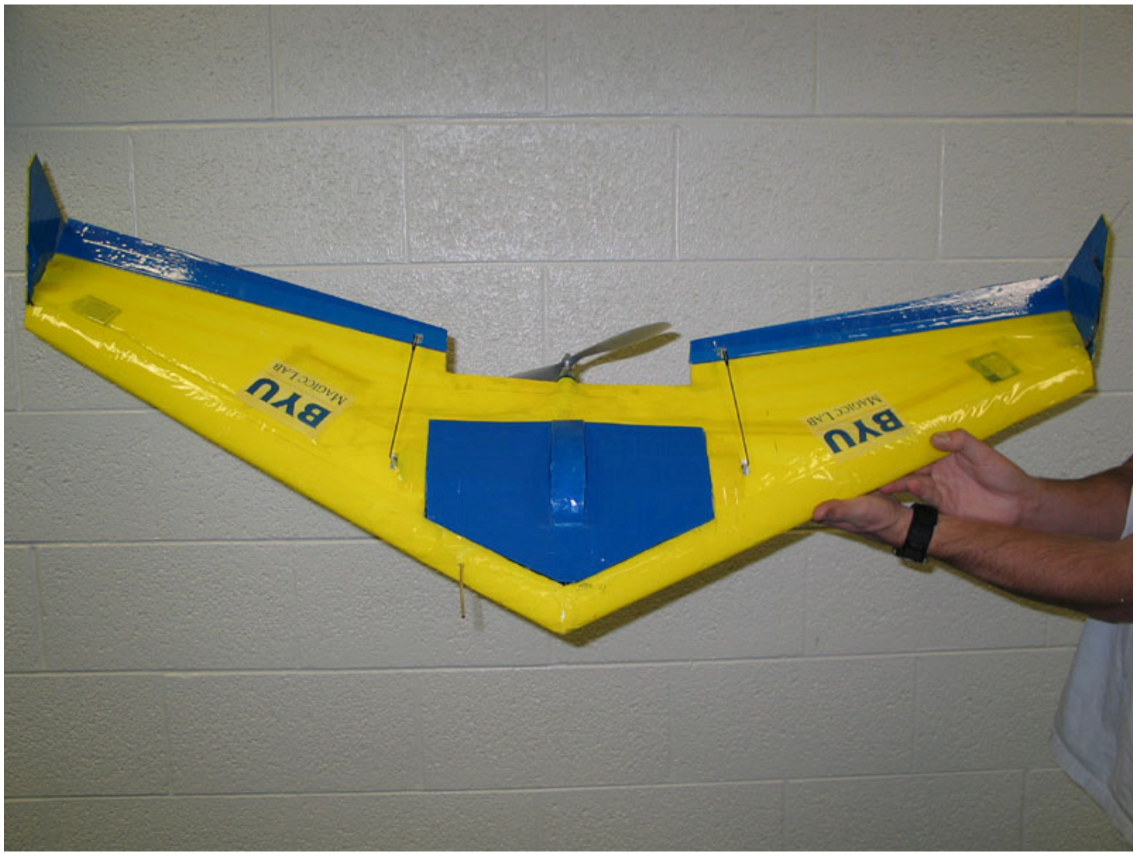


Figure 3.4: Completed Zagi THL flight platform with electric conversion.

3.3 AVIONICS

Figure 3.5 shows the avionics subsystem. The avionics consist of the autopilot board (1), GPS receiver (2), the communications hardware (3), and the Bypass circuit (4). The following discussion of the avionics components will begin by outlining the autopilot main board. Next, the GPS unit and digital communication link will be covered. The last section will detail the need for and development of the Bypass circuit.

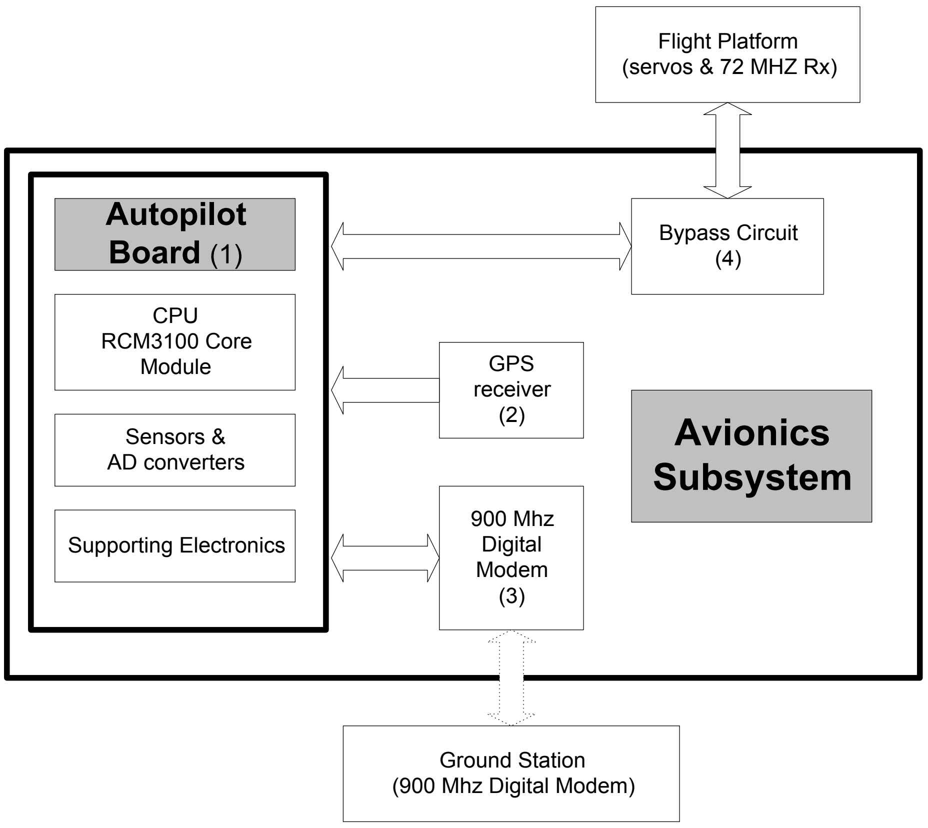


Figure 3.5: Avionics hardware diagram.

3.3.1 AUTOPILOT BOARD

The autopilot board is arguably the most important and complex piece of hardware in the UAV system. The function of the autopilot is to control the aircraft using the aircraft states, the user-programmed mission, and the pre-programmed fail-safe functions. The autopilot is a compact unit that contains: the Central Processing Unit (3.6(2)), sensors to measure the aircraft states (3.6(1)), input/output ports to accommodate the payload, GPS and communication devices (3.6(4)), and electronic components to support these devices (3.6(3)). This section will discuss the elements of the autopilot hardware. First will come discussion of CPU selection, followed by discussion of the various sensors and hardware needed to determine the aircraft states. It will end by detailing the interfaces that support the GPS, radio modem, and payload.

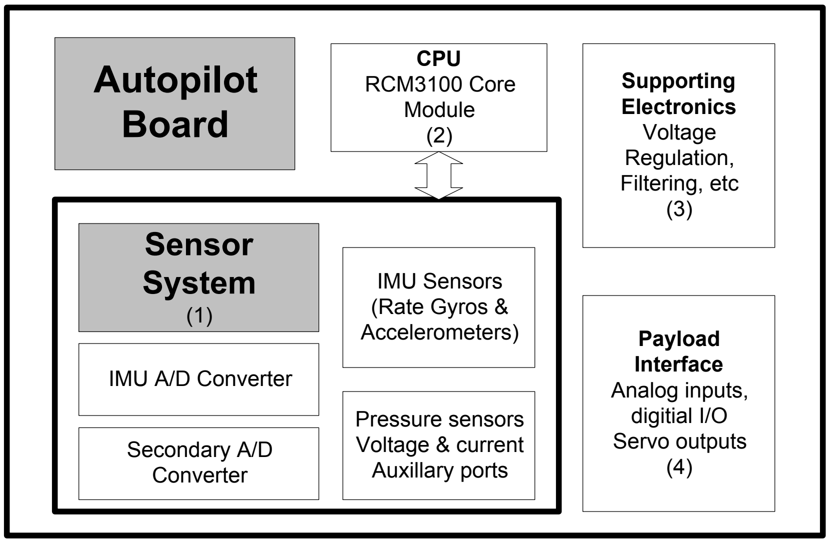


Figure 3.6: Autopilot main board block diagram.

The CPU (3.6(2)) is the heart of the autopilot. It is responsible for processing sensor data, handling I/O to the GPS, modem, and Bypass circuit, running the low-level control algorithms, and handling communications with the ground station. Based on these requirements, the CPU must have abundant serial and digital I/O ports, and be capable of rapid 32-bit floating point operations. It must also have sufficient RAM and flash memory for autopilot source code storage and runtime execution. The chosen processor, the RCM3100 core module, has 6 serial ports, 50 general-purpose I/O ports, 512 Kbytes RAM and 512 Kbytes of flash memory, software floating-point support, and extensive C libraries. It is also small and lightweight. While the RCM3100 has many benefits, two drawbacks worth mentioning are the 29 MHz clock speed and 8-bit operation. While fast enough to run the sensor processing and control algorithms, the RCM3100 does not have the computing power for complex operations such as video processing.

SENSORS

To control the aircraft effectively, an accurate, timely estimate of aircraft states is needed. The purpose of this section is to outline the method used to gather the aircraft state information. The commonly used notation for the 12 aircraft state variables follows:

1. X = inertial position of the UAV along x1 north
2. Y = inertial position of the UAV along y1 east
3. H = altitude of the aircraft
4. Vp = pressure airspeed (indicated airspeed)
5. Beta = sideslip angle
6. Alpha = angle of attack
7. ϕ = roll angle
8. θ = pitch angle
9. ψ = yaw angle (heading)
10. P = body fixed roll rate
11. Q = body fixed pitch rate
12. R = body yaw rate

To implement the control algorithms discussed in Chapter 2, only 10 of the 12 state variables must be measure or estimated: X,Y,H, Vp, ϕ, θ, ψ, P, Q, and R. Direct sensing of these aircraft states is ideal because it is fast and generally accurate. However, some states are difficult to measure directly. At this time, sensors to measure roll (ϕ), pitch (θ), and yaw (ψ) are not suitable because of high price, large size, and weight. Therefore, these states are estimated by combining information from several sensors. This section will discuss the sensors used to directly measure and estimate the aircraft states.

GPS

The first states of interest are X and Y, which represent the aircraft's inertial position in the north direction and east direction, respectively. These states are measured directly using the GPS receiver.

RATE GYROS

The next states of interest are the body fixed rotational rates, P, Q, and R. There are several low-cost, lightweight piezo gyros available for this purpose. One gyro, the Tokin LD-16, accurately senses the aircraft states but suffers from a high drift rate, and susceptibility to interference from the onboard 900 MHz digital modem. Another gyro, the Adxrs300, is more ideal, having a low drift rate, small physical size, and a built-in temperature sensor. The Adxrs300 has the additional advantage of a maximum measurable rotational rate of 300 degrees/second, as compared to the Tokin's 90 degrees/second rate. For these reasons, the Adxrs300 rate gyro is used to measure P, Q, and R.

PRESSURE SENSORS

Altitude and Airspeed (H and Vp) can be measured using pressure sensors. Altitude is measured using the Motorola MPX3000 absolute pressure sensor. Airspeed is measured using the Motorola MP4015 differential pressure sensor. Both of these sensors have internal temperature compensation and signal conditioning. Because of the small changes in pressure resulting from changes in altitude, the signal from the absolute pressure sensor is amplified before being read by the A/D converter. The differential pressure sensor output is also amplified. A first-order, low pass filter is employed on the output of both sensors to attenuate high frequency noise. The GPS also produces an altitude estimate. However, the GPS altitude is not suitable for controlling the aircraft because it suffers from high latency and low sensitivity to small changes in altitude. However, the GPS altitude measurement is useful as it can be used to calibrate the absolute pressure sensor. The GPS can also be used to calibrate the differential pressure sensor, but this is only useful in an environment of low wind, when the GPS-measured ground speed is directly related to airspeed.

ATTITUDE ESTIMATION

The earth referenced Euler angles, ϕ, θ, and ψ, are the most difficult states to measure. Decisions presented on methods and hardware to measure attitude stem from considerable research and investigation. Of the three approaches researched, the first two (involving off-the-shelf devices) proved unsuitable for our application because of cost and implantation issues. The third approach involved building an in-house inertial measurement unit using MEMS accelerometers and rate gyros. The in-house unit was successful in that it produced usable estimates of ϕ, θ, and ψ. The unit weighs less than an ounce and costs less than $150 to produce. We will discuss the initial approaches, and then present the sensors used in the in-house IMU.

The first approach is to use an off-the-shelf inertial measurement unit (IMU). Popular off-the-shelf IMUs combine rate gyro information with accelerations and magnetic field measurements to form an attitude estimate. These IMU options were eliminated for various reasons. The Crossbow IMU300 produces accurate estimates of ϕ, θ, and ψ, but its \$6000 price and weight are unacceptable. The 3DM-G-M from MicroStrain, Inc. is low-cost (\$1,195) and lightweight (27.5 grams without enclosure). However, though it produces an accurate measurement in static conditions, this unit produces inaccurate results when placed in a coordinated turn condition. Because coordinated turns are common in normal UAV flight, the 3DM-G-M is unsuitable for a flight application.

The next approach is to use an off-the-shelf wing-leveling device manufactured for radio-controlled aircraft hobbyists. One such device, Co-Pilot Dave from FMA Direct, contains a set of orthogonal passive infrared sensors that use the difference in temperature between the ground and sky to produce an error signal. When the aircraft deviates from level flight attitude, this error voltage is produced proportional to the deviation. The microcontroller in the unit then applies feedback to the elevator or aileron actuator to level the aircraft. As the autopilot application only requires the sensing unit portion of Co-Pilot Dave, a test was performed in which the sensor unit was interfaced directly into the A/D converter on the autopilot board. The CPU sampled the sensor and converted the voltage into roll and pitch angle estimates. With proper calibration, the unit produced roll and pitch measurement, accurate within 7 degrees. However, the calibration routine was lengthy and required second-order curve fitting. The first- and second-order coefficients changed with the varying temperatures of the ground and sky. On some days, the sensor had to be recalibrated hourly to produce accurate outputs. While Co-pilot Dave provided a useable estimate of pitch and roll, the extensive calibration and re-calibration made it impractical for use in the final product.

The third approach involves building an IMU using off-the-shelf piezo rate gyros and accelerometers. Several groups have published information on the construction of IMUs and the processing of the sensor outputs to produce an attitude estimate. One such group is the GNU Autopilot Project [12]. Their approach uses three axis rate gyros and three axis accelerometers combined with an extended Kalman filter to form an estimate of ϕ, θ, ψ. The in-house IMU uses the same basic hardware and a simplified version of GNU Autopilot Project's Kalman Filter algorithm to estimate attitude. The algorithm will be discussed in Chapter 4. ADXRS300 rate gyros used to directly measure P, Q, and R provide the angular rate measurements, and two ADXL202E two axis MEMS accelerometers from Analog Devices provide acceleration measurements. The rate gyros and accelerometers are mounted orthogonally such that rotational rates and accelerations about the body axes can be sensed. Figure 3.7 shows the orthogonally mounted rate gyros and accelerometers used to estimate aircraft attitude.

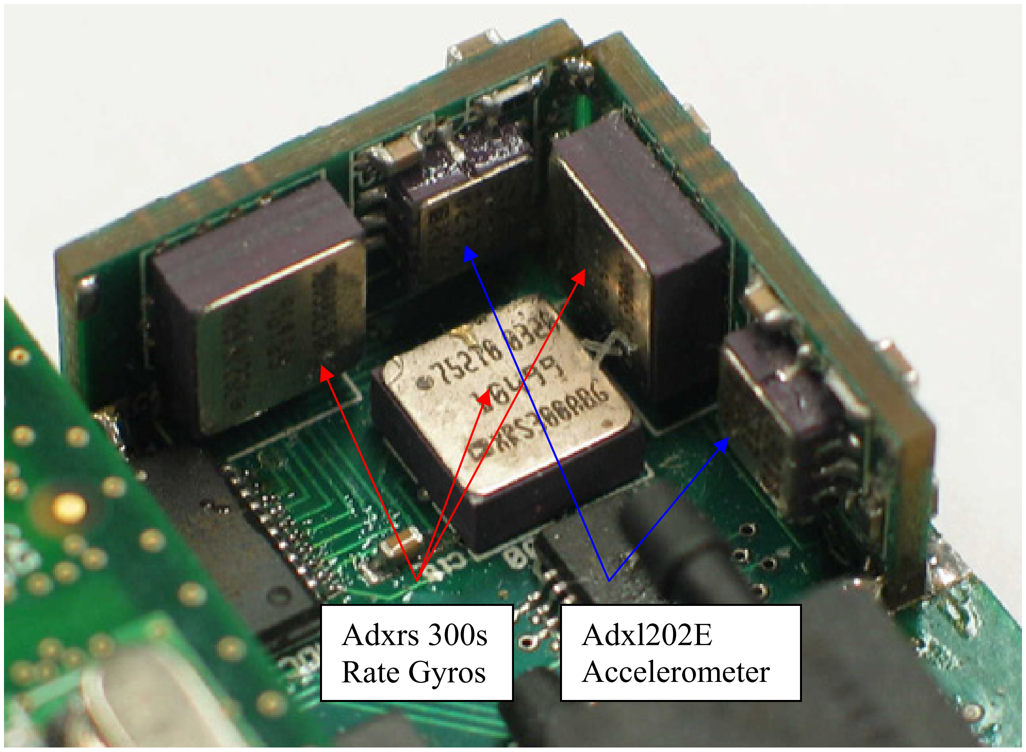


Figure 3.7: In-house IMU hardware.

SENSOR INTERFACE HARDWARE

This section discusses the analog signal conditioning and analog-to-digital conversion of the sensor outputs. First the analog techniques used to minimize noise in the sensor readings will be presented, followed by the selection of the analog to digital converters (ADCs).

Several important factors must be considered when digitizing analog signals. First, the power to the analog sensor lines and ADCs should be carefully filtered to reduce internal noise. Second, mixed-signal techniques should be used when laying out the circuit to prevent noise from the digital circuitry from crossing over to analog traces.

Third, the analog sensor signals should be filtered prior to the analog to digital conversion. The autopilot design incorporates all three techniques.

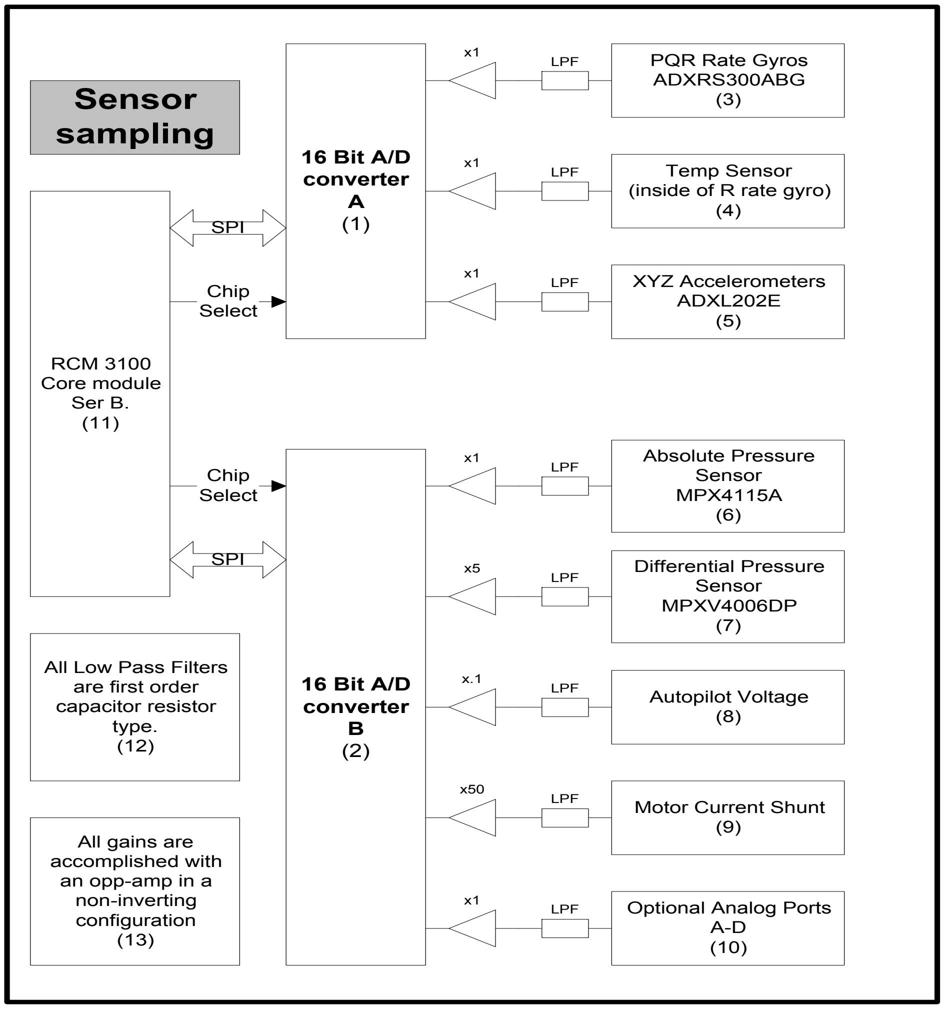

Sensor sampling
RCM 3100 Core module Ser B. (11)
SPI
Chip Select
16 Bit A/D converter A (1)
Chip Select
SPI
16 Bit A/D converter B (2)
All Low Pass Filters are first order capacitor resistor type. (12)
All gains are accomplished with an opp-amp in a non-inverting configuration (13)
x1
LPF
PQR Rate Gyros ADXRS300ABG (3)
x1
LPF
Temp Sensor (inside of R rate gyro) (4)
x1
LPF
XYZ Accelerometers ADXL202E (5)
x1
LPF
Absolute Pressure Sensor MPX4115A (6)
x5
LPF
Differential Pressure Sensor MPXV4006DP (7)
x.1
LPF
Autopilot Voltage (8)
x50
LPF
Motor Current Shunt (9)
x1
LPF
Optional Analog Ports A-D (10)

Figure 3.8: Sensor Sampling Diagram.

Suitable analog-to-digital converters must convert the autopilot's conditioned analog signals to digital values for the RCM3100 microprocessor. The Texas Instruments ADS8344 A/D converter is suitable because of low noise characteristics, 16 bits of precision, support of the SPI interface used by the RCM3100. The SPI interface simplifies the development of the A/D interface code on the CPU and lowers the amount of CPU time required for each conversion. There are 12 analog signals on the autopilot to be read for normal operation. Two ADS8344 analog to digital converters, with eight signal lines each, are used to accommodate this number of signals. Figure 3.8 shows the layout of the sensor and sampling hardware. The additional four analog lines allow for future development or payload interface. Table 3.4 summarizes the information on the sensors used on the autopilot.

Table 3.4: Autopilot sensors.

Sensor	**Direct states**	**Estimated States**	**Manufacturer**	**Part Number**	**Interface**
Roll Rate Gyro	P	Phi,Theta,Psi	Analog Devices		Analog
Pitch Rate Gyro	Q	Phi,Theta,Psi	Analog Devices		Analog
Heading Rate Gyro	R	Phi,Theta,Psi	Analog Devices		Analog
X accelerometer	Ax	Phi,Theta,Psi	Analog Devices		Analog
Y Accelerometer	Ay	Phi,Theta,Psi	Analog Devices		Analog
Z Accelerometer	Az	Phi,Theta,Psi	Analog Devices		Analog
Differntial Pressure	Vp		Motorola		Analog
Absolute Pressure Sensor	H		Motorola		Analog
GPS Lat	X		Furuno		Aysnch serial
GPS Lon	Y		Furuno		Aysnch serial
GPS Altitude	H		Furuno		Aysnch serial
GPS Heading	Velocity Vector	Phi,Theta,Psi	Furuno		Aysnch serial
GPS Velocity	Ground Speed		Furuno		Aysnch serial

3.3.2 I/O

The autopilot board must communicate with several devices during normal operation. These devices include the GPS, digital modem, Bypass circuit, and payload. The GPS, digital modem, and Bypass circuit use asynchronous serial communication. The payload interface, however, varies depending on user requirements. In order to support a variety of payloads, the autopilot was designed with several different interfaces that may that may be used by the payload. These optional interfaces are accessible via a set of connectors on the front of the autopilot board. These connectors are shown in Figure 3.9. Table 3.5 further details the interface ports and their functions.

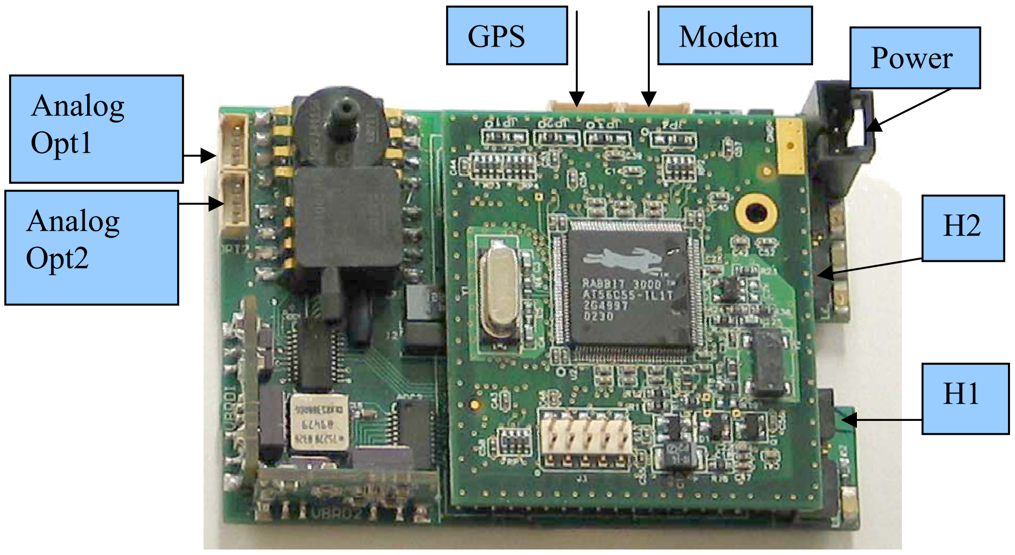


Figure 3.9: Autopilot board showing interface ports.

Table 3.5: Autopilot external interface ports.

Connector Name	Type	Function
Opt1	Analog Input	Optional Analog Input
Opt2	Analog Input	Optional Analog Input
GPS	Serial	GPS Port
Modem	Serial	Modem Port
Power	Power	Main Autopilot Power
H1	Digital I/O	Payload and Bypass Interface
H2	Digital I/O	Payload interface

3.3.3 GPS RECEIVER

The GPS receiver used on the autopilot is the Furuno GH-80. This receiver suits the autopilot application for several reasons. First, it has an integrated antenna, which makes it self-contained and low-maintenance. Second, it is very small and lightweight, weighing only 17 grams. Third, it has desirable electronic characteristics. These include: a 12-channel receiver, low fix time, and the ability to accept differential corrections. Figure 3.10 shows the Furuno GPS installed in the UAV. The Furuno suffers from the standard limitations of non-differential GPS. First, its accuracy is limited to around 10 meters. The second limitation is the 1 Hz update rate. The position, heading, velocity, and altitude are updated only once a second. Because of the aircraft's low velocity, this is sufficient for most of the control algorithms. The autopilot control could be improved if a filter were used to provide a position and heading estimate between GPS samples. This is also an area of future work.

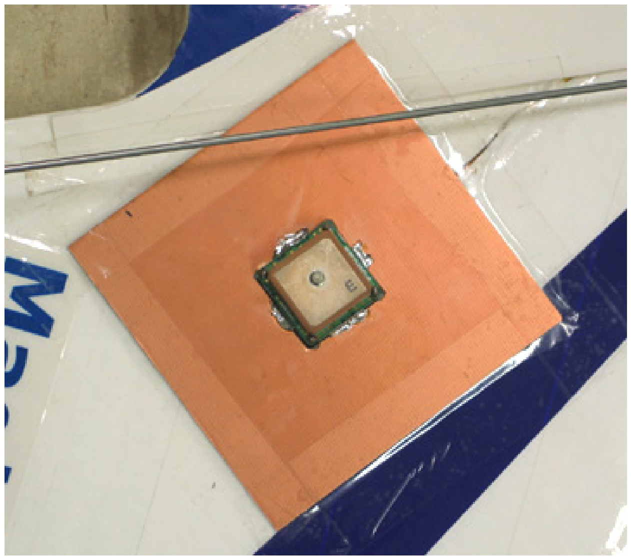


Figure 3.10: Furuno GH-80 GPS installed.

3.3.4 COMMUNICATION HARDWARE

An essential part of the autopilot avionics is the digital communication link. The digital communications link provides several important functions. First, it provides real-time status updates to the user. Second, it allows the autopilot to be dynamically configured in-flight. (This allows for in-flight gain tuning and sensor monitoring.) Third, it provides a means for commands to be sent to the autopilot. Fourth, it provides a method for Pilot-in-the-Loop aircraft control. The digital communications link can be used in place of the 72 MHz RC link to control the aircraft if desired. This section will discuss the specifications in choosing the digital modem for the communication link. The two digital modems that were tested will be presented, followed by detailing the integration of the digital modem into the airframe.

The digital modem has five specifications. First, the modem must support the data rate needed to control and monitor the aircraft. Experience with mobile robots and autonomous aircraft suggests that 9600 baud is adequate, while higher rates are more desirable. Second, the modem must have a 5 km range using an omni-directional antenna. Third, the modem must weigh less than 1 ounce. The fourth factor is cost. In order to keep the avionics cost below $2000, the digital communication link must cost less than a few hundred dollars. Finally, latency (time between data arriving at the interface port on the transmitter, and the corresponding data arriving at the serial interface on the receiving modem) must be low to achieve responsive manual control over the digital communications link. This is a desirable characteristic for Human-in-the-Loop control during autopilot development.

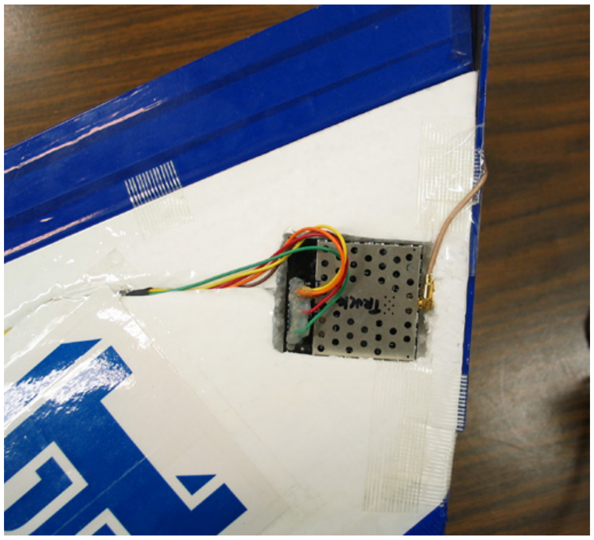


Figure3.11: Aerocomm AC4490-500-M3 digital modem installed in a UAV.

Two modems that fit the size, weight, range, cost, and bandwidth specifications were purchased for further testing. The 9XStream OEM module by Maxstream was selected for testing, in addition to the Aerocom AC4490-500 OEM module (this modem is shown in Figure 3.11). Both modems operate in the 900 MHz spread spectrum band, and are similar in size and weight. We tested both modems on the Zagi UAV. The basic setup for the test is as follows. The modems were placed one at a time into the wing of the UAV. A custom-built ¼-wave dipole antenna was mounted vertically in the wing tip. We will talk more about this antenna in the next section. The manufacturer-supplied antennas were used on the ground station side. Maxstream provided a 3-dB gain dipole while Aerocom provided a 2-dB gain dipole. The UAV was configured to transmit 40 bytes of data, 10 times a second. The ground station was programmed to acknowledge the UAV packets by sending 5-byte acknowledgment packets. The Aerocomm modem was configured to transmit at 1/3 maximum power output. The Maxstream was configured to transmit at full power. The UAV was flown straight and level away from the ground station at 350 feet altitude. The UAV was programmed to continue its flight until no acknowledgment packets were received from the ground for 3 seconds. If this condition did not occur within 5 km from the ground station, the UAV was programmed to fly home. This test was conducted several times. When tested with the Maxstream modem, the UAV consistently turned around at the 2 km point. When tested with the Aerocomm, the UAV was able to fly the entire 5 km without interruption in communications. Based on these results, the Aerocomm modem was selected for the final design. The Aerocomm has the additional advantages of higher over-the-air data rate (57600 baud vs. 9600 baud) and slightly smaller size. The higher power output means the Aerocomm will use more battery power, but the amount used is dwarfed by that consumed by the UAV propulsion system.

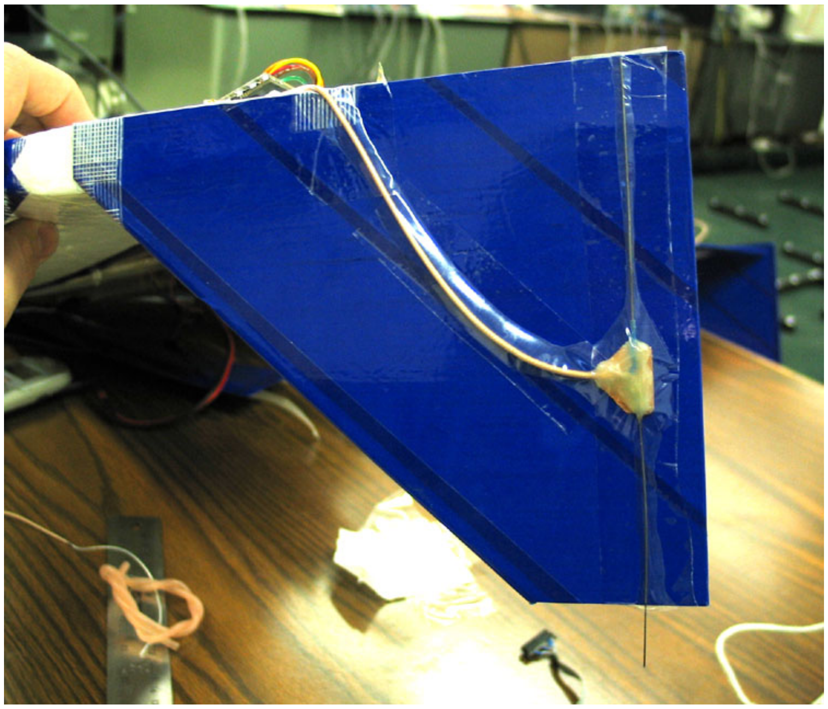


Figure 3.12: 900 MHz digital communications dipole antenna installed in a UAV.

An important component of the digital communication link is the antenna. Omni-directional antennas are used on the UAV and ground station because they do not require readjustment during flight. The UAV antenna is a custom-built dipole with ¼-wave radiating elements. This antenna is shown in Figure 3.12. The antenna is mounted on the UAV in a vertical orientation to match the vertical polarization of the ground station antenna in level flight. When the aircraft banks, there is a degradation in signal strength as the polarization of the UAV antenna changes with respect to the ground station. The UAV is designed to function normally during these brief periods of degraded communication. Lost communication behavior will be discussed in Chapter 4. A manufacturer-supplied ¼-wave 2-dB dipole antenna is used on the ground station (WCP-2400-MMCX).

3.3.5 BYPASS CIRCUIT

The Bypass circuit is an essential component of the onboard avionics during autopilot development. Should a problem arise during testing that puts the aircraft in danger, a human safety pilot can immediately take control of the aircraft using a switch on the RC transmitter. The human pilot can either land the aircraft, or fly the aircraft until the problem is resolved and control returned to the autopilot. This approach to autopilot development allowed testing of new algorithms and hardware with minimal risk to the aircraft or people on the ground. This section will discuss the design of the Bypass circuit. A photo of the Bypass circuit is shown in Figure 3.13.

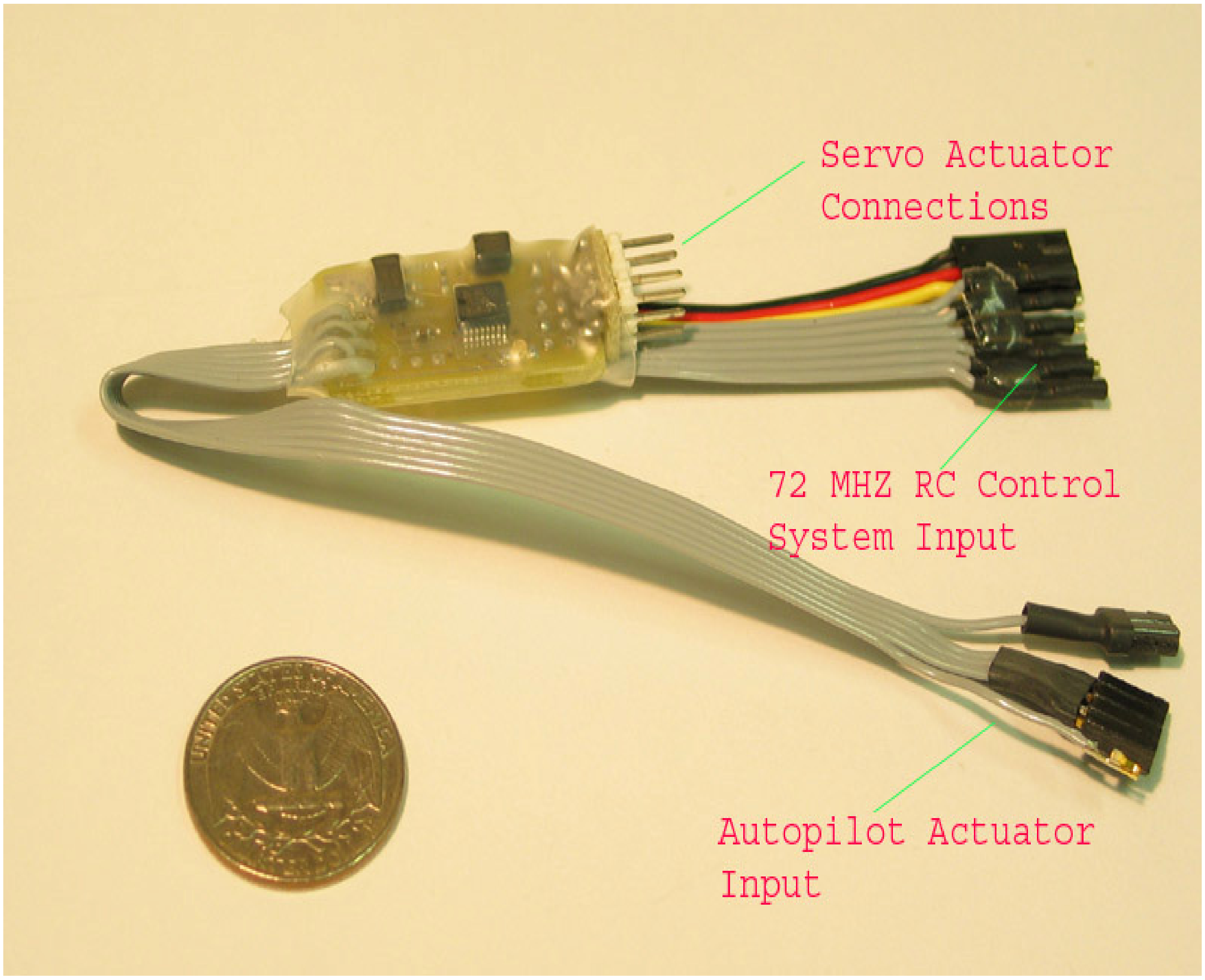


Figure 3.13: Photo of the completed Bypass circuit.

When a human pilot takes control of the airplane via the RC transmitter, this switches control of the UAV actuators from the autopilot to the onboard 72 MHz control system. The Bypass circuitry decodes the servo position signals from the 72 MHz RC receiver. This allows the RC transmitter stick positions to be used as flight control inputs, and also allows the stick positions to be logged for system identification and flight analysis.

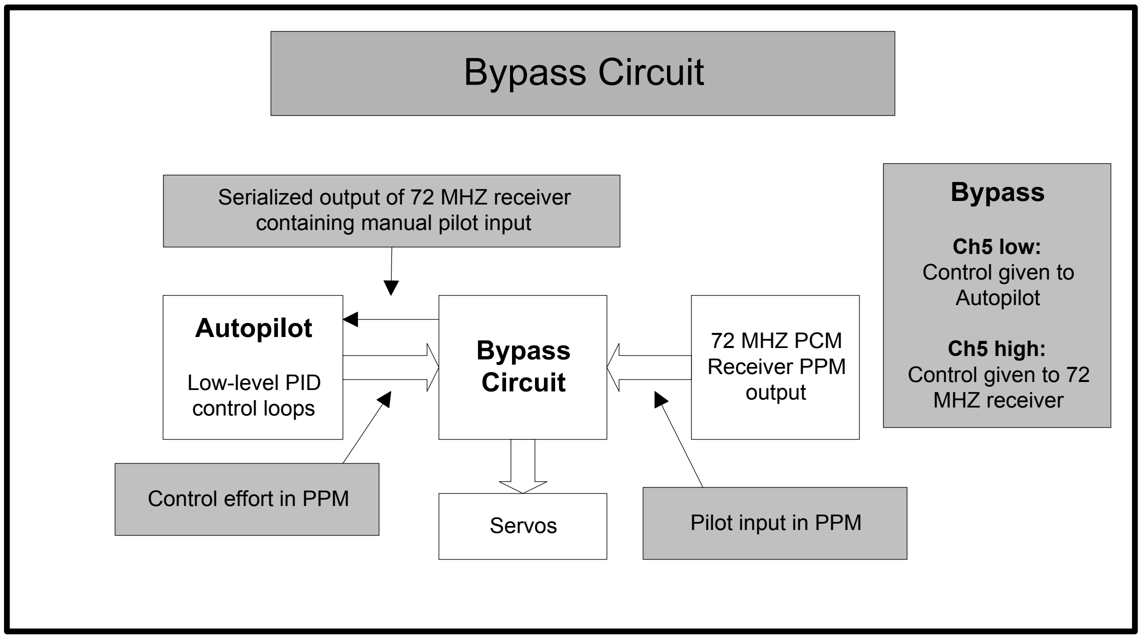

Bypass Circuit
Serialized output of 72 MHZ receiver containing manual pilot input
Bypass
Ch5 low:
Control given to Autopilot
Ch5 high:
Control given to 72 MHZ receiver
Autopilot
Low-level PID control loops
Bypass Circuit
72 MHZ PCM Receiver PPM output
Control effort in PPM
Servos
Pilot input in PPM

Figure 3.14: Bypass circuit diagram.

The Bypass circuit contains a PIC microcontroller and a digital multiplexer. The microcontroller decodes the pulse position modulation servo position signals from the RC receiver. Channels 1-4 and 6-8 are decoded into values representing the servo pulse high time. Channel 5 is decoded separately and is used as the control signal. If the Channel 5 pulse falls below a set threshold, the microcontroller switches the digital multiplexer to connect the RC receiver servo outputs directly to the servos. This mode is known as Pilot-in-Control mode (PIC), as the aircraft is controlled by a human pilot through the 72 MHZ control link. If the channel 5 pulse rises above the threshold, the digital multiplexer connects the autopilot servo outputs to the servos. This is known as Computer-in-Control mode (CIC) as the autopilot has control of the aircraft. The decoded servo positions and PIC/CIC mode information are sent the autopilot over an asynchronous serial port at 20 Hz. The data is sent regardless of control mode. During

PIC mode, the autopilot logs the servo control information which can be used for flight analysis. In CIC mode, the autopilot can be configured to use the servo position signals as control inputs to the low level PID loops for autopilot-assisted flight. The autopilot can also be configured to use the control signals in conjunction with the control efforts of the PID loops to test disturbance rejection. Figure 3.15 is a diagram of the Bypass circuit. Figure 3.15 shows the Bypass circuit connected to the autopilot, servos, and RC receiver.

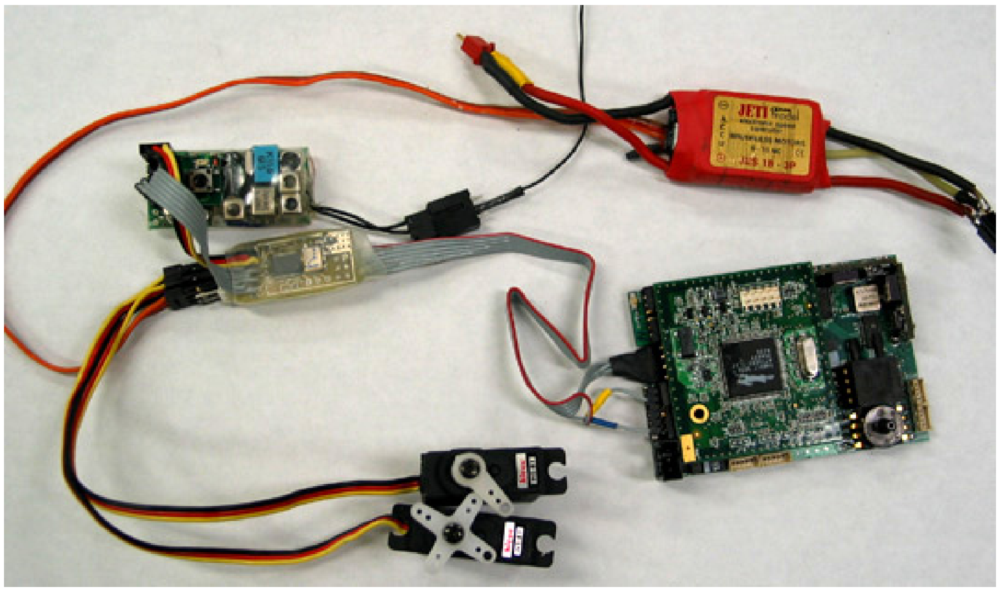


Figure 3.15: Bypass circuit connected to the autopilot, servos, and RC receiver.

3.3.6 PAYLOAD

The autopilot is designed to accommodate and interface with a wide variety of payloads. To demonstrate the effectiveness of the UAV as an observation platform, a video surveillance payload is used for testing. The video system consists of the airborne camera and transmitter, and the ground-based receiver and video display. The discussion of the video system components is divided into two sections. The first covers the airborne components. The second is the ground-based equipment to receive and display the video.

The first component of the airborne system is the camera. The Panasonic GP-CX171 hi-resolution CCD camera is used because of its high resolution and low weight. The second component is the video transmitter. The Felsweb TX612 is used because of its low cost, 20 gram weight, small size, and 500 mW power output. The Felsweb transmitter, when used with a gain antenna on the receiving side, is capable of a 5-km range. The final component of the airborne video system is the antenna. Because the orientation and position of the aircraft with respect to the ground constantly changes, an omni-directional antenna must be used on the aircraft. A ¼-wave dipole is used for this purpose. Figure 3.16 shows the video camera and transmitter installed in the UAV.

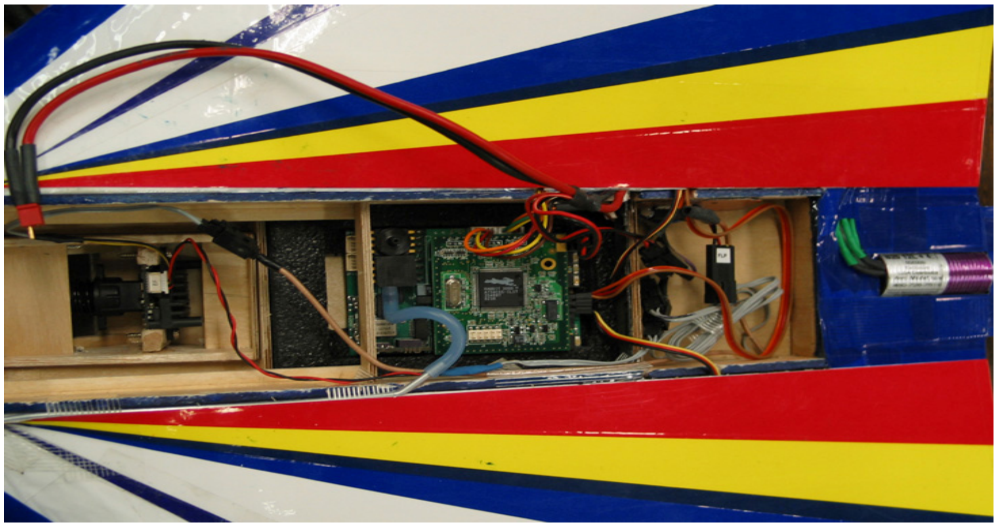


Figure 3.16: Video payload installed in UAV.

3.4 GROUND STATION HARDWARE

The final section of this chapter deals with the hardware used in the ground station. The purpose of the ground station hardware is to control and monitor the UAV and payload. The ground station hardware can be divided into two systems. The first is the autopilot-specific hardware. The second is the payload-specific hardware.

There are three components that make up the autopilot-specific ground station hardware. The first is the digital modem that communicates with the digital modem in the UAV. A matching Aerocomm AC4490 digital modem is used. Because the Aerocomm modems have a 5 km range without the use of directional antennas, the Aerocomm ¼-wave omni-directional antenna is used for the ground station communications antenna. The third component of the ground station is the user interface. A laptop computer running Microsoft Windows is used. An in-house graphical user interface controls the autopilot. The development of the graphical user interface will be discussed in Chapters 4 and 5. The ground station hardware is shown in Figure 3.17.

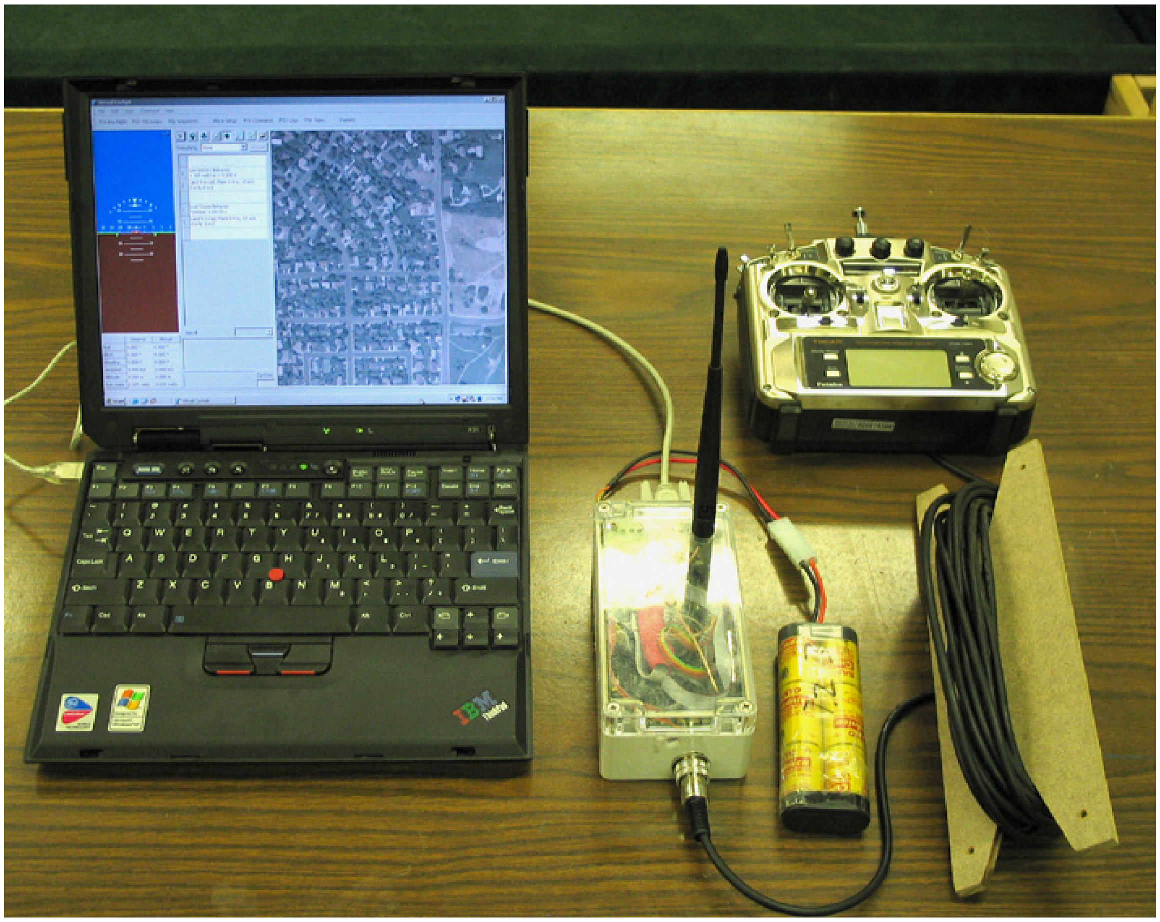


Figure 3.17: Ground station hardware.

The payload-specific portion of the ground station consists of the ground-based hardware necessary to make use of the payload carried by the UAV. In the case of the video surveillance payload, a video receiver, antenna, and video display are needed. The Felsweb RX612 receiver receives the video from the airplane. A high-gain directional patch antenna from Hyper Link Technologies is necessary to achieve a 5 km range. A Sony GV-D1000 Mini DV cassette recorder is used to display and record the video.

CONCLUSION

The purpose of this chapter was to outline the development of the hardware that makes up the autopilot system. The chapter began with a discussion on the Zagi airframe. The next section detailed the system avionics. The chapter concluded with a brief description of the hardware that makes up the ground station.

CHAPTER 4

AUTOPILOT SOFTWARE

The purpose of this chapter is to discuss the software on the autopilot and in the ground station. The autopilot software is responsible for attitude estimation, processing sensor data, parsing GPS data, controlling the aircraft, and handling communications with the ground station. The autopilot software is written in Dynamic C, a language developed by Rabbit Semiconductor for use with Rabbit Semiconductor Core Modules. The Core Module used on the autopilot is the RCM3100. The ground station software's purpose is to provide an easy-to-use graphical interface to the autopilot. The ground station software runs on a Windows-based laptop computer. The primary goal of this thesis was to develop the autopilot; the ground station software was developed by a different group to support the autopilot effort. As such, the discussion on the ground station software will be limited in scope. The chapter will proceed as follows: first, the autopilot software structure and flow will be discussed. Next, the sensor processing, attitude estimation and sensor calibration will be detailed. The following sections will discuss the control and fail-safe algorithms. The chapter will finish with an overview of the communications protocol.

4.1 AUTOPILOT SOFTWARE STRUCTURE AND FLOW

The autopilot software is written with optimization, functionality, development, and compactness in mind. As shown in Figure 4.1, the autopilot software is divided into three sections. The first is the pre-main section. The pre-main code is responsible for initialization of the autopilot hardware, declaration of global variables, and initialization of global variables and structures. The pre-main code will be discussed in Section 4.1.1. The next section of the autopilot code is the main loop. The main loop runs continuously at approximately 130 Hz, and is responsible for gathering and processing sensor data and computing the low-level control algorithms. The main loop will be discussed in Section 4.1.2. The costates, processes that run in parallel with the main loop, comprise the remaining section of the autopilot code. The costate functions execute at specified intervals or on an event-triggered basis. Costates will be discussed in Section 4.1.3.

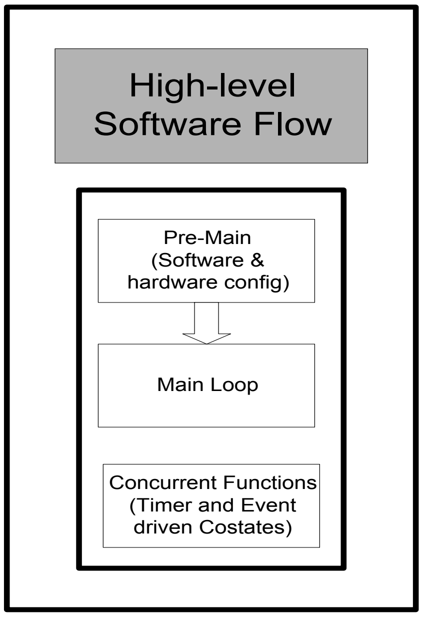

High-level Software Flow
Pre-Main (Software & hardware config)
Main Loop
Concurrent Functions (Timer and Event driven Costates)

Figure 4.1: Autopilot software high level diagram.

4.1.1 PRE-MAIN AUTOPILOT SOFTWARE

The pre-main autopilot code is responsible for declaring the global variables and structures, initializing them, and configuring the autopilot hardware. Figure 4.2 shows the sequence of events in the pre-main code. The global variables will be discussed first, hardware initialization second.

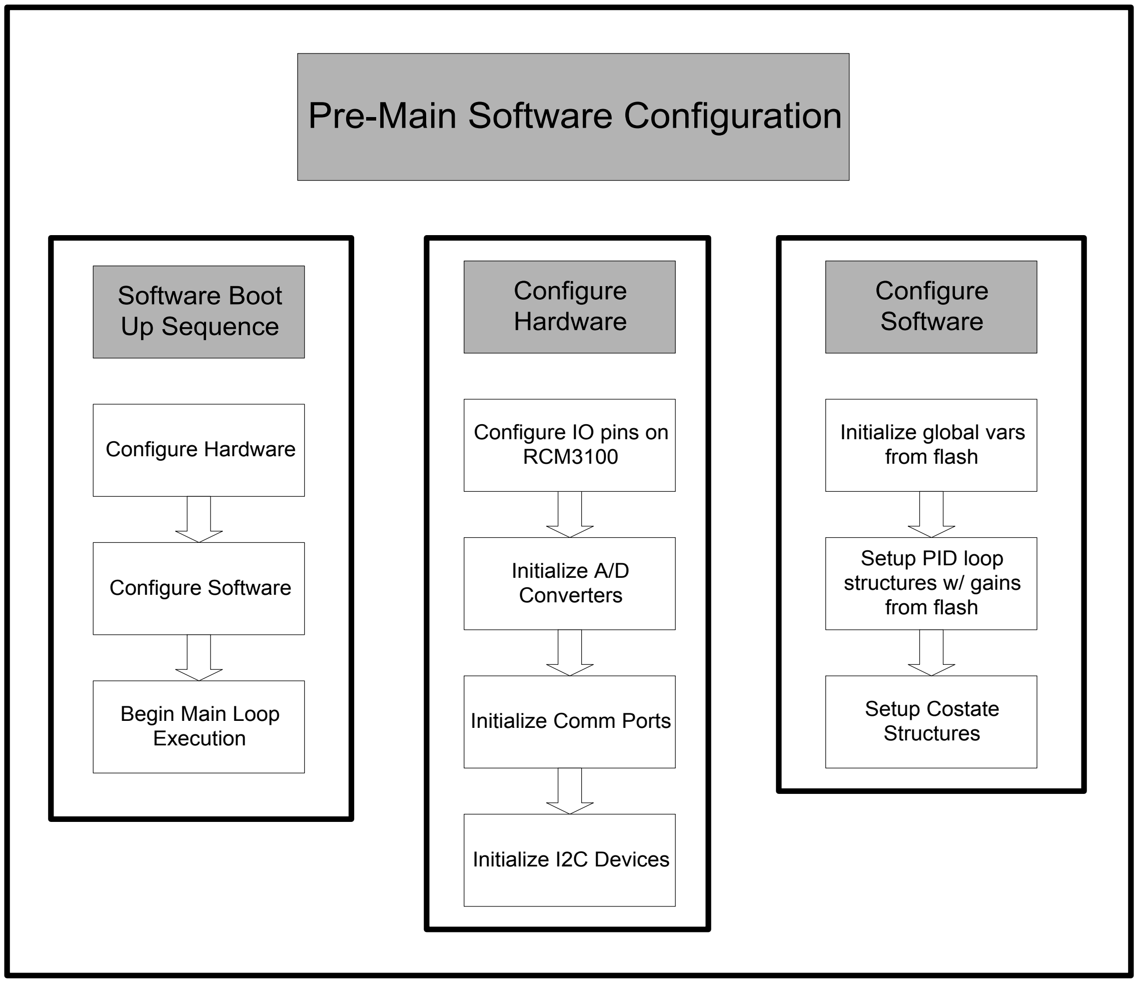

Pre-Main Software Configuration
Software Boot Up Sequence
Configure Hardware
Configure Software
Begin Main Loop Execution
Configure Hardware
Configure IO pins on RCM3100
Initialize A/D Converters
Initialize Comm Ports
Initialize I2C Devices
Configure Software
Initialize global vars from flash
Setup PID loop structures w/ gains from flash
Setup Costate Structures

Figure 4.2: Pre-main software flow.

GLOBAL VARIABLES AND STRUCTURES

The autopilot uses several sets of RAM-based and flash memory-based global variables and structures. The RAM-based variables are used during autopilot execution to store global type data accessible to all the autopilot functions. The flash memory serves to back up a subset of these global variables and structures. Figure 4.3 shows the organization of the global variables and structures.

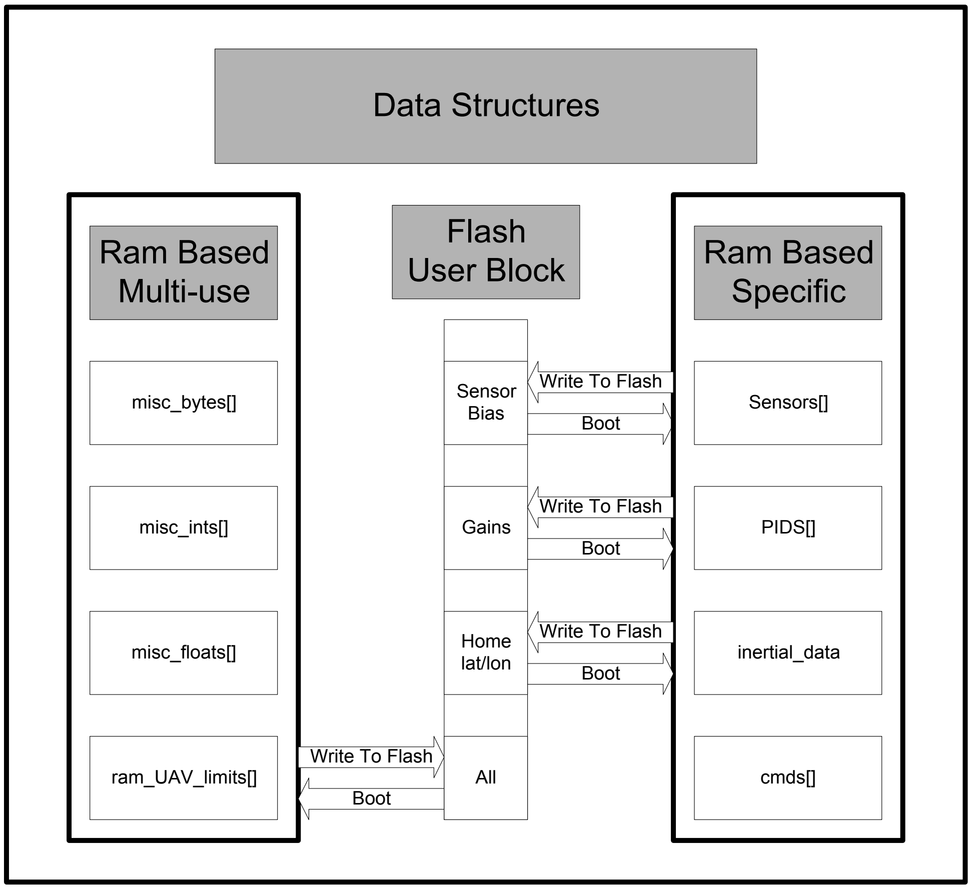


Figure 4.3: Global variables and structures.

The global variables are organized into four arrays. The purpose of the array organization is to allow the variables to be viewed and modified from the ground station to facilitate autopilot development and configuration. The first three arrays are known as Operational Variables, hereafter referred to simply as variables, as they contain most of the constant and dynamic data used by the autopilot during operation. The variables are not backed up in the nonvolatile flash memory. The variables are organized into arrays of 8-bit signed character values, 16-bit signed integer values, and 32-bit floating point values. The last array is known as the UAV Parameters array. The UAV Parameters, hereafter referred to as parameters, are constants that govern the behavior of the autopilot. These constants are designed to be manipulated by the user to configure the autopilot for different behaviors and airframes. The parameters are 32-bit floating point values that are backed up in nonvolatile memory.

Global structures are designed to organize the large amount of autopilot data into hardware- and function-based values that can be written to and read from the ground station. There are three types of structures. The first is the GPS structure. This structure contains static and dynamic GPS specific information. The second structure is the sensor structure. The sensor structure contains raw, biased, and processed sensor information. The bias portion of the sensor structure is backed up in nonvolatile memory. An instance of the sensor structure exists for each sensor. The last structure is the PID structure. The PID structure contains static and dynamic information for each PID loop. The PID gains and efforts are stored in this structure. The PID gains are backed up in nonvolatile memory. An instance of the PID structure exists for each PID loop.

The global structures and variables are instanced and initialized in the pre-main portion of the autopilot code. The variables are initialized to hard-coded constants stored in the source code. These constants were determined during autopilot development.

They can be changed temporarily from the ground station, but the changes must be placed in the source code to be permanent. The parameters are initialized by nonvolatile memory. They may be changed from the ground station and the changes may be written to the nonvolatile memory using the "Write to Flash" button in the ground station. These parameters will be initialized to the new values the next time the pre-main function is executed. The global structures are initialized in the same manner as the global variables. The sensor biases and PID gains are initialized from the nonvolatile memory, while the remaining structure members are initialized to values hard-coded in the autopilot source code.

HARDWARE INITIALIZATION

The second purpose of the pre-main function is to set up and initialize the autopilot hardware. The I/O devices on the RCM3100 must be initialized, as well as the analog-to-digital converters. First, the pins on the RCM3100 associated with the payload I/O pins, GPS serial port, digital modem serial port, and the synchronous serial port used with the analog to digital converters are configured. Next, the serial ports are initialized to the correct baud rates and parity.

4.1.2 MAIN LOOP

The main loop executes the core algorithms responsible for low-level control and sensor processing. The main loop is executed more frequently than any other portion of the autopilot code and executes at a rate of approximately 130 Hz. The top of the main loop executes the functions which gather and process the sensor data. The attitude estimation algorithms are executed as part of the sensor processing functions. These functions will be discussed in detail in Sections 4.3 and 4.4. The central portion of the main loop executes the PID loop calculations. The PID loops will be discussed in detail in Section 4.4. At the bottom of the main loop, the low-level PID efforts and Pilot-in-the-Loop commands are written to the servo actuators. Figure 4.4 shows the flow of the main loop, as well as the costates.

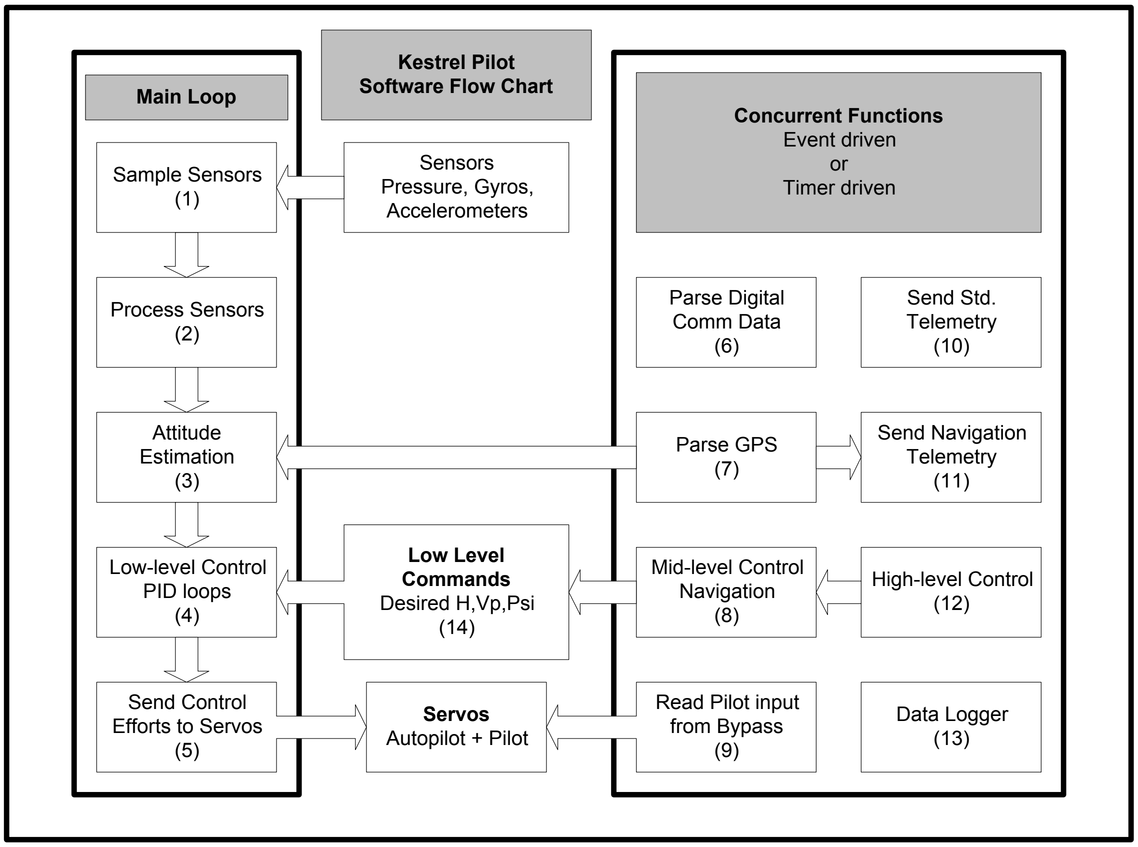

Kestrel Pilot Software Flow Chart
Main Loop
Sample Sensors (1)
Process Sensors (2)
Attitude Estimation (3)
Low-level Control PID loops (4)
Send Control Efforts to Servos (5)
Sensors Pressure, Gyros, Accelerometers
Low Level Commands Desired H,Vp,Psi (14)
Servos Autopilot + Pilot
Concurrent Functions Event driven or Timer driven
Parse Digital Comm Data (6)
Send Std. Telemetry (10)
Parse GPS (7)
Send Navigation Telemetry (11)
Mid-level Control Navigation (8)
High-level Control (12)
Read Pilot input from Bypass (9)
Data Logger (13)

Figure 4.4: Main loop flow and autopilot costates.

4.1.3 COSTATES

The costates are a set of functions that run concurrently with the main loop. The costates are written using Dynamic C costate function calls. The Dynamic C costate functions allow the costates to be executed by timer-driven and I/O-driven events. This organizational structure allows several independent tasks to function at the same time as the main loop code. There are 8 costates in the autopilot code. These are visible in (4.4). The costates perform the following functions: parse data from the digital modem, send standard telemetry packets to the ground station, parse data from the GPS, send navigation telemetry to the ground station, execute mid-level navigation functions, execute high-level failsafe functions, read Pilot-in-the-Loop data from the Bypass circuit, and run the Data Logger. These costates will be discussed in the following paragraphs.

The costate which parses the digital communication data is triggered when data is received on the digital modem serial port. The function reads the data in, checks it for errors by examining the checksum, and then calls a function appropriate to the data received.

The Standard Telemetry costate is run at the interval specified in the variable Standard Telemetry Send Time (see Section 5.2). The purpose of this costate is to send aircraft and autopilot status information to the ground station. The contents of the standard telemetry packet are listed in the protocol definition in Appendix B.

The Parse GPS costate is run when data is received on the GPS serial port. The costate checks that the data is the correct length, and then extracts the Longitude, Latitude, GPS altitude, GPS velocity, and GPS heading. A function is called which calculates the relative position in meters east and north of the GPS Home Position is called. The data is placed into the Inertial Data global structure. The method for gathering the GPS Home Position will be discussed in Section 4.4.

The Navigation Telemetry costate is executed once a second. Its purpose is to send navigation and GPS information to the ground station. The contents of the navigation status packet are listed in the protocol definition in Appendix B.

The costate which runs the mid-level control algorithms is run at the interval specified in the variable Navigation Execution Period (see Section 5.2). This value is specified in milliseconds. The purpose of this costate is to execute the Altitude Tracker script and the Waypoint Navigation script. These will be discussed in further detail in Section 4.6.

The High-Level control costate executes the fail-safe functions. This costate runs at 10 Hz. Its purpose is to execute scripts which check for communication failure and then execute a pre-defined behavior in the event of a communication failure. The Fail-safes will be discussed in detail in Section 4.7.

The Bypass circuit costate executes when data is received on the Bypass circuit serial port. The costate checks the data for errors and then extracts Pilot-in-the-Loop control information, which is sent to the Bypass circuit via the RC controller.

The Data Logger costate runs at the frequency specified in the data log setup packet. The purpose of the Data Logger costate is to collect data on the autopilot for later retrieval. The Data Logger can be configured to collect any of the global variables or structures at rates up to 130 Hz. The amount of data is limited to 150 kilobytes. The data can be retrieved using the ground station. The specifics of the Data Logger communication packets are listed in the protocol definition in Appendix B.

4.2 SESNOR PROCESSING

The purpose of the sensor processing function is to sample the 16 analog-to-digital channels and convert the data to useable values that represent states of the aircraft, autopilot status information, and payload information. There are ten sensors, system voltage, and four payload analog channels that are measured. The analog inputs are sampled from the A/D converter, compensated for temperature, bias shifted, and then processed. The processing involves the conversion of the compensated analog-to-digital number to a meaningful floating point value in the correct units. The data collection, temperature compensation, and bias removal will be discussed in Section 4.2.1. The sensor processing will be discussed in Section 4.2.2. Figure 4.5 shows the flow of the sensor processing.

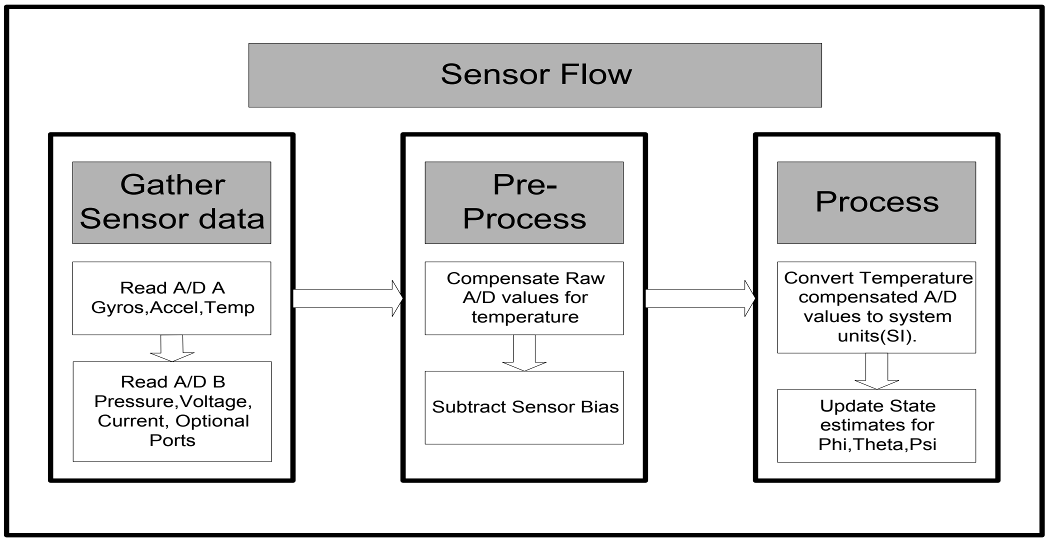

Sensor Flow
Gather Sensor data
Read A/D A Gyros,Accel,Temp
Read A/D B Pressure,Voltage, Current, Optional Ports
Pre-Process
Compensate Raw A/D values for temperature
Subtract Sensor Bias
Process
Convert Temperature compensated A/D values to system units(SI).
Update State estimates for Phi,Theta,Psi

Figure 4.5: Sensor processing flow.

4.2.1 DATA COLLECTION

The data collection takes place in three steps: data sampling, temperature compensation, and bias removal.

The analog-to-digital converters communicate with the Rabbit Processor via the SPI serial interface. Each of the sixteen analog to digital channels are read and the corresponding 12-bit values are stored in the AD_Value portion of the appropriate sensor structure.

The raw analog values are then compensated for temperature drift using

$$Compensated\_value = Sensor_{Raw\_AD} + C_{sensor} \cdot \Delta T\ . \tag{4.1}$$

Where $Sensor_{Raw\_AD}$ is the raw analog-to-digital converter output, C_{sensor} is the temperature compensation coefficient for the sensor, and ΔT is the difference in temperature from the time of calibration. The base temperatures are collected during the calibration process of each sensor. This is discussed further in Section 4.2.3. The temperature is read from the temperature sensor built into the Z rate gyro. Because only one temperature sensor is used, and every sensor has different temperature drift characteristics, the coefficient must be determined using an empirical method. Appendix C lists temperature drift coefficients for the prototype autopilot. The autopilot is placed in a freezer at 0º C. The autopilot is then removed from the freezer and allowed to warm up to room temperature. The outputs of the sensors are logged with temperature. The temperature drift coefficients are derived using a first-order approximation of the sensor output verses temperature. The drift coefficients are stored in parameters 40 through 50 (see Section 5.2). The temperature-corrected value replaces the AD_value in the sensor structures.

The next step is to remove the sensor bias. The sensor biases are gathered during sensor calibration. This will be discussed further in Section 4.3. The biases are simply subtracted from the temperature-compensated analog value.

4.2.2 SENSOR PROCESSING

The purpose of sensor processing is to convert the temperature-compensated, bias-shifted value into the correct units. This is done using the appropriate formula based on the sensor. There are eleven sensors on the autopilot that must be processed. The four payload analog ports will not be discussed here as they are not part of the autopilot core. The discussion on sensor processing will discuss the following: absolute pressure sensor, differential pressure sensor, rate gyros, accelerometers, current shunt, and system voltage.

The absolute pressure sensor used on the autopilot is the Motorola MPX 4115a. It outputs a voltage based on the absolute atmospheric pressure. This pressure is relative to an internal vacuum reference in the sensor. The absolute pressure is used to measure pressure altitude. This is the main altitude reference for the autopilot. The sensor units are meters. The altitude is relative to the altitude at which the pressure sensor bias was stored. In general this is the altitude of the ground station. Thus, the calibrated value for the absolute pressure sensor is in meters above the ground station. This is discussed in further detail in Section 4.3. The formula to convert from the raw output of the analog to digital converter was derived using empirical data. The sensor output was measured at ground level and then measured again at 300 feet. The coefficient was then backed out using a linear fit of altitude versus analog output. $H = Abs_{Comp\_AD} \cdot 0.9144$ is used to convert the temperature-compensated, bias-shifted output of the absolute pressure sensor to meters above the ground station.

The differential pressure sensor used on the autopilot is the Motorola MPX 5000. It outputs a voltage based on the difference in pressure between its two external ports. The pitot tube is connected to one of the ports and the other is left open to the ambient air. The flow of air against the pitot tube causes a pressure difference proportional to the speed of the air. The corresponding voltage produced by the sensor is used to calculate the airspeed of the UAV. The units are knots indicated airspeed (KIAS). KIAS is a unit that represents that pressure of the air and as such it is not compensated for pressure altitude or air density. $Vp = Diff\_press_{Comp\_AD} \cdot 0.044704$ is used to convert the temperature-compensated output of the sensor to KIAS. The constant is obtained using a wind tunnel.

The autopilot uses three Analog Devices ADXL300S rate gyros to measure the angular rates P, Q, and R. These values are in radians/second. A constant multiplier is used to convert the output of the rate gyros to radians/second. This is demonstrated by the following equation:

$$\begin{Bmatrix} P \\ Q \\ R \end{Bmatrix} = \begin{bmatrix} RollGyro_{Tcomp\_AD} \\ PitchGyro_{Tcomp\_AD} \\ YawGyro_{Tcomp\_AD} \end{bmatrix} \cdot 0.044380.$$

The constant is determined by placing the rate gyro on a rate table spinning at a constant velocity. The output of the analog-to-digital converter is divided by the known rate of the table to produce the constant 0.044380.

Two Analog Devices ADXL200E two-axis accelerometers are used to measure the body fixed accelerations Ax, Ay, and Az. Because only relative accelerations are needed, the accelerations remain in temperature-compensated, bias-shifted units. Empirical data suggests that approximately 270 analog-to-digital converter counts are equal to the gravity acceleration due to gravity. The relative accelerations are used in an arctangent function in the attitude estimation filter to produce reference values for roll and pitch.

The autopilot temperature is derived from the built-in temperature sensor in the Q rate gyro. This value is never converted to a temperature reading in conventional units. It is kept in biased analog-to-digital converter units for use in the temperature compensation algorithm.

The current shunt is a resistive device used to measure the current delivered to the propulsion motor. The voltage drop across the shunt is converted from bias-compensated analog-to-digital converter units using $I = Shunt_{Raw\_AD} \cdot z \cdot y$. The constant y is the conversion from voltage drop in millivolts across the current shunt to current in amps through the shunt. It is dependent on the resistance of the current shunt used. The constant z converts the analog-to-digital converter output to millivolts. This value is 0.17, which is based on the gain of the current shunt circuit.

The system voltage is sampled from the main power connector on the autopilot board. The raw analog output of the measuring digital-to-analog converter is converted to volts using $V = Vin_{Raw\_AD} \cdot y$. The constant y is the conversion from analog-to-digital units to volts and is based on the gain of the system voltage circuit.

A summary of the equations used to calculate the calibrated values for each sensor is available in Appendix D.

4.3 ATTITUDE ESTIMATION

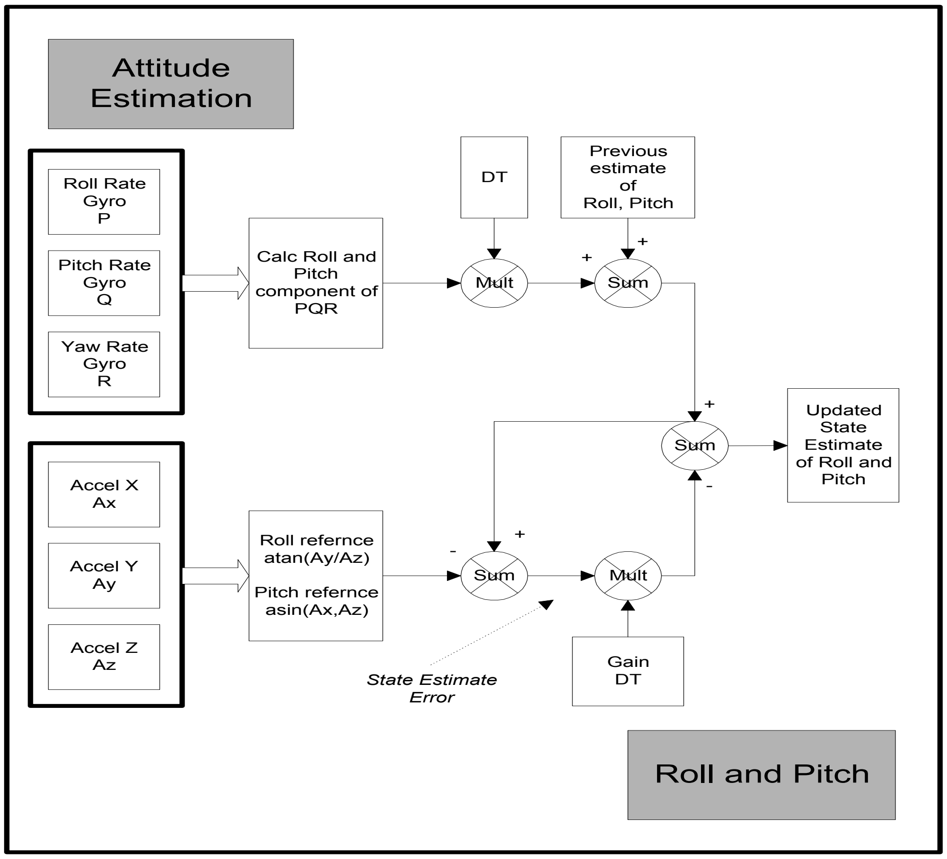
The autopilot uses two separate attitude estimation filters to determine the aircraft attitude. The first is used to estimate the roll and pitch angles. The technique employed in this filter uses integrated values from the rate gyros and estimated roll and pitch angles produced by the accelerometers. The purpose of this technique is to produce estimates of roll and pitch that have low drift and decent accuracy. The second filter is used to produce a heading estimate that has a faster response than GPS heading. The roll and pitch filter will be discussed in Section 4.3.1. The heading estimation filter will be discussed in 4.3.2.

4.3.1 ESTIMATING ROLL AND PITCH

The process for estimating roll and pitch is shown in Figure 4.6. The first step in estimating the roll and pitch angles is to derive a roll and pitch reference based on the arctangent of the aircraft's acceleration vectors and the gravity vector. This is accomplished using

$$\begin{Bmatrix} \overline{\phi} \\ \overline{\theta} \\ \overline{\psi} \end{Bmatrix} = \begin{bmatrix} \tan^{-1}(\frac{Ay}{Ax}) \\ \tan^{-1}(\frac{Ax}{-Az\cdot\cos(\phi)-Ay\cdot\sin(\phi)}) \\ GPS\_Heading \end{bmatrix}, \tag{4.2}$$

where $\overline{\phi}$ is used as a roll angle reference and $\overline{\theta}$ is used as a pitch angle reference. Both of these values are in radians. They suffer from inaccuracy in coordinated turns where the net acceleration of the aircraft is in the negative Z direction. In this case, (4.2) will produce a 0 degree roll angle. Experience shows that the resulting error can be limited during extended coordinated turns by giving minimal weight to the reference roll angle produced by (4.2) relative to that produced by integrating the rate gyros. A similar error in the reference pitch occurs during a coordinated pull-up where (4.2) will show zero pitch. However, because coordinated pull-ups are transient events in normal flight, they are not a significant factor.



Attitude Estimation
Roll Rate Gyro P
Pitch Rate Gyro Q
Yaw Rate Gyro R
Calc Roll and Pitch component of PQR
DT
Mult
Previous estimate of Roll, Pitch
Sum
Accel X Ax
Accel Y Ay
Accel Z Az
Roll refernce atan(Ay/Az)
Pitch refernce asin(Ax,Az)
Sum
Mult
Gain DT
State Estimate Error
Sum
Updated State Estimate of Roll and Pitch
Roll and Pitch

Figure 4.6: Attitude estimation flow.

The next step is to update the current state estimate using the roll and pitch components of P, Q, and R rate gyros. The pitch and roll components of the angular rates are calculated by

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & Q\cdot\sin(\phi) & \cos(\phi)\cdot\tan(\theta)\\ 0 & \cos(\phi) & -\sin(\phi)\\ 0 & \sin(\phi)\cdot\cos(\theta) & 0\end{bmatrix}\cdot\begin{bmatrix}P\\ Q\\ R\end{bmatrix}. \tag{4.3}$$

The integration is accomplished using the Euler approximation

$$\begin{bmatrix}\hat{\phi}\\ \hat{\theta}\\ \hat{\psi}\end{bmatrix}_{k+1} = \begin{bmatrix}\hat{\phi}\\ \hat{\theta}\\ \hat{\psi}\end{bmatrix}_{k} + \Delta t\cdot\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}_{k}. \tag{4.4}$$

Where $\hat{\phi}$, $\hat{\theta}$, and $\hat{\psi}$ are the state estimates of roll, pitch, and yaw. The updated state values of pitch and roll produced by (4.4) are then subtracted from the reference values for pitch and roll computed in (4.2). The corresponding value is the state estimation error. This error value is multiplied by a gain and subtracted from the state estimate to form the new state estimate. This process is shown in the following equations:

$$\begin{bmatrix}\tilde{\phi}\\ \tilde{\theta}\\ \tilde{\psi}\end{bmatrix} = \begin{bmatrix}\bar{\phi}\\ \bar{\theta}\\ \bar{\psi}\end{bmatrix} - \begin{bmatrix}\hat{\phi}\\ \hat{\theta}\\ \hat{\psi}\end{bmatrix}, \tag{4.5}$$

$$\begin{Bmatrix}\phi\\ \theta\\ \psi\end{Bmatrix} = \begin{bmatrix}\hat{\phi}\\ \hat{\theta}\\ \hat{\psi}\end{bmatrix} - \Delta t\cdot\begin{bmatrix}\tilde{\phi} & 0 & 0\\ 0 & \tilde{\theta} & 0\\ 0 & 0 & \tilde{\psi}\end{bmatrix}\cdot\begin{bmatrix}k_{\phi}\\ k_{\theta}\\ k_{\psi}\end{bmatrix}. \tag{4.6}$$

Where $\tilde{\phi}$, $\tilde{\theta}$, and $\tilde{\psi}$, are the state errors. This process minimizes the integration error caused by rate gyro drift. Gains $\begin{bmatrix} k_\phi \\ k_\theta \\ k_\psi \end{bmatrix}$ are determined empirically in flight: the gains are tuned until the roll and pitch estimates correspond visually with the airplane's behavior during level flight and in coordinated turns. The accuracy of the state estimates in flight has not been determined, as it is not now possible to obtain state sensors small enough and light enough to fly on our aircraft.

4.3.2 ESTIMATING HEADING

The same process that is used to estimate roll and pitch (4.2 – 4.6), is used to estimate heading. The only difference is in obtaining the reference value. The reference heading is obtained from the GPS heading. However, because GPS heading is inaccurate when the GPS velocity is low, or when there is poor GPS lock, the heading estimate is only updated when the GPS velocity is above a certain threshold and the GPS has a valid lock. The velocity threshold is determined to be 2 m/s, using empirical data gathered from the Furuno GH-80 GPS receiver. At values above 2 m/s, the GPS heading produced by this receiver is accurate. Figure 4.7 shows the process used to estimate heading.

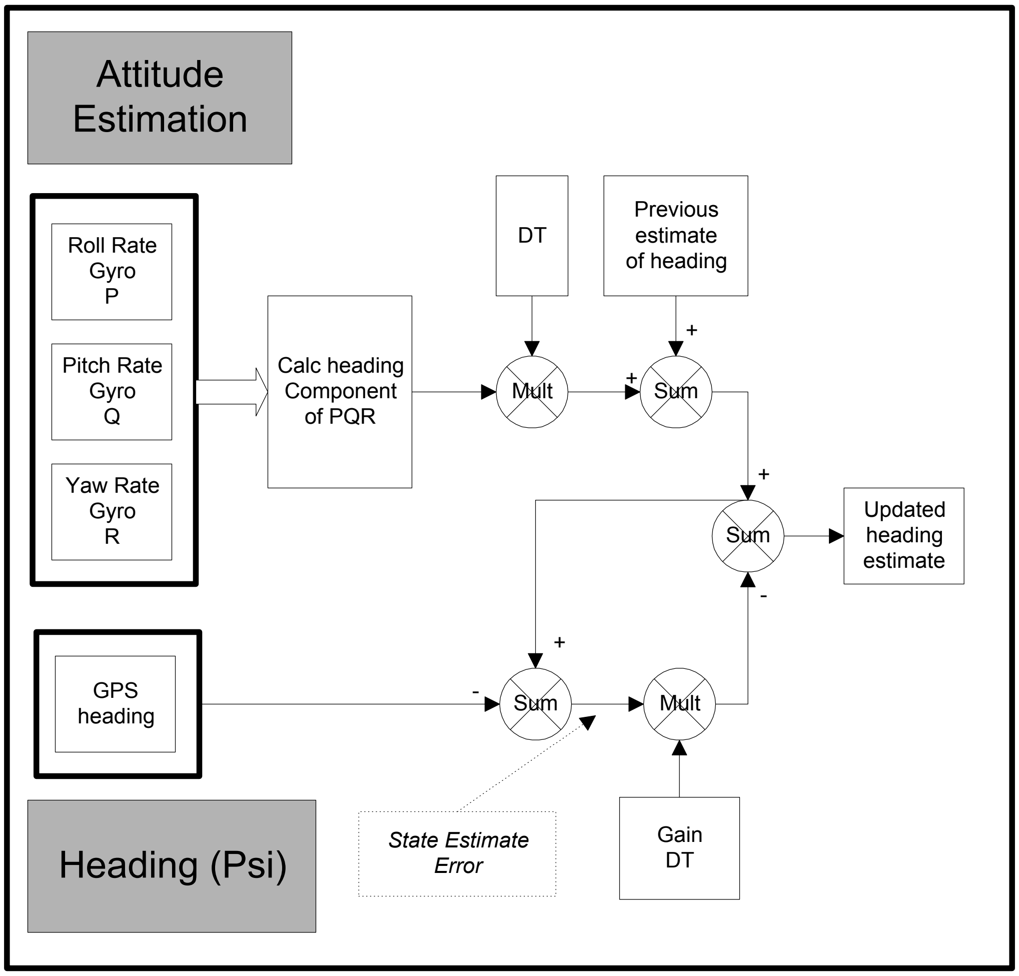


Figure 4.7: Heading estimation flow.

4.4 SENSOR CALIBRATION

To produce accurate values, most of the autopilot sensors and the GPS receiver must be calibrated. The calibration routines determine the bias or steady-state value of the sensor in an unexcited state; the calibration routine of the GPS receiver produces the GPS Home Position. The sensors should be calibrated after the autopilot board temperature has stabilized. This usually occurs within 5 minutes of power on. The discussion on sensor calibration will proceed in the following order: absolute and differential pressure sensors, rate gyros, accelerometers, and GPS.

4.4.1 AUTOPILOT SENSORS

The absolute pressure sensor produces a measurement of altitude in meters relative to the calibration point. The pressure sensor should be calibrated at the location where the aircraft will be flown. Because atmospheric conditions vary with time, the calibration should take place before each flight. The calibration routine for the absolute pressure sensor gathers 50 samples of the absolute pressure channel on the analog-to-digital converter. The 50 samples are averaged and stored in the sensor structure as the bias value. This value is subtracted from all future readings of the sensor to remove the sensor bias.

The calibration routine for the differential sensor uses the same 50-sample method as the routine for the absolute pressure sensor. The routine which acquires the bias for the differential and absolute pressure sensors is called Calibrate Pressure Sensors and can be called using a button on the ground station. This should take place before each flight.

The rate gyros are drift prone. Because the output of the rate gyros is integrated to produce the roll, pitch, and heading estimates, small offsets in the calibrated rate gyro outputs can cause significant error in the attitude estimate. To minimize this error, it is important to determine an accurate value for the rate gyro biases. The rate gyro biases are gathered using the same technique explained above for the absolute pressure sensor. It is critical that, while the rate gyro biases are being gathered, the airplane is not moved. Moving the airplane will excite the P, Q, and R rate gyros, the excited values will be averaged into the bias values, which will cause significant error in the calibrated values.

Calibrating the accelerometers is a two-step process. The first step is to acquire the biases for the X and Y accelerometers. The bias value is the value output from the sensor when the sensor is not excited. The X and Y accelerometers are not excited when their axes are perpendicular to the gravity vector and when the autopilot is not experiencing acceleration of any kind. This occurs when the autopilot is lying flat (with the bottom down) and still. The next step is to acquire the bias for the Z accelerometer. The Z accelerometer is not excited when its sensing axis is perpendicular to gravity. This occurs then the autopilot is tipped up on its edge (such that the Z axis is perpendicular to Earth's gravity). The same 50-sample technique used to acquire the absolute pressure sensor bias is used to acquire the accelerometer bias. The accelerometer biases should be acquired once the autopilot is installed in the aircraft.

4.4.2 CALIBRATING THE GPS

The GPS calibration is used to obtain the GPS Home Position. The GPS Home Position is comprised of the latitude, longitude, and altitude of the home position. This is the position that the relative positions of the aircraft (in meters east and meters north) are calculated from. The following formula is used for this calculation:

$$X = (longitude - h\_longitude) \cdot \cos(\frac{h\_latitude}{60\,\mathrm{min/deg}} \cdot \frac{\pi}{180\,\mathrm{deg}}) \cdot 1853.2\,\mathrm{min}/meter$$

$$Y = (Latitude - h\_latitude) \cdot 1853.2\,\mathrm{min}/meter.$$

The longitude and latitude used in the equation are in minutes. The X and Y position are in meters east and north of GPS Home Position, respectively. The GPS altitude is also referenced from the altitude stored in the GPS Home Position.

The home position is also the set of coordinates to which the aircraft will fly, should it lose communication with the ground station. This is typically the coordinates of the ground station. The GPS Home Position is acquired automatically after GPS lock is first acquired. The first five samples after lock are averaged to form the home longitude, home latitude, and home altitude. Because the GPS position tends to drift for some time after GPS lock is acquired, it is advisable to re-acquire the GPS home position once the GPS position has stabilized. This can be accomplished using the button labeled "Gather GPS Home Position" in the ground station.

4.5 LOW-LEVEL CONTROL ALGORITHMS

Low-level control of the aircraft is accomplished using the PID loop structure outlined in Chapter 2. This is a collection of servo PID loops that produce aileron, elevator, throttle, and rudder outputs and a collection of outer PID loops that produce commanded values for the servo PID loops. Figure 4.8 shows the basic PID loop structure as implemented in the autopilot code. The purpose of the low-level control algorithms is to control the aircraft in the roll and pitch axis, hold a commanded velocity, hold a commanded heading, and hold a commanded altitude. The PID loops are implemented in the autopilot software using Dynamic C structures. The structures contain the proportional, integral, and derivative gains, along with saturation limits on the outputs of the PID loops. The PID structures also contain pointers to access autopilot sensors and actuators. Figure 4.9 shows how the aileron PID loops are implemented using this structure. In this way the PID loops can be reconfigured to use any sensor on the autopilot as the reference value, and write the effort to any actuator or variable in the autopilot code. The gains and effort limits can be updated during flight using the ground station software. The gains and effort limits are backed up in the nonvolatile memory of the Rabbit Processor.

A PID structure is instantiated for each PID loop in the pre-main portion of the autopilot code. The structures are initialized with the gains and effort limits and stored in the nonvolatile memory as well as with the hard coded values representing the correct sensors and actuators for each loop. The PID loop efforts are calculated in the main loop, which executes at approximately 130 Hz. Standard PID calculation techniques are used to implement the effort calculations in the source code.

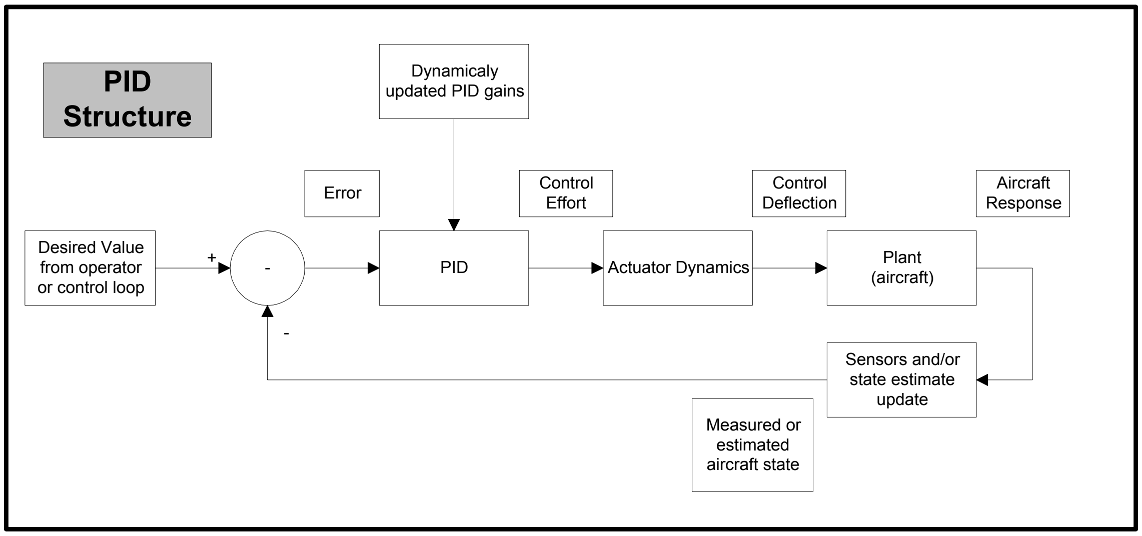

PID Structure
Dynamicaly updated PID gains
Error
Control Effort
Control Deflection
Aircraft Response
Desired Value from operator or control loop
+
-
PID
Actuator Dynamics
Plant (aircraft)
-
Sensors and/or state estimate update
Measured or estimated aircraft state

Figure 4.8: Basic PID loop structure as implemented in the autopilot source code.

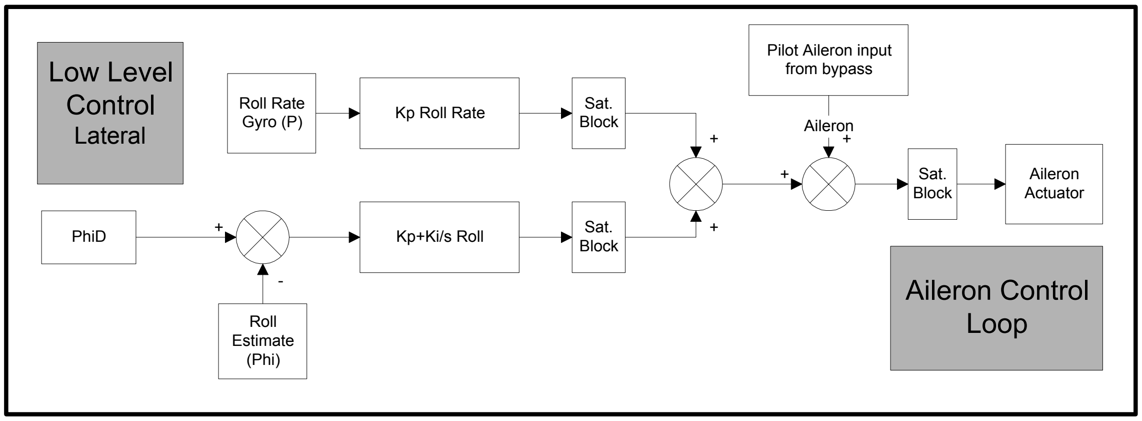


Figure 4.9: Aileron control loop as implemented in the autopilot source code.

4.6 MID-LEVEL CONTROL ALGORITHMS

There are two mid-level algorithms on the autopilot. The first is the Altitude Tracker. The purpose of the altitude tracker is to enable the correct PID loops to maintain the commanded altitude in an efficient manner. The Altitude Tracker will be discussed in Section 4.6.1. The second mid-level control algorithm is the Waypoint Navigation script. The Waypoint Navigation script executes the set of navigation commands uploaded to the autopilot by the user. The waypoint navigation commands were developed and optimized for the Zagi flying wing aircraft. The Waypoint Navigation script will be discussed in Section 4.6.2.

4.6.1 ALTITUDE TRACKER

The purpose of the Altitude Tracker is to maintain the aircraft's commanded altitude in an efficient and safe manner. At the heart of the Altitude Tracker are the Throttle from Airspeed and Pitch from Altitude PID loops. When the aircraft is near its commanded altitude, the aircraft's altitude can be controlled easily with small changes in pitch. The altitude error is small in this region; therefore, the commanded pitch angles will also be small and within the maximum angle of attack of the airframe. However, if altitude error is large, the Pitch from Altitude loop will saturate and possibly produce a pitch angle that exceeds the maximum angle of attack of the airframe. A stall will result. The Throttle from Airspeed loop may also have difficulty maintaining airspeed at these high pitch angles, depending on the thrust available from the motor. In the case where the commanded altitude is significantly below the actual altitude, the Pitch from Altitude loop will command a large negative pitch which may cause the aircraft to exceed its maximum structural airspeed of the airframe.

The solution to these problems is to reconfigure the PID loops based on the altitude error. This is a technique used in general aviation aircraft. When the altitude error is large, the aircraft pitch is trimmed for a safe and efficient climb or descent airspeed and the throttle is used to control the rate of climb or descend. In the case of a large positive altitude error, the autopilot is configured to use the Pitch from Airspeed PID loop, which regulates the airspeed using pitch. The throttle is set to full for maximum climb performance. As the airspeed is regulated, the aircraft will never enter a stall situation. Once the altitude error is reduced to a set threshold, the Pitch from Altitude and Throttle from Airspeed loops are re-enabled and the Pitch from Airspeed loop is disabled. In the case of a large negative altitude error, the same technique is used to lower the aircraft at a constant airspeed to the desired altitude. The throttle is held at idle and the aircraft is allowed to descend at a safe airspeed. The window in which the altitude error is considered small is set using the parameter Altitude Error Window (See Section 5.2). This value is in meters and is set to 50 meters by default.

4.6.2 WAYPOINT NAVIGATION SCRIPT

The Waypoint Navigation script executes a set of navigation commands uploaded to the autopilot by the user. Its purpose is to navigate the airplane based on the specific command being executed. There are 8 different waypoint commands that can be executed. The Waypoint Navigation script uses the low-level control algorithms to navigate the airplane. It does this either by commanding heading, altitude, and velocity or by commanding roll angle, altitude, and velocity. The method used is determined automatically by the waypoint command being executed. Waypoints are executed sequentially in the order they are uploaded by the user. When the autopilot finishes the last command, it will re-execute the first command.

WAYPOINT TYPES

There are six types of waypoint commands.

1. *Goto XY*: The *Goto XY* waypoint command is used to navigate to a position specified in meters east and meters north of the GPS Home Position. When executing this command, the autopilot commands a heading based on the bearing to the waypoint. The command terminates when the magnitude of the distance to the waypoint is less than the value specified in the parameter Waypoint Radius. The *Goto XY* command requires the following parameters: meters east, meters north, altitude (meters), airspeed (meters/second).

2. *Goto Legal*: The *Goto Legal* waypoint command is used to navigate to a position specified in degrees longitude and latitude. When executing this command, the autopilot commands heading based on the bearing to the waypoint. The command

terminates when the magnitude of the distance to the waypoint is less than the value specified in parameter Waypoint Radius. The *Goto Legal* command requires the following parameters: degrees longitude, degrees latitude, altitude (meters), airspeed (meters/second).

3. *Loiter XY*: The *Loiter XY* waypoint command executes an orbit around a position specified in meters east and meters north of the GPS Home Position for a specified period of time. When executing this command, the autopilot commands a roll angle based on the aircraft's distance from the desired loiter path and the tangential heading error. The command is finished when the amount of time specified in the orbit time field has passed. If a value of 0 is specified for the orbit time, the aircraft will orbit indefinitely. The *Loiter XY* command requires the following parameters: meters east, meters north, altitude (meters), airspeed (meters/second), orbit time (seconds), and orbit radius (meters).

4. *Land XY*: The *Land XY* waypoint command executes a landing sequence around a position specified in meters east and meters north of the GPS Home Position. The landing sequence orbits the aircraft around the landing position while holding airspeed. The autopilot then cuts throttle and allows the aircraft to descend in the orbit while regulating the airspeed using the Pitch from Airspeed PID loop. When the aircraft descends to the altitude specified by the Flair Height, the autopilot attempts to level the aircraft by commanding a zero roll angle to the Aileron from Roll PID loop. If the flair height is set to zero, the autopilot will not attempt to flair the aircraft. The *Land XY* command requires the following parameters: meters east, meters north, altitude (meters), airspeed (meters/second), orbit radius (meters), and flair height (meters).

5. *Take Off*: The *Take Off* waypoint command executes a take-off sequence at the current GPS position. The take-off sequence commands the aircraft to hold a constant pitch (specified in the parameter Take Off Pitch) while using the Aileron from Roll PID loop to hold a 0 degree roll angle. The autopilot commands full throttle. Once the aircraft reaches the airspeed specified in the rotate airspeed input field, the autopilot enables the Altitude Tracker and commands an orbit at the current GPS location with the orbit radius specified in the orbit radius input field. If the aircraft does not achieve the rotate velocity within the period of time specified in the parameter Take Off Rotate Time Out, the take-off sequence is aborted and the autopilot commands 0 throttle in an attempt to land the aircraft. If the aircraft reaches the altitude specified in the altitude input field, the *Take Off* Command is finished, and the Waypoint Navigation script executes the next waypoint command. This command takes the following parameters: command completion altitude (meters), rotate airspeed (meters/second), and climb out orbit radius (meters).

6. *Repeat*: The *Repeat* waypoint command causes the Waypoint Navigation script to execute the waypoint command specified in the waypoint number input field. This command is used to repeatedly fly sequences of waypoints. As an example, if 2 is used as the repeat parameter, the autopilot will begin executing waypoint 2 once the repeat command is executed. The *Repeat* command takes the following parameter: waypoint number.

WAYPOINT COMMAND EXECUTION

The Waypoint Navigation script uses a state machine to execute the waypoint commands. The state machine has eight states. The state of the state machine is known as the Waypoint Navigation Mode, or Nav Mode. Figure 4.10 shows the flow of the state machine.

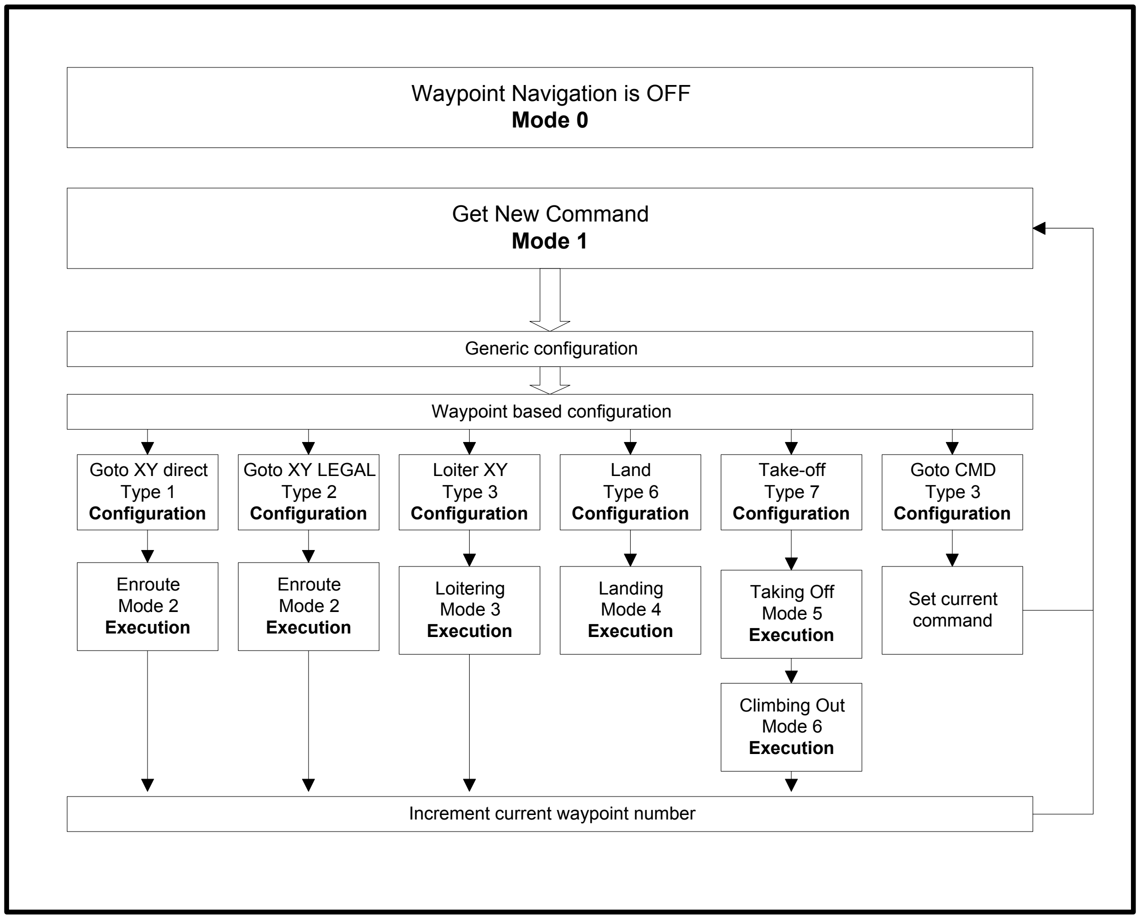

Waypoint Navigation is OFF
Mode 0
Get New Command
Mode 1
Generic configuration
Waypoint based configuration
Goto XY direct Type 1 Configuration
Goto XY LEGAL Type 2 Configuration
Loiter XY Type 3 Configuration
Land Type 6 Configuration
Take-off Type 7 Configuration
Goto CMD Type 3 Configuration
Enroute Mode 2 Execution
Enroute Mode 2 Execution
Loitering Mode 3 Execution
Landing Mode 4 Execution
Taking Off Mode 5 Execution
Set current command
Climbing Out Mode 6 Execution
Increment current waypoint number

Figure 4.10: Waypoint navigation script flow.

The states of the Waypoint Navigation script are as follows:

1. State 0: Waypoint Navigation script is OFF. In this state, no navigation commands are executed.
2. State 1: Get next command and configure. In this state, the autopilot copies the next command out of the waypoint array and configures the autopilot PID loops appropriately for the command. The PID loops are configured by modifying the Feedback Loop Configuration variable or FLC. The next state is determined by the command type. The following list shows the configuration and next state for each type of waypoint:
    a. *Goto XY*
        i. FLC: The following values will be added to the FLC using the bitwise OR operator. PID loops not mentioned will be left in their current state.
            1. Aileron from Roll
            2. Roll from Heading
        ii. Other:
            1. Desired heading rate = 0
            2. Nav Mode = 2
    b. *Goto Legal*
        i. FLC: The following values will be added to the FLC using the bitwise OR operator. PID loops not mentioned will be left in their current state.
            1. Aileron from Roll

            2. Roll from Heading

      ii. Other:

            1. Desired heading rate = 0

            2. Nav Mode = 2

c. *Loiter XY*

      i. FLC: The following values will be added to the FLC using the bitwise OR operator. PID loops not mentioned will be left in their current state.

            1. If (!Orbit Clockwise&0x02) Aileron from Roll. The second bit of Orbit Clockwise determines what PID loop is used to control the orbit.

            2. If (Orbit Clockwise&0x02) NOT Aileron from Roll

            3. Roll Fixed

      ii. Other:

      iii. Desired heading rate = 0

      iv. Nav Mode = 3

d. *Land XY*

      i. FLC: The following values will be added to the FLC using the bitwise OR operator. PID loops not mentioned will be left in their current state.

            1. Pitch Fixed

            2. Throttle Fixed

            3. Pitch from Airspeed

        4. Roll From Heading
        5. Aileron from Roll
    ii. Other:
        1. Desired heading rate = 0
        2. Fixed Throttle = parameter Landing Throttle
        3. Nav Mode = 4

e. *Take Off*
    i. FLC: The following values will be added to the FLC using the bitwise OR operator. This means, that PID loops not mentioned will be left in their current state
        1. Pitch Fixed
        2. Throttle Fixed
        3. Roll Fixed
    ii. Other:
        1. Desired heading rate =0
        2. Fixed Throttle = UAV Parameter: Take Off Throttle
        3. Waypoint Navigation Mode = 5
        4. Desired Pitch = UAV Parameter: Take Off Pitch
        5. Desired Roll = 0

f. *Repeat*
    i. Current Command = Current Command +1;
    ii. Nav Mode =1

3. State 2: En-Route. This state is used for the *Goto XY* and *Goto Legal* commands. In this state, the autopilot computes a heading from the bearing to the waypoint. The autopilot also sets the desired altitude and velocity to the values specified in the waypoint command. The state machine will stay in this state until the distance to the waypoint is less than the value specified in the parameter Waypoint Radius. At that point the current waypoint variable will be incremented and the state will be set to state 1 (get next command and configure).
4. State 3: Loitering. In this state, the autopilot is orbiting around an inertial referenced point at a fixed radius. The autopilot attempts to maintain a constant radius by adjusting the roll angle of the aircraft. The orbit algorithm computes the roll angle based on the loiter radius error and the heading angle error. The heading angle error is the difference between the tangent to the circular orbit path and the airplane's actual heading. The loiter radius error is the difference between the desired loiter radius and the actual distance of the aircraft from the loiter point, or the center of the orbit circle. Two gains are used in the roll angle calculations. The first is the orbit angle gain (parameter `Orbit Angle Gain`). This gain is the weight placed on the tangential heading error in the roll angle calculation. The second gain is the orbit position gain (parameter `Orbit Pos Gain`). This gain is the weight placed on the loiter path error in the roll angle calculation. Also used in the roll angle calculations are two UAV parameters. The first is the trim angle (parameter `Phi Trim`). This value is summed into the roll angles produced by the loiter radius error and tangential heading error. This value should be set to the roll angle necessary to produce a circular flight path

with the desired radius under trim conditions. The loiter algorithm is fairly insensitive to this value, and as such this value can be set to a nominal value in the range of the desired radius. The second parameter is the maximum contribution of loiter radius error (parameter `Max Xtrack Error`). This value limits the maximum contribution of the loiter radius error to the commanded roll angle. This value is in meters and should be set to a value that is approximately 125 percent of the average commanded loiter radius. The waypoint navigation state machine will stay in this state until the orbit timer expires. At that point, the current waypoint variable will be incremented and the state will be set to state 1 (get next command and configure). If the orbit timer is set to 0 the loiter will be indefinite and the state machine will stay in state 3 indefinitely or until manually changed by the user.

5. State 4: Landing. This state is used for the landing command. In this state, the autopilot descends at a constant airspeed while orbiting around an inertial position. The orbit is kept at a constant radius using the loiter algorithm described in state 3. The object of this state is to land the aircraft safely on the ground. The method employed is to glide the aircraft slowly to the ground while circling around a point on the ground. The glide speed is set at 120 percent of the aircraft's stall speed in order to limit the descent rate while avoiding a stall. The glide is achieved by cutting the power to the propulsion system and regulating the airspeed using the Pitch from Altitude PID loop. The danger of this method is that the aircraft may reach the ground in a non-level flight attitude because of the roll angles commanded by the orbit algorithm. An area of future work is to

employ a height-above-ground sensor to sense the distance from the ground and zero the roll angle when a minimum distance is reached. The absolute pressure sensor on the aircraft cannot be used because of varying terrain heights and insufficient sensor resolution. The current algorithm works well with a flying wing UAV in obstacle-free, flat areas 100 meters square or larger. The waypoint navigation state machine will stay in this state until manually commanded by the user. The Waypoint Navigation script will not execute the next command when this command is finished, it is assumed that this will be the last command of the flight.

6. State 5: Take-off. This state is used to launch the aircraft under autopilot control. It is the first state in the auto launch sequence. In this state, the autopilot holds the aircraft level in the roll axis and at a specified pitch angle. The pitch angle is set by the user in parameter Take Off Pitch (the optimum value for the tested flying wing aircraft is 20 degrees). Full power is applied to the propulsion system. At this point, the aircraft is launched by the user. The autopilot holds the constant pitch angle with the wings level in an attempt to build airspeed with minimum altitude loss. Once the aircraft's airspeed reaches the value specified in the rotate velocity of the *Take Off* command input, the state machine switches to the climb-out state (state 6). This sequence works well with our flying wing type aircraft.
7. State 6: Climb-Out. This state is the second state in the auto launch sequence. In this state, the aircraft circles around the inertial coordinates specified in the *Take Off* command at a constant radius. The constant-radius circle is achieved by

commanding roll angles produced by the orbit algorithm. The airspeed is maintained using the Pitch from Airspeed PID loop. The state machine stays in this state until the take off altitude specified in the *Take Off* command is reached. At that point the current waypoint variable will be incremented and the state will be set to state 1 (get next command and configure).

4.7 HIGH-LEVEL CONTROL

There are two types of high level control algorithms used by the autopilot. The first is the Lost Communication Go Home fail-safe, or Go Home fail-safe. The purpose of this fail-safe is to utilize the navigation capabilities of the autopilot to fly the aircraft back to the GPS Home Position in the event of a communication failure between the aircraft and ground station. The second high level control algorithm is the Pilot-in-Command Lost Communication fail-safe, or PIC fail-safe. The purpose of this failsafe is to utilize the low-level control and navigation capabilities of the aircraft to safely fly the aircraft in the event of a lost communication scenario when the autopilot is in Pilot-in-Command mode (PIC). The Go Home Fail-Safe will be discussed in Section 4.7.1 and the PIC fail-safe will be discussed in Section 4.7.2.

4.7.1 GO HOME FAIL-SAFE

It is possible that the communication link between the aircraft and the ground station may be interrupted or lost completely in flight. Several events could cause this. First, the aircraft may simply fly out of range on its way to a waypoint or while being piloted via the video link. Second, the line-of-sight communication path may be temporarily obstructed by obstacles or terrain. Both of these scenarios could potentially result in the loss of the aircraft, as without the communication link there is no way to know the position of the aircraft and no ability to command the aircraft home. The Go Home fail-safe is designed for the lost communications scenario.

The Go Home fail-safe operates under the following premises. First, the autopilot will send a navigation status packet to the ground station every time the autopilot receives a packet from the GPS receiver (about once a second). Second, the ground station will send a packet back to the autopilot acknowledging the navigation status packet. Third, the autopilot will have GPS lock while navigating. Under these premises, there will be two-way communication between the autopilot and ground station at least once per second. The Go Home fail-safe algorithm keeps track of the number of navigation status packets that have been sent since the last navigation status acknowledgment was received. If that number exceeds a threshold specified in parameter Lost Communication Go Home Timeout, then it is assumed that the communication link is down and the aircraft is commanded to fly to the GPS Home Position. Once the GPS Home Position is reached, the autopilot is configured to orbit at a constant radius around the GPS Home Position. Flying the aircraft to the home position is intended to reduce the line-of-sight distance to the aircraft and allow for communication with the aircraft to be re-established. At the very least, the user knows where the aircraft is and can retrieve it once the batteries run out.

4.7.2 PIC FAIL-SAFE

Section 4.7.1 discusses scenarios that could lead to a loss of communication between the aircraft and ground station. We also discussed the Go Home fail-safe, which will fly the aircraft to the GPS home position if communications are lost while the autopilot is enabled. This section will discuss what happens if communications are lost when the autopilot is not enabled.

One method of Pilot-in-the-Loop control of the aircraft is to send the raw elevator, aileron, throttle, and rudder commands generated by the pilot through the communication link. The advantage of this over other methods which use the onboard Bypass circuit and RC receiver is that the Bypass circuit and RC receiver are not required on the aircraft, saving weight and space which can be used for additional payload. The disadvantage is that if the digital communication link between the aircraft and ground station is interrupted for any reason, the pilot will lose control of the aircraft. To address this situation, the PIC fail-safe mode was developed. This fail-safe is active only when Pilot-in-the-Loop commands are being sent through the communications link (PIC mode). If communications are interrupted, the autopilot will enable itself (switch to CIC mode) and fly itself home if communications are not re-established after a short period of time. There are two modes in this fail-safe. The first is the level mode. This mode simply engages the autopilot (CIC mode) and holds a nose-up, level flight attitude. The second mode is the go-home mode. If communications are not re-established during level mode during a specific period of time, the aircraft flies to the GPS Home Position.

Level mode is initiated when no pilot commands are received in the period specified by parameter RC over Comm Level Timeout. This mode is intended to level the aircraft during periods of brief communication interruption. The level mode will exit automatically when communications are re-established. In the level mode, the following occurs:

1. Commanded roll angle is set to zero.
2. Commanded pitch angle is set to 15 degrees.
3. Pitch and roll PID loops are enabled.
4. Autopilot is switched to CIC mode.
5. Throttle is set to zero.

If the period of interrupted communication extends past the value specified in the parameter RC Over Comm Go Home Timeout, then the autopilot will enter the second stage of the PIC fail-safe. At that point, the autopilot will enable the appropriate PID loops and navigate to the GPS Home Position using the Waypoint Navigation script. To exit PIC fail-safe go-home mode, the user must switch the autopilot to CIC mode, and then back to PIC mode.

4.8 AUTOPILOT COMMUNICATION

The communication protocol is designed with two goals in mind: to facilitate autopilot development and to allow the user to receive status updates from and send commands to the aircraft. These goals are achieved through the use of a comprehensive set of communication packets. Every command sent to or from the autopilot is embedded into a packet that begins with 0xFF and ends with a checksum byte followed by 0xFE. The first byte after 0xFF is the packet type. As of this writing, there are 50 different packet types. The comprehensive listing of packets is contained in Appendix B.

There are three types of autopilot communication. The first is unsolicited communication. The purpose of unsolicited communication is for the autopilot to provide status updates to the ground station at regular intervals. Sending of the navigation status and standard telemetry data are examples of unsolicited communications. Navigation status is sent once a second when the GPS has a valid fix. Standard telemetry is streamed at 10 Hz. Appendix B contains a detailed description of the contents of these packets. The second type of communication consists of autopilot commands. These are packets sent from the ground station to the autopilot for the purpose of requesting information or executing commands. Examples of this include PID gain packets, waypoint upload packets, and calibration command packets. The third type of autopilot communication is acknowledgment. The autopilot sends a packet acknowledging the receipt of most commands. In some cases the acknowledgment will be in the form of the requested data.

4.9 AUTOPILOT PERFORMANCE

The purpose of this section is to present data that shows the actual performance of the autopilot when used on the Zagi airframe. Steps in heading, altitude, airspeed, and position are shown in Figures 4.11 through 4.14. The aircrafts actual state is shown next to the aircrafts commanded state. The plots show that the autopilot is able to command the aircraft to the desired values for heading, altitude, airspeed, roll, and pitch.

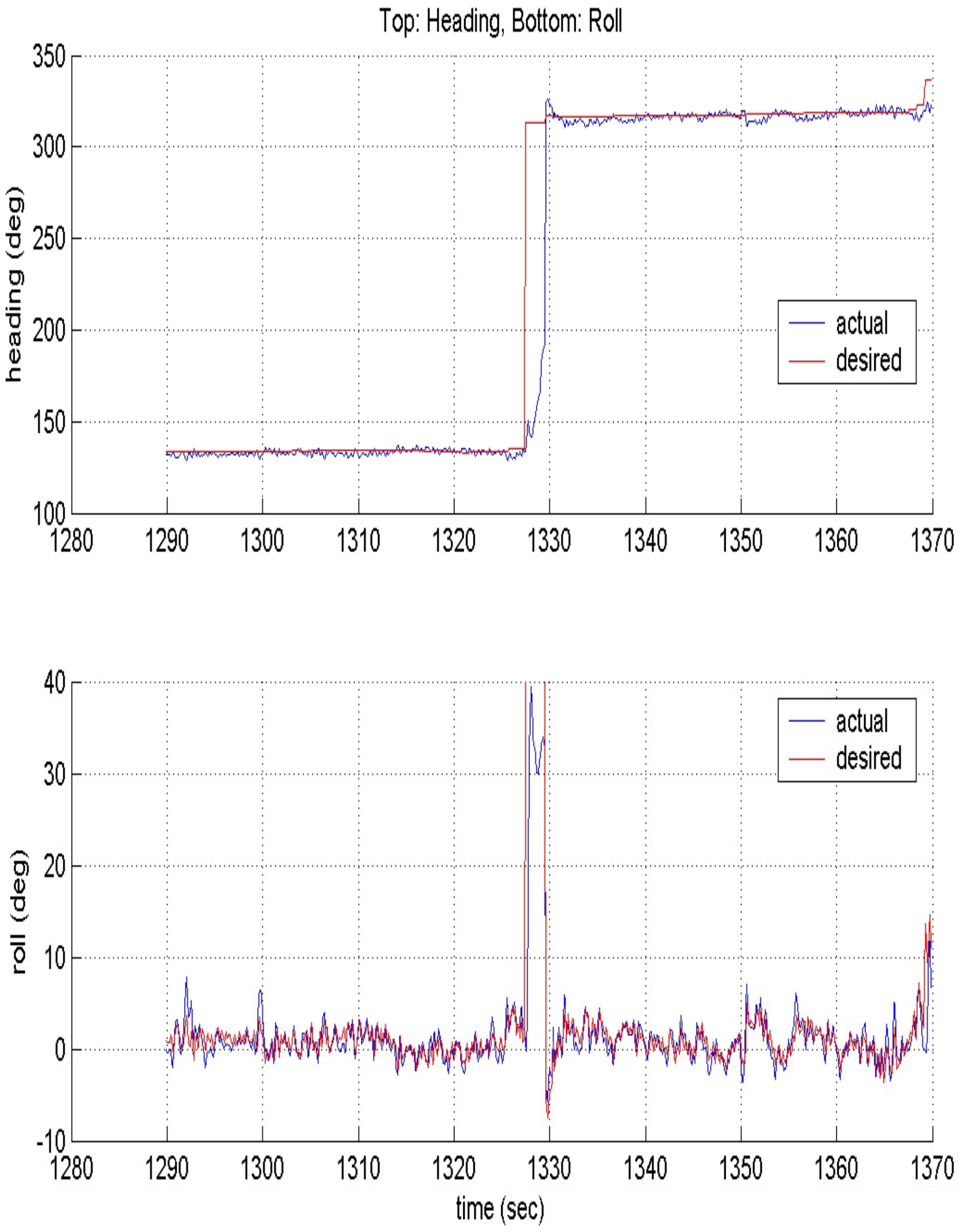


Figure 4.11: Closed loop response to 180 degree step in heading.

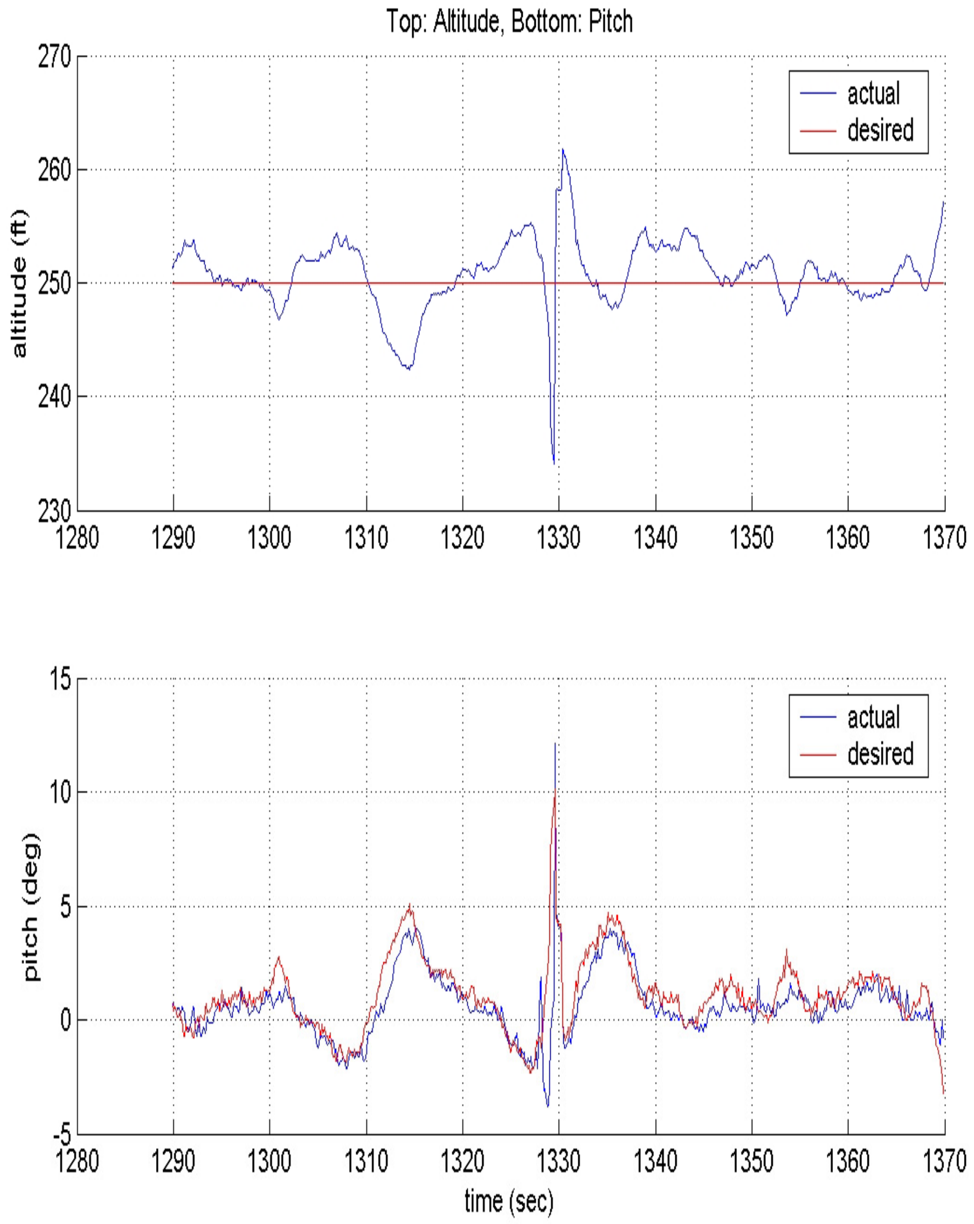


Figure 4.12: Closed loop altitude hold performance to constant commanded altitude.

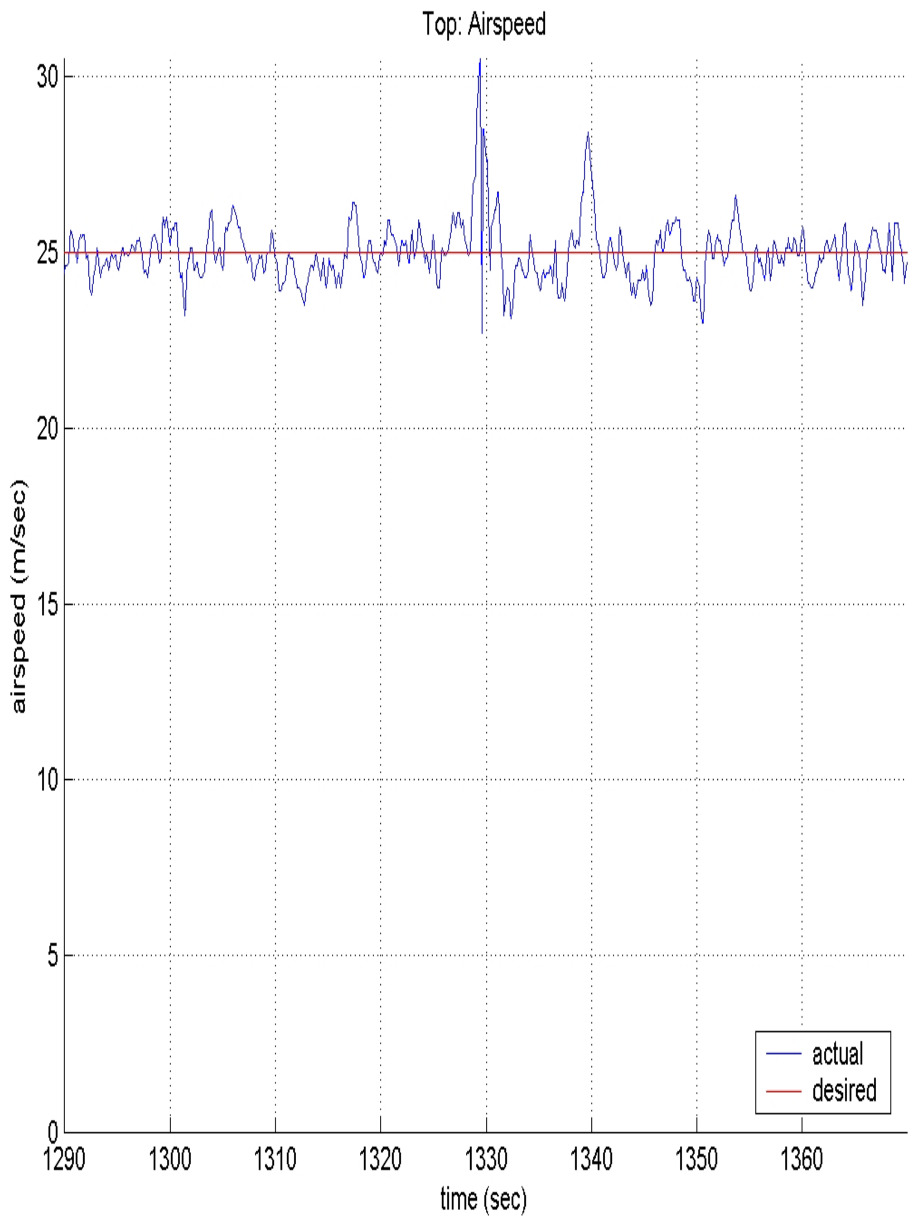


Figure 4.13: Closed loop airspeed hold performance..

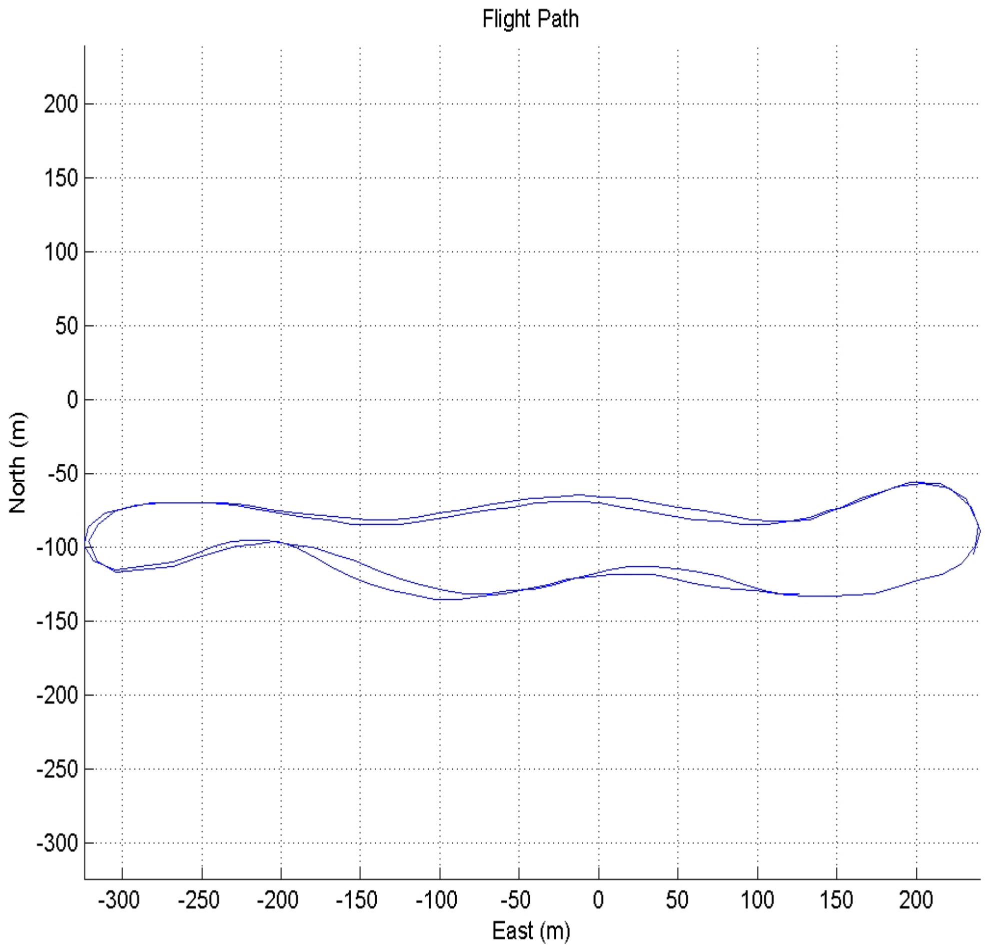


Figure 4.14: Closed loop waypoint tracking performance.

CONCLUSION

The purpose of this chapter was to discuss the software used in the autopilot system. The chapter began with a description of the autopilot software structure. Next the algorithms used in sensor processing were detailed. The attitude estimation filter was covered next. A detail of the low-level, mid-level, and high-level control algorithms was given. Next, the communications protocol was detailed. The chapter finishes with plots showing the closed loop performance of the autopilot.

CHAPTER 5

AUTOPILOT USERS MANUAL

The Kestrel autopilot system consists of the autopilot, GPS, digital communication link, pitot system, payload, ground station hardware, and ground station software. For the system to function correctly, it is important that it is installed correctly, configured for the airframe, and programmed correctly for the desired mission. The purpose of this chapter is to guide the user through this setup process. This chapter is a user manual for the autopilot system. The user manual will discuss the following:

1. Hardware system description
2. Virtual Cockpit software user manual
3. Autopilot installation
4. Autopilot setup
5. First flight
6. Second flight
7. Pre-flight
8. In-flight monitoring
9. Landing

5.1 AUTOPILOT SYSTEM

The autopilot system can be divided into airborne components and ground station components. The airborne components consist of the autopilot, GPS, and digital modem. The ground station components are the ground station hardware and ground station software. Figure 5.1 serves to illustrate the individual components of the autopilot hardware.

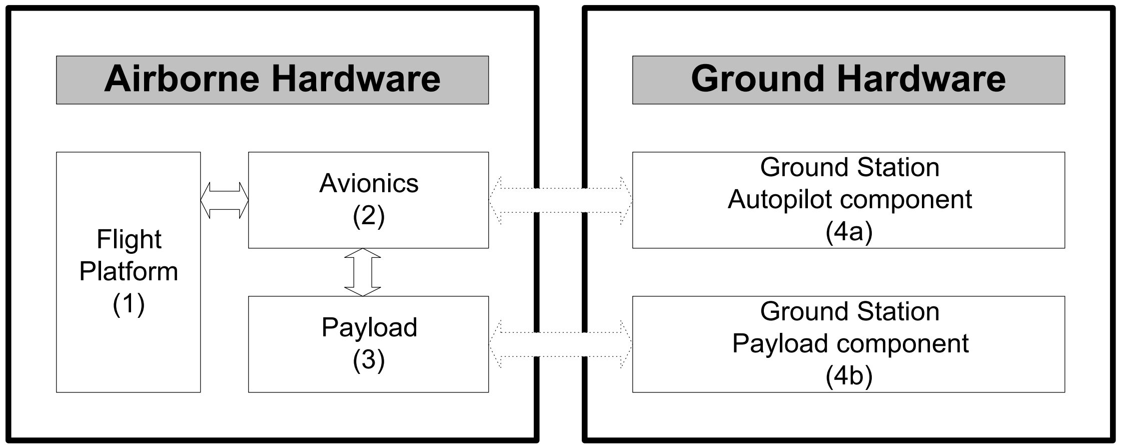

Airborne Hardware
Ground Hardware
Flight Platform (1)
Avionics (2)
Payload (3)
Ground Station Autopilot component (4a)
Ground Station Payload component (4b)

Figure 5.1: Autopilot hardware diagram.

5.1.1 AIRBORNE COMPONENTS

The airborne components of the autopilot system are the hardware devices that are carried in the aircraft. They include the GPS, autopilot, and digital modem. Figure 5.2 shows the airborne hardware.

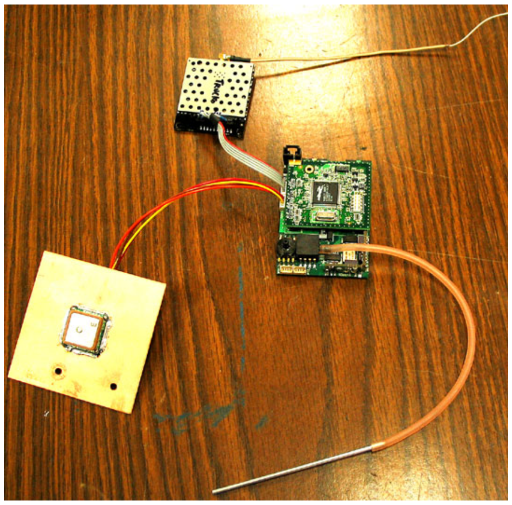


Figure 5.2: Airborne Hardware.

AUTOPILOT

The autopilot is the heart of the Kestrel system. It is powered by an 8-bit 29 MHz processor. The autopilot board contains a suite of sensors used by the autopilot software to measure and estimate the states of the aircraft. The autopilot interfaces directly to the digital communication link, which enables it to send real-time status telemetry to the ground station and receive commands in-flight. The GPS plugs into the autopilot board and provides inertial navigation information to the autopilot. It also has several additional interface ports to support payloads. The autopilot controls the aircraft through four standard RC hobby servo ports.

GPS

The Kestrel system supports two GPS interfaces: Furuno Binary and standard NMEA. The autopilot is programmed by default for Furuno gh-80 GPS. A different Furuno or NMEA GPS can be used by making simple modifications to the autopilot source code. The GPS provides position, heading, and velocity information necessary for waypoint navigation. The autopilot does not require a GPS for operator-assisted modes in which pilot input is used to navigate the aircraft.

DIGITAL COMMUNICATION LINK

The autopilot supports communication with a digital modem running at 115 Kbaud. Any digital modem which supports a 115 Kbaud asynchronous interface can be used. The standard modem used with the Kestrel autopilot system is the Aerocomm AC4490-5-m3. This modem has a 115 Kbaud interface and supports a 57600 Kbaud over-the-air baud rate and a 500 mW power output. The Aerocomm modem uses a server/client infrastructure that allows several aircraft to be controlled from one ground station. The aircraft can communicate with each other as well as the ground station. Using a ¼-wave dipole on the aircraft and ground station, communication ranges of greater than 10 kilometers can easily be achieved. As of the writing of this document, the modems have been tested out to a range of 5 kilometers with minimal packet loss.

PAYLOAD

The standard payload used in the Kestrel system is a video payload. The video payload consists of the Panasonic GX-171 CCD camera and the Felseweb 500 mW 2.4 GHz video transmitter. This system is capable of a range greater than 5 km when a directional antenna is employed on the receiving side. Figure 5.3 shows the Panasonic video camera gimbal mounted in a flying wing type airframe.

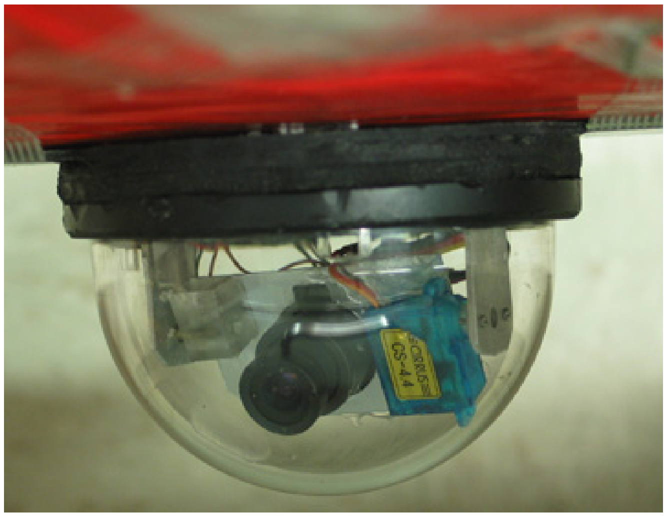


Figure 5.3: Gimbaled video camera as part of the video payload.

5.1.2 GROUND STATION COMPONENTS

The ground station components consist of the ground station hardware, a Futaba-compatible RC transmitter, and ground station software. The Futaba transmitter is only used during development and is not required for autonomous flight.

GROUND STATION HARDWARE

The ground station hardware consists of the main board, digital modem, and GPS. The main board has an 8-bit CPU that decodes the pulse train from the Futaba transmitter into controls for the airplane. It also passes data between the digital modem and the ground station computer, and parses position information from the GPS. The ground station has 3 main purposes. The first is to send up RC transmitter stick information to the aircraft for Pilot-in-the-Loop control. The second is to handle communications between the ground station computer and the aircraft. The third is to pass the position information from the ground station GPS to the ground station computer. Figure 5.4 shows the ground station hardware.

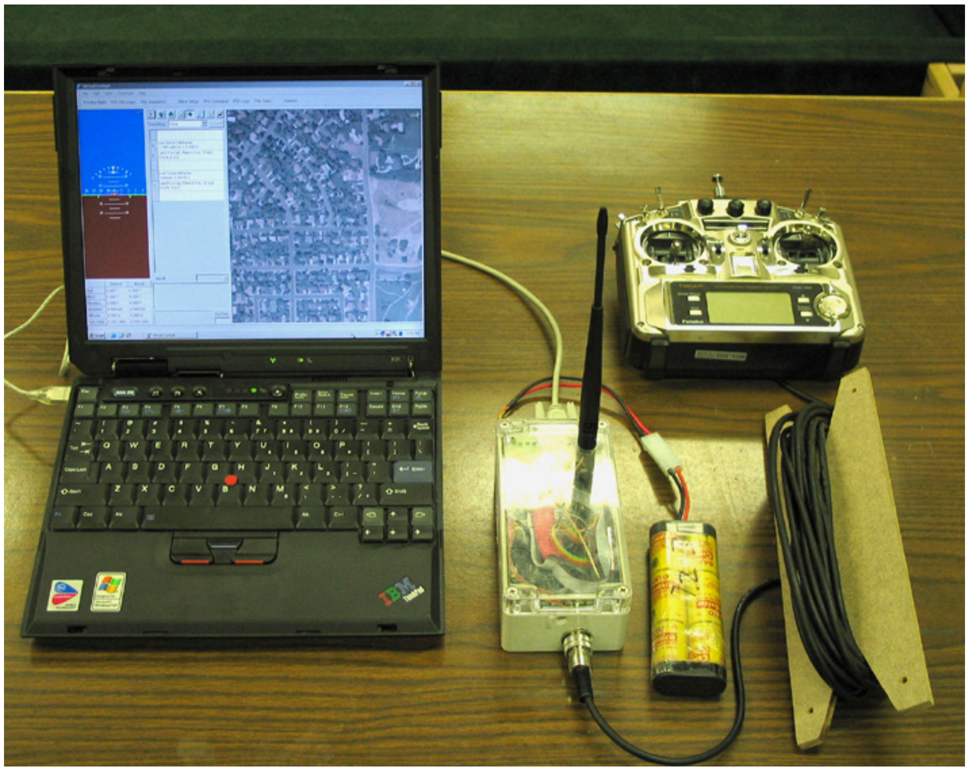


Figure 5.4: Ground station hardware.

FUTABA TRANSMITTER

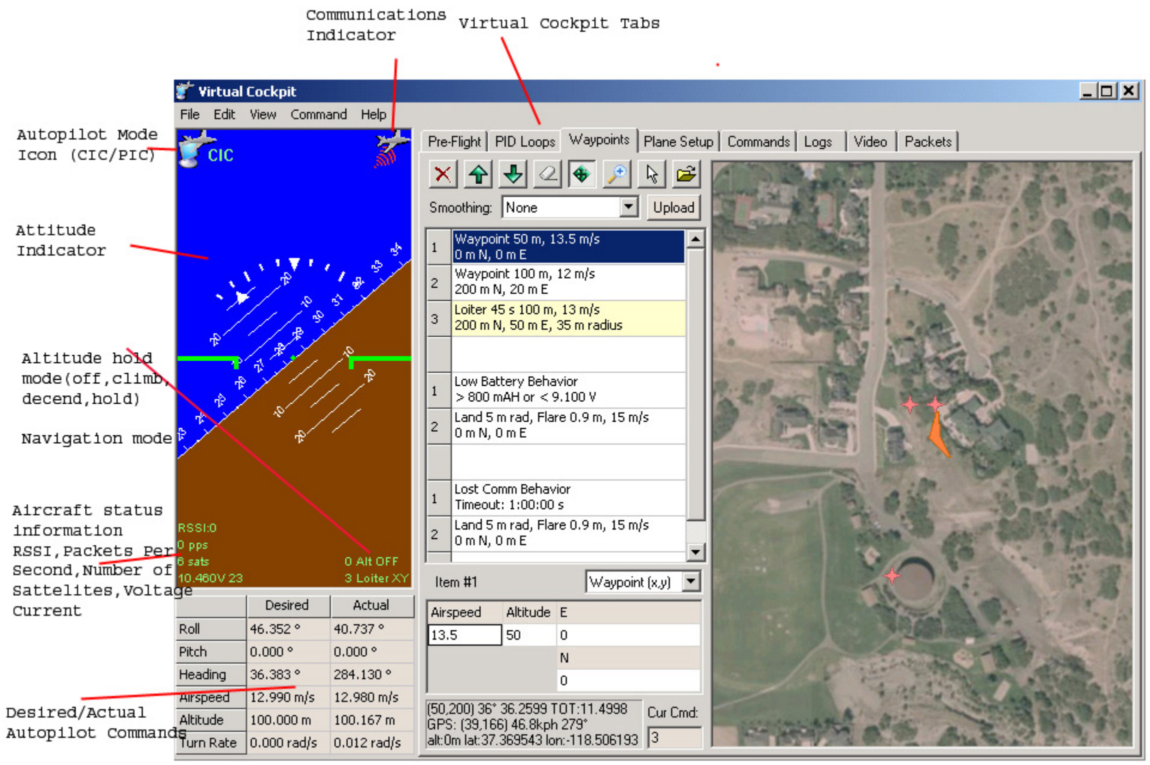
Any Futaba transmitter of 5 channels or more can be used with the ground station. The aircraft is flown using this transmitter when the autopilot is off and during Pilot-in-the-Loop modes. The channel 5 landing gear switch is used to turn the autopilot on and off. When the channel 5 pulse is longer than 1500 μs, the autopilot is enabled (Computer-in-Command mode), when the pulse is shorter than 1500 μs, the autopilot is disabled (Pilot-in-Command mode). The transmitter is not needed for autonomous modes and can be unplugged when the autonomous modes are enabled.

5.2 GROUND STATION SOFTWARE

A graphical interface was written to control the autopilot. The graphical interface software is known as the Virtual Cockpit. The Virtual Cockpit is a complete system that is used to configure, debug, program, and monitor the autopilot.

THE VIRTUAL COCKPIT

The Virtual Cockpit contains several screens accessible by tabs and a Status screen that is always visible. There are several options available in the dropdown menus. The purpose of the Status window is to give the user an indication of the aircraft's status and health. The Virtual Cockpit has four tab windows, which contain specific control and setup information. The discussion on the Virtual Cockpit will begin with the Status window, and then proceed to the four tab windows: PID Loops, Waypoints, Plane Setup, and Pre-flight.



	Desired	Actual
Roll	46.352 °	40.737 °
Pitch	0.000 °	0.000 °
Heading	36.383 °	284.130 °
Airspeed	12.990 m/s	12.980 m/s
Altitude	100.000 m	100.167 m
Turn Rate	0.000 rad/s	0.012 rad/s

1	Waypoint 50 m, 13.5 m/s 0 m N, 0 m E
2	Waypoint 100 m, 12 m/s 200 m N, 20 m E
3	Loiter 45 s 100 m, 13 m/s 200 m N, 50 m E, 35 m radius
1	Low Battery Behavior > 800 mAH or < 9.100 V
2	Land 5 m rad, Flare 0.9 m, 15 m/s 0 m N, 0 m E
1	Lost Comm Behavior Timeout: 1:00:00 s
2	Land 5 m rad, Flare 0.9 m, 15 m/s 0 m N, 0 m E

Airspeed	Altitude	E
13.5	50	0
		N
		0

(50,200) 36° 36.2599 TOT:11.4998
GPS: (39,166) 46.8kph 279°
alt:0m lat:37.369543 lon:-118.506193

Figure 5.5: Virtual Cockpit main window.

5.2.1 STATUS WINDOW

The Status window shown in Figure 5.5 is always visible and contains a collection of vital aircraft status information. The purpose of the Status window is to give the user an indication of the aircraft's status and health. The Status window is divided into two windows: the Attitude Indicator and the Desired/Actual window.

ATTITUDE INDICATOR

The most prominent feature of the Status window is the attitude indicator (see Figure 5.5). The attitude indicator is a visual representation of the aircraft's pitch, roll, and heading.

Heading information is displayed on the centerline of the attitude indicator, while pitch and roll can be derived from the location of the “horizon” as seen from the plane.

The attitude indicator has important aircraft status information overlaid on it. The overlaid information will be discussed in detail below. The discussion will proceed as follows: Upper left: autopilot mode, Upper right: comm indicator, Lower right: general status information, Lower left: navigation and altitude hold modes.

AUTOPILOT MODE

The upper left corner displays the autopilot mode. The autopilot has two modes: Pilot-in-Command (PIC) and Computer-in-Command (CIC). The mode is controlled by the channel 5 switch on the RC transmitter. If the channel 5 pulse is longer than 1500 μsec, the autopilot switches to CIC mode. If the channel 5 pulse is shorter than 1500 μsec, the autopilot switches to PIC mode. If no bypass is present, or the Futaba RC controller is not plugged into the ground station, the autopilot defaults to CIC mode.

In PIC mode, the autopilot actuator outputs are disabled and the aircraft is controlled either via the stick positions of the RC controller plugged into the ground station, or via the servo control signals from the Bypass circuit (if present). The autopilot does not control the aircraft in this mode.

In CIC mode, the aircraft is commanded by the control efforts generated by the autopilot. Pilot inputs from the Bypass circuit, or Futaba RC controller plugged into the ground station, are summed into the autopilot control efforts. This is useful in autopilot gain tuning and development as it provides a method to test disturbance rejection by giving the pilot some control over the aircraft even when the autopilot is enabled. The pilot inputs can serve as disturbance inputs as well as control inputs.

COMMUNICATIONS INDICATOR

The upper right corner displays the digital communications link status with the autopilot. The autopilot sends two types of unsolicited packets. The first type consists of standard telemetry packets. By default, these packets are sent 10 times a second, and contain basic aircraft state and status information. The second type of unsolicited packet is the Navigation Status packet. These packets are only sent when the autopilot has a valid 2-D GPS fix. They are sent once a second, and contain GPS and navigation status information. During normal autopilot operation, the ground station receives 11 unsolicited packets from the autopilot every second. If more than 1.5 seconds have passed since the ground station has received an autopilot packet, the bars on the communication indicator disappear, indicating a loss of communication.

GENERAL STATUS INFORMATION

General aircraft status information is displayed in the lower right corner of the attitude indicator. The status information is updated 10 times a second by the standard telemetry packets.

The first line displays the Received Signal Strength Indication. The next line displays the Futaba RC controller status. This is indicated by the number of stick position packets received by the autopilot every second (pps). When the Futaba RC controller is plugged into the ground station and turned on, the stick positions are sent to the autopilot 20 times a second. The actual number received by the autopilot is sent to the ground station in the standard telemetry packet. It serves as an indication of communication link quality and RC controller status, which is especially important in Pilot-in-Command mode. PIC mode should only be attempted when at least 10 stick position packets per second are being received by the autopilot.

The next line shows the number of satellites acquired by the airborne GPS receiver. Autonomous flight should not be attempted with less than four satellites acquired.

The last line displays the aircraft main battery voltage and current. (The current is only available when a current shunt is used and plugged into pin 3 of the main power header.) The autopilot will not function properly on a voltage below 6.5 volts.

NAVIGATION AND ALTITUDE HOLD STATUS INFORMATION

The text on the lower right corner of the attitude indicator displays navigation and altitude hold mode information.

The bottom line displays the navigation mode. The autopilot has seven navigation modes:

1. Get new command
2. En route to waypoint
3. Loiter
4. Land
5. Takeoff
6. Climbout

The top line displays the altitude hold mode. There are 4 modes: Off, Hold, Climb, and Descent. When enabled, the altitude tracker selects the correct PID loops to control the aircrafts altitude. Navigation and status modes will be discussed in Section 5.2.3).

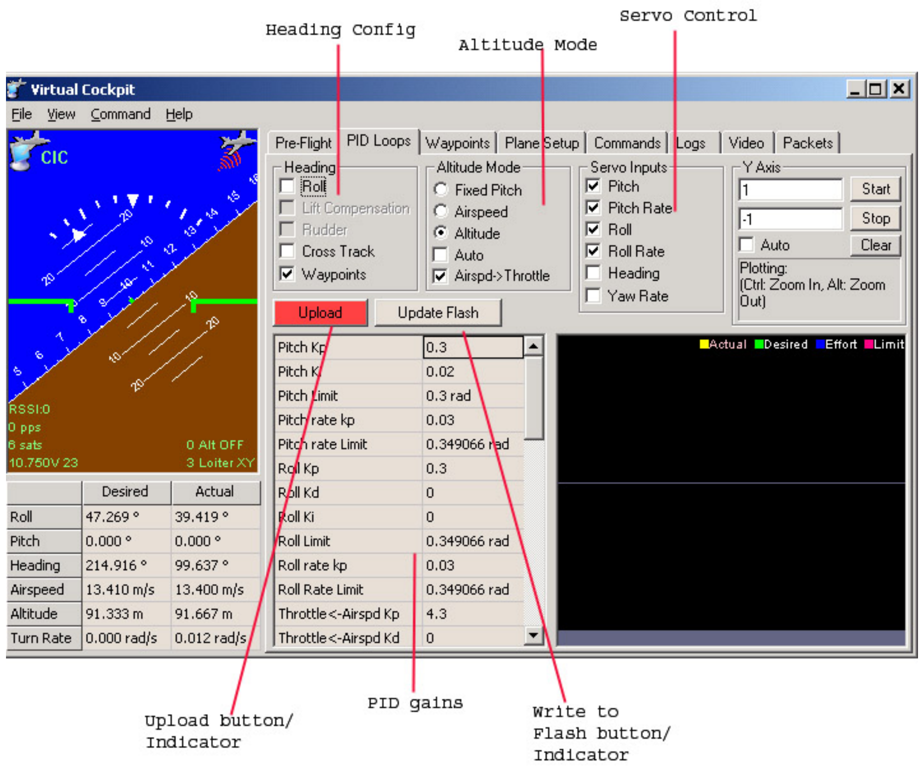
DESIRED AND ACTUAL INFORMATION

The lower portion of the Status window is occupied by the Desired/Actual state window. This window is used to view and command the aircraft states. The states shown are pitch, roll, heading, altitude, velocity, and turn rate. The window is updated 10 times a second by unsolicited standard telemetry packets from the autopilot. Display units can be changed between English and SI via the view dropdown menu. The autopilot can be commanded by simply typing in the new value in the desired window. If the appropriate PID loops are enabled, the autopilot will attempt to control the airplane to match the desired value. If waypoint navigation or the altitude tracker is enabled, the desired values will be overwritten by the autopilot navigation and altitude hold routines.

5.2.2 PID LOOPS TAB

The PID Loops tab is used to configure the PID loops on the aircraft. Here the loops can be configured, and gains can be set. The PID Loops tab is shown in Figure 5.6. The PID Loops tab is divided into two sections: the PID Control window and the PID Gains window. The PID Control window is used to enable and disable the PID loops on the autopilot. The PID Control window is further divided into three sections: Servo

Inputs window, Altitude Mode window, and Heading Mode window. The PID Gains window is used to set the PID gains on the autopilot



Heading: [ ] Roll, [ ] Lift Compensation, [ ] Rudder, [ ] Cross Track, [✓] Waypoints

Altitude Mode: ( ) Fixed Pitch, ( ) Airspeed, (●) Altitude, [ ] Auto, [✓] Airspd->Throttle

Servo Inputs: [✓] Pitch, [✓] Pitch Rate, [✓] Roll, [✓] Roll Rate, [ ] Heading, [ ] Yaw Rate

Y Axis: 1, -1, [ ] Auto

	Desired	Actual
Roll	47.269 °	39.419 °
Pitch	0.000 °	0.000 °
Heading	214.916 °	99.637 °
Airspeed	13.410 m/s	13.400 m/s
Altitude	91.333 m	91.667 m
Turn Rate	0.000 rad/s	0.012 rad/s

Pitch Kp	0.3
Pitch Ki	0.02
Pitch Limit	0.3 rad
Pitch rate kp	0.03
Pitch rate Limit	0.349066 rad
Roll Kp	0.3
Roll Kd	0
Roll Ki	0
Roll Limit	0.349066 rad
Roll rate kp	0.03
Roll Rate Limit	0.349066 rad
Throttle<-Airspd Kp	4.3
Throttle<-Airspd Kd	0

Figure 5.6: PID Loop configuration tab.

PID CONFIGURATION WINDOW

Low-level aircraft control is accomplished by a collection of nested PID loops. The inner loops control the servos directly; the outer loops control the inner loops. The autopilot can receive commanded values for any PID loop. However, if an outer loop is enabled, any commanded value for its inner loops will be overwritten by the outer loop's control effort. Table 5.1 lists the various autopilot PID loops. The servo control loops are accessible in the Servo Input window. The altitude and velocity loops are controlled via the Altitude Mode window. The Heading loops and Waypoint Navigation script (hereafter referred to as Nav script) are controlled in the Heading window. Each of these windows will be discussed in further detail in this section.

Table 5.1: Autopilot PID loops (grey fill = inner, no fill = outer).

loop name	Input	output	Sensor	Function
Elevator Pitch Angle	Pitch Angle	Elevator Servo Position	Estimated Pitch (Theta)	Control aircraft pitch
Eelvator Pitch Rate	Fixed Zero	Elevator Servo Position	Pitch Gyro (Q)	Dampen aircraft pitch rate
Aileron Roll Angle	Roll Angle	Aileron Servo Position	Estimated Roll (Phi)	Control aircraft pitch
Aileron Roll Rate	Fixed Zero	Aileron Servo Position	Roll Gyro (P)	Dampen aircraft pitch rate
Rudder Yaw Rate	Fixed Zero	Rudder Servo Position	Yaw Gyro R	Dampen aircraft yaw rate
Rudder Heading Rate	Heading Rate	Rudder Servo Position	Estimated Heading Rate (PSIDOT)	Control aircraft heading rate
Throttle Velocity	Airspeed	Throttle Servo Position	Pressure Airpseed (Vp)	Control aircraft airspeed
Pitch Airspeed	Airspeed	Pitch Angle	Pressure Airpseed (Vp)	Control aircraft airspeed
Pitch Altitude	Altitude	Pitch Angle	Pressure Altitude (H)	Control aircraft altitude
Roll heading	Heading	Roll Angle	Estimated Heading (PSI)	Control aircraft heading

SERVO CONTROL WINDOW

The Servo Control window is used to enable and disable the inner PID loops that are connected directly to servos that control the aircraft. If a box is checked, its loop is enabled. The command to enable or disable loops is not sent to the aircraft until the "Upload" button is clicked. This button will turn red when changes in PID loop setup have not been sent to the aircraft. Upon receipt of an acknowledgment from the aircraft, the button color will return to grey. It is important to note that the appropriate inner loops must be enabled for the outer loops to control the aircraft. For example, in order to use the Pitch from Altitude loop to control the aircraft altitude, the inner Elevator from Pitch loop must be enabled. For instructions on enabling the correct loops for different flight modes, refer to the PID tuning and Pre-flight sections of this document.

ALTITUDE MODE WINDOW

The Altitude Mode window is used to control the autopilot velocity and altitude hold modes. The current altitude mode is visible in the lower right hand corner of the Attitude Indicator.

The first three radio buttons represent the outer PID loops that generate desired Pitch angles for the inner Elevator from Pitch loop. If "Fixed Pitch" is selected, then the desired Pitch is held constant at the value commanded by the user in the Desired/Actual window of the Status window. This value defaults to 0 degree pitch at power-up. If the "Airspeed" radio button is selected, then the desired Pitch will be generated using the Pitch from Airspeed PID loop. This loop generates the desired pitch angle from the difference between the commanded airspeed and the measured airspeed. If the airspeed is below the desired, the autopilot commands the aircraft to pitch down, to increase the airspeed. If the aircraft is going too fast, the autopilot commands the aircraft to pitch up. The airspeed is commanded by the user in the Desired/Actual window of the Status window or, if Waypoint Navigation is enabled, the airspeed for the current waypoint is used.

The "Auto" check box enables (when checked) or disables the Altitude Tracker. When the Altitude Tracker is enabled, the autopilot automatically selects the appropriate loops to control airspeed and altitude based on the altitude error. When the "Altitude Tracker" box is checked, the user does not need to manage any of the PID loops inside the Altitude Mode window as the autopilot will take care of this automatically. The altitude tracker has four modes: Off, Climb, Descent, and Hold. The current mode is displayed on the lower right of the Attitude Indicator. In the Off mode, the user selects the appropriate PID loops to control altitude. In Climb mode, the autopilot regulates the airspeed using the Pitch from Airspeed loop and the throttle is set to full. In Descent mode, the autopilot regulates the airspeed using the Pitch from Airspeed loop and the throttle is set to zero percent. In the Hold mode, the throttle is regulated using the Throttle from Airspeed PID loop and the altitude is regulated using the Pitch from Altitude loop. During normal flight, the Altitude Tracker should be enabled.

The “AirSpd->Throttle” check box is used to enable and disable the Throttle from Airspeed inner PID loop. When checked, the autopilot regulates the airspeed of the aircraft by adjusting the throttle. The airspeed is commanded by the user in the Desired/Actual window of the Status window or, if Waypoint Navigation is enabled, the airspeed for the current waypoint is used. If the Altitude Tracker is enabled, the autopilot will enable and disable this PID loop as necessary.

HEADING CONTROL WINDOW

The Heading Control window is used to enable and disable the PID loop that controls aircraft heading, and the Nav script.

The “Roll” check box is used to enable and disable the Roll from Heading PID loop. When checked, the autopilot controls the heading of the aircraft by commanding a desired Roll angle. The inner Aileron from Roll angle loop affects the commanded Roll angle by moving the aileron actuator. The Aileron from Roll angle loop must be enabled for the Roll from Heading loop to control the aircraft Heading. The aircraft heading is commanded by the user in the Desired/Actual window of the Status window or, if Waypoint Navigation is enabled, the commanded Heading will be computed automatically based on the relative position to the current waypoint. If Waypoint Navigation is enabled, the Roll from Heading PID loop will be enabled and disabled automatically as needed for proper aircraft navigation. If the Roll from Heading loop is disabled, the roll angles for the Aileron from Roll PID loop can be commanded by the user in the Desired/Actual window of the Status window.

The "Cross Track" check box is used to enable and disable the Cross Track controller. This box only has an effect when Waypoint Navigation is enabled. When the Cross Track controller is enabled, the Nav script will attempt to fly the course between waypoints. When Cross Track is disabled, the Nav script will control the aircraft Heading using the bearing to the current waypoint. Cross Track controller should be enabled for normal Waypoint Navigation.

The "Waypoints" check box enables and disables Waypoint Navigation. When checked, the Nav script is enabled. When Waypoint Navigation is enabled, the desired Roll angle, altitude, Heading, and velocity are commanded by the autopilot. The commanded values can be viewed in the Desired/Actual window of the Status window. For Waypoint Navigation to work correctly, the Elevator from Pitch and Aileron from Roll PID loops must be enabled. The Altitude Tracker or Pitch from Altitude and Throttle from Airspeed must also be enabled. The Aileron from Roll Rate and Elevator from Pitch rate can also be enabled to provide additional rate damping. The Nav script will automatically enable and disable the Roll from Heading loop as needed. The Waypoint Navigation status can be viewed on the lower right hand portion of the Attitude Indicator inside the Status window. The aircraft's position and additional navigation information can be view in the Navigation Status window inside the Waypoints tab window. The Waypoint Navigation modes of the aircraft will be described in more detail in Section 5.2.3.

PID GAINS WINDOW

The PID Gains window is used to view and modify the current PID loop gains. When communication is first established with the autopilot, the Virtual Cockpit requests the PID gain information from the autopilot. To request the current PID gains at a later time, select drop-down option “File->Request All Values.”

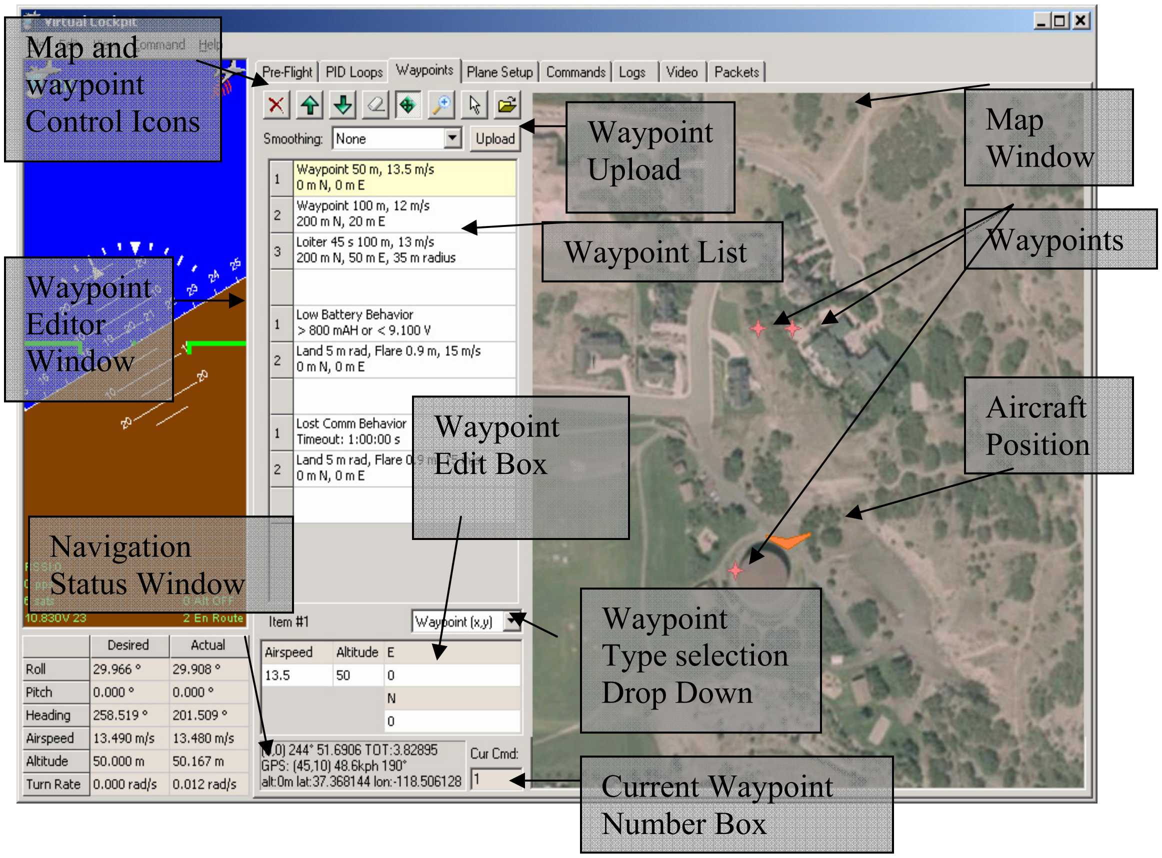
To change PID gains on the airplane, place the cursor in the appropriate field and enter the new gain. Press “Enter” to send the new value to the aircraft. When the aircraft acknowledges the new gain, the field background will change from white to grey. Table 5.2 lists gains that may be changed. The gain change is effected by default in the RAM of the autopilot, and will be lost when the autopilot is powered off. To effect the change in the autopilot’s nonvolatile memory, use the “Update Flash” button (once clicked, this button will turn red until the Virtual Cockpit receives an acknowledgment from the aircraft). The Update Flash function takes up to a second to occur on the aircraft. During this time, the autopilot cannot control the aircraft. Because of this, do not update the flash when the aircraft is airborne.

Table 5.2: User programmable PID loops.

PID Loop	Value	Description
Elevator from Pitch	Kp	Proportional Gain
Elevator from Pitch	Ki	Integral Gain
Elevator from Pitch	Limit	Absolute Saturation Limit in Radians of Elevator deflection
Elevator from Pitch Rate	Kp	Proportional Gain
Elevator from Pitch Rate	Limit	Absolute Saturation Limit in Radians of Elevator deflection
Aileron from Roll	Kp	Proportional Gain
Aileron from Roll	Ki	Integral Gain
Aileron from Roll	Limit	Absolute Saturation Limit in Radians of Aileron deflection
Aileron from Roll Rate	Kp	Proportional Gain
Aileron from Roll Rate	Limit	Absolute Saturation Limit in Radians of Aileron deflection
Throttle from Airspeed	Kp	Proportional Gain
Throttle from Airspeed	Kd	Derivative Gain
Throttle from Airspeed	Ki	Integral Gain
Throttle from Airspeed	Limit	Absolute Saturation Limit in Percent(0 to 100) Throttle
Rudder from Yaw Rate	Kp	Proportional Gain
Rudder from Yaw Rate	Limit	Absolute Saturation Limit in Radians of Rudder deflection
Rudder from Heading Rate	Kp	Proportional Gain
Rudder from Heading Rate	Kd	Derivative Gain
Rudder from Heading Rate	Ki	Integral Gain
Rudder from Heading Rate	Limit	Absolute Saturation Limit in Radians of Rudder deflection
Pitch from Airspeed	Kp	Proportional Gain
Pitch from Airspeed	Kd	Derivative Gain
Pitch from Airspeed	Ki	Integral Gain
Pitch from Airspeed	Limit	Absolute Saturation Limit in Radians of command Pitch to inner loop
Pitch from Altitude	Kp	Proportional Gain
Pitch from Altitude	Kd	Derivative Gain
Pitch from Altitude	Ki	Integral Gain
Pitch from Altitude	Limit	Absolute Saturation Limit in Radians of command Pitch to inner loop
Roll from Heading	Kp	Proportional Gain
Roll from Heading	Kd	Derivative Gain
Roll from Heading	Ki	Integral Gain
Roll from Heading	Limit	Absolute Saturation Limit in Radians of command Roll to inner loop

5.2.3 WAYPOINT TAB WINDOW

The Waypoint tab window contains autopilot Waypoint Navigation information. The principle components of the Waypoint tab window are the Map window, the Waypoint Editor window, and the Navigation Status window. Figure 5.7 shows the Waypoint tab window.



Smoothing: None | Upload

1	Waypoint 50 m, 13.5 m/s 0 m N, 0 m E
2	Waypoint 100 m, 12 m/s 200 m N, 20 m E
3	Loiter 45 s 100 m, 13 m/s 200 m N, 50 m E, 35 m radius
1	Low Battery Behavior > 800 mAH or < 9.100 V
2	Land 5 m rad, Flare 0.9 m, 15 m/s 0 m N, 0 m E
1	Lost Comm Behavior Timeout: 1:00:00 s
2	Land 5 m rad, Flare 0.9 m 0 m N, 0 m E

Item #1 | Waypoint (x,y)

Airspeed	Altitude	E
13.5	50	0
		N
		0

	Desired	Actual
Roll	29.966 °	29.908 °
Pitch	0.000 °	0.000 °
Heading	258.519 °	201.509 °
Airspeed	13.490 m/s	13.480 m/s
Altitude	50.000 m	50.167 m
Turn Rate	0.000 rad/s	0.012 rad/s

```
(0,0) 244° 51.6906 TOT:3.82895
GPS: (45,10) 48.6kph 190°
alt:0m lat:37.368144 lon:-118.506128
```

Cur Cmd: 1

Figure 5.7: Waypoint tab window.

MAP WINDOW

The Map window displays the position of the aircraft and the waypoints. The map can be manipulated using the Map Control icons.

The Pan Icon allows the map to be panned by clicking on the map with the left mouse button and moving the mouse pointer.

The Zoom Icon allows the map to be zoomed by clicking on the map with the left mouse button and moving the mouse pointer. Moving the mouse pointer up will decrease the zoom level. Moving the mouse pointer down will increase the zoom level.

WAYPOINT EDITOR WINDOW

The Waypoint Editor window is used to display and edit the aircraft's flight plan, which is made up of waypoint commands. The Waypoint Editor window is composed of the Waypoint Control icons (Top), the Waypoint List, the Waypoint Type Selection drop-down menu, and the Waypoint edit box.

The Waypoint List displays the waypoints in the flight plan. To download the flight plan from the airplane, select “Edit->Download.” To upload the flight plan to the airplane, click the “Upload” button above the flight plan. The button will change from red to grey when the flight plan is done uploading. The waypoints are displayed in the order they will be flown by the airplane. The first waypoint is displayed at the top of the list and is recognized by the autopilot as waypoint 1. The waypoint currently being flown by the aircraft will be highlighted in yellow on the Waypoint List. The waypoints will also appear in the Map window.

The Waypoint Control Icons, located at the top of the Waypoint List, are used to change the ordering and remove the waypoints in the flight plan. The Delete icon, deletes the selected waypoint from the flight plan. The waypoint ordering can be changed by selecting a waypoint with the mouse, and then using the Move Up icon, or the Move Down icon.

The Waypoint edit box is used to add new waypoints to the flight plan, and change waypoint in the flight plan. To add a new waypoint to the flight plan, click on the empty box below the last waypoint. Next select the waypoint type from the Waypoint Type Selection drop-down menu. Once the appropriate waypoint type is selected, the parameters for that waypoint type are entered in the Waypoint edit box.

WAYPOINT TYPES

There are seven types of waypoints:

1. *Goto XY*: The *Goto XY* waypoint command is used to navigate to a position specified in meters east and meters north of the GPS Home Position. The *Goto XY* command requires the following parameters: meters east, meters north, altitude (meters), airspeed (meters/second).

2. *Goto Legal*: The *Goto Legal* waypoint command is used to navigate to a position specified in degrees longitude and latitude. The *Goto Legal* command requires the following parameters: degrees longitude, degrees latitude, altitude (meters), airspeed (meters/second).

3. *Loiter XY*: The *Loiter XY* waypoint command executes an orbit around a position specified in meters east and meters north of the GPS Home Position for a specified period of time. The *Loiter XY* command requires the following parameters: meters east, meters north, altitude (meters), airspeed (meters/second), orbit time

(seconds), and orbit radius (meters). If 0 is entered into the orbit time field, the autopilot will command to orbit the aircraft indefinitely around the specified point.

4. *Loiter Legal*: The *Loiter Legal* waypoint command executes an orbit around a position specified in degrees longitude and latitude for a specified period of time. The *Loiter Legal* command requires the following parameters: degrees longitude, degrees latitude, altitude (meters), airspeed (meters/second), orbit time (seconds). If 0 is entered into the orbit time field, the autopilot will command to orbit the aircraft indefinitely around the specified point.

5. *Land XY*: The *Land XY* waypoint command executes a landing sequence around a position specified in meters east and meters north of the GPS Home Position. The landing sequence orbits the aircraft around the landing position while holding airspeed. The autopilot then cuts throttle and allows the aircraft to descend in the orbit while regulating the airspeed using the Pitch from Airspeed loop. When the aircraft descends to the altitude specified by the Flair Height parameter entered for this command, the autopilot attempts to level the aircraft by commanding a 0° roll angle to the Aileron from Roll loop. If the Flair Height is set to zero, the autopilot will not attempt to flair the aircraft. The *Land XY* command requires the following parameters: meters east, meters north, altitude (meters), airspeed (meters/second), orbit radius (meters), and flair height (meters).

6. *Take Off*: The *Take Off* waypoint command executes a take-off sequence at the current GPS position. The take-off sequence commands the aircraft to hold a constant Pitch (specified in the Take Off Pitch UAV Limit field) while using the Aileron from Roll loop to hold a 0 degree roll angle. The autopilot commands full throttle. Once

the aircraft reaches the airspeed specified in the Rotate Airspeed field, the autopilot enables the Altitude Tracker and commands an orbit at the current GPS location with the orbit radius specified in the Orbit Radius field. If the aircraft does not achieve the Rotate Velocity within the period of time specified in the parameter Take Off Rotate Time Out, then the take-off sequence is aborted, and the autopilot commands 0 percent throttle in an attempt to land the aircraft. If the aircraft reaches the altitude specified in the Altitude field, the *Take Off* Command is finished, and the Nav script executes the next waypoint command. The *Take Off* command takes the following parameters: command completion altitude (meters), rotate airspeed (meters/second), and climb out orbit radius (meters).

7. *Repeat*: The *Repeat* waypoint command causes the Nav script to execute the waypoint command specified in the Waypoint Number field. This command is used to repeatedly fly sequences of waypoints. The *Repeat* command takes the following parameter: waypoint number.

EDITING A WAYPOINT

To edit a waypoint, first select the waypoint that is to be edited in the Waypoint List. Then edit the waypoint using the Waypoint Type Selection drop-down menu and the Waypoint Edit box.

Once the flight plan is completed, it can be uploaded to the aircraft using the "Waypoint Upload" button.

NAVIGATION STATUS WINDOW

The Navigation Status window (see Figure 5.8) displays GPS and Waypoint Navigation status information. The first line displays navigation status information and the second and third lines display GPS information.

Waypoint Navigation status information is valid only when Waypoint Navigation is enabled. (The Waypoint Navigation state can be viewed in the lower right portion of the Attitude Indicator in the Status window.) The Navigation Status window also contains the Current Waypoint Number box. This is the box in the lower right hand corner of the Waypoint Edit window. This window displays the Waypoint Command Number that is currently executing. To command the Nav script to execute a different Waypoint Command Number, place the mouse cursor in the box, and enter the new waypoint number. The background will change from white to grey when the ground station receives acknowledgment from the autopilot that the new command number was received.

CURRENT WAYPOINT COMMAND NUMBER

The Current Waypoint Command Number window indicates the Waypoint Command currently being executed by the Nav script. The Nav script begins execution at command number 1. When all commands have been executed, the autopilot starts over at the first command. The user-definable waypoints start at 1 and end at 230. Waypoints 231 through 250 are reserved commands, and cannot be changed by the user. The reserved commands are used internally by the autopilot to fly pre-defined commands.

These commands can be executed by the user using the Autopilot Commands drop-down menu. The reserved commands follow:

1. Waypoint 231 is the *Go Home* command. This command is used by the autopilot to navigate the aircraft to the GPS Home Position. This command may not be executed directly by the user by setting the current command to 231. The user may execute this command by sending a *Go Home Now* command to the autopilot (Packet 50 type 2). This command is also used by the internal autopilot fail-safes to fly the airplane home in case of lost communication or low battery. When command 231 completes, the autopilot executes command 232, which is an indefinite loiter about home.
2. Waypoint 232 is a *Loiter Indefinitely about the GPS Home Position* command. This command may not be executed directly by the user by setting the current command to 232. This command is used by the internal autopilot fail-safes to fly the airplane home in-case of lost communication or low battery and is executed after a command 231 has completed.
3. Waypoint 233 is a *Loiter Indefinitely at the Current GPS Position* command. This command may not be executed directly by the user by setting the current command to 233. This command is executed by sending a Packet 50 type 2 with the loiter-now flag set. In the future, there will be a button in the Virtual cockpit for this function.

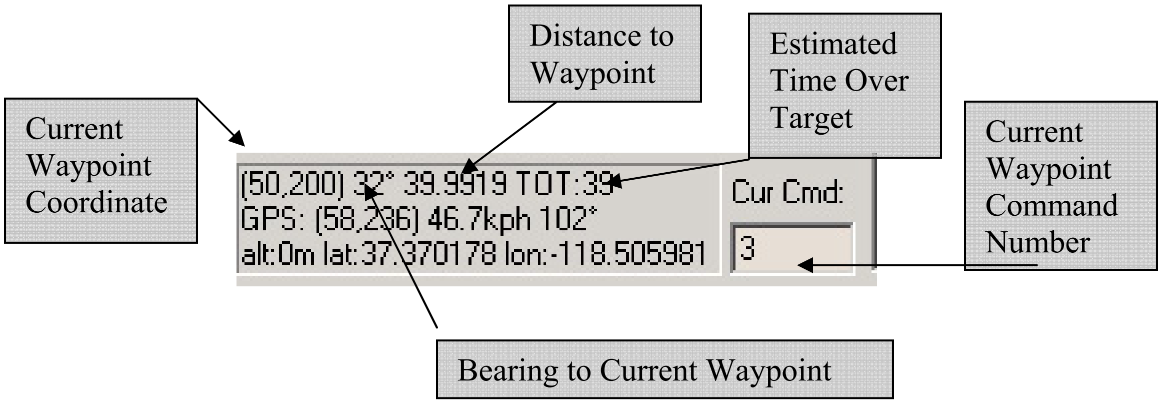
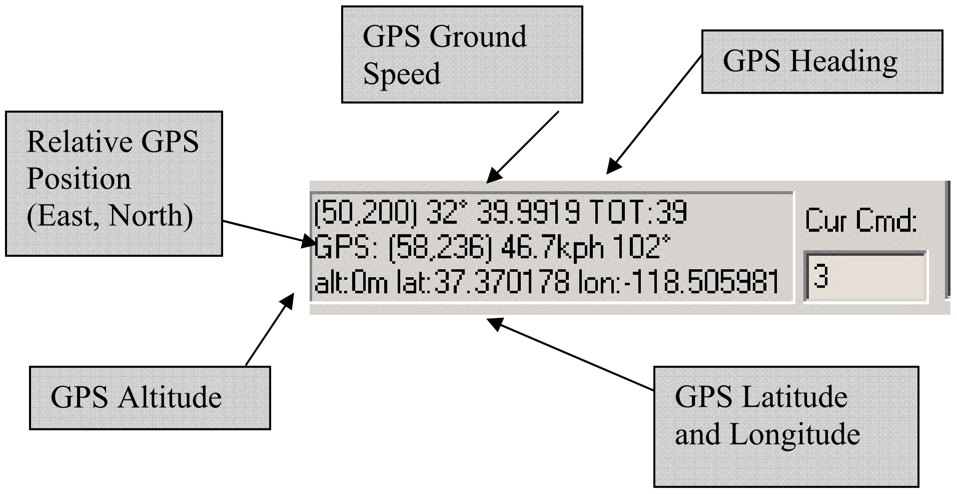

```
[50,200] 32° 39.9919 TOT:39
GPS: [58,236] 46.7kph 102°
alt:0m lat:37.370178 lon:-118.505981
```
Cur Cmd:
3
Current Waypoint Coordinate
Distance to Waypoint
Estimated Time Over Target
Current Waypoint Command Number
Bearing to Current Waypoint

Figure 5.8: Navigation Status Information box.

The Navigation Status Information box is located at the bottom of the Waypoint Navigation tab. Figure 5.8 shows the Navigation Status Information box. The first line contains Waypoint Navigation status information. The second and third lines contain GPS information. The following Navigation status information is displayed:

1. Current waypoint coordinates in meters east and meters north of the GPS Home Position. This is the autopilot's current destination. In the case of a *Loiter* command, this is the orbit point.
2. Bearing to current waypoint. This is the heading command by the autopilot to arrive at the current waypoint.
3. Distance to current waypoint. This is the distance between the current GPS position of the aircraft and the current waypoint coordinates.
4. Estimated time over target. This is the estimated time of arrival at the current waypoint coordinates. In the case of a *Loiter* command, it is the time in seconds remaining in the orbit. If the loiter is indefinite, this value will display 255.


GPS Ground Speed
GPS Heading
Relative GPS Position (East, North)
GPS Altitude
GPS Latitude and Longitude
Cur Cmd:
3

```
(50,200) 32° 39.9919 TOT:39
GPS: (58,236) 46.7kph 102°
alt:0m lat:37.370178 lon:-118.505981
```

Figure 5.9: GPS information in the Navigation Status box.

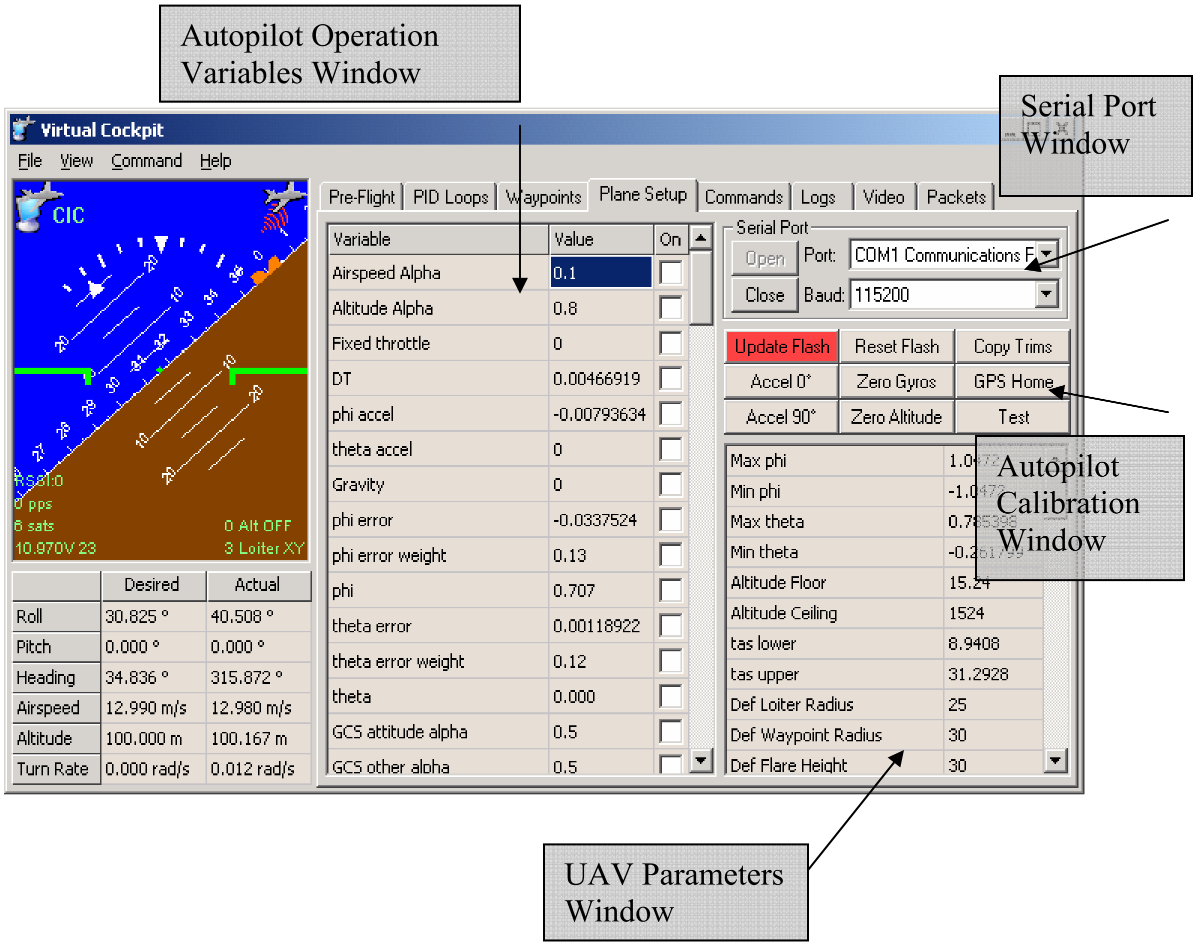
Figure 5.9 details the GPS status information visible in the Navigation Status box. The following GPS status information is displayed:

1. GPS Position in meters east and meters north of the GPS Home Position.
2. GPS Ground Speed in kilometers/hour.
3. GPS Heading in degrees (0 to 360).
4. GPS Altitude in meters above the GPS Home Position
5. GPS Latitude and Longitude in degrees.

5.2.3 PLANE SETUP TAB

The Plane Setup tab is used to configure the Virtual Cockpit serial interface, calibrate the autopilot, and display and edit Autopilot Operation Variables (hereafter referred to as variables) and NonVolatile UAV Parameters (hereafter referred to as parameters). The Plane Setup tab is composed of the following windows: Autopilot

Operation Variable window, Serial Port window, Autopilot Calibration window, and UAV Parameters window. Figure 5.10 shows the Plane Setup tab window.



	Desired	Actual
Roll	30.825 °	40.508 °
Pitch	0.000 °	0.000 °
Heading	34.836 °	315.872 °
Airspeed	12.990 m/s	12.980 m/s
Altitude	100.000 m	100.167 m
Turn Rate	0.000 rad/s	0.012 rad/s

Variable	Value	On
Airspeed Alpha	0.1	
Altitude Alpha	0.8	
Fixed throttle	0	
DT	0.00466919	
phi accel	-0.00793634	
theta accel	0	
Gravity	0	
phi error	-0.0337524	
phi error weight	0.13	
phi	0.707	
theta error	0.00118922	
theta error weight	0.12	
theta	0.000	
GCS attitude alpha	0.5	
GCS other alpha	0.5	

Port: COM1 Communications P
Baud: 115200

Max phi	1.0472
Min phi	-1.0472
Max theta	0.785398
Min theta	-0.261799
Altitude Floor	15.24
Altitude Ceiling	1524
tas lower	8.9408
tas upper	31.2928
Def Loiter Radius	25
Def Waypoint Radius	30
Def Flare Height	30

Figure 5.10: Plane Setup tab window.

AUTOPILOT OPERATION WINDOW

The Autopilot Operation Variable window is used to display and edit the Autopilot Operation Variables. These RAM-based variables are used internally in the autopilot, and prove useful for debugging autopilot operation and for temporarily changing a mode of operation. These variables should only be changed by a user familiar with the internal operation of the autopilot. The Virtual Cockpit requests these values from the autopilot when communication with the autopilot is first established. The user can request the entire set of values later by selecting “File->Request All Values.” To continually request the current value of a variable, check the box to the right of the variable name (it will be updated 5 times a second). There are 3 types of variables: Floating point, Integer, and Byte. These values are stored internally in the autopilot as 4-byte floating point values, 2-byte integer values, and 1-byte character values. The following is a list of the variables and their significance.

FLOATING POINT AUTOPILOT OPERATION VARIABLES

0. `Airspeed Alpha`: The low pass Alpha Filter constant used to filter the output of the differential pressure sensor.
1. `Altitude Alpha`: The Alpha Filter constant used to filter the output of the absolute pressure sensor.
2. Not used.
3. `DT`: The current time step of main autopilot loop (in seconds). During normal operation this value should range between .005 and .012 Seconds.
4. `Phi Accel`: This value represents the Euler roll angle (in radians), as measured by the accelerometers. This value is used by the Attitude Estimation Filter. This value is useful for debugging the accelerometers and finding errors in drift and attitude estimation.

5. `Theta Accel`: This value represents the Euler pitch angle (in radians), as measured by the accelerometers. This value is used by the Attitude Estimation Filter. This value is useful for debugging the accelerometers and finding errors in drift and attitude estimation.
6. `Gravity`: The magnitude of the total acceleration vector of the airplane. This value is in Accelerometer Units from the Analog to Digital converter (1 g is approximately 231 Accelerometer Units).
7. `Phi Error`: This is the difference measured in radians, between the roll estimate produced by the Attitude estimation filter and the Euler roll angle as measured by the accelerometers (see variable 5).
8. `Phi Error Weight`: This is the weighting factor placed on the `Phi Error` used to update the Phi estimate in the Attitude Estimation Filter. A higher value will place more weight on the Euler Phi angle measured by the accelerometers relative to that measured by the integrated rate gyros. If this value is set to 0, the accelerometers are not used to correct the roll state estimate produced by integrating the rate gyros.
9. `Phi`: The Euler roll angle in radians as estimated by the Attitude Estimation Filter.
10. `Theta Error`: This is the difference (measured in radians) between the pitch estimate produced by the Attitude Estimation Filter and the Euler pitch angle as measured by the accelerometers (see variable 6).
11. `Theta Error Weight`: This is the weighting factor placed on the P Error used to update the ϕ estimate in the Attitude Estimation Filter. A higher value will

place more weight on the Euler pitch angle measured by the accelerometers relative to the integrated rate gyros. If this value is set to 0, the accelerometers are not used to correct the pitch state estimate produced by integrating the rate gyros.

12. `Theta`: The Euler pitch angle in radians as estimated by the Attitude Estimation Filter.
13. `GCS Attitude Alpha`: The Alpha Filter constant used to filter the output of the Attitude Estimation filter before being sent to the Virtual Cockpit in the Standard Telemetry Stream. This value only applies to the roll, pitch, and yaw as viewed in the Virtual Cockpit. It has no effect on the internal values of roll, pitch, and yaw.
14. `GCS Other Alpha`: The Alpha Filter constant used to filter the altitude, airspeed, and battery voltage before being sent to the Virtual Cockpit in the Standard Telemetry Stream. This value only applies to these values as viewed in the Virtual Cockpit. It has no effect on the internal values of altitude, airspeed, and battery voltage.
15. `Psi`: The Euler heading angle in radians as estimated by the Attitude Estimation Filter.
16. `Phi K`: The Euler roll angle estimate in radians used internally by the Attitude Estimation Filter.
17. `Theta K`: The Euler pitch angle estimate in radians used internally by the Attitude Estimation Filter.
18. `GPS East Position`: The GPS position of the aircraft (in meters east of the GPS Home Position).

19. `GPS North Position`: The GPS position of the aircraft (in meters north of the GPS Home Position).
20. `GPS Speed`: The magnitude of the velocity vector (in meters/second) of the aircraft as reported by the GPS.
21. `GPS Heading`: The direction (in radians) of the velocity vector of the aircraft as reported by the GPS.
22. `GPS Altitude`: The GPS position of the aircraft (in meters above the GPS Home Position).
23. `Opt Roll Slope Right`: Not used.
24. `Opt Roll Slope Left`: Not used.
25. `Opt Pitch Slope Down`: Not used.
26. `Opt Roll Slope Up`: Not used.
27. `Psi Error`: This is the difference measured in radians between the heading estimate produced by the Attitude Estimation Filter and the GPS heading (see variable 21).
28. `Psi Error Weight`: This is the weighting factor placed on the Psi error used to update the heading estimate in the Attitude Estimation Filter. A higher value will place more weight on the GPS heading relative to the integrated rate gyros. If this value is set to 0, the GPS heading is not used to correct the heading state estimate produced by integrating the rate gyros.
29. `Psi K`: The Euler heading angle estimate in radians used internally by the Attitude Estimation Filter.

30. `Inertial Data Simulated East Pos`: When the `GPS Sim` variable (see Byte Autopilot Operation Variables) is set to 1, this value is used to simulate the position of the aircraft in meters east of the GPS Home Position. This is useful for debugging the Nav script.
31. `Inertial Data Simulated North Pos`: When the `GPS Sim` variable (see `GPS Sim` variable) is set to 1, this value is used to simulate the position of the aircraft in Meters North of the GPS Home Position. This is useful for debugging the Nav script.
32. `Inertial Data Simulated Velocity`: When the `GPS Sim` variable is set to 1, this value is used to simulate the GPS velocity of the aircraft. This is useful for debugging the Nav script.
33. `Inertial Data Simulated Heading`: When the `GPS Sim` variable is set to 1, this value is used to simulate the GPS Heading of the aircraft. This is useful for debugging the Nav script.
34. `Inertial Data Simulated Altitude`: When the `GPS Sim` variable is set to 1, this value is used to simulate the GPS altitude of the aircraft in meters above the GPS Home Position. This is useful for debugging the Nav script.
35. `Inertial Data Simulated Latitude`: When the `GPS Sim` variable is set to 2, this value is used to simulate the position of the aircraft in Degrees Latitude. This is useful for debugging the Nav script.
36. `Inertial Data Simulated Longitude`: When the `GPS Sim` variable is set to 2, this value is used to simulate the position of the aircraft in degrees longitude. This is useful for debugging the Nav script.

37. `Inertial Data Simulated GPS Home Latitude`: When the `GPS Sim` variable is set to 2, this value is used to simulate the GPS Home Position of the aircraft in degrees latitude. This variable is also used as the starting position for the GPS simulator when variable `GPS Sim` is set to 3. This is useful for debugging the Nav script.
38. `Inertial Data Simulated GPS Home Longitude`: When the `GPS Sim` variable is set to 2, this value is used to simulate the GPS Home Position of the aircraft in degrees longitude. This variable is also used as the starting position for the GPS simulator when `GPS Sim` is set to 3. This is useful for debugging the Nav script.
39. `Loss of Lift`: The value (in radians) of elevator deflection to be summed into the elevator actuator output as produced by the turn compensation algorithm. This elevator deflection is produced by the turn compensation algorithm to compensate for the loss of lift caused by aircraft banking. This algorithm is controlled by the parameter `Loss of Lift Gain`.
40. `RC Comm Packets`: The number of RC Stick Position packets received by the airplane from the ground station since power-up.
41. `Heading Rate`: The rate of change of the heading estimate produced by the Attitude Estimation filter. This value is used by the Rudder from Heading Rate loop.
42. `Roll Gyro No Drift Compensation`: The output of the roll gyro (in radians/second). This value is has been compensated for temperature, but not for drift.

43. `Pitch Gyro No Drift Compensation`: The output of the pitch gyro (radians/second). This value is has been compensated for temperature, but not for drift.
44. `Heading Gyro No Drift Compensation`: The output of the yaw gyro (in radians/second). This value is has been compensated for temperature, but not for drift.

INTEGER AUTOPILOT OPERATION VARIABLES

1. `Pitch Trim`: The elevator position (in radians*1000) summed into the PID loop efforts and RC controller effort that is sent to the elevator actuator. When the PID loops connected to the elevator actuator are disabled and the RC control position (if present) is 0, this value represents the commanded elevator position.
2. `Roll Trim`: The aileron position (in radians*1000) summed into the PID loop efforts and RC controller effort that is sent to the aileron actuator. When the PID loops connected to the aileron actuator are disabled and the RC control position (if present) is 0, this value represents the commanded aileron position
3. `PID Tuning Send Interval`: The time interval (in milliseconds) between PID tuning packets sent by the autopilot when PID tuning mode is enabled. (This mode is enabled by sending a PID tuning enable packet to the autopilot.) When PID tuning is enabled for a PID loop, the autopilot sends the desired, actual, and effort values to the Virtual Cockpit at the rate specified by this value.
4. `Status to GCS Send Time`: The time interval (in milliseconds) between Standard Telemetry packets sent by the autopilot to the Virtual Cockpit. The

Standard Telemetry stream contains basic autopilot and aircraft state information and cannot be disabled.

5. `FLC`: A 2-byte value representing the Feedback Loop Configuration. The Feedback Loop Configuration contains the status of all the PID loops (enabled and disabled). The FLC can be set using the PID Loop Control windows in the Setup window, or manually by sending a Set FLC packet (Packet 30). For bit mask information on the FLC, check "Feedback Loop Configuration" in Table 5.6.

6. `System Status`: A 2-byte value representing various Boolean flags: They are listed from low order bit. Table 5.3 shows the system status bits.

    a. Servo Bias Init: This flag is set to 1 when the autopilot has a valid set of servo trims. The servo trims are stored in nonvolatile memory so they are not lost at power down. Servo trims should be re-initialized any time the trim state of the aircraft changes. The trims are initialized using the "Copy Trims" button in the Autopilot Calibration window on the Autopilot Setup tab. The trims are read through the Bypass Circuit or from the average of the next 10 RC Controller packets received from the ground station. For more information, see the description of the Autopilot Calibration window.

    b. Pressure init: This flag indicates whether the differential and absolute pressure sensors have been calibrated. This flag is cleared on power up. The pressure sensors are calibrated using the "Calibrate Altitude" button in the Autopilot Calibration window on the Autopilot Setup tab. When this button is clicked, the autopilot stores the average of the next 50 samples of the differential and absolute pressure sensors as the biases of these two sensors. The pressure altitude is then referenced from this pressure. (This is the zero pressure altitude point.) The autopilot should not be flown until this flag is 1.

c. GPS Home Position init: This flag indicates that the autopilot has a valid GPS Home Position. (This flag defaults to zero on power up.) There are two ways to set the GPS Home Position. The first is to use the "GPS Home Position" button in the Autopilot Calibration window on the Autopilot Setup tab. When this button is clicked, the autopilot sets the Home Position as the average of the next 5 samples from the onboard GPS. The second method is to send the Home Position to the airplane using the Set GPS Home Position packet. The autopilot should only be flown if this flag is 1, as all east/north navigation commands are referenced to the GPS Home Position. The GPS Home Position is an average of the latitude, longitude, and GPS altitude.

d. GPS fix: This flag is 1 when the onboard GPS has a valid 2-D or 3-D position fix. This flag is dependent on the current state of the GPS and can change from 1 to 0 based on the current GPS fix. The aircraft should not be flown if the GPS does not have a valid fix.

e. Gyro init: This flag is 1 if the autopilot has a valid set of gyro biases. This flag defaults to 0 on power up. In most cases the gyros do not need to be calibrated before each flight, so it is reasonable to fly the autopilot if this flag is 0. If there is significant error in the attitude estimation, the gyros should be re-calibrated using the “Gyro Calibrate” Button in the Autopilot Calibration window on the Autopilot Setup tab. For more information, see the description of the Autopilot Calibration window.

f. PIC/CIC: This flag is 1 if the autopilot is in PIC mode and 0 if the autopilot is in CIC mode.

g. Data log Complete: This flag is 1 when the user-configurable autopilot data logger has finished. This flag only pertains to the custom data logger and is not important for normal flight operations. This flag is set to 0 once the log has been downloaded to the ground station. For more information, see the description of the User Configurable Data Logger.

h. Heading init: This flag is set to 1 once the autopilot has initialized the Heading component of the Attitude Estimation Filter using the GPS Heading. This occurs automatically in flight. Once the Heading has been initialized, the Attitude Estimation Filter will use the GPS Heading in heading estimation. This is a state flag and cannot be changed by the user, but can be used to verify correct heading estimation.

i. Flash write needed: This flag goes to 1 when there is flashable data in RAM. On autopilot power up, the contents of nonvolatile memory (including sensor biases, servo biases, UAV parameters, and PID gains) are copied to RAM. During normal operation, the RAM copies are used and modified. This flag indicates that the RAM copy is out of sync with the non-volatile memory. The “Write Flash” Button in the Autopilot

Calibration window on the Autopilot Setup tab will send a sync command to the autopilot. This command should not be used in the air as the autopilot cannot control the aircraft while flash is being written. Therefore, this command should only be issued after landing.

j. Bad Communications: This flag is 1 when the autopilot has not received any communication packets from the ground station for a period exceeding the parameter Lost Communications Go Home Timeout (by default, 10 seconds). When in this mode, the autopilot will attempt to fly the airplane to the GPS Home Position and orbit there. The ground station's Current Waypoint Navigation command will become 231 (en route to GPS Home Position) or 232 (orbit around Home Position). Once communications are re-established, this flag will be reset automatically. The Nav script must be reset manually by using the "Current Command" window on the waypoint tab to send a Change Current Waypoint command packet to the autopilot.

k. Low Autopilot Battery: This is 1 when the autopilot battery voltage drops below a threshold set in the UAV Parameters. This feature is not implemented at this time.

Table 5.3: System status bits.

Bit	Flag Name	Bit	Flag Name
0	Servo bias init	8	Flash write needed
1	Pressure init	9	Bad communications
2	GPS home position init	10	Low autopilot battery
3	GPS fix	11	
4	Gyro init	12	
5	PIC/CIC	13	
6	Datalog complete	14	
7	Heading init	15	

7. `Command Period`: This is the time period (in milliseconds) between executions of the Nav script. If a GPS with an update rate greater than 1 Hz is used, this period should be shortened.
8. `Fixed Throttle`: The throttle position, in percent (0 to 100), that is summed into the control effort sent to the throttle actuator. When the PID loops connected to the throttle actuator are disabled and the RC control position (if present) is 0, this value represents the commanded throttle value.
9. `Rudder Trim`: This value represents the rudder position (in radians*1000) that is summed into the control effort sent to the rudder actuator. When the PID loops connected to the rudder actuator are disabled and the RC control position (if present) is 0, this value represents the commanded rudder position.
10. `Time since last communication`: This value represents the time (in seconds) since the last Navigation Status Acknowledgment packet was received by the autopilot from the Virtual Cockpit. When this value exceeds the value of the `Lost Comm Go Home Timeout` parameter set by the user, the autopilot enters a lost communications go home state.

11. `Time Since Last Bypass`: This value represents the time (in seconds/5) since a packet was received by the autopilot from the Bypass circuit. If this value exceeds the value of the parameter `Bypass Timeout` set by the user, the autopilot switches to Computer-in-Command mode. If the timeout value is reached, the autopilot ignores future communications from the Bypass circuit.
12. `Time Since Last RC Over Comm`: This value represents the time (in seconds) since the last RC Controller packet was received by the autopilot from the Ground Station. This value is used by the autopilot when the `Use RC Over Comm UAV parameter` is set to 1 and the autopilot is in PIC mode. If these conditions are met, the autopilot uses the value of variable `Time Since Last RC over Comm` to determine the quality of the Pilot-in-the-Loop control link. If this value exceeds the `RC Over Comm Timeout` parameter, then the autopilot goes into PIC fail-safe leveling mode. In this mode, the autopilot switches to CIC mode and holds the roll angle at 0° and the pitch angle at 15°. If communication with the ground station are not re-established within the value set in the parameter `RC Over Comm Go Home Timeout` the autopilot will switch on the Nav script and the PID loops to needed to navigate and enable the Go-Home fail-safe. It will then execute waypoint command 231 which will attempt to fly the aircraft to the GPS Home Position. Once the Go-Home fail-safe has been entered, the pilot must cycle the channel 5 switch on the Futaba RC transmitter from PIC to CIC and back to PIC to put the autopilot back into PIC mode.
13. `Channel 6 Trim`: The channel 6 servo position (in radians*1000) summed into the PID loop efforts and RC controller effort that is sent to the channel 6 actuator.

When the PID loops connected to the channel 6 actuator are disabled and the RC control stick position (if present) is 0, this value represents the commanded channel 6 position. The channel 6 actuator is only available with the optional I2C servo interface.

14. `Channel 7 Trim`: The channel 7 servo position (in radians*1000) summed into the PID loop efforts and RC controller effort that is sent to the channel 7 Actuator. When the PID loops connected to the channel 7 actuator are disabled and the RC control stick position (if present) is 0, this value represents the commanded channel 7 position. The channel 7 actuator in only available with the optional I2C servo interface.

BYTE AUTOPILOT OPERATION VARIABLES:

1. `PID Tuning Enable`: This is a boolean flag representing the status of the PID tuning telemetry stream. When set to 1, the autopilot sends a PID Tuning packet (Packet 240) containing the actual, desired, and PID effort of the PID loop specified in variable `PID Tuning Loop ID`. The packet is streamed at the rate set by variable `PID Tuning Send Time`. The PID tuning telemetry stream can be configured and turned on using packet 239.
2. `PID Tuning Loop ID:` This value represents the PID loop ID used to generate the PID tuning telemetry stream. When the variable `PID Tuning Enable` is set to 1, the autopilot sends a PID Tuning packet (Packet 240) containing the actual, desired, and PID effort of the PID loop specified by this variable. The packet is

streamed at the rate set by the variable `PID Tuning Send Time`. The PID tuning telemetry stream can be turned on and configured using packet 239.

Table 5.4: PID loop ID numbers.

Loop ID	Loop Name	Loop ID	Loop Name
0	Elevator Pitch	8	Pitch Altitude
1	Elevator Pitch Rate	9	Roll Heading
2	Aileron Roll	10	Test Loop 1
3	Aileron Roll Rate	11	Test Loop 2
4	Throttle Airspeed	12	Test Loop 3
5	Rudder Yaw Rate	13	Test Loop 4
6	Rudder Heading Rate	14	Test Loop 5

3. `PIC/CIC`: This value is 1 when the autopilot is in Pilot-in-Command mode and 0 when the autopilot is in Computer-in-Command mode. This flag is set to 1 by the channel 5 switch on the RC controller (through the Bypass circuit or the ground station). If no RC controller is present, this value defaults to 0 (CIC mode).
4. `Servo Bias Init`: This variable is used to acquire the autopilot servo trims. If this variable is 1, the autopilot has a valid set of servo trims. If this value is set to 0 by the user, the autopilot will re-initialize the servo trims. (The servo trims are stored in nonvolatile memory.) Servo trims should be re-initialized any time the trim state of the aircraft changes. To initialize the servo trims, use the “Copy Trims” button in the Autopilot Calibration window in the Autopilot Setup tab. The trims are read through the Bypass circuit or from the average of the next 10 RC Controller packets received from the ground station. For more information, see the description of the Autopilot Calibration window.

5. `GPS Link Quality`: This is the GPS link quality as reported by the autopilot GPS.
6. `GPS Fix Quality`: This is the GPS fix quality as reported by the autopilot GPS. A value greater than 1 indicates a valid GPS fix.
7. `GPS Number of Satellites`: This is the number of GPS satellites acquired by the GPS receiver.
8. `GPS Home Init:` This is a boolean flag that indicates the autopilot has a valid GPS Home Position. If this flag is set to 0 by the user, the autopilot will resample the GPS Home Position from the onboard GPS receiver. The standard methods of setting the GPS Home Position are to use the "GPS Home Position" button (in the Autopilot Calibration window on the Autopilot Setup tab) or to send the GPS Home Position to the airplane using the Set GPS Home Position packet. This flag is reset to zero on power up. The autopilot should only be flown if this flag is 1 as all East/North Waypoint Navigation commands are referenced to the GPS Home Position.
9. `Attitude Reference:` This byte is used to change the heading mode in the Attitude Estimation Filter. When set to 1, the autopilot heading (ψ) is taken directly from the GPS heading. When set to 2, the autopilot heading is taken from output of the Attitude Estimation Filter. This value defaults to 2, and should be kept at 2 for normal operation.
10. `Airborne:` This is a Boolean, read-only flag that is 1 if the aircraft is airborne, and 0 if it is not. This flag is used by the autopilot in conjunction with the fail-safes. If a loss of communication is detected while the airborne flag is set, the

autopilot will enable the velocity loop, turning on the aircraft's propulsion motor. If the airborne flag is zero, the autopilot will not enable the motor. (This is a safety feature to prevent the motor from turning on the aircraft propeller inadvertently on while the aircraft is on the ground). The `Airborne` variable is set to 1 when all of the following conditions are true:

   a. GPS velocity > 3 meters/second
   b. Pressure altitude > 50 feet
   c. Airspeed > 3 meters/second

11. `Altitude AI mode`: This byte represents the state of the autopilot's altitude tracking algorithm (the Altitude Tracker). When enabled, the Altitude Tracker uses the Pitch from Airspeed, Pitch from Altitude, and Throttle from Airspeed loops to manage the aircraft's airspeed and altitude. There are 4 possible modes. The current mode is visible in the lower right hand portion of the Aircraft Attitude Indicator in the Status window. The altitude window is set in meters using the parameter `Altitude AI Window`. Great care should be taken not to inadvertently enable the Altitude Tracker on the ground as the propeller could spin up, causing injury. The Altitude Tracker is enabled in the Aircraft PID Control window in the Airplane Setup tab. The modes are as follows:
   a. Mode 0: Altitude Tracker is off; the altitude and velocity PID loops must be managed by the user.
   b. Mode 3: Altitude Hold. The altitude error (difference between the commanded altitude and the actual altitude) is less than the altitude

window. In this mode, airspeed is controlled by the Throttle from Airspeed loop. Altitude is controlled by the Pitch from Altitude loop.

   c. Mode 4: Climb mode. The aircraft is below the commanded altitude and the altitude error is greater than the altitude window. In this mode, the throttle is fixed to 100 percent, and the airspeed is controlled by the Pitch from Airspeed loop.
   d. Mode 5: Climb mode. The aircraft is above the commanded altitude and the altitude error (Difference between the commanded altitude and the actual altitude) is greater than the altitude window. In this mode the throttle is fixed to 0% and the airspeed is controlled by the Pitch from Airspeed loop.

12. `Number of Commands`: This byte represents the total number of waypoints. Waypoints are uploaded to the aircraft using the Waypoint tab in the Virtual Cockpit. The autopilot can store a maximum of 230 waypoints at once.
13. `Current Command`: This byte represents the waypoint currently being executed by the Nav script. This value only has significance when the Nav script is enabled. The waypoint numbers are internally zero-referenced. (This means that the first waypoint is command 0). For display purposes in the Virtual Cockpit, a 1 is added to the waypoint number so that the first waypoint is displayed as 1. If this byte is set to a number greater than the total number of waypoint commands, the autopilot will execute the first waypoint. To change this byte, a Set Current Command packet (52) should be used. Waypoints 0 through 230 are user-

programmable waypoints. Waypoints 231 through 250 are reserved. The following list explains the reserved waypoint commands:

a. Command 232: Navigate to the GPS Home Position at the current desired altitude or the `Default Go Home Altitude` parameter (whichever is greater), and at the `Default Go Home Airspeed` parameter.
b. Command 233: Initiate an *Orbit XY* command at the GPS Home Position at the current desired altitude or the `Default Loiter Altitude` parameter (whichever is greater) and at the `Default Loiter Airspeed` parameter. The orbit radius is set by the `Default Loiter Radius` parameter.
c. Command 234: *Orbit Now*. This command can only be executed by sending an Orbit Now command packet (Packet 50 type 2). The aircraft will orbit at the current GPS location, with a radius defined by the `Default Loiter Radius` parameter, at current desired altitude or `Default Loiter Altitude` parameter (whichever is greater), and at an airspeed defined by the `Default Loiter Airspeed` parameter. When a Continue command packet is sent to the autopilot, the Nav script will return to executing the command being executed by the Nav script prior to receiving the *Orbit Now* command.

14. `New GPS`: This is a boolean variable that indicates that the last GPS sentence received from the onboard GPS receiver was the first GPS sentence since valid GPS fix. This byte is used by the Attitude Estimation Filter to initialize the heading estimate (ψ).

15. `GPS Update`: Not used.

16. `OPT Proc`: Legacy code dealing with optical attitude estimation – not used.

17. `Opt Pitch Cal Up:` Legacy code dealing with optical attitude estimation – not used.

18. `Opt Pitch Cal Down`: Legacy code dealing with optical attitude estimation – not used.

19. `Opt Pitch Cal Right`: Legacy code dealing with optical attitude estimation – not used.

20. `Opt Pitch Cal Left`: Legacy code dealing with optical attitude estimation – not used.

21. `Restore Flash Defaults`: This Boolean flag is used to command the autopilot to restore the nonvolatile flash memory to its default state. If this flag is 1, a Commit to Flash command packet (Packet 70 type 0) will cause the nonvolatile flash memory to be overwritten with the default values set in the autopilot source code. This command should be used with caution, as it will overwrite all PID gains, temperature calibration coefficients, parameters, sensor biases, and servo trim values.

22. `RC Over Comm Bad Level`: This value represents the timeout (in seconds) before the PIC level fail-safe mode is initiated by the autopilot. This mode is designed to turn the autopilot on in the event of a Pilot-in-the-Loop lost communications scenario. In this mode, the autopilot holds the aircraft roll at 0 degrees, and aircraft pitch at 15 degrees. All of the following must occur to enter this mode:

   a. PIC mode
   b. `RC Over Comm` parameter = 1
   c. Variable `Time Since Last RC Over Comm` must be greater than parameter `RC over Comm Level Timeout`.

23. `RC Over Comm Bad Go Home:` This value represents the timeout in seconds after lost communications before the Go-Home fail-safe mode is initiated by the autopilot. In this mode the autopilot enables the appropriate PID loops and attempt to navigate the aircraft to the GPS Home Position. This mode is designed to fly the aircraft home if communication is lost during Pilot-in-the-Loop operations. Once the Go-Home fail-safe has been entered, the pilot must cycle the channel 5 switch on the RC transmitter from PIC to CIC and back to PIC to put the autopilot back into PIC mode. All of the following must occur to enter this mode:
    a. PIC mode
    b. `RC Over Comm` parameter = 1
    c. The autopilot must be in the PIC Level fail-safe.
    d. Variable `Time Since Last RC Over Comm` must be greater than the `RC Over Comm Timeout parameter`.
24. `Debug GPS`: When set to 1, the autopilot will send the GPS sentence as received from the GPS receiver over the programming port.
25. `GPS simulation`: Used to control the GPS simulation environment
    a. Mode 0: Off. GPS information is received from the onboard GPS receiver.

b. Mode 1: GPS relative east and north positions are calculated from user programmable variables. The user may change these variables to simulate GPS aircraft movement:

   i. `INERTIAL_DATA_SIM_GPS_HOME_LAT`
   ii. `INERTIAL_DATA_SIM_GPS_HOME_LON`
   iii. `INERTIAL_DATA_SIM_LAT`
   iv. `INERTIAL_DATA_SIM_LON`
   v. `INERTIAL_DATA_SIM__HEADING`
   vi. `INERTIAL_DATA_SIM_VEL`
   vii. `INERTIAL_DATA_SIM_ALT`

c. Mode 2: GPS relative east and north positions set to variables `Inertial Data Sim X` and variable `Inertial Data Sim Y`. The user may change these variables to simulate GPS aircraft movement:

   i. `INERTIAL_DATA_SIM_X`
   ii. `INERTIAL_DATA_SIM_Y`
   iii. `INERTIAL_DATA_SIM__HEADING`
   iv. `INERTIAL_DATA_SIM_VEL`
   v. `INERTIAL_DATA_SIM_ALT`

d. Mode 3: Enables rough second-order autopilot behavioral simulation. To use this mode, the GPS simulation byte must first be set to 1 to initialize the GPS Home Position to the values in the `GPS Sim Home Position` variables. Then, set this byte to 3. The pitch, roll, altitude, and heading PID loops will be simulated. This mode is useful for debugging the Nav script and for becoming familiar with ground station operation. While the autopilot is in this mode, the sensor outputs will be overwritten with simulated values.

26. `Debug IMU:` When set to 1, the autopilot will send debugging information relating to the Attitude Estimation Filter over the programming port.
27. `Debug Temperature Compensation`: When set to 1, the autopilot will send debugging information relating to sensor temperature compensation over the programming port.
28. `Debug Gyro Drift Estimation:` Not used.
29. `Ground Station Select`: When set to 0, the legacy Standard Telemetry and Navigation Status packets will be sent. This mode is used to support VC 1.0. When set to 1, the status packets will be sent in Standard International Units compatible to the Virtual Cockpit 1.1
30. `Enable Ultrasound`: This byte is used to enable or disable ultrasound. This value should be zero during normal operation. The following modes are supported:
    a. Mode 0: Ultrasound sensors are not polled.
    b. Mode 1: I2C Ultrasound sensor is polled at 10 Hz.
    c. Mode 2: Analog Ultrasound sensor is polled at 10 Hz.
31. `HIL Byte`: This byte is used to control the Hardware-in-the-Loop simulation environment. The following modes are supported as of March 2004.
    a. Mode 0: off – normal autopilot operation.
    b. Mode 1: on – HIL is enabled.

32. `Use Cross Tracking`: When enabled, this the Nav script uses cross track error in its heading computations. This byte can be set manually by the user, or set by checking the "Cross Track" check box in the Aircraft PID Control portion of the Setup tab. The Cross Track algorithm attempts to keep the aircraft on the course between waypoints. It steers the airplane to a location that is in front of the aircraft's GPS position, tangential to the desired course. The gain used to tune the cross track control algorithm is the parameter `Xtrack Hand Distance`.
33. `RPV mode`: When enabled, the roll, pitch, and throttle positions sent by the RC controller (via the Bypass circuit or the ground station) are used as commanded values for the Aileron from Roll, Altitude, and Velocity PID loops. The values are clamped according to the limits specified in the `Phi, Altitude`, and `Velocity` parameters. The aircraft must be in CIC mode for this to function.

UAV PARAMETERS WINDOW

The UAV Parameters window is used to display and modify UAV Parameters. The UAV Parameters are 32-bit floating-point variables used to govern various aircraft-specific behaviors of the autopilot. The Kestrel autopilot is capable of flying a variety of different airframes. The UAV Parameters allow the user to tune the autopilot for the airframe and operating scenario. On power up, the UAV Parameters are copied from nonvolatile flash memory to RAM. When a UAV parameter is changed by the user, the value is only changed in the autopilot RAM. To effect the change in the autopilot nonvolatile memory, use the "Update Flash" button in the Autopilot Calibration window. The "Update Flash" button should not be used in flight. The UAV Parameter list is requested from the autopilot when the Virtual Cockpit first establishes communications. After that time, the user can request the list of UAV Parameters by selecting the "File->Request all" in the ground station. The following list explains each UAV Parameter and its function.

UAV PARAMETERS

0. `Phi Lower Limit:` This value is the lower limit (in radians) on the value of the commanded roll angle. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from the RC controller in RPV mode, and the values sent to the autopilot in Joystick command packet (Packet 45). If a commanded value is under this limit, the commanded value on the autopilot will be set to the limit. This does not apply to values generated by PID loops connected to the Aileron from Roll loop.
1. `Phi Upper Limit:` This value is upper limit (in radians) on the value of the commanded roll angle. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from the RC controller in RPV mode, the values sent to the autopilot in Joystick command packet (Packet 45). If a command value is sent that exceeds this limit, the command will be set to the limit. It does not apply to values generated by PID loops connected to the Aileron from Roll loop.
2. `Theta Lower Limit`: This value is lower limit on the value of the commanded pitch angle. This limit is in radians. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from

the RC controller in RPV mode, the values sent to the autopilot in Joystick command packet (Packet 45). If a command value is sent that exceeds this limit, the command will be set to the limit. It does not apply to values generated by PID loops connected to the Elevator from Pitch loop.

3. `Theta Upper Limit`: This value is upper limit on the value of the commanded pitch angle. This limit is in radians. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from the RC Controller in RPV mode, the values sent to the autopilot in Joystick command packet (Packet 45). If a command value is sent that exceeds this limit, the command will be set to the limit. It does not apply to values generated by PID loops connected to the Elevator from Pitch loop.
4. `Alt Lower Limit`: This value is lower limit on the value of the commanded altitude in meters. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from the RC controller in RPV mode, the values sent to the autopilot in Joystick command packet (Packet 45), and the values commanded in waypoints. If a commanded value is sent that exceeds this limit, the command will be set to the limit.
5. `Alt Lower Limit:` This value is upper limit on the value of the commanded altitude in meters. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from the RC controller in RPV mode, the values sent to the autopilot in Joystick command packet (Packet 45), and the values commanded in waypoints. If a commanded value is sent that exceeds this limit, the command will be set to the limit.

6. `TAS Lower Limit`: This value is lower limit on the value of the commanded velocity in meters / second. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from the RC controller in RPV mode, the values sent to the autopilot in Joystick command packet (Packet 45), and the values commanded in waypoints. If a commanded value is sent that exceeds this limit, the command will be set to the limit.
7. TAS Upper Limit: This value is upper limit on the value of the commanded velocity in meters / second. This limit applies to values commanded by the user in the Desired/Actual window of the Status window, the values sent from the RC controller in RPV mode, the values sent to the autopilot in Joystick command packet (Packet 45), and the values commanded in waypoints. If a commanded value is sent that exceeds this limit, the command will be set to the limit.
8. `Default Loiter Radius`: This is the orbit radius in meters used in the following scenarios.
    a. *Orbit Now* Command
    b. *Land Now* Command
9. `Default Flair Height`: This is the flair height in meters used by the Nav script. When the aircraft descends to this pressure altitude during a *Land Now* command, the Nav script commands a 0 roll angle to level the plane in an attempt to minimize airframe damage at impact. The parameter `Default Flair Height` is used in the following scenario:
    a. *Land Now* command

10. `Waypoint Radius`: The distance in meters used by the Nav script to determine if the aircraft has arrived at a waypoint. When the aircraft is within the circle scribed by this radius, the Nav script executes the next command. If this value is set too small, the aircraft may have difficulty reaching a waypoint and may circle the waypoint continually in an attempt to reach it.
11. `Altitude AI Window:` This is the size (in meters) of the error window that governs the mode switching of the Altitude Tracker. When the altitude error is less than this value, the Altitude Tracker switches to the altitude hold mode.
12. `Landing Throttle`: This is the throttle value in percent sent to the throttle actuator during a *Land* command. At times it may be desirable to set this to a nonzero value to flatten the landing glide slope. If this value is set to high, the aircraft may not be able to achieve a negative rate of decent. Parameter `Landing Throttle` is used In the following scenarios:
    a. *Land Now* command
    b. *Land* command
13. `Default Landing Velocity:` This is the velocity in meters/second commanded by the autopilot during a *Land Now* command. The parameter `Default Landing Velocity` is used in the following scenario:
    a. *Land Now* command
14. `Take Off Throttle`: This is the throttle value in percent sent to the throttle actuator during a *Take Off* command. At times it may be desirable to set this to a value below 100 percent to minimize battery drain during climb out. If this value

is set to low, the aircraft may not be able to achieve a positive climb rate during take off. Parameter `Take Off Throttle` is used In the following scenario:

   a. *Take Off* command

15. `Take Off Pitch`: This is the pitch angle commanded to the Elevator from Pitch loop by the Nav script during the first segment of a *Take Off* command. The aircraft is commanded to this pitch angle until the aircraft airspeed reaches the value set in the `Rotate Velocity` parameter. Once this airspeed is achieved the Nav script switches to the climb out segment of the *Take Off* command. Parameter `Take Off Pitch` is used in the following scenario:

    a. *Take Off* command

16. `Rotate Velocity`: When the aircraft airspeed reaches this value during a *Take Off* command, the Nav script switches to the climb out segment of the *Take Off* command. During the climb out segment, the throttle is held at the `Take Off Throttle` parameter and the airspeed and altitude are controlled by the Altitude Tracker. The heading is controlled by the Orbit function.
17. `Lost Communications Go Home Timeout`: This is the timeout in seconds used by the lost communications detection routine. If the autopilot does not receive a Navigation Status Acknowledgment packet from the Virtual Cockpit in the period of time set by this variable, the autopilot enters a lost communications go home state (Go-Home fail-safe). In this state, autopilot will enable the Nav script and attempt to navigate the aircraft to the GPS Home Position.
18. `Climb Velocity`: Not Used.
19. `Decent Velocity`: Not Used.

20. `Default Loiter Velocity`: This is the orbit radius in meters used in the following scenarios.
    a. *Orbit Now* command
    b. *Home Orbit* command issued by the Go-Home fail-safe
21. `Default Loiter Altitude`: This is the orbit altitude in meters used by the Nav script during an *Orbit Now* command or a *Home Orbit* during a lost communication scenario. If the current commanded altitude is greater than the parameter `Default Loiter Altitude`, the current commanded altitude is used in place of the parameter `Default Loiter Altitude`. The `Default Loiter Altitude` parameter is considered when it is greater than the current commanded altitude in the following scenarios.
    a. *Orbit Now* command
    b. *Home Orbit* command issued by the Go-Home fail-safe
22. `Default FLC`: The lower two bytes of this field are used as the power up `Feedback Loop Configuration variable`(See variable FLC). The bit masks to program these values are shown in Table 5.6.

Table 5.6: FLC bit mask selection.

Value - Controller	Source	Bit Index	Mask
Throttle	Fixed	1-0	0x0000
	Airspeed		0x0001
	Altitude		0x0002
Elevator	Fixed	3-2	0x0000
	Pitch		0x0004
	Pitch Rate		0x0008
	Pitch + Pitch Rate		0x000C
Ailerons	Fixed	5-4	0x0000
	Roll		0x0010
	Roll Rate		0x0020
	Roll + Roll Rate		0x0030
Rudder	Fixed	7-6	0x0000
	yawrate		0x0040
	Heading Rate		0x0080
Pitch & Pitch Rate	Fixed	10-8	0x0000
	Airspeed		0x0300
	Altitude		0x0400
Roll & Roll Rate	Fixed	11	0x0000
	Heading		0x0800

23. `Effort Reversal`: This is a low level control variable used to reverse the sign on the control efforts of the inner PID loops connected to the aileron, elevator, rudder, and throttle actuators. When the bit mask value is 1, the corresponding control effort is multiplied by -1. This value should be used for development only. If the control directions are incorrect for the aircraft, the `Servo Scalar` parameters signs should be negated instead of using parameter `Effort Reversal`. Table 5.7 shows the bit masks that correspond to each PID loop.

Table 5.7: Effort reversal bit masks.

Actuator	Bitmask
Aileron	0x0001
Elevator	0x0002
Throttle	0x0004
Rudder	0x0008

24. `Servo Mixing`: This value controls the various servo mixing functions on the autopilot. To enable a particular mixing modes use the following bit masks. When the bit mask value is 1, the corresponding mixing is enabled. When 0, the corresponding mixing is disabled. Currently the only servo mixing available is elevon mixing.

Table 5.8: Servo mixing bit mask.

Mixing Type	Bit Mask
Elevon	0x0001

25. `Aileron Scalar`: This is the conversion factor from radians of aileron deflection to servo Pulse Position Modulation (PPM) high time values. The servo PPM value representing the aileron deflection is then added to the aileron servo bias and sent to the aileron servo actuator. This value is based on the mechanical linkages between the aileron actuator and the aileron control surfaces. The sign on this value should be set such that the Aileron from Roll and Aileron from Roll Rate loops move the ailerons in the correct direction. For example if the aircraft is commanded into a left bank in CIC mode, the aileron actuator should move the aileron in the direction that will cause the aircraft to bank left. This value can be

determined using an oscilloscope to measure the pulse width on the aileron servo and a protractor to measure the corresponding aileron control surface movement in radians.

26. `Elevator Scalar:` This is the conversion factor from radians of elevator deflection to servo Pulse Position Modulation (PPM) high time values. The servo PPM value representing the elevator deflection is then added to the elevator servo bias and sent to the elevator servo actuator. This value is based on the mechanical linkages between the elevator actuator and the elevator control surfaces. The sign on this value should be set such that the Elevator from Pitch and Elevator from Pitch Rate loops move the elevator in the correct direction. For example if the aircraft is commanded into pitch up condition in CIC mode, the elevator actuator should move the elevator in the direction that will cause the aircraft to pitch up. This value can be determined using an oscilloscope to measure the pulse width on the elevator servo and a protractor to measure the corresponding elevator control surface movement in radians.
27. `Throttle Scalar`: This is the conversion factor from percent (0 to 100) of throttle actuator deflection to servo Pulse Position Modulation (PPM) high time values. The servo PPM value representing the throttle deflection is then added to the throttle servo bias and sent to the throttle servo actuator. This value is based on the mechanical linkages between the throttle actuator and the throttle control surfaces. The sign on this value should be set such that the Throttle from Velocity loop moves the throttle actuator in the correct direction. For example if the aircraft is commanded to increase velocity in CIC mode, the throttle actuator

should move the throttle in the direction that will cause the aircraft throttle increase. This value can be determined using an oscilloscope to measure the pulse width on the throttle servo and a protractor to measure the corresponding percentage of carburetor opening (in the case of a glow or gas engine).

28. `Rudder Scalar`: This is the conversion factor from radians of rudder deflection to servo Pulse Position Modulation (PPM) high time values. The servo PPM value representing the rudder deflection is then added to the rudder servo bias and sent to the rudder servo actuator. This value is based on the mechanical linkages between the rudder actuator and the Rudder control surfaces. The sign on this value should be set such that the Rudder from Yaw Rate and Rudder from Heading Rate loops move the rudder in the correct direction. For example if the aircraft is commanded a negative heading rate (left turn) in CIC mode, the rudder actuator should move the rudder in the direction that will cause the aircraft to achieve a negative rate of change of heading. This value can be determined using an oscilloscope to measure the pulse width on the rudder servo and a protractor to measure the corresponding rudder control surface movement in radians.
29. `UAV INITS:` Not Used.
30. `Bypass Time Out`: The timeout in seconds/5 used to determine that there is no Bypass circuit present. If the autopilot does not receive any communication from the Bypass circuit for the time specified by this variable then future packets from the Bypass circuit will be ignored. I the parameter `Use RC Over Comm is` set to 0, the autopilot will also enter CIC mode.

31. `Minimum Throttle`: This is the minimum throttle value in percent (0 to 100) that can be commanded by an autopilot PID loop. Values generated by PID loops that are below this value will be set to this value. This is to prevent a PID loop from inadvertently killing a gas or glow engine in flight. The throttle actuator can be set to values below this value by using the variable `Fixed Throttle`.
32. `Use RC Over Com`: This boolean value enables or disables the fail-safe algorithms associated with Pilot-in-the-Loop control using the RC transmitter connected to the ground station. When set to 1, the PIC fail-safes are enabled. There are two PIC fail-safes: Level, and Go Home. See the Fail-safe section of the autopilot manual for further explanation. This value should be set to 0 when the Pilot-in-the-Loop control using the ground station hardware is not used. This value should also be set to zero when a Bypass circuit is plugged in.
33. `RC over Comm Level Timeout`: The timeout value in seconds before the autopilot enters the PIC Level fail-safe mode. In this mode, the autopilot switches itself to CIC mode and holds a 0 degree roll angle and 15 degree pitch angle. See the fail-safe section of this manual for more information. This timer is only active under the following circumstances:
    a. `RC Over Comm` parameter = 1
    b. The autopilot is in PIC mode.
34. `RC over Comm Go Home Timeout`: The timeout value in seconds before the autopilot enters PIC Go-Home fail-safe mode. See the fail-safe section of this manual for more information. This timer is only active under the following circumstances:

    a. `RC Over Comm` parameter = 1
    b. The autopilot is in PIC mode.

35. `Differential Pressure Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for the differential pressure sensor, and should be determined for each autopilot individually.
36. `Absolute Pressure Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for the absolute pressure sensor, and should be determined for each autopilot individually.
37. `Roll Gyro Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for the Roll Gyro, and should be determined for each autopilot individually.
38. `Pitch Gyro Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for the Pitch Gyro, and should be determined for each autopilot individually.
39. `Yaw Gyro Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for the Yaw Gyro, and should be determined for each autopilot individually.
40. `X Accelerometer Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for the X Accelerometer, and should be determined for each autopilot individually.
41. `Y Accelerometer Temperature Compensation Coefficient:` This value is used to generate a temperature correction factor for the Y Accelerometer, and should be determined for each autopilot individually.

42. `Z Accelerometer Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for the Z Accelerometer, and should be determined for each autopilot individually.

43. `Optional Port A Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for an optional analog device connected to Optional Analog Input A. This coefficient is autopilot specific and should be determined for each autopilot individually.

44. `Optional Port B Temperature Compensation Coefficient:` This value is used to generate a temperature correction factor for an optional analog device connected to Optional Analog Input B. This coefficient should be determined for each autopilot individually.

45. `Optional Port C Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for an optional analog device connected to Optional Analog Input C. This coefficient should be determined for each autopilot individually.

46. `Optional Port D Temperature Compensation Coefficient`: This value is used to generate a temperature correction factor for an optional analog device connected to Optional Analog Input D. This coefficient should be determined for each autopilot individually.

47. Default Go Home Altitude: This is the Go Home Altitude (in meters) used by the Nav script during a Go Home Now command or a GPS Go Home Command initiated by a Lost Communications scenario. If the current commanded altitude

is greater than the Default Go Home Altitude, the current commanded altitude is used in place of the Default Go Home Altitude.

48. `Default Go Home Velocity`: This is the velocity (in meters/second) commanded by the autopilot during a *Go Home Now* command, or when the Go-Home fail-safe is enabled.

49. `Gyro Zero Temperature`: The temperature reading, in analog-to-digital converter (ADC) units, used to calculate temperature compensation values for the P, Q, and R rate gyros. The value is read by the autopilot when a Calibrate Gyros or Calibrate All command packet is received, and should not be edited directly by the user.

50. `Level Accelerometer Zero Temperature`: The temperature reading (in ADC units) used to calculate temperature compensation values for the X and Y accelerometers. This value is read by the autopilot when a Calibrate Level Accelerometers command packet is received, and should not be edited directly by the user.

51. `90 Accelerometer Zero Temperature`: The temperature reading (in ADC units) used to calculate temperature compensation values for the Z accelerometer. This value is read by the autopilot when a Calibrate 90 Accelerometer command packet is received, and should not be edited directly by the user.

52. `Orbit Angle Gain:` This value is used by the orbit algorithm to calculate a control effort from the heading error. When orbiting, the aircraft should follow a circular path centered on the GPS orbit point, with a specified orbit radius. The heading error is the difference between the current heading (ψ) of the airplane

and the heading tangential to the circular path. The `Orbit Angle Gain` parameter tunes the aircraft's ability to fly a circular path when orbiting, and has no effect on the actual radius of the orbit. When tuning the orbit algorithm, this gain should be tuned first, with the `Orbit Position Gain` parameter set to 0.

53. `Orbit Position Gain`: This value is used by the orbit algorithm to calculate a control effort from the cross track error. (The cross track error is the distance between the aircraft's current position and the position tangential to the aircraft on the circumference of the orbit path.) This gain tunes the aircraft's ability to orbit at a fixed distance from the GPS orbit point. This is gain should be set to zero when tuning the `Orbit Angle Gain` parameter. It should only be set after the `Orbit Angle Gain` parameter is tuned.
54. `Orbit Radius`: This value is not used. The orbit radius is determined from the waypoint command currently being executed by the Nav script.
55. `Orbit Phi Trim`: This is the steady-state output of the orbit algorithm. The units are radians if specifying roll angle, or radians/second if specifying heading rate. This is the output commanded by the orbit algorithm when there is no cross track error and no tangential angle error. This value should be set to the approximate roll angle or rate of turn to achieve the desired orbit radius. The more accurate this value is, the better the orbit algorithm will work.
56. `Maximum Orbit Roll`: This is absolute value of the saturation block on the output of the orbit algorithm. If the orbit algorithm calculates a control effort that exceeds this value, the control effort will be set to this value. This value should

be set such that the maximum performance specifications of the airframe are not exceeded.

57. `Maximum Cross Track Error`: This value limits the contribution of the cross track error into the control effort of the orbit algorithm. This value should be set such that when the aircraft is far from the orbit point, the significant cross track error will not influence the output of the orbit algorithm. If this value is too small, the orbit algorithm will have difficulty minimizing the orbit cross track error.
58. `Orbit Clockwise`: This determines the direction of the orbit algorithm. Bit 0 (bit mask 0x0001) determines the orbit direction. When set to 0, the aircraft will orbit counter clockwise. When set to 1, the aircraft will orbit clockwise.
59. `PID Derivative Corner Frequency`: This value sets the 3 dB corner frequency (in radians/second) of the Alpha filter used on the derivative portion in the PID loop calculations. This value is used for development purposes only.
60. `Loss of Lift Gain`: This value is the gain used by the lift compensation algorithm, which reduces the altitude loss in turns due to the aircraft's bank angle. This is accomplished by generating elevator deflections based on the estimated roll angle (ϕ) of the aircraft. These elevator deflections are summed into the PID control effort, the RC controller offset, and the elevator trim, which are fed to the elevator actuator. The parameter `Loss of Lift Gain` should be set to 0 during PID gain tuning. It should only be set to a non-zero value once the user is familiar with the closed-loop flight characteristics of the aircraft.
61. `Cross Track Hand Distance`: This value is used by the Nav script cross track algorithm. The parameter `Cross Track Hand Distance` is the distance

between the aircraft's position and the steer point. Smaller values will cause the autopilot to fly the course more closely. This value only has an effect when the cross track algorithm is enabled.

62. `Waypoint Navigation Configuration`: This value is used to configure the Nav script. Bit zero is the LSB. The following list shows the function of each bit:
    a. Bit 0: When set to 0, the Nav script will not command altitude and velocity. When set to 1, the Nav script will command the desired altitude and velocity for each waypoint.
    b. Bit 1: Determines the orbit output mode. When set to 0, the orbit algorithm produces a roll angle to be fed into the Aileron from Roll loop. When set to 1, the orbit algorithm produces a heading rate to be fed into the Rudder from Heading Rate loop. Aircraft without ailerons should always use the heading rate mode.
    c. Bit 2: Determines the output for en route navigation. When set to 0, the Nav script produces a heading based on the bearing to the current waypoint. When set to 1, the Nav script produces a heading rate based on the bearing to the current waypoint.

AUTOPILOT CALIBRATION WINDOW

The Autopilot Calibration window contains buttons to manage the nonvolatile flash memory, and to calibrate the autopilot sensors, actuators, and GPS. All buttons turn red when clicked, indicating that the command has been sent to the autopilot. When the autopilot finishes executing the command, it sends a command acknowledgment packet. When the Virtual Cockpit receives this packet, the button color will change back to grey. Figure 5.11 shows a close-up of the Autopilot Calibration window. The following list contains a description of each button.

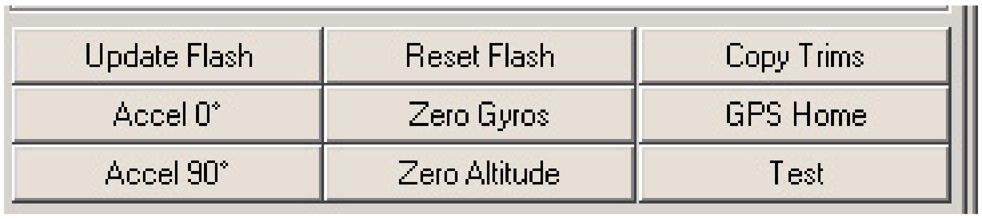


Figure 5.11: Autopilot Calibration window.

1. Update Flash: The "Update Flash" button sends a Commit to Flash command packet to the autopilot. The autopilot copies a selected set of operating variables to the nonvolatile portion of memory. On power up, the autopilot copies the variables from nonvolatile memory to RAM, where they can be changed. All changes are lost when the autopilot is powered down unless the "Update Flash" button is clicked prior to disconnecting the power. Any time a change is made to any of the variables backed up in the nonvolatile memory, the "Update Flash" button will turn red, indicating the RAM is out of synchronization with the nonvolatile memory. The "Update Flash" button should not be clicked while the aircraft is in flight. The following variables are copied to nonvolatile memory when the "Update Flash" button is clicked:
    a. PID gains
    b. UAV Parameters

    c. Sensor Biases

    d. Actuator Trims

2. Accel 0: The "Accel 0" button sends a command to calibrate the X and Y accelerometers. The command is designed to capture the biases of these accelerometers in an orientation in which their sensing axes are perpendicular to the force of gravity (the aircraft is level). When the autopilot receives the command, it collects 50 samples each from the X and Y accelerometers and stores the averages as the X and Y accelerometer biases. These biases will be subtracted from all future X and Y accelerometer readings such that only the actual X and Y accelerations are measured. The following steps should be followed to calibrate these accelerometers. (1) First, with the autopilot mounted correctly in the aircraft, power on the autopilot and allow it to warm up for at least 5 minutes. (2) Second, orient the aircraft so it is level. (3) Third, click the "Accel 0" button. The aircraft must be level and still during this procedure. When the button turns back to grey, the procedure is finished, and the autopilot may be moved. The biases from this calibration may be written to nonvolatile memory by clicking the "Update Flash" button. The accelerometers do not need to be calibrated before each flight. They only need to be calibrated if the autopilot installation is changed, or the temperature changes significantly. A faulty calibration of the X or Y accelerometers can cause incorrect roll or pitch estimations. If the roll or pitch estimates are incorrect even after the gyros are re-calibrated, then the X and Y accelerometers should be re-calibrated.

3. Accel 90: The "Accel 90" button sends a command to calibrate the Z accelerometer. The command is designed to capture the bias of the Z accelerometer in an orientation where the sensing axis is perpendicular to the force of gravity (the aircraft is at 90° pitch or roll). When the autopilot receives the command, it collects 50 samples from the Z accelerometer and stores the average as the Z accelerometer bias. The bias will be subtracted from all future Z readings such that only the actual Z acceleration is measured. The following steps should be followed to calibrate the Z accelerometer. (1) First, with the autopilot mounted correctly in the aircraft, power on the autopilot and allow it to warm up for at least 5 minutes. (2) Second, hold the aircraft at 90° roll or pitch in any direction. (3) Third, with the aircraft held still at 90°, click the "Accel 90" button. When the button turns back to grey, the procedure is finished, and the autopilot may be moved. The bias from this calibration may be written to nonvolatile memory by clicking the Update Flash button. The Z accelerometer should not need to be calibrated before each flight. It should only need to be calibrated if the autopilot installation is changed, or the temperature changes significantly. A faulty calibration of the Z accelerometer can cause incorrect roll or Pitch estimations for larger values of pitch and roll. If the roll or Pitch estimates appear inaccurate when the aircraft is held at angles between 40° and 75°, then the Z accelerometer may need to be re-calibrated.
4. Reset Flash: The "Reset Flash" button sends a command to the autopilot to restore the nonvolatile memory to the default values compiled into the source code. Use caution, as this command will set all PID gains, servo actuator trims,

and UAV parameters (including temperature drift correction coefficients) to their default states. This button is mainly used for debugging purposes.

5. Zero Gyros: The “Zero Gyros” button sends a command to the autopilot to calibrate the P, Q, and R rate gyros. When the autopilot receives the command, it collects 50 samples each from the P, Q, and R rate gyros and stores the average as the P, Q, and R gyro biases. The biases will be removed from all future rate gyro readings such that only the actual rotational rates are measured. Rate gyros drift with temperature and time, so it is a good idea to calibrate them periodically. If the pitch and roll are off by more than 5°, re-calibrating the rate gyros is suggested. The following steps should be followed to calibrate the rate gyros. (1) First, with the autopilot mounted correctly in the aircraft, power on the autopilot and allow it to warm up for at least 5 minutes. (2) Second, set the aircraft down, such that it is not moving. (3) Third, click the “Zero Gyros” button. When the button turns back to grey, the procedure is finished, and the autopilot may be moved. The bias may be written to nonvolatile memory by clicking the “Update Flash” button.
6. Zero Altitude: The “Zero Altitude” button sends a command to the autopilot to calibrate the differential and absolute pressure sensors. When the autopilot receives the command, it collects 50 samples each from the differential and absolute pressure sensors and stores the average as the differential and absolute pressure sensor biases. The biases will be removed from all future pressure readings. This command is important in that the relative pressure altitude on the autopilot is referenced from this point. As the atmospheric pressure changes day

by day, it is important to re-calibrate the pressure sensors before each flight. The following steps should be followed to calibrate the pressure sensors. (1) First, with the autopilot mounted correctly in the aircraft, power on the autopilot and allow it to warm up for at least 5 minutes. (2) Second, click the "Zero Altitude" button. When the button turns back to grey, the procedure is finished, and the autopilot may be moved. The bias may be written to nonvolatile memory by clicking the "Update Flash" button.

7. Copy Trims: The "Copy Trims" button sends a command to the autopilot to store the current stick position as the level flight actuator trim position. When the autopilot receives this command, it averages the next 20 RC Stick Position packets from the RC controller (via the Bypass circuit or the ground station). The averaged value is stored as the servo trim position for servo actuator channels 1-8. These channels normally correspond to the aileron, elevator, throttle, rudder, ch6, and ch7 actuators. Because the actual pulse width for the center position of each servo is different, and every airframe has a slightly different trim condition, the autopilot needs this trim value to generate the correct servo pulses. To do this, the Bypass circuit must be plugged into the autopilot or the RC controller must be plugged into the ground station. The following steps should be followed to copy trims. (1) First, fly the aircraft in Pilot-in-Command mode using the RC controller. (2) Once satisfied that the trims are set correctly for hands-off level flight, click the "Copy Trims" button. It is important that there are no pilot stick inputs during the next few seconds (or at least 1 second after the button color changes back to green), as they will distort the averaged trim value. Because it

can take a few seconds to capture the trim values, it may be easier to land the aircraft and then copy the trim values. Once the trim values are copied, they should be written to the nonvolatile memory using the "Update Flash" button.

8. GPS Home: The "GPS Home" button sends a command to store the current value of the autopilot as the GPS Home Position. All relative east/north positions and GPS altitude are referenced from the GPS Home Position. The GPS Home Position is gathered automatically, immediately after GPS Lock is acquired. It is a good idea to re-acquire the GPS Home Position once the GPS position has stabilized. Latitude, Longitude, and GPS altitude are averaged from five GPS samples to form the Home Position. (The GPS Home Position may also be set using the Update GPS Home Position packet.) Because the Home Position averages 5 GPS samples, it is important to not move the airplane, or to click the "GPS Home" button twice within the same five second period. This will cause significant error in the relative east, north, and GPS altitude values. The GPS Home Position should be updated before each flight.
9. Test: Does nothing.

5.3 AUTOPILOT SYSTEM INSTALLATION

The autopilot, pitot system, GPS, and radio modem must be installed correctly in the aircraft to function properly. The purpose of this section is to guide the user through the installation process.

5.3.1 INSTALLING THE AUTOPILOT

The autopilot must be installed flat on its bottom with the front forward in the direction of flight. It is important that it be mounted in an area that does not experience significant vibrations from the motor. It is also advisable to place foam padding around it for protection in the event of a crash or hard landing. Figure 5.12 shows the autopilot installed in the airframe.

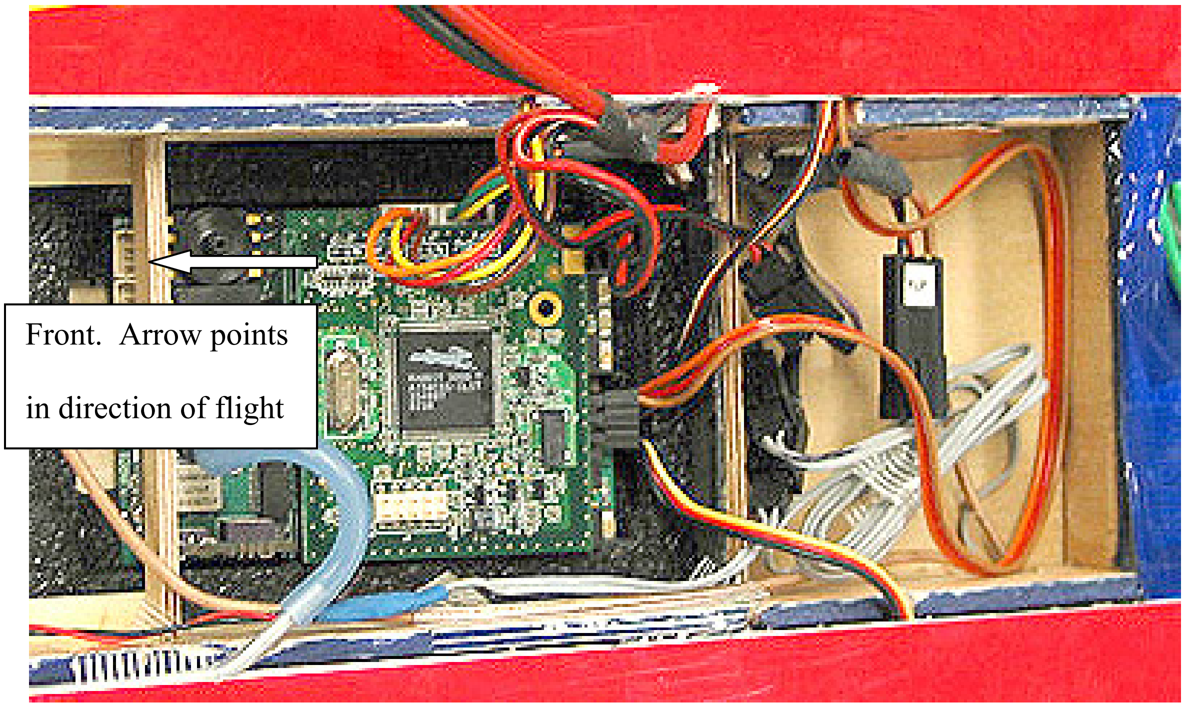


Figure 5.12: Autopilot installed in the aircraft.

5.3.2 INSTALLING THE GPS

The GPS receiver is the most important autopilot sensor. Without GPS reception, the aircraft cannot estimate its position or heading. Loss of GPS reception during flight can result in the loss of the aircraft, as the autopilot cannot navigate.

The GPS unit should be mounted such that the antenna has a clear "view" of the sky. It should be as far as possible from radio transmitting devices such as the radio modem, or video transmitting equipment that could potentially interfere with satellite reception. The GPS receiver performance can be improved significantly by adding a ground plane under the antenna. A ground plane 2 inches by 2 inches will noticeably improve performance, although a larger plane is better. The GPS receiver is connected to the autopilot via a 4-wire cable with MOLEX connectors on either end. Pin 1 is ground, Pin 2 is power, Pin 3 is autopilot Tx, and Pin 4 is autopilot Rx. The default protocol is Furuno Binary at 9600 baud, 3.3 volt TTL. If another GPS receiver is to be used, the autopilot source code must be modified. Once installed, the GPS receiver should be tested to verify that it can receive at least 5 satellites with a clear view of the sky. Items that could potentially cause interference, such as the radio modem and video transmitter, should be turned on during the test. Figure 5.13 shows an example GPS installation with a copper ground plane.

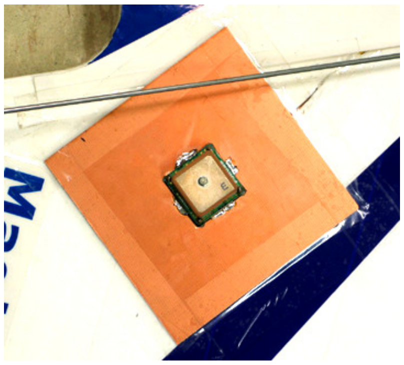


Figure 5.13: Sample GPS installation and ground plane.

5.3.3 INSTALLING THE RADIO MODEM

The digital radio modem is a key element of the autopilot system. Without the digital modem, there is no way to communicate with the aircraft in flight. Any 115k baud capable TTL level digital modem can be used with the autopilot, but the standard modem is the Aerocomm ac4490-500. This modem is lightweight and, with a decent antenna, can achieve ranges greater than 10 kilometers. Because the modem emits RF, it should be mounted as far as possible from radio receiving devices such as the RC receiver or GPS receiver. Figure 5.14 shows an example radio modem installation.

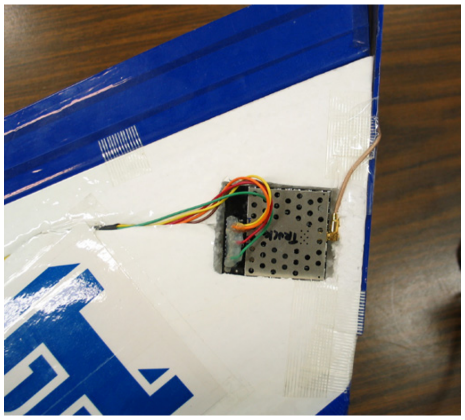


Figure 5.14: A sample modem installation.

If the ac4490 modem is used, it should be programmed with the following parameters:

- Baud rate: 115200 bps
- Client/Acknowledge mode
- Auto destination
- Tx retries: 7

- Power output (15 on the -500 model, 60 on the -200 model)
- Full Duplex

If the ac4490-500 is used, ensure that the autopilot modem voltage jumper is set to 3.3 volts, or the modem will be damaged. The jumper should be set to the 5.0 volt position for the ac4490-200. An effort should also be made to keep the coax between the modem and antenna as short as possible (ideally, less than four inches) to prevent signal attenuation. A simple dipole antenna can be constructed that is lightweight and provides good radiation characteristics. A suitable dipole is shown in Figure 5.15. The dipole antenna should be mounted vertically, and away from any conducting surfaces.

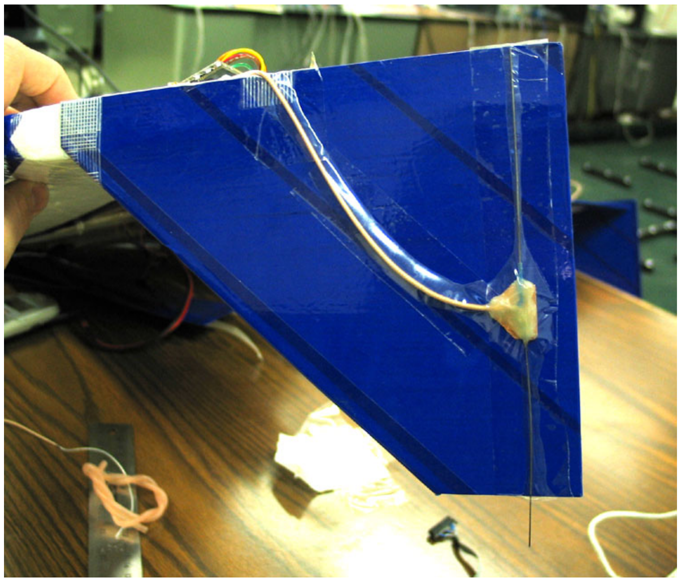


Figure 5.15: A suitable dipole antenna installed in the winglet of a flying wing UAV.

5.3.4 INSTALLING THE PITOT SYSTEM

The pitot system is critical for correct airspeed and altitude readings. The altitude port can be left open if the autopilot is mounted in an area where the pressure will not change significantly during flight. If more accurate altitude measurements are desired, the static port can be connected to a static port on the outside of the aircraft. The airspeed measurement is derived from a differential pressure sensor that measures the difference between static pressure and the pressure from the forward airspeed. The pressure port on the differential pressure sensor should be connected, using silicon fuel tubing, to a rigid tube at least 1/16 inch in diameter. This tube should be mounted such that the open end is in the path of the forward air stream in the direction of flight. Care should be taken not to mount the tube in an area that will be exposed to the propeller wash or significant turbulence; the airspeed tube should protrude at least 1 inch in front of the airframe to ensure it is clear of boundary layer turbulence. Figure 5.16 shows a suitable pitot tube installation. The airspeed sensor can be tested for functionality by lightly blowing into the pitot tube and noting the airspeed indicator in the Virtual Cockpit.

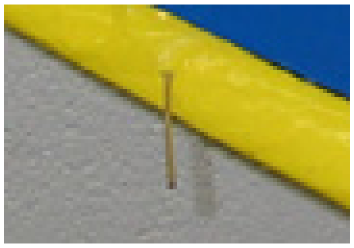


Figure 5.16: Pitot tube installation in a flying wing aircraft.

5.4 AUTOPILOT SETUP

The autopilot must be configured correctly before attempting autonomous flight for the first time. The following guide is intended to lead the user through this process. Most crashes involving the autopilot occur during the first few flights, and are due to incorrect autopilot setup, installation, or calibration. If all the steps in this manual are followed, there is a much higher chance of successful flight.

5.4.1 CONTROL SURFACE SETUP

The aircraft control surfaces should be configured using a standard RC receiver and transmitter before being flown for the first time. Program the RC transmitter for the correct control surface deflections and directions. (The RC controller also needs to be programmed for the desired control surface mixing.) Ideally, the aircraft will then be test flown, without the autopilot equipment installed, to determine proper trim settings. Once the control surfaces are set up for Pilot-in-Command flight, the autopilot and Bypass circuit should be connected to the RC receiver. If ground station hardware is used that supports Pilot-in-Command flight via the digital communications link, a Bypass circuit is not needed.

As a first step, the autopilot must be set up for the desired mode of Pilot-in-Command RC control. The two methods are a Bypass circuit or an RC controller plugged into the ground station. If the Bypass circuit is used, the parameter `Use RC Over Comm` must be set to 0. If the ground station will be used, the parameter `Use RC Over Comm` should be set to 1.

In step 2, switch the RC controller to PIC mode (switch toward pilot on most models) and verify smooth control deflection in the proper directions. If a direction is reversed, it should be fixed in the RC controller.

Step 3 is to set up the autopilot for the correct servo mixing. The parameter `Servo Mixing` should be set according to Table 5.9.

Table 5.9: Servo mixing flag.

Mixing Type	Bit Mask
Elevon	0x0001

Step 4 is to configure the autopilot for the correct control surface directions. This is accomplished in 5 steps:

1. With the RC controller sticks in neutral positions and the throttle set in the down position, click the "Copy Trims" button in the Autopilot Calibration window. This will generate an initial set of trim values in the autopilot.
2. Disable the following PID loops in the Servo Control and Altitude Control windows:
    a. Elevator from Pitch Rate
    b. Elevator from Pitch
    c. Aileron from Roll Rate
    d. Aileron from Roll
    e. Rudder from Yaw Rate
    f. Rudder from Heading
    g. Throttle from Airspeed
    h. Altitude Tracker (uncheck "Auto")
3. Place the autopilot in Computer-in-Command (CIC) mode by toggling the channel 5 switch on the RC controller toward the pilot. Verify the autopilot is in CIC mode by checking the autopilot mode in the upper right portion of the Attitude Indicator in the Virtual Cockpit.

4. As no PID loops are enabled, the servos should be in almost the same positions as they were in PIC mode. Verify correct control surface movement by moving each of the control surfaces using the RC controller and noting the correct movement of the corresponding control surface(s). The control surfaces should move smoothly and in the correct directions. If the ground station is used to transmit the RC control signals, there may be a slight latency in the controls. This latency should not exceed 0.25 seconds. This is a ballpark estimate.
5. Using the Pre-Flight window in the Virtual Cockpit, program the autopilot for correct control effort direction. The autopilot should be in CIC mode and the RC controller sticks should be in their neutral positions, with the throttle stick in the down position. As no PID loops are enabled, the Servo Sliders should be centered, and the Throttle slider all the way to the left. Set the autopilot control effort directions using the following steps:
    a. Move the aileron stick to the left (the aileron slider should move to the left). If it moves to the right, reverse the sign on the parameter `Aileron Scalar` in the Plane Setup tab. Move the aileron stick to the right; the aileron slider should move to the right.
    b. Move the elevator stick down (nose up), and the elevator slider should move to the left. If it moves to the right, reverse the sign on the parameter `Elevator Scalar`. Move the elevator stick up (nose down); the elevator slider should move to the right.

c. Advance the throttle stick to full throttle. The throttle slider should move to the right. If it does not, reverse the sign on the parameter `Throttle Scalar`.

d. Move the rudder stick to the left, the rudder slider should move to the left. If it moves to the right, reverse the sign on the parameter `Rudder Scalar`. Move the rudder to the right; the rudder slider should move to the right.

e. If the any UAV Parameters were changed, use the "Write to Flash" button to commit the changes to nonvolatile memory.

The control surface directions and mixing are now set up. These directions should be checked before each flight using the abbreviated procedures outlined in the Pre-Flight instructions.

5.4.2 SENSOR CALIBRATION AND TESTING

A control system is only as good as its sensors. For this reason, great care should be taken to ensure the sensors are functioning correctly and are properly calibrated. The sensors should be tested with the autopilot installed in the aircraft and the pitot system installed.

1. Differential Pressure Sensor: The differential pressure is used by the autopilot to determine airspeed. To test this sensor:
   a. Zero the differential pressure sensor using the "Calibrate Pressure" button.
   b. Lightly blow into the Pitot tube and verify a change in measured airspeed in the Actual/Desired window.

2. Absolute Pressure Sensor: The absolute pressure is used by the autopilot to measure relative pressure altitude. To test this sensor:
    a. Zero the absolute pressure sensor using the "Calibrate Pressure" button
    b. Connect a tube to the differential pressure sensor and create a vacuum by lightly sucking on the tube. The Altitude indicator in the Actual/Desired window should show a significant increase in altitude.
3. Rate Gyros: The P, Q, and R rate gyros are extremely important sensors. They sense the rotational rates in the aircraft body coordinate system. The rate gyro readings are used by the Attitude Estimation Filter and many PID loops. Because the rate gyros are integrated, it is important that they be calibrated correctly. Even the slightest bias offset caused by movement during calibration can lead to significant errors in the estimated attitude of the aircraft. To test the rate gyros:
    a. Power on the autopilot and let it warm up for 5 minutes.
    b. Calibrate the gyros using the "Zero Gyros" button in the Virtual Cockpit.
    c. Roll Gyro (P):
        i. In the Autopilot Operational Variable window inside the Plane Setup tab, check the "Request Value" box to the right of variable `Roll Gyro No Drift Compensation`. The Virtual Cockpit will request the output of the roll gyro several times per second.
        ii. Gently roll the autopilot to the right, and verify a positive rate.
        iii. Gently roll the autopilot to the left, and verify a negative rate.
        iv. Set the autopilot down and verify that the value has returned to approximately zero.

d. Repeat the same procedure for the pitch gyro (Q). A positive rate should occur when the autopilot is pitched in the nose up direction.

e. Repeat the same procedure for the yaw gyro (R). A positive rate should occur when the autopilot is rotated right along the yaw axis.

4. Accelerometers: The X, Y, and Z accelerometers measure the accelerations along the body fixed X, Y, and Z axes. They are used by the Attitude Estimation Filter. To test the accelerometers:

    a. Power on the autopilot and let it warm up for 5 minutes.

    b. Use the procedure outlined in the Autopilot Calibration window section to calibrate the X, Y, and Z accelerometers.

    c. In the Plane Setup tab, check the "Request Value" box to the right of variable `Phi Accelerometer`. The Virtual Cockpit will request the output of the roll estimate produced by the X and Z accelerometers several times per second.

    d. Gently tilt the autopilot to the right to produce a positive value for variable `Phi Accelerometer`.

    e. Gently tilt the autopilot to the left to produce a negative value for variable `Phi Accelerometer`.

    f. Hold the autopilot level; the value for variable `Phi accelerometer` should be approximately zero.

    g. Repeat steps c through f for variable `Theta Accelerometer`. Tilt the autopilot in the pitch axis. A positive pitch angle should produce a positive value for variable `Theta Accelerometer`.

5. Attitude Estimation Filter: The Attitude Estimation Filter is one of the most critical components of the autopilot. It produces estimates for roll, pitch, and heading, which are used by the inner PID loops to stabilize the aircraft. If the attitude is not estimated correctly, the autopilot cannot control the aircraft. To test the Attitude Estimation Filter:
    a. Power on the autopilot and let it warm up for 5 minutes.
    b. Place the aircraft in which the autopilot is mounted on a level surface.
    c. Use the procedure outlined in the Autopilot Calibration window section to calibrate the X, Y, and Z accelerometers.
    d. Use the procedure outlined in the section on the Autopilot Calibration window to calibrate the rate gyros.
    e. With the aircraft level, the Attitude Indicator in the Virtual Cockpit should indicate level. The pitch and roll values in the Desired/Actual window should be within a few degrees of level.
    f. Roll the aircraft right 45° and verify that the actual value for the roll angle is approximately 45°.
    g. Roll the aircraft left 45° and verify that the actual value for the roll angle is approximately -45°.
    h. Pitch the aircraft 45 ° nose up and verify that the actual value for the pitch angle is approximately 45°.
    i. Pitch the aircraft 45° nose down and verify that the actual value for the pitch angle is approximately -45°.

j. Yaw the aircraft 90° to the right and verify that the heading increases by 90°.

k. Yaw the aircraft 90° to the left and verify that the heading decreases by 90°.

l. Note that, as the heading drift is controlled by the GPS heading, the heading estimate may drift when there is no GPS lock or the aircraft is not moving.

m. Return the aircraft to a level attitude and verify that the pitch and roll angles return to within a few degrees of zero. If not, re-calibrate the X and Y accelerometers using the "Cal Accel 0" button.

5.4.3 INITIAL PID LOOP SETUP

The PID loops are responsible for the stabilization and control of the aircraft. They produce control efforts using the difference between commanded aircraft states and current aircraft state information produced by the sensors, GPS, and Attitude Estimation Filter. The control efforts are sent to the servo actuators, which control the aircraft. Because the correct PID gains are a function of the airframe, each aircraft will have a slightly different set of gains. The gains should be set to approximate values before the first flight. If the user is familiar with the airframe and the sensitivity of the control surfaces, an approximate set of gains can be determined empirically. Otherwise, the approximate gains should be determined by using a simulation environment such as MATLAB. The PID gains should then be systematically tuned during the first flight. This is done by enabling the inner PID loops one at a time, and moving the aircraft in the appropriate axis. The gains are adjusted until the magnitudes of the control surface deflections appear reasonable. To verify that loops and control surface deflections are set correctly:

1. Place the aircraft on a level surfaced with the autopilot powered on.
2. Enable the PID loops one at a time to check the control effort surface deflection directions. (The autopilot should be in CIC mode and the RC controller sticks should be in their neutral positions, with the throttle stick in the down position.) To check the deflection directions:
    a. Ailerons:
        i. Enable the Aileron from Roll loop in the PID Loop Setup window.
        ii. Enter a commanded roll angle of 30 degrees in the Desired/Actual window in the Virtual Cockpit.
        iii. Verify that the ailerons move in the correct direction for a right bank.
        iv. Enter a commanded roll angle of -30 degrees.
        v. Verify the ailerons move in the correct direction for a left bank.
        vi. Disable the Aileron from Roll loop.
    b. Elevator:
        i. Enable the Elevator from Pitch loop.
        ii. Enter a commanded pitch angle of 30 degrees in the Desired/Actual window in the Virtual Cockpit.

iii. Verify that the elevator moves in the correct direction to cause the aircraft to pitch up.

iv. Enter a commanded pitch angle of -30 degrees.

v. Verify that the elevator moves in the correct direction to cause the aircraft to pitch down.

vi. Disable the Elevator from Pitch loop.

5.4.4 WAYPOINT NAVIGATION SCRIPT SETUP

Several airframe dependent parameters must be configured for Nav script to function correctly. The first is the parameter `Waypoint Radius`. The faster the aircraft flies, the larger this number needs to be. The default of 30 meters works well for aircraft flying in the range of 8 to 15 meters/second. The next parameter `Waypoint Navigation Configuration`, governs the commanded altitude and velocity during waypoint navigation. If bit 0 (Bit Mask 0x0001) of this parameter is set to 1, then the Nav script sets the commanded altitude and velocity according to the values programmed in the waypoint when navigating waypoints. If that bit is set to 0, the commanded altitude and velocity values in the Desired/Actual window will be used. Bit 1 of this parameter determines the method used by the Nav script to control the aircraft. The control type can be set independently for waypoint navigation and the orbit routine. If bit 1 (Bit Mask 0x0002) of the parameter `Waypoint Navigation Configuration` is set to 1, then the Nav script will control the aircraft in orbit using the Rudder from Heading Rate loop (use this if the aircraft uses only a rudder or V-tail for heading control). If that bit is set to 0, the Roll from Heading loop will be used (use this if the aircraft has ailerons). If bit 2 (Bit Mask 0x0004) of parameter `Waypoint Navigation Configuration` is set to 1, then the Nav script will control the aircraft between waypoints using the Rudder from Heading Rate loop. If that bit is set to 0, the Roll from Heading loop will be used.

5.4.5 WAYPOINT NAVIGATION SCRIPT PID CONFIGURATION

The Nav script produces desired headings or heading rates based on the bearing to the next waypoint. In order for the aircraft to fly these bearings certain PID loops must be enabled by the user. The following loops must be enabled in the PID Loop Setup window:

1. Elevator from Pitch
2. Elevator from Pitch Rate
3. Aileron from Roll
4. Aileron from Roll Rate
5. Altitude Tracker

The Nav script is enabled and disabled using the "Waypoints" check box in the Heading Control window.

5.4.6 AIRCRAFT LIMITS

The autopilot has user-definable limits to help prevent damage to the airframe. There are limits on values commanded by the user in the Actual/Desired window and limits on the values commanded by the Nav script and PID loops. These limits should be set by the user based on the performance limits of the airframe and mission parameters.

These values should be stored in the nonvolatile memory using the "Update Flash" button.

USER COMMANDED LIMITS

The following is a list of limits on the values commanded by the user in the Actual/Desired window. If the user commands a value outside the limit, the autopilot will set the commanded value to the limit.

1. Parameter `Phi Upper` – This is the upper limit on the roll angle (radians).
2. Parameter `Phi Lower` – This is the lower limit on the roll angle (radians).
3. Parameter `Alt Upper` – This is the upper limit on the altitude (m).
4. Parameter `Alt Lower` – This is the lower limit on the altitude (m).
5. Parameter `TAS Upper` – This is the upper limit on the airspeed (m/s).
6. Parameter `TAS Lower` – This is the lower limit on the airspeed (m/s).

SCRIPT COMMANDED LIMITS

The following is a list of limits on the values commanded by the Nav script.

1. Parameter `Max Roll Angle`: This value is determined by the effort limit on the Roll from Heading loop. This can be set in the PID Gains window. This value is used by the autopilot when navigating between waypoints or flying user-commanded headings.
2. Parameter `Max Orbit Angle`: This is the maximum roll commanded by the orbit algorithm. This value is used during orbits.

5.4.7 FAIL-SAFE BEHAVIOR

The autopilot is programmed with routines designed to protect the aircraft during unusual circumstances caused by equipment or communication failure. These behaviors should be set up and tested prior to each flight. There are 2 fail-safe routines: Lost Communications Go Home (Go-Home fail-safe) and RC-Over-Comm Lost Communication (PIC fail-safe).

LOST COMMUNICATIONS GO HOME

The Go-Home fail-safe, will attempt to fly the aircraft home if communications are interrupted for a specified amount of time. When enabled, the aircraft will enable the Elevator and Aileron PID loops, enable the Altitude Tracker, and turn on the Nav script. The Nav script will be sent 2 waypoint commands. The first is a *Goto XY* command with the coordinates (0, 0). The second is an *Orbit XY* command with the same coordinates. The aircraft commanded altitude will be set to the current desired altitude or the altitude specified by parameter `Default Go Home Altitude`, whichever is greater. The commanded velocity during is determined by the parameter `Default Go Home Velocity`. Once the aircraft arrives at the 0, 0 coordinate, it will execute the orbit command. The orbit altitude and velocity are determined by the parameters `Default Orbit Altitude and Default Orbit Velocity`.

The Go-Home fail-safe is only enabled when the aircraft has GPS lock. The routine is turned on when the aircraft cannot communicate with the ground station for the period determined in parameter `Lost Communications Go Home Timeout` (specified in seconds). Once the fail-safe has been enabled, it must be turned off manually by the user by changing the current waypoint in the Current Command window. To test this script, simply unplug the ground station to simulate a communications failure. When the ground station is plugged back in and communication re-established, the aircraft should be in the correct mode to fly home and the current waypoint should be the *Go Home* waypoint number 230 or the *Home Orbit* waypoint 231.

RC-OVER-COM LOST COMMUNICATION

This fail-safe is designed to protect the aircraft during Pilot-in-Command mode when the RC control commands are sent over the digital communication link. If the Bypass circuit is used for Pilot-in-the-Loop control, or Pilot-in-Command mode is not used, this fail-safe should be disabled. If Pilot-in-the-Loop control is used via the RC controller plugged into the ground station, this fail-safe should be enabled. To enable, set the parameter `Use RC over Comm` to 1. There are two timers used for this fail-safe. The timers are enabled reset when the autopilot receives a valid RC control packet from the ground station. When the timers reach their set points, the corresponding fail-safe is enabled. The first set point is governed by the parameter `RC Over Comm Level Timeout`. If no RC control packets are received for the length of time in seconds specified by this parameter, the autopilot enables the PIC level fail-safe. In this mode, the autopilot attempts to hold the aircraft level by switching itself to Computer-in-Command mode and commanding 10 degrees of pitch and 0 roll angle. The second timer is governed by the parameter `RC Over Comm Go Home Timeout`. If no RC control packets are received for the length of time in seconds specified by this parameter, the autopilot enables the PIC Go-Home fail-safe. In this mode, the autopilot attempts to fly the aircraft to the GPS Home Position by switching to Computer-in-Command mode and enabling the Nav script. The Nav script begins executing the *Go Home* waypoint command (waypoint number 230). After the *Go Home* command is completed, the autopilot executes the *Home Orbit* command (waypoint number 231). To exit this mode, the user must switch to CIC then back to PIC using the Autopilot Mode Switch on the RC controller. To test these fail-safe modes, place the autopilot in PIC mode and turn off the ground station to simulate a loss of communication.

5.5 FIRST FLIGHT

At this point, the following items should be completed:

1. Autopilot installed
2. Control surfaces tested
3. Sensors calibrated
4. PID loop gains set
5. Nav script set up
6. Fail-safes tested

If these items have been completed according to this manual, then it is time for the first flight with the autopilot. The purpose of the first flight is to capture a trim setting for the aircraft, and to verify the sensors are functioning correctly. The following procedure should be followed for the first flight.

1. Pre-flight the aircraft according to the Pre-flight guide
2. Launch the aircraft in Pilot-in-Command mode.

3. Once the aircraft has reached a safe altitude use the RC controller to maneuver the aircraft to test the sensors. The following maneuvers are a good starting point:
    a. Shallow bank while maintaining altitude and airspeed - verify the attitude indicator corresponds to the bank angle. When the aircraft is returned to level, the attitude indicator should return to within a few degrees of level.
    b. Fly the aircraft at various pitch angles and verify the attitude indicator tracks.
    c. Fly the aircraft at several different airspeeds and verify the airspeed indicator tracks.
    d. Fly the aircraft at various heading and verify the indicated heading reads correctly.
    e. Fly the aircraft at several different altitudes and verify the indicated altitude tracks.
    f. Make sure the GPS position updates and the GPS ground speed is reasonable.
4. Trim the aircraft for hands-off level flight.
5. Land the aircraft and with the RC controller still on, use the "Copy Trims" button, followed by the "Update Flash" button to store the trim values in the non-volatile memory.
6. The aircraft is ready for gain tuning.

5.6 SECOND FLIGHT SESSION

The purpose of this flight session is to tune PID gains, test the Nav scripts, and test the fail-safes. The fail-safes and waypoint navigation rely on the PID loops functioning correctly. Thus, it is important to tune the gains as soon as possible.

5.6.1 TUNING THE PID GAINS

The gains should be tuned one loop at a time with a competent RC pilot standing by to take control of the aircraft, should it go unstable. The basic procedure is that the aircraft is given a standard pre-flight and then launched in Pilot-in-Command Mode. Once the aircraft is flying at a safe altitude, the PID tuning can begin. The proportional gain should be tuned first with the derivative and integral gain set to 0. The proportional gain should be set low to begin with and slowly increased until instabilities are introduced. The gain should then be reduced and/or derivative gain added until the instabilities disappear. Integral gain should be added cautiously to minimize steady state error. Many loops can be tuned each flight. The pilot may take control by placing the aircraft in PIC mode while the appropriate loops are configured on the autopilot. The pilot then gives control to the autopilot by switching to CIC mode. The loops should be tuned using standard PID tuning techniques. The inner loops should be tuned first in the following order:

1. Elevator from Pitch Rate only
2. Elevator from Pitch and Pitch Rate (Adjust the Pitch gains only)
3. Aileron from Roll Rate only
4. Aileron from Roll and Roll Rate (Adjust the Roll gains only)

5. Rudder from Yaw Rate
6. Rudder from Heading Rate
7. Throttle from Airspeed

Once the inner loops are tuned, the outer loops should be tuned in the following order:

1. Pitch from Altitude
2. Pitch from Airspeed
3. Roll from Heading

Once the outer loops are tuned, the Altitude Tracker may be tested. This done by checking the "Auto" check box in the Aircraft Control tab. At this point, the autopilot is ready to test the Nav script.

5.6.2 TESTING THE NAVIGATION SCRIPT AND ALTITUDE TRACKER

The Nav script relies on the PID gains to be properly set. The following procedure was developed using the Zagi THL airframe. It should be modified appropriately for the airframe. The aircraft should be given a standard pre-flight first. To test the Nav script, use the following procedure:

1. Place the autopilot in PIC mode.
2. Upload two *Goto XY* waypoint command:
    a. Waypoint 1:
        i. Altitude: 75 m
        ii. Airspeed: 12 m/sec
        iii. East: 150 m

        iv. North: 50 m

    b. Waypoint 2:

        i. Altitude: 75 m

        ii. Airspeed: 12 m/sec

        iii. East: 50 m

        iv. North: 50 m

3. Enable the following PID loops:

    a. Elevator from Pitch and Pitch Rate

    b. Aileron from Roll and Roll Rate

    c. Altitude Tracker enabled

4. Enable the Nav script using the "Waypoints" check box.

5. Verify the autopilot is navigating to the first waypoint using the Navigation Status Screen in the waypoint tab. The following items should be checked:

    a. GPS position is approximately 0,0 indicating the GPS position has not drifted significantly from the GPS Home Position

    b. Current waypoint = 1

    c. Desired East/North = 150,50

    d. Desired Altitude (Desired/Actual window) = 75 m

    e. Desired Airspeed (Desired/Actual window) = 12 m/sec

    f. Waypoint Navigation Mode = 2 (indicating *Goto XY*)

    g. Bearing to waypoint is correct

    h. Distance to waypoint is approximately 158 m

6. Launch the aircraft under Pilot-in-Command mode.

7. Pilot the aircraft to an altitude of approximately 75 m and switch the autopilot to CIC mode. Move the throttle on the RC controller to the throttle off position.
8. Verify that the autopilot turns the aircraft toward the first waypoint and the distance to waypoint is decreasing.
9. Verify the aircraft is maintaining 75 m altitude and 12 m/sec airspeed.
10. Once the aircraft reaches the first waypoint, it should reverse direction and fly to the second waypoint. Once the second waypoint is reached, it will return to the first waypoint.
11. Now add the following third waypoint command:
    a. Waypoint type: *Orbit XY*
    b. Altitude: 75 m
    c. Airspeed: 12 m/sec
    d. East: 50 m
    e. North: 50 m
    f. Radius 35 m
    g. Time: 0
12. Upload the waypoints. The aircraft should immediately fly to the first waypoint. Verify this in the Navigation Status window.
13. Once the aircraft reaches the third waypoint, it will begin to orbit.
14. The orbit should be tuned using the following UAV Parameters:
    a. Orbit Angle Gain
    b. Orbit Position Gain

15. Once the orbit is smooth and circular, change the orbit time to 45 seconds and upload the waypoints again.

To test that the Altitude Tracker is functioning correctly edit the waypoints so they are at different altitudes and airspeeds. The altitude tracker should maintain the altitude and airspeed between waypoints

5.6.3 TESTING THE FAIL-SAFES

The final step in preparing the aircraft for autonomous flight is testing the fail-safes. Only the Go-Home fail-safe will be discussed at this point. First test the fail-safes on the ground using the procedure outlined in the section on configuring the fail-safes. To test the fail-safes, use the procedure outlined in steps 1-10 of Testing the Waypoint Navigation Script. Once the aircraft is flying between the two waypoints, turn off the ground station by removing power to the ground station hardware. Within the number of seconds specified in the parameter `Lost Comm Go Home Timeout`, the aircraft should turn towards the GPS Home Position. Turn on the ground station again. Verify the following in the Navigation Status window:

1. Current waypoint 230 or 231
2. Nav Mode: 2 or 3
3. Desired East/North (0,0)

Exit the fail-safe mode by entering 1 into the Current Waypoint window. The aircraft should return to navigating between waypoints 1 and 2.

5.7 PRE-FLIGHT

The autopilot is a complex system of sensors and software which must be setup and calibrated to function correctly. Because of the risks associated with UAV flight, it is important to carefully test the critical components of the system before each flight. It is much easier to detect and fix a small problem before flight, then pick up the pieces after a crash. The purpose of this section is to outline a pre flight procedure that is quick, and thorough. This procedure should be carefully followed and checked off a step at a time before each UAV flight.

5.7.1 POWER-UP

1. Power on the laptop computer and load the Virtual Cockpit software.
2. Power on the RC controller
3. Set the throttle stick to 0 throttle
4. Switch the channel 5(landing gear) to the PIC position(away from the pilot)
5. Power on the ground station hardware.
6. Power on the autopilot
7. Verify communications with the ground station by checking the Communications Indicator in the upper right hand portion of the attitude indicator.
8. Verify the autopilot is in PIC mode by checking the Autopilot Mode indicator in the upper left hand portion of the attitude indicator.
9. Place the aircraft in a location where the GPS receiver has a clear view of the sky so that it may start gathering satellites for GPS lock.

5.7.2 CONTROL SURFACES

With the autopilot in PIC mode check the control surfaces for proper trim and movement. Use the RC controller to move each control surface independently while checking the direction of deflection and the travel. With the sticks centered and throttle cut, verify the control surfaces are in their trim position and the Servo Position Sliders are centered. The throttle slider should be positioned to the far left. Check the following control surfaces:

1. Aileron
2. Elevator
3. Throttle
4. Rudder

5.7.3 GPS

Verify the GPS has a valid 3-D fix. The GPS indicator on the attitude indicator will turn green when the GPS has a 3-D fix. Do not attempt to fly the aircraft with fewer than four acquired satellites. Once the GPS has a position lock, verify the following information on the GPS line of the Navigation Status window.

1. The GPS relative position within should be within 20 m of 0,0. If the GPS has drifted more than this amount, re-acquire the GPS Home Position using the "GPS Home Position" button.
2. The reported GPS longitude and GPS latitude for the aircraft is correct.

5.7.4 SENSORS

1. Place the aircraft on the ground in a level attitude.
2. Calibrate the pressure sensors with the “Calibrate Pressure” button.
3. If the roll angle or pitch angle are more than 5 degrees from level, re-calibrate the rate gyros with the Calibrate Gyros button.
4. Slowly bank the aircraft approximately 30 degrees to the right and than to the left and verify the roll angle reads accordingly. Positive roll angle is to the right.
5. Slowly pitch the aircraft approximately 30 degrees nose up (positive angle) and than 30 degrees nose down (negative angle) and verify the pitch angle reads correctly.
6. Slowly yaw the aircraft clockwise (positive angle change) and than counterclockwise (negative angle change), and verify the heading changes accordingly. The heading angle does not need to align with the magnetic heading of the earth. The autopilot will calibrate its heading with the GPS heading after launch.
7. Lightly blow into the pitot tube and verify a sharp increase in airspeed. The airspeed should return to zero.
8. Verify the altitude reads approximately zero.
9. Re-calibrate the sensors as needed. If the roll or pitch angles read more than 5 degrees from zero even after re-calibrating the rate gyros, the accelerometers may need to be re-calibrated.

5.7.5 WAYPOINT PROGRAMMING

1. Create a flight plan using the Waypoint Editor.
2. Upload the flight plan to the aircraft.
3. Enable Waypoint Navigation by checking the “Waypoints” check box and clicking “Upload.”
4. Verify the Nav script is active and navigating to the correct waypoint. Do this using the following steps.
    a. Verify the correct waypoint navigation status in the attitude indicator.
    b. Verify that the current waypoint is set to waypoint 1 in the Current Waypoint window.
    c. Verify the correct relative waypoint coordinates in the Navigation Status window.
    d. Verify the distance to waypoint is correct in the Navigation Status window.

5.7.6 PID LOOPS

Enable the PID loops necessary for waypoint navigation in the Setup window.

1. Elevator from Pitch
2. Elevator from Pitch Rate
3. Aileron from Roll
4. Aileron from Roll Rate

5. Altitude Tracker ON (Caution: Enabling the Altitude Tracker while in CIC mode may cause the propeller to spin up. Make sure the autopilot is in PIC mode before enabling.)

5.7.7 AUTOPILOT COMMANDS

The Nav script produces a commanded heading, altitude, and velocity. The altitude and heading loops produce a commanded roll and pitch angle. Before launch note these commands so that you will have an idea of how the aircraft will behave once control is given to the autopilot. These commanded values are visible in the Desired/Actual window. Pay special attention to the following commands.

1. Desired Altitude
2. Desired Airspeed

5.7.8 LAUNCH

Prior to launching the aircraft do a final check of the following:

1. Battery Voltage
2. Number of GPS satellites
3. Waypoint Navigation Mode
4. Current waypoint
5. PID loops

If those check out correctly use the following procedure to launch the aircraft:

1. With your fingers clear of the propeller, flip to CIC mode and verify the propeller spins up.

2. Flip back to PIC mode.
3. Launch the aircraft and fly it to a safe altitude under manual control.
4. With the aircraft airborne, do another check of the following:
    a. Battery Voltage
    b. Number of GPS Satellites
    c. Waypoint Navigation Mode
    d. Current waypoint
    e. Desired Altitude
    f. Desired Airspeed
    g. Distance to waypoint
    h. Roll angle
    i. Pitch angle
    j. Altitude
    k. Airspeed
5. If the above parameters are correct, enable the autopilot by flipping to CIC mode. Immediately cut the throttle on the RC controller.
6. As the aircraft begins to turn towards the first waypoint and climb to the desired altitude, pay special attention to the following:
    a. Does the actual airspeed match the desired airspeed?
    b. Is the aircraft climbing to the desired altitude?
    c. Is the distance to the waypoint getting smaller as the aircraft flies towards the waypoint?

7. If there is a problem, re-enable manual control of the aircraft by switching to PIC mode, and diagnose the problem.

5.8 IN-FLIGHT MONITORING

The autopilot is designed to provide a high level of autonomy and neglect tolerance. However, to insure the safety of the aircraft and people on the ground, the aircraft should be carefully monitored while in-flight. There are several important indicators of autopilot performance and aircraft health. These indicators can alert the user of potential problems so the appropriate measures can be taken. The following list contains items that should be monitored every few seconds:

1. Battery Voltage. If the voltage drops below 6 volts, the autopilot will not function. The battery voltage should never be allowed to approach that point.
2. Altitude. Is the aircraft tracking altitude?
3. Airspeed. Is the aircraft tracking airspeed?
4. Number of GPS satellites. This value should always be greater than five. If GPS lock is lost, the aircraft will not be able to navigate.
5. Distance to waypoint. As the aircraft navigates towards the current waypoint, this number should be decreasing.
6. Pitch. Pitch should stay between 0 and 15 degrees during level flight.
7. Roll. Is roll staying within the limits set in the Roll from Heading PID loop and the parameter `Orbit Phi Trim`?

5.9 LANDING

The aircraft can be landed using the Land command or in PIC mode. To land the aircraft, simply upload a *Land* waypoint command. After the aircraft lands, use caution when retrieving the aircraft as it may enter a lost communication fail-safe mode which could cause the propeller to turn on. This can happen when the landing area is not in line of sight with the ground station, causing a loss of communication with the ground station. To land the aircraft manual in PIC mode, flip to PIC mode and land the aircraft. Keep the autopilot in CIC mode until the power is removed to ensure the propeller does not spin up.

APPENDIX

APPENDIX A

ZAGI AIRCRAFT DYNAMICS COEFFICIENTS

Mass (slug)	0.0369018
I_xx (slug-ft^2)	0.0208188
I_yy (slug-ft^2)	0.00449063
I_zz (slug-ft^2)	0.1252959
I_xz (slug-ft^2)	0.00000
Reference chord (ft)	0.833333
Reference span (ft)	4.00000
Reference area (ft^2)	3.33333
CD reference speed (ft/s)	19.6850
Baseline alpha_0 (rad)	0.00000
Baseline Cm_0 at alpha_0	0.1100395
Baseline CL_0 at alpha_0	0.132896
Upper stall limit CL	1.10000
Lower stall limit CL	-0.800000
Profile CD	0.025000
CD Re-scaling exponent	-0.500000
Lift slope	4.08115
Pitch stability	-0.422680
Sideforce due to sideslip	-0.107990
Roll due to sideslip	-0.0524900
Yaw stability	0.0213716
Lift due to pitch rate	4.90714
Pitch damping	-1.68929
Sideforce due to roll rate	-0.0385900
Roll damping	-0.425470
Yaw due to roll rate	-0.000882313
Sideforce due to yaw rate	.0595832
Roll due to yaw rate	0.0442800
Yaw damping	-0.0122373
Lift due to elevator	0.491440
Pitch due to elevator	-0.210653
Sideforce due to rudder	0.00000
Roll due to rudder	0.00000
Yaw due to rudder	0.00000

Sideforce due to aileron	0.0204900
Roll due to aileron	0.133270
Yaw due to aileron	-0.00725753
Effective span	0.85
Deformation drag	0.02
Drag due to aileron deflection	0.04
Drag due to elevon deflection	0.04

APPENDIX B

COMMUNICATION PROTOCOL

PACKET DEFINITIONS

General Packet		
Index	Var.	Description
0	0xFF	1
1	cmd/data	Descriptor
…	…	
n-1	cksum	
n	0xFE	

Descriptors			
Value	Symbol	Description	arg type
1	ACK	Acknowledge	
2	NAK	Negative ACK	
10	Set / Response	PID	float
11		Sensor	float
12		misc float	float
13		misc int	int
14		misc byte	unsigned char
15		uav limit (flashable)	float
20	Get	PID	float
21		Sensor	float
22		misc float	float
23		misc int	int
24		misc byte	unsigned char
25		uav limit (flashable)	float
30	set	Set Feedback loop config.	int
34	Recal universal	zero universal	
35	Recal	Zero sensors(p,q,r,v,h,gps)	

36		zero servos (copy servo trim)
38		zero accel x,y (@level)
39		zero accel z (@90)
40	set/response	servo trims
41	get	servo trims
50	set/response	cmd upload
51	get	cmd download
52	set	change currently executing cmd
70	set	commit uav limits to flash
180	set	rc stick position packet
201	get	get datalog packet
202	set	start new datalog
230	set	set desired values
237	get	get PID 4 value set
238	set	set PID 4 value set
239	PID tuning	PID tunning On/Off select
240		PID tunning pkt.
248	telem	Nav/GPS telem
249		Standard telemetry
250	set	digitial output toggle

Descriptor Number	**Specific Packet Definition**	
Index	Value	Description

1-2	**ACK, NAK** - recognition of last message	
2	lm_descr	Descriptor to which responding

10-11	**Set / Response** - Used to set PID / Sensor values or in response to value requests.	
2	struct_id	Struct identifier
3	index	Value index

4		Value (USB)
5	value	Value
6		Value
7		Value (LSB)

12-15	**Set / Response** - Used to set values or in response to value requests.	
2	index	Array index
3		Value (USB)
	value	
n		Value (LSB)

20-21	**Get** - Used to request PID / Sensor values.	
2	struct_id	Struct / Array Identifier
3	index	Value Index

22-25	**Get** - Used to request values.	
2	index	Array index

26	**request standard telemetry / responds with standard telem packet**

27	**request nav telemetry - responds with nav telem packet**

28	**request code version/ response from autopilot - responds with string of length n**	
2	byte	number of bytes
3-n		string containing version

30	**FLC** - Set feedback loop configuration.	
2		Value (lower byte)
3		Value (upper byte)

34	**Recal** - universal - Zero sensors. (general, servos,accel x,y, accel z)	
2	byte	type 0=pres,1=gyro,2=ac 0,3=ac 90,4=gps ,5=servos

35-39	**Recal** - Zero sensors. (general, servos,accel x,y, accel z)

none…

40	**Set / Response** - servo trims - used to set the servo bias or tirms - value in uS high time		
2	int	ch 1 - ail	int US
4	int	ch2 - elev	int US
6	int	ch3 - thr	int US
8	int	ch4 - rud	int US
10	int	ch6	int US
12	int	ch7	int US

43	**Set GPS home pos**	
2	int	lat degrees
4	float	lat minutes
8	int	lon degrees
12	float	lon minutes
16	float	altitude

41	**Get Servo Trims -** requests servo trims

none…

45	**Send Joystick cmds (roll,pitch,throttle)** governed by phi,theta,min throttle uav_limits		
2	signed byte	roll (radians -1.65 rad -> 1.65 rad)	byte*75
3	signed byte	pitch (radians -1.65 rad -> 1.65 rad)	byte*75
4	unsigned byte	throttle (percent 0->100)	%

50	**Command Set / Response** - Used to upload commands to autopilot and to and get command response.	
2	byte	command type
3	byte	command number
4	byte	total number of commands
5…	depends on command	

51	**Get command -** requests command from autopilot	
2	byte	command number

52	**Set Current Command-** used to change the currently executing command on the autopilot	
2	byte	command number

53	**Edit Command-** used to edit a specific command on the on the autopilot (0-231)	
2	byte	cmd type
3	byte	command number
4…	depends on command	

70	**Commit to Flash-** comiits uav flashable limits to flash	
2	byte	0=normal write 1=restore default write

200-202	**Datalog Packets -** see datalog section
none…	

230	**set desired!**	
2	float	desired pitch
6	float	desired roll
10	float	desired heading
14	float	desired alt
16	float	desired vel
20	float	desired turn rate

231	**set desired - Indexed**	
2	byte	0=Alt 1=vel 2=roll 3=pitch 4=heading 5=turnrate

232	**set desired universal -**variable length packet based on byte 2, send values in same order as bitmask is listed. Only include values that are 1 on bitmask	
2	byte	(bit: 0=pitch, 1=roll,2=heading, 3=alt, 4=vel, 5=turn rate)
3	float	desired Pitch
7	float	desired Roll
11	float	desired Heading
15	float	desired alt
19	float	desired vel
23	float	desired turn rate

237	**PID 4 Value Get** - Used to request Kp, Kd, Ki, and effort limit values of a specified controller.	
2		PID struct number (control block)

238	**PID 4 Value Set** - Used to set Kp, Kd, Ki, and effort limit values of a specified controller.		
2	byte	PID struct number (control block)	
3	int	Proportional	int * 10000
5	int	Derivative	int * 10000
7	int	Intergral	int * 10000
9	int	Effort Limit	int

238	**PID 4 Value Set** float - Used to set Kp, Kd, Ki, and effort limit values of a specified controller.	
2	byte	PID struct number (control block)
3	float	Proportional
7	float	Derivative
11	float	Intergral
15	float	Effort Limit

239	**PID ON/OFF Select** - Used to select current PID loop to tunning. (select 250 to turn off)	
2	byte	PID struct number (control block)

240	**PID tunning pkt.** - Relates to controller being tunned.	
2		PID struct number (control block)
3	Effort	int *10
5	Desired	int *10
7	Actual	int *10

247	**Raw GPSTelemetry.** - raw data from gps - only sent when debug is 2	
2	byte	raw data

248	**GPSTelemetry.non-si** - Sent back at "gen. tele. period". LEGACY		
2	int	GPS Velocity (uint meters/sec*100)	int *10
4	int	GPS Alt (uint meters*6 ASL)	int * 10
6	int	GPS heading *10 deg	int*10
8	float	GPS lat (float minuetes)	float
12	float	GPS long (float minuetes)	float
16	int	relative x (signed int- meters)	int
18	int	relative y (signed int- meters)	int
20	byte	current cmd (index)	byte

21	byte	nav state(see structures)	byte
22	int	desired x (signed int- meters)	int
24	int	desired y (signed int- meters)	int
26	int	time over tartget (unsigned int seconds)	int
28	int	distance from target (unsigned int meters)	int
30	int	heading to target *10	int*10
32	int	FLC	int

249	**Standard Telemetry non si**- LEGACY		
2		Altitude (meters)	meters* 10
4		Velocity (km/h)	km/h *10
6		Roll (deg +/-)	deg * 10
8		Pitch (deg +/-)	deg *10
10		Heading (deg 0->360)	deg * 10
12		Climb Rate	int *10
14		Battery Voltage mv	int *10
16	int	System Status	int
18	byte	GPS num sats (byte)	byte
19	byte	alt hold AI state	byte
20	int	desired alt	int * 10
22	int	desired vel	int * 10
24	int	desired roll	int * 10
26	int	desired pitch	int * 10
28	int	desired heading	int * 10
30	int	desired climb	int * 10
32	int	aileron out (signed int * 200)	int *200
34	int	elevator out (signed int * 200)	int *200
36	int	throttle out (signed int * 200)	int *200
38	int	rudder out (signed int * 200)	int *200
		throttle from 0 to 100 (* 200), everything else -35 to 35 (* 200)	

248	**GPSTelemetry.** - SI		
2	int	GPS Velocity (uint meters/sec*100)	int *100
4	int	GPS Alt (uint meters*6 ASL)	int*6
6	int	GPS heading *1000 rad (0->2pi)	int*1000
8	int	GP lat degress int	int
10	float	GPS lat (float minuetes)	float

14	int	GPS lon int degrees	int
16	float	GPS long (float minuetes)	float
20	float	relative east (meters)	float
24	float	relative north (meters)	float
28	byte	current cmd (index)	byte
29	byte	nav state(see structures)	byte
30	float	desired meters east	float
34	float	desired meters north	float
38	float	time over target seconds	float
42	float	distance from target meters	float
46	int	heading to target rad*1000 (0->2*pi)	int *1000
48	int	FLC	int

249	**Standard Telemetry.** - SI Sent back at "gen. tele. period". Used for virtual cockpit instruments.		
2	int	Altitude (meters*6)	meters* 6
4	int	Velocity (m/s *100)	int*100
6	int	Roll (rad +/-) rad* 1000	int*1000
8	int	Pitch (rad +/-) rad* 1000	int*1000
10	int	Heading (0->2pi)	int*1000
12	int	Turn rate rad/sec*1000	int *1000
14	byte	RSSI	byte
15	byte	RC com packets per second	byte
16	int	Current - amps*100	int*100
18	int	Battery Voltage Volts*100	int*100
20	int	System Status (int)	int
22	byte	GPS num sats (byte)	byte
23	byte	alt hold AI state (byte)	byte
25	int	desired alt meters*6	int * 6
26	int	desired vel m/s*100	int * 100
28	int	desired roll rad* 1000	int * 1000
30	int	desired pitch rad* 1000	int * 1000
32	int	desired heading rad* 1000	int * 1000
34	int	desired turn rad/sec *1000	int * 1000
36	int	aileron out rad* 1000	int *1000
38	int	elevator out rad*1000	int *1000
40	int	throttle out percent*1000 (.00->1)	int*1000
42	int	rudder out rad*1000	int *1000
44	int	ch6 out rad*1000	int *1000
46	int	ch 7 out rad*1000	int *1000

250	**Digtial output toggle - pin 12 H2**

2	byte	0=pg5 low, 1=pg5 high, 2=toggle

100	**Hardware in the loop -** Request from autopilot for sensor information

100	**Hardware in the loop -** Response packet from matlab containing sensor information	
1	float	DiffPres (airspeed)
5	float	Abs Pres (pressure alt)
9	float	Roll gyro (P) (rad/sec)
13	float	Pitch Gyro(Q) (rad/sec)
17	float	Yaw Gyro (rad/sec)
21	float	Accel X (m/s^2)
25	float	Accel Y m/s^2)
29	float	Accel Z m/s^2)
33	float	DT - time step (seconds)
37	int	g.lat_degrees (east)
39	int	g.lon_degrees (north)
41	float	g.lat_minutes
45	float	g.lon_minutes
49	int	g.num_satellites
51	float	g.altitude
55	float	g.velocity (ground speed m/s)
59	float	g.heading (rad 0->2pi)

101	**Hardware in the loop -** Packet sent from the autopilot to matlab containing servo positions	
2	float	aileron deflection (rad -pi->pi)
6	float	elevator deflection (rad -pi->pi)
10	float	throttle deflection (0->1)
14	float	rudder deflection (rad -pi->pi)

102	**Hardware in the loop -** Start/stop Hardware in the loop simulation mode - put the autopilot in HIL mode(stop gathering sensor data)	
2	byte	1=on 0=off

110	**Ground station -** status packet	
2	float	battery voltage
6	byte	Packets rx/second
7	byte	packets tx/second
8	byte	PIC/CIC
9	byte	ground station status (GS_status)
10	byte	num_sats
11	byte	fix quality
12	int	degrees lat
14	float	min lat
18	int	degrees lon
20	float	min lon
24	float	alt
28	float	velocity
32	float	heading
36	float	RSSI
40	int	pilot address

111	**Ground station -** Set Pilot address (**send to ground station**)	
2	unsigned int	airplane pilot address

111	**Ground station -** Set Pilot address (**response from ground station**)	
2	unsigned int	airplane pilot address

DATALOG PACKETS

201	**Get Datalog Packet-** used to request a datalog packet from autopilot	
2	int	packet number
4	byte	datalog packet size

202	**Start new Datalog -** used to initiate a datalog on the autopilot	
2	byte	number of samples
4	int	sample period
6	byte	number floats

7	byte	number sensor floats
8	byte	num pid floats
9	byte	num servo floats
10	byte	number ints
11	byte	number sensor ints
12	byte	number bytes
13	floats	float array
n	floats	sensor float array
n	floats	pid float array
n	floats	servo float
n	ints	int array
n	ints	sensor in array
n	bytes	byte array

201	**Get Datalog Packet response (packet 0)-** autopilot response for a datalog packet 0 request.

2		packet number (1)
4		number active sensors
5-n		steady state value of all active sensors (2 bytes/sensor)

201	**Get Datalog Packet response (packet 1)-** autopilot response for a datalog packet 1 request.

2	int	packet number (unsigned int) (0)
4	int	number of samples(2 bytes(
6	int	data log sample size
8	byte	num floats
9	byte	num sens floats
10	byte	num PID_floats
11	byte	num servo_floats
12	byte	num ints
13	byte	num sens ints
14	byte	num bytes
15		float array bytes
n		sensor float array (bytes) 2*#sf length byte 1=sensor# byte 2=index (valid index=2->6)
n		pid float array byte 1=pid loop byte 2=index
n		servo float byte1=servo byte 2=index
n		int array (bytes)
n		sensor int array (bytes) #si length
n		misc byte array

201	**Get Datalog Packet response (packet 2-n)-** autopilot

	response for a datalog packet 2-n request.	
2	int	packet number (unsigned int) (0)
4-n		data start

Datalog Sample	
0	sample number
2	DT (LSB fisrt) Little endian
4	misc floats
n	sensor floats
n	pid floats
n	servo floats
n	misc ints
n	sensor ints
n	misc bytes

	PID Struct:		
Index	**Value**	**Description**	**Log?**
0	kp	Gains	don't log
1	kd		don't log
2	ki		don't log
3	fp	Efforts	log
4	fd		log
5	fi		log
6	effort	Total effort	don't log
7	desired	Desired	log
8	actual		log
9	error	Total error	log
10	effort_limit	Saturation Value	don't log
11	loop_id	PID ID	don't log
12	enable		don't log
13	d_LPF_corner	Derivatitve low-pass corner freq.	log
14	(legacy)	(legacy)	don't log
15	error_old		don't log
16	local effort		log

APPENDIX C

TEMPERATURE DRIFT COEFFICIENTS FOR PROTOTYPE AUTOPILOT

Name (Field)	**Sample Autopilot Temperature Drift Coefficients**
Differential Pressure (35)	0.2775
Absolute Pressure (36)	0.0396
P: Roll Rate (37)	0.0722
Q: Pitch Rate (38)	0.1588
R: Yaw Rate (39)	0.1208
Ax: X-Axis Fixed Body Acceleration (40)	-0.0737
Ay: Y-Axis Fixed Body Acceleration (41)	-0.0796
Ax: Z-Axis Fixed Body Acceleration (42)	0.0352

APPENDIX D

SUMMARY OF SENSOR CALIBRATION EQUATIONS

$$Compensated\_value = Sensor_{Raw\_AD} + C_{sensor} \cdot \Delta T$$

$$H = Abs_{Comp\_AD} \cdot .9144$$

$$Vp = Diff\_press_{Comp\_AD} \cdot .044704$$

$$\begin{Bmatrix} P \\ Q \\ R \end{Bmatrix} = \begin{bmatrix} RollGyro_{Tcomp\_AD} \\ PitchGyro_{Tcomp\_AD} \\ YawGyro_{Tcomp\_AD} \end{bmatrix} \cdot 0.044380$$

$$I = Shunt_{Raw\_AD} \cdot z \cdot y$$

$$V = Vin_{Raw\_AD} \cdot y$$

$$X = (longitude - h\_longitude) \cdot \cos(\frac{h\_latitude}{60} \cdot \frac{Pi}{180}) \cdot 1853.2$$

$$Y = (Latitude - h\_latitude) \cdot 1853.2$$

BIBLIOGRAPHY

[1] Tim Crosby, "RQ-1 Predator," *The Warfighter Encyclopedia,* California, 2004.

[2] Andreas Parsch, "AeroVironment FQM-151 Pointer*," Directory of U.S. Military Rockets and Missiles,* 2004, http://www.designation-systems.net/dusrm/m-151.html.

[3] Nuke Newcome, "News Room," *UAV Forum,* SRA International, 4 Oct. 2003, http://www.uavforum.com/library/news.htm.

[4] John Pike, "Dragon Eye" *Intelligence Resources,* GlobalSecurity.org, 21 Dec. 2003, http://www.globalsecurity.org/intell/systems/dragon-eye.htm.

[5] Sara Waddington, "Commercial and civil missions for public service agencies: are UAVs a viable option?," *Unmanned Vehicles magazineBusiness Analysis Forecast*, UAV World. Brass Trading Ltd., Herts, United Kingdom, 29 Apr. 2004, http://www.uavworld.com/civil.htm.

[6] Jon Dougherty, "Border Patrol choppers called unsafe," *WorldNetDaily News*, WorldNetDaily.com, Inc, Oregon, 8 Oct. 2000, http://www.worldnetdaily.com/news/article.asp?ARTICLE_ID=15543.

[7] Howard Loewen, *MP2028g Installation and Operation,* MicroPilot, Manitoba, Canada, 2004.

[8] Ross Hoag, *A highly integrated UAV avionics system*, Cloud Cap Technology, Oregon,10 Apr. 2003.

[9] UAV Flight Systems, *AP50 Autopilot,* Colorado, 2003.

[10] Robert Nelson, *Flight Stability And Automatic Control,* WCB/McGraw-Hill, Ohio, 1998.

[11] Danny Chapman, ”Slope Soaring Simulator,” 29 Feb 2004, http://www.rowlhouse.co.uk/sss/.

[12] Trammell Hudson, “autopilot: Do it yourself UAV,” *SourceForge.net,* 9 Sep 2001, http://autopilot.sourceforge.net/.