CHAPTER 1: INTRODUCTION

Unmanned Aerial Vehicles (UAV) or Autonomous Flying Vehicles are vehicles that can fly with minimal human intervention. Therefore they are useful for tasks that require large areas to be covered with minimum use of manpower. These tasks can be classified as military or civilian in nature depending of the use they are put into. For example, military applications include surveillance of national borders, reconnaissance of enemy territory (drones), providing air support, remote targeting of potential enemy targets, and aerial targets for anti-aircraft gunnery practice. Civilian applications include tracking speeding vehicles (such as a getaway vehicle after a robbery), humanitarian missions to providing relief supplies after natural disaster (such as earthquakes), and crop dusting. There are also scientific applications to UAVs such as sending one into the eye of a hurricane, and thunderstorms to gather data without putting oneself in danger. UAVs are increasingly becoming more independent with numerous intelligence and decision making capability being programmed into them. This is made possible with the advent of low cost commercial off the shelf (COTS) electronics items including sensors, embedded controllers actuators, and recently entire autopilot and navigation platforms. Therefore current UAV research deals less with low level flight control algorithms or auto piloting (since they are already understood very well), and more on the following themes: intelligent and adaptive mission capability including path planning through an unfamiliar 3D terrain coupled with obstacle avoidance, multi-UAV formation flights, and severe aerial maneuvers (aerial aerobatics). This is an emerging "hot" area of research with many universities around the world tackling increasingly complicated problems and implementing innovative solutions to UAV applications. In this chapter an overview of the American University of Sharjah Unmanned Aerial Vehicle (AUS-UAV) background will be given, ending with the thesis objectives and statement detailing my contribution.

1.1 Background

Previous graduate research work (thesis) in this area has been completed by the AUS alumni Hadi [1]. He designed, built and tested a low cost Avionics Unit consisting of an embedded system integrated with the sensors and actuators. The

embedded system consists of a PID controlled Dynamic Flight Controller System (DFCS) or Stability Augmentation System (SAS), a Longitudinal Autopilot created using Discrete State Space Controller implementing Linear Quadratic Regulator (LQR), and a Lateral Autopilot created using PID controller. The SAS is able to maintain stability of the aircraft from various disturbances and also extricate itself from rough transitions or turns. The Longitudinal Autopilot maintains a reference speed and altitude within tolerable limits. The Lateral Autopilot is used to turn the aircraft to a given direction. The embedded system implementing the autopilots also initiates a wireless link with a Ground Station to send sensor data at regular interval for future flight performance analysis. The Embedded Controller also implements a text based configuration menu to set reference gains and offset trims, to adapt the aircraft to different flight environments.

The second area of research in the UAV field carried out in this university was an undergraduate design project, which involved the creation of a Graphical User Interface (GUI) based Ground Station created using Visual Basic [2]. This particular Ground Station was passive in nature and can only be utilized to monitoring the status of the UAV including its roll, pitch and yaw angles, it x, y, and z rates, acceleration, and position. The Ground Station was able to receive data serially through a wireless transceiver. The design also incorporated the possibility of using a database server to store a continuous stream of incoming flight data (ground based black box). Finally the project considered (but have not implemented) the use of the Ground Station to monitor and manage the UAV up to fifty miles away by utilizing the internet to send flight data.

1.1.1 Avionics Unit

The Avionics Unit is the integration of all the necessary sensors needed to monitor the flight parameters of the aircraft. Figure 1-1 below shows the Avionics Unit constructed in [1] using low cost Commercial off the Shelf (COTS) items. This dimension of the Avionics Unit has been made to fit in the fuselage of the TRI-60 RC aircraft.

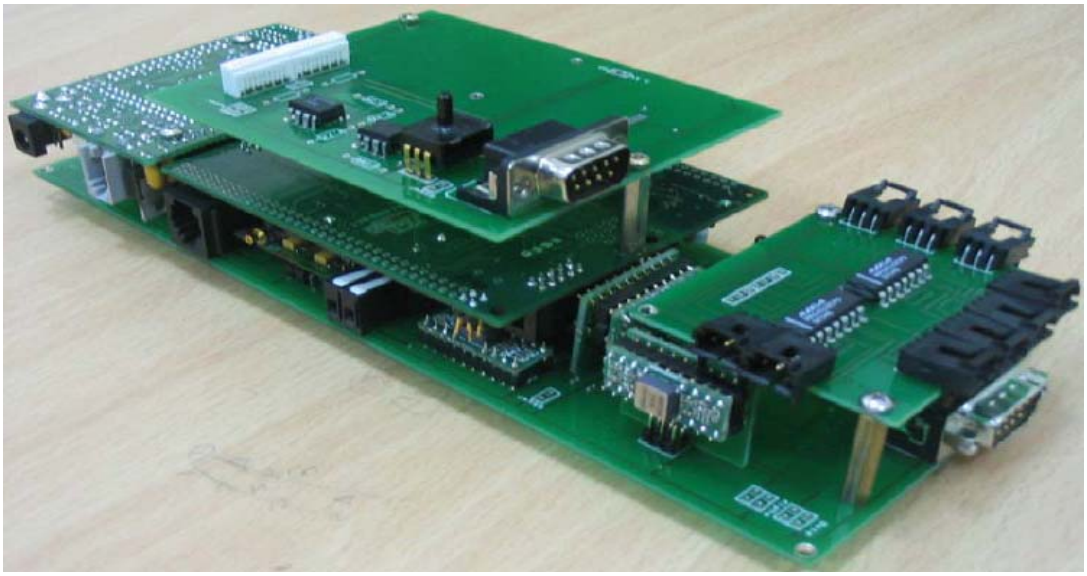


Figure 1-1: The Avionics Unit constructed in [1] by Hadi

The Avionics Unit consists of several analog inertial sensors such as tilt sensors, rate gyros, and accelerometers. Tilt sensors measure the Euler angles such as roll and pitch angles (attitude of the aircraft), rate gyros measure the xb, yb, and zb body axes rates, and accelerometers measure the specific forces acting on the aircraft in the xb, yb, and zb direction. Other analog sensors used are pressure sensors and angle of attack sensor. The gauge pressure sensor is used to measure the body frame forward airspeed (u), while the absolute pressure sensor is used to measure the altitude (h) of the aircraft. The angle of attack sensor is used to measure the alignment of the aircraft with the wind.

The Avionics Unit also consist of two serial sensors such as the HMR 3000 Compass to measure the heading of the aircraft, and the Garmin 15H/L GPS sensor to measures the latitude and longitude of the aircraft. Future sensors will include sonar to measure altitude accurately during takeoff and landing. Sonar sensors or IR sensors will also be used to detect obstacles in flight in future. The Avionics Unit also consists of a manual override circuitry that will be used during emergencies to switch to manual control of the aircraft. An important part of the Avionics Unit is the embedded microcontroller that is used to integrate the sensors an actuator system. The embedded microcontroller collects sensor data, and also implements the Dynamic Flight Controller System (DFCS) consisting of the Stability Augmentation System (SAS), and the Longitudinal and Lateral Autopilots. Finally the embedded system

creates a wireless link to the Ground Station and sends sensor data in periodic intervals to keep track of the flight record during a mission.

1.1.2 Ground Station

Figure 1-2 shows the high level Ground Station architecture, and Figure 1-3 shows the Ground Station GUI developed in [2]. The Ground Station designed can display to the all the relevant flight parameters such attitude using the artificial horizon, heading using the compass, altitude using the altimeter, speed using the odometer. The Ground Station unfortunately does not display the location of the aircraft in a map. To test the Ground Station, the project team has utilized Aerosim to create and run continuously a simulated aircraft model in flight condition.

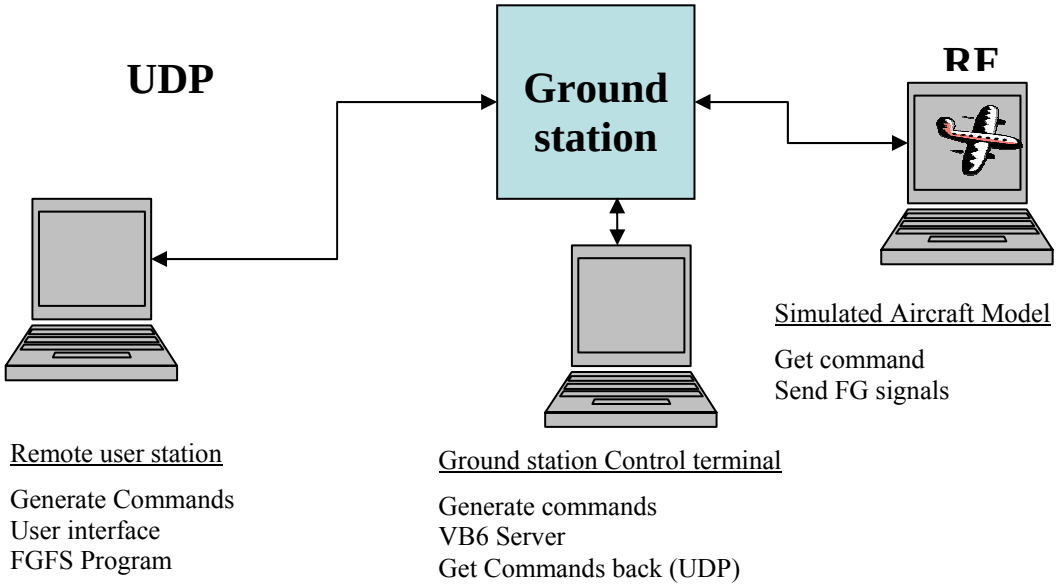


Figure 1-2: High Level overview of Ground Station Architecture implemented in [2]

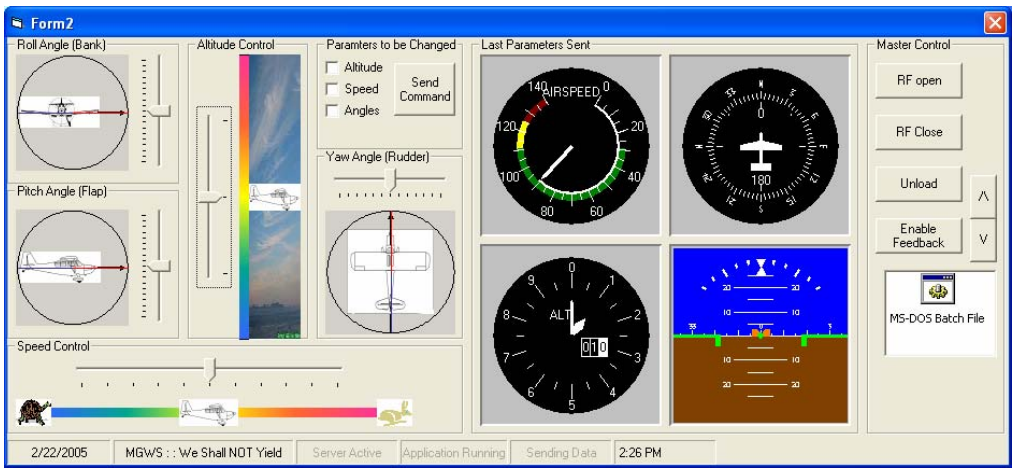


Figure 1-3: The Ground Station GUI designed in [2]

The Ground Station sends periodic requests to the aircraft model via the RF transceivers, upon which the model sends back the state of the aircraft at that moment. It achieves this by momentarily stopping the simulation of the aircraft in flight, and then sending back sensor data such as roll, pitch, heading, altitude, and speed at that moment. Once the transfer is over, the simulation continues from where it stopped. The method of communication between the Ground Station and the Simulated Aircraft Model is Serial using RS232 protocol. Once the Ground Station receives the aircraft data, it refreshes the GUI and at the same time sends the information through the LAN network to a Flight Gear Flight Simulator System (FGFS) for visual display of aircraft attitude.

1.1.3 Limitations of Previous Research

The research done towards AUS-UAV project to date is significant, but has a lot of limitation to its ability. Analysis of the existing capability of the UAV concludes that although once in autopilot mode, the UAV can fly with relative stability, it can only cruise straight. The autopilots cannot change heading, altitude or speed on their own, and they need to be instructed to do so. The trajectory controller that is necessary to command the autopilots is not implemented in the embedded system. The trajectory controller is also necessary to guide the UAV between two waypoints using the smoothest possible flight path. Furthermore, the UAV cannot accept missions as a sequence of waypoint that it must follow. This is a serious disadvantage, in the sense that the UAV is “mission incapable”. Therefore, the guidance of the UAV must be done manually via the wireless link from the Ground Station. To make the aircraft increase altitude, the user must send a reference altitude to the autopilot through the wireless link. To turn the aircraft turn to a specific heading, the user must again input the reference heading through the wireless link. Each specific action must be initiated by the user. Therefore the autopilot is “auto” only in the sense that the user do not need to control the joystick like that of an remote control unit (RCU); but input commands, and then watch the aircraft try to automatically follow that command while keeping the aircraft stable.

Communication between the UAV and the Ground Station (user) currently occurs through a Text Based Interface (TBI) via HyperTerminal. The embedded system facilitates this by providing the text menu. This mode of communication is

not intuitive, time consuming, and inefficient. The UAV can either be in the autopilot mode or in communication mode, not both. This is because, during the communication mode, the embedded system is busy supplying the menu and accepting user input. Thus the UAV needs to be manually controlled during the communication phase. This menu allows the users several options including software calibration of sensors or control gains, starting and stopping the autopilot mode, and changing the default positions of the elevators, ailerons, and rudders. The menu may be comprehensive, but this method of interaction is tedious, slow, and error prone. Table 1-1 lists in detail the shortcoming of the current method of communication.

These shortcomings need to be addressed in any new design that attempts to extend the capabilities of the UAV. A GUI based approach utilizing a Ground Station highly simplifies this process and addresses several of the shortcomings listed. However the GUI system designed so far is not suitable since it is can only monitoring the status of UAV during flight. The user must also be able to send missions to the UAV. Therefore, the Ground Station need to have a "map" of the terrain, and should have the ability to allow the user to select waypoints within that map, and to transmit them to the UAV. Part of this thesis is to overcome these particular problems to enable flexible mission capability, and the discussion for them will be done later in the chapter.

Table 1-1: Shortcoming of the TBI and the menu system of interaction

Shortcomming	Explanation
Time Consuming	The user needs to go through several menus before being able to send information to the UAV
Impractical	The user needs to memorize the menu hierarchy, else waste valuable flight time trying to find the desired selection
Error prone	This is due to the fact that the TBI offers no validation of user input. Erroneous unintentional input can send the UAV into an unstable condition. Since the user is allowed to change low-level information such as elevator, rudder and ailerons angles, gains, and offsets, the chances of this happening are high. Once sent error values cannot be corrected fast enough through the menus
Tedious	It is easier and more enjoyable to control the UAV manually, rather than by menus, which are visually unappealing

1.2 Path Planning Literature

Making the UAV "mission flexible" is the main theme of this thesis, and is achieved by means of path planning. Path planning in a UAV provides the level of autonomy by having minimal ground control. In an abstract term, path planning involves creating a plan to guide a point-like object from its initial position to a destination waypoint. Along the way, there may be a set of regions to visit and a set of regions to avoid. In addition, the traveling object may have certain motion constraints. Path planning strategy could either be a static or dynamic depending on whether the path-planning problem is to create a path in static or dynamic environment [3]. Good path planning routines will attempt to create paths that are minimum length and fully consistent with the physical constraints of the aircraft. Path planning comes in two forms, one that is pre-generated to show the approximate route that will be taken by the aircraft while it navigates between two waypoints, and the other is the controller that guides the aircraft between two waypoints while tracking a route that was pre-generated. The former is implemented in the Ground Station, while the latter is implemented in the embedded controller of the Avionics Unit on top of the autopilots. There is numerous path planning algorithms that have been researched to date, and the following section attempts to give a brief overview of these technologies.

1.2.1 Approaches to Path Planning Strategies

In [3] a behavioral approach to path planning is considered by creating a set of flying modes (or behavior) that will guide the aircraft to its next waypoint. This approach works best for helicopters, since they have the hover mode. Behavior includes different flying modes such as take off, cruise, turn and landing, which can be used to compose an entire flight path. An example flight path using behavior control uses this set of instructions to guide the UAV to its next waypoint: Take off → Cruise 500m → Turn Left 90 degrees → Cruise for 10 seconds → Turn Right to face East → Cruise 300m → Circle waypoint. These same behaviors can be modeled for a fixed wing aircraft as well by creating dedicated algorithms such as Pitch Attitude Hold (PAH), Pitch Rate Hold, Velocity Hold, Altitude Hold, Heading Hold, Roll Attitude Hold (RAH), Roll Rate Hold, and Turn Rate Hold (TRH) [4]. Figure 1-4 shows a schematic diagram of a behavioral based approach.

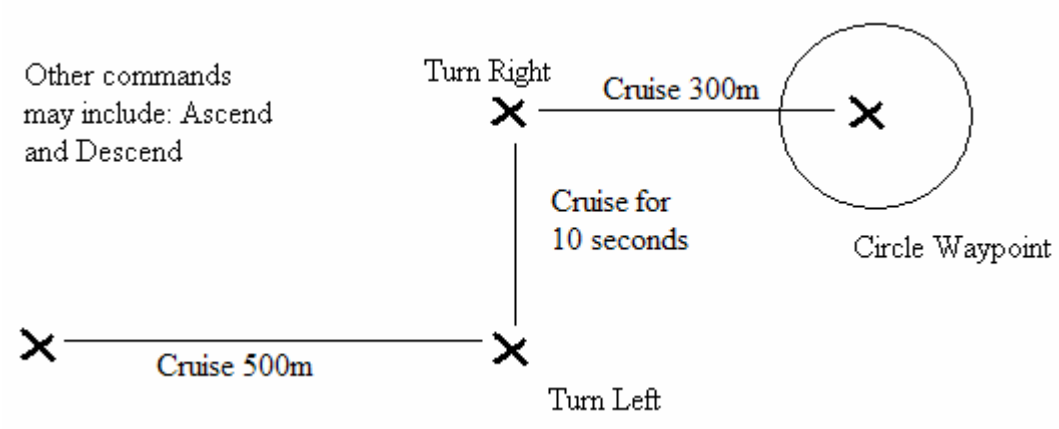


Figure 1-4: Schematic Description of a Behavioral Based Approach to Path Planning

In [5] a concurrent constraint programming (CCP) was used as the main tool for the design and the implementation of a software path planner. CCP is a very high level and complex heuristic path planner that takes into account obstacle avoidance, shortest and optimum flight path, and weighed regions. Weighed regions are regions with abnormally low or high pressure, wind speeds, or any other factor affecting flight, such as elevation of the land. Most path planning follows an approach where the path planning, trajectory smoothing, and flight stability are separated into separate layers as shown in Figure 1-5 [4, 6, and 7]. The path planner merely plots a series of set points between two waypoints, navigating through obstacles, whereas the trajectory controller guides the UAV through a smooth path through those set points. The stability controller runs in the background, and keeps the aircraft stable through flight.

This kind of layered hierarchy is followed in this paper with the Ground Station as the top layer where missions are planned. The Trajectory Tracker, which is also the guidance controller, is in the second layer. The Trajectory Tracker forms the interface between the lower levels which are intimately connected with maintaining flight stability and the upper level where missions are planned. The third layer will consist of the lateral and Longitudinal Autopilots, whose tasks are keeping the UAV stable during lateral and longitudinal motions respectively. These autopilots form a suitable abstraction layer between the DFCS and the Trajectory Tracker. They are also intermixed with the DFCS, but accept reference inputs from the Trajectory Tracker. The autopilot then tries to maintain the aircraft stability at the new reference states. The lowest layer will be the DFCS, which is the stability controller and keeps the UAV stable in general during flight or while transit between two reference states.

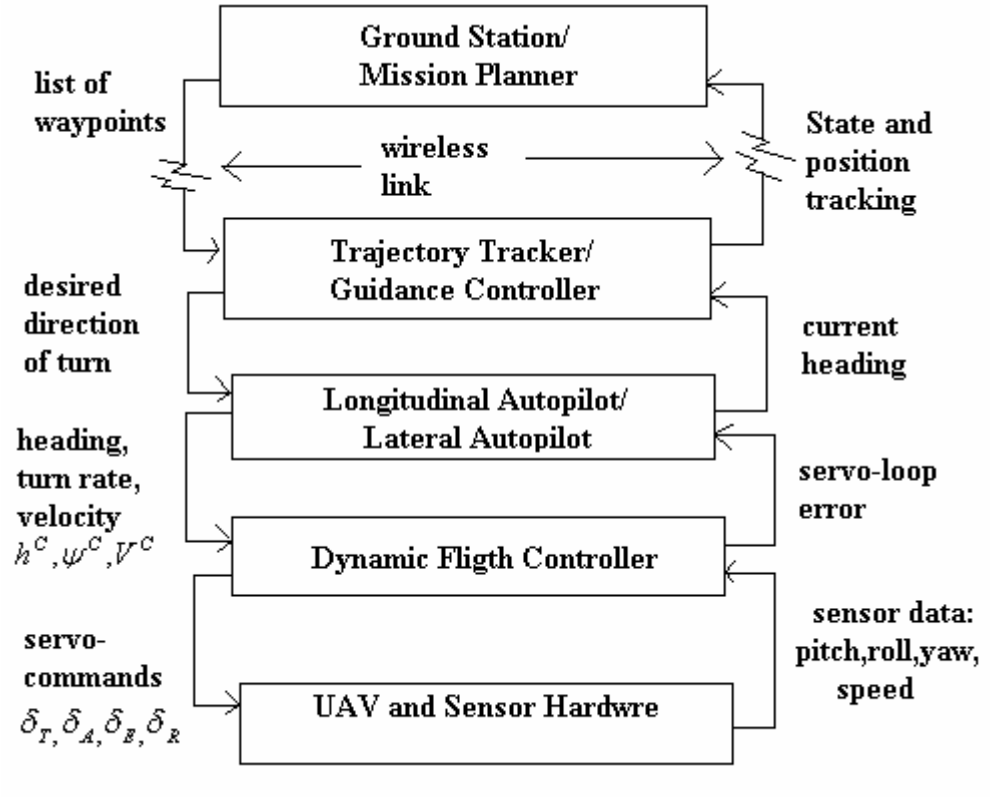


Figure 1-5: Schematic Representation of a Layered Architecture

A final issue that needs to be addressed is the position of the aircraft during flight. The GPS unit onboard the UAV receives a position fix once every second. With a UAV top speed of 25-30 m/s, the trajectory controller does not get the current GPS coordinates fast enough to guide the UAV, therefore the UAV can stray from its path during flight. To correct this problem the UAV needs a faster and an accurate method of determining its position for up to a minute relying on GPS signals.

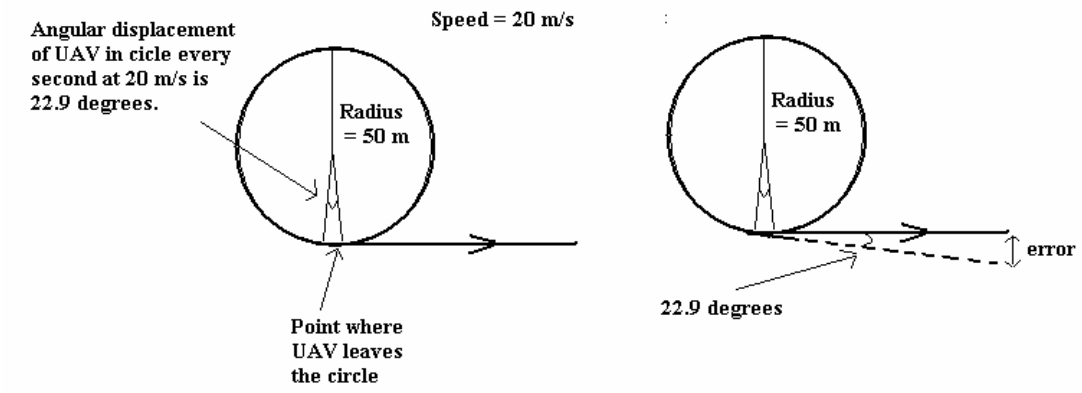


Figure 1-6: Schematic Representation of Error due to Low Position Update Frequency

A dead reckoning navigation system based on the fusion of inexpensive Inertial Navigation System (INS), air data, and magnetic sensors to is described in [8]. This GPS/IMU combination is also used in [9] to create Guidance, Navigation, and Control (GNC) algorithm. Figure 1-6 shows an approximation of the worst case track error that can occur when navigation depends only on low frequency (1 Hz) GPS updates. Dead reckoning usually depends usually solves this problems by relying on inertial sensor data and the most recent GPS position fix to calculate the current position, velocity and heading of the aircraft. Depending on the complexity of the algorithm, position fixes can occur at 10 to 20 times the rate of a GPS update. A dead reckoning navigation algorithm using extended Kalman filtering is given in [10], [22]. [23].

1.2.2 Path Planning Strategies

An important component of creating flexible missions is the generation of a suitable path for the UAV to follow. This is no mean task, since it requires complex algorithm to generate paths that meet all possible constraints placed on it. Several path-planning algorithms exists, among them the Dijkstra's Algorithm [11], which calculates all possible paths from a point and then chooses the shortest path available at that point. However Dijkstra's Algorithm does not give a global optimum, since the path look ahead is only one point. [12] describes a highly autonomous and flexible guidance system that utilizes onboard flight path prediction in combination with numerical optimization routines to guide the vehicle. For this thesis, the following algorithms were researched:-

- A guidance strategy based on the UAV instantaneous velocity vector, the position vector of the target from the UAV, and instantaneous turn rate. This strategy (called the Feedback Guidance Strategy as shown in Figure 1-7) can steer the UAV from a given initial position and heading to a specified destination, in an obstacle free 2-D (constant altitude) environment, provided the algorithm is given continuous position and heading updates [13]. Since one of the goals is to allow the UAV to fly between waypoints without changing altitude, this algorithm may be suitable. This is the strategy used by the path planner to create shortest flight path. An extension of this strategy will be used to control the direction of the UAV once it reaches the waypoint, and also to control the altitude while the UAV

moves in a straight line path. All of this will be done in the shortest possible flight path.

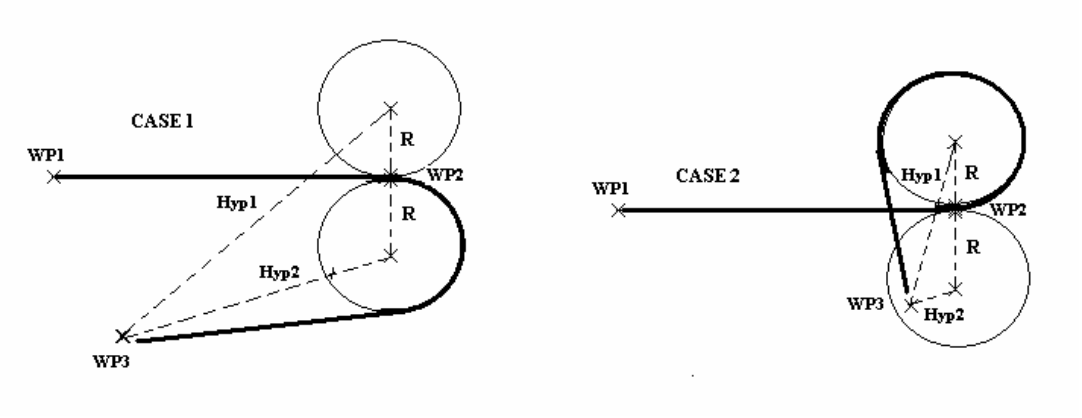


Figure 1-7: Path taken by UAV when entering a curve using Feedback Guidance Strategy

- In [14], a technique for planar trajectory is proposed where the trajectory is modeled as a planar Spline. The goal is to have the UAV follow a reference trajectory specified by a B-cubic Spline, which is said to be easy to construct using minimal user input and requires minimal computation. The Spline is also developed in a constant altitude plane.
- In [15], the trajectory has been created using a "Clothoid" or "Cornu-Spiral" algorithm as shown in Figure 1-8. Clothoids are arcs whose curvature increases with arc length. The justifications for using Clothoids were low computation time, and the fact that objects moving in a straight line (zero curvature) cannot instantaneously enter into an arc with a certain curvature.

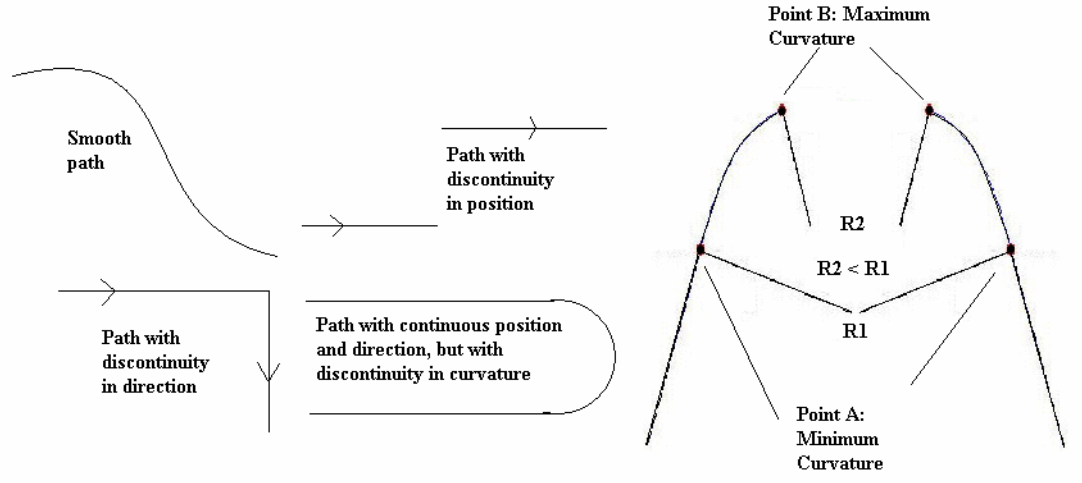


Figure 1-8: Schematic Representation of Path Created using Clothoids

- In [4, 7], a κ-trajectory is used to join two straight-line segments to provide a dynamically feasible path for the UAV. The implementation of the κ-trajectory is shown below in Figure 1-9, and is another means of resolving discontinuities that occur when an aircraft enters and exits from a circle

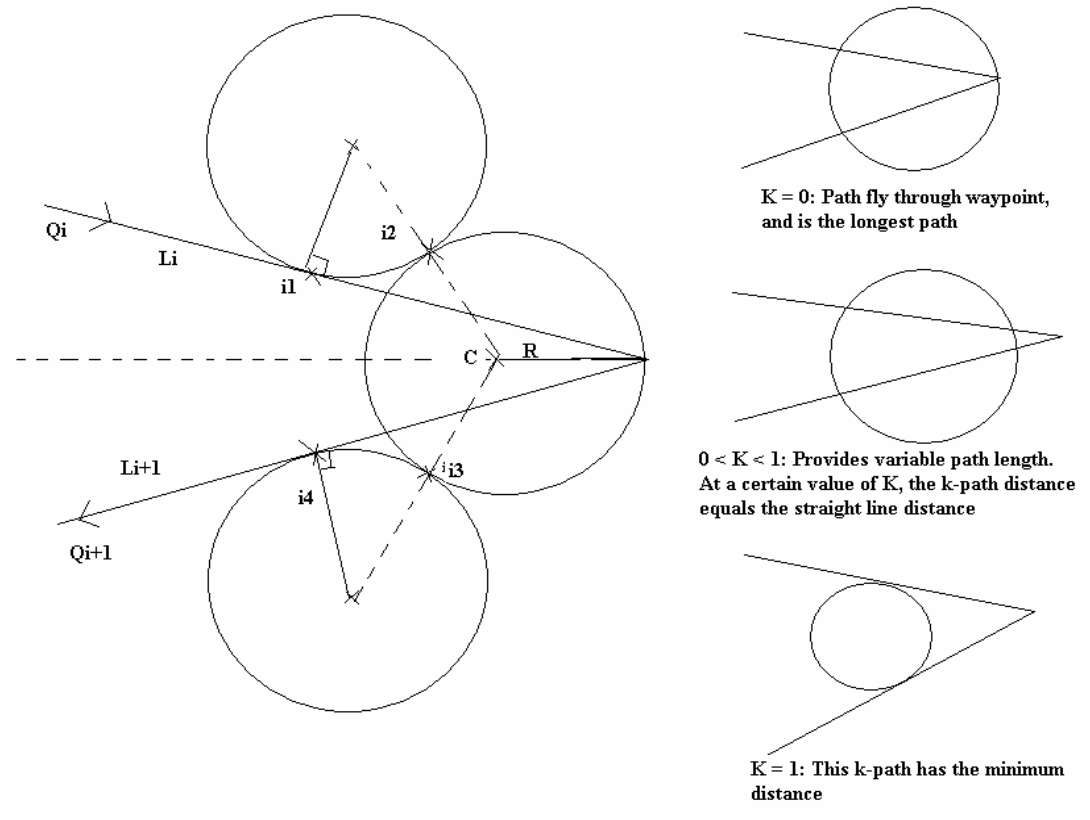


Figure 1-9: Schematic Representation of k-trajectory Flight Path

- In [16], a sinusoidal path was developed for the UAV to track a slow moving ground vehicle. This allows the UAV whose velocity is much faster to keep up with the ground vehicle. This ensures the shortest possible path and therefore saves a lot in terms of fuel consumption. A surveillance algorithm in the form of a rose curve, shaped like an '8' is also described. This is useful since an airplane, unlike a helicopter cannot hover over a position indefinitely.
- In [20], another novel approach has been implemented for a fixed wing aircraft to track a helicopter for coast guard applications. Where as the helicopter can come to a standstill and then turn without changing position, the aircraft must keep flying at all times. Even so the aircraft must be able to detect deceleration in the part of the helicopter and the direction is about to turn and immediately follow a

set trajectory so that it intercept a point close to the helicopter after the turn. This is accomplished through the use of a decentralized hybrid controller and a mode switching path planner to generate the optimized tracking trajectory.

1.2.3 Three Dimensional Path Strategies

Most of the strategies presented above do not deal with the three dimensional path planning. It is not an easy task to control the aircraft in a three dimensional environment, while at the same time executing the path algorithm. When the aircraft is moving straight, controlling the height and the speed of the aircraft is very easy because of two reasons: there are fewer variables involved, and the variables are highly independent of each other. But during a turn the height and speed of the aircraft is not controlled. Keeping the speed steady during a turn will result in decrease in the altitude, while keeping the height steady requires increases in the speed. Since the aircraft must move at a close to constant speeds during a turn to ensure that the radius of turn is kept constant, the height of the aircraft suffers as a result. The only solution is to rectify the error in the height while the aircraft is moving in a straight line.

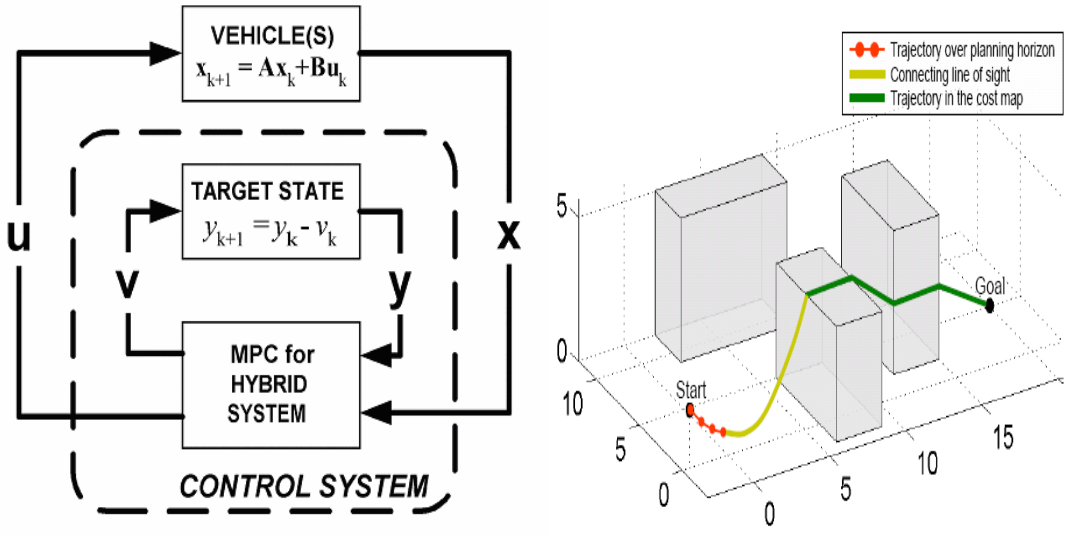


Figure 1-10: Overview of the MPC Control Scheme in [18] and its use in [19] as RH

There are several algorithms that can be utilized to control the aircraft maneuver so that it does exactly what is asked of it. In [18] a formulation for Model Predictive Control is applied to vehicle maneuvering problems in which the target regions do not contain equilibrium points. The algorithm is an extension of previous MPC algorithms which were formulated for control around steady state conditions. In

[18] Mixed Integer Linear Programming (MILP) is also used in combination with MPC to solve the trajectory optimizations. Analytical proofs are given to show that the problem will always be completed in finite time and that, subject to initial feasibility, the optimization solved at each step will always be feasible in the presence of a bounded disturbance. MPC is a feedback control scheme in which a trajectory optimization is solved at each time step. The first control input of the optimal sequence is applied and the optimization is repeated at each subsequent step. Because the on-line optimization explicitly includes the operating constraints, MPC can operate closer to constraint boundaries than traditional control schemes. For this thesis, the use of MPC and Kalman filtering to handle the non-linearity of the system is explored.

Likewise three dimensional motion planning have been utilized in [19] and [20]. [19] Utilizes the same MILP algorithm used in [18] to optimize a trajectory over three dimensional spaces. The aircraft flies close to the ground and only deals with obstacles by flying over them. Path planning is done initially in 3D without regards for any obstacles. Once an obstacle presents itself in front of the aircraft, MILP is used to calculate an optimized trajectory to fly over the obstacle based on the receding horizon principle. [20] deals with navigating a small automated drone through a dense 3D terrain.

1.2.4 Ground Station Design

A "secondary theme" of this research is to create a working model of a Ground Station with bidirectional communication facility between itself and the UAV that allows the user to track the UAV on a map, and also direct the UAV to precise destinations by having it follow a set of waypoints; i.e. path planning. Ground stations are very important when it comes to monitoring the status of flying objects such as satellites, missiles, airplanes, and UAVs to name a few. For a UAV, the Ground Station provides important "attitude" information such as pitch, roll, yaw, angle of slip, and angle of attack. The Ground Station also provides information regarding the heading, location, altitude, and velocity. Very often, it is not possible to track a UAV in flight visually due to distant location; therefore, a Ground Station can be valuable for both the user and UAV. Ground Station serves as a virtual cockpit in addition to mission planning and monitoring. Passive Ground Stations merely monitors the UAV, whereas an active Ground Station is able to direct and guide the

UAV through its missions, such as surveillance, path planning, calibration and other functions. Ground stations are implemented in [2, 4], with [4] presenting an example of a good intuitive design and user friendly interface. In [4], PDA is also used to control the attitude of the UAV. The PDA is interfaced with the Laptop through a Bluetooth connection. For this thesis, an improvement on [2] has been considered.

To address shortcomings of previous research track in a suitable Ground Station for an UAV, a more recent project have been undertaken for this thesis to create a Ground Station that can also command the UAV to follow a set of waypoints (mission planning). The Ground Station architecture will be detailed in Chapter 3, but the basic framework has been revamped, and it allows for easy and intuitive usage. The user is spared the need to know any low-level flight dynamics, and instead can spend his/her time completing high-level mission objectives such as selection of waypoints, and other mission parameters. The Ground Station will allow flexible mission planning capabilities through the dynamic path-planning feature that has been incorporated, through which the user can view an approximate path that the UAV will follow for a given set of waypoints, even before these waypoints are sent to the UAV. The Ground Station also allows for repeat mission, without the need to reset or reprogram the UAV or to bring the UAV down.

1.3 Thesis Outline

This thesis report is split up into the four main components; namely the background, theory, hardware/software setup, and results and analysis. This first chapter gives the user the overall background of the AUS-UAV research track and also presents the user with a review of the literature and research being conducted in other universities. Chapter 2 presents the objectives of the current research, contributions being made to the AUS-UAV research track, and its significance. It also proposes a general solution to overcome previous research shortcoming and ends with an overview of the design and methodology. In short Chapter 1 and 2 addresses the background component of this thesis report.

Chapter 3 addresses the theory component to this thesis and details the path creation algorithm and methodology and explains the constraints that must be adhered to during trajectory generation. Path constraints include the mechanical constraints such as the turn rate constraints, and the control constraints of the lateral and longitudinal autopilots. It then does a mathematical analysis of path creation utilizing

the 2D feedback guidance algorithm and extending it to 3D. Alternatives path creation methods that allow the UAV to reach a waypoint and at the same time face the next waypoint are also described. Chapter 2 will end with a brief description of the Dead Reckoning algorithm.

Chapter 4, 5, 6 and 7 describes the individual HILS components and their overall integration to test the flexible mission architecture. For example Chapter 4 deals with the Ground Station GUI framework including the ability to select waypoints, generate approximate path, communicate with the UAV, and display UAV flight status. The path algorithm implemented in the Ground Station is also described.

Chapter 5 deals with the embedded system design and the real time implementation of the Trajectory Tracker, Autopilots, and Dead Reckoning. It also describes the communication aspects between the two microcontrollers that makes up the embedded system, and the communication with the Ground Station. The chapter ends with a detailed description of the calibration and tuning of the inertial sensors and the air data sensors that forms a part of the Avionics Unit.

Chapter 6 describes the design of the "Simulation Only" Setup, while Chapter 7 describes the HILS setup. These two chapters have more of a "technical documentation" feel, and are designed to let a future research easily set up the HILS up and running quickly.

Chapters 8 deal with the results and analysis of the HILS and "Simulation Only" test runs. This chapter also describes the limitations of this research and offer scopes for improvement, and deals with real time issues which were problematic. Finally Chapter 9 looks at the problems and limitations of this thesis and scope for future improvements that can be made and concludes this thesis.

CHAPTER 2: PROBLEM STATEMENT

2.1 AUS-UAV Research Requirements

Research into UAVs in AUS started with simple stability algorithms that would keep the UAV level (zero pitch and roll) during flight and be able to counter disturbances (rate dampers). The first Avionics Unit consisted of a compass (with Heading, Pitch, and Roll output), GPS, one rate gyro, and two accelerometers. Demonstration would include elevator response to control pitch, aileron response to control roll, and rudder control to control yaw rate.

Since then research progressed into Ground Station Design and a complete Avionics Unit design. The current Avionics Unit is professionally made PCB that integrates sensors, controller and actuator together into one unit. The sensor can measure all the flight parameter data such as the Euler angles, angular rates, acceleration, velocities and position in three axes. The controller is modifiable to apply different control laws, and the actuator system has an inbuilt manual override system. The Ground Station allows the user in the Ground to monitor the UAV in flight and also to give mission waypoints. Currently demonstrable is the ability to remain stable in flight and with the completion of this thesis the ability to navigate to various waypoint coordinates in 3D.

The **overall goal** of the AUS-UAV research is the ability to carry out **generic civil missions** (such as monitoring remote locations, surveillance) or any other client specified missions using **an autonomous and cooperative team of ground vehicles and aerial vehicles** (consisting of both **helicopters** and **fixed wing aircraft)**, moving in **coordination** by **2010**. Each individual UAV must be capable of **take off** and **landing** from a specified runway.

Landing requires very accurate control not only because the UAV must be brought down lightly so as not to damage its payload, but also because it must land in the runway, which requires the UAV to follow a rigid trajectory during the landing run, in order not to overshoot the runway or land elsewhere.

Formation flight requires the UAVs to be able to avoid obstacles during flight and maintain a specific distance from each other despite any disturbances due to wind and be able to communicate with each other and with the Ground Station. The UAVs trailing the leading aircraft must be able to manage the atmospheric disturbances

caused by the flight of the leading aircraft, and should not diverge from the track followed by the leading aircraft.

Reconnaissance involves the ability to loiter over a target location or the ability to track a ground based moving object. Reconnaissance requires utilizing of a camera to get an image of a location and the use of image processing to "zero in" on the desired object. The camera position control should then be able to keep the object centered in the image by actuating the pivot motors despite the movement of the UAV or the ground object.

The ability to manage landing or formation flight requires the need for accurate position information from the navigational computer. To achieve this, the NAVCOM should be able to predict position based on inertial data, which requires the design of an accurate GPS/IMU unit. The current Avionics Unit is not suited for inertial navigation due to shoddy IMU design, and lack of temperature or humidity compensation.

To enable proper research ability into UAVs, an important requirement is the design of a 6 DOF rate platform that will extend the validity of the HILS simulation by allowing the inertial sensors in the Avionics Unit to be used instead of having to simulate the sensors. This is not only good for trying out Dead Reckoning algorithms but also good for calibrating the inertial sensors. Also needed is a room sized wind tunnel setup to test and calibrate the air-data sensors on the Avionics Unit. A final requirement is the design of an indigenous aircraft body-frame that has more aerodynamic stability than commercially available models. The advantage of such a design is that aircraft stability derivates can be controlled due to design parameters, and various aircrafts can be designed such as delta wing models.

2.2 Thesis Requirements

2.2.1 Objectives

The primary objective of this thesis is to demonstrate via Hardware in the Loop Simulation that the existing UAV test-bed can be utilized to perform waypoint navigation in three dimensions. Waypoint navigation consists of getting the aircraft to fly to within 50 meters over subsequent waypoints or position coordinates. The waypoints will be selected by the Ground User by clicking on the map in the Ground Station, and then sent to the UAV via the wireless link thus allowing "flexible

mission" capability. The path taken by the UAV to navigate between waypoints should be smooth and minimum time and should not violate the flight constraints of the UAV. Once the HILS is proved then the actual UAV test-bed, consisting of the TRI-60 RC aircraft and the Avionics Unit hardware, must be flown over a minimum of three waypoints in 3D.

To accomplish this objective, the Longitudinal and Lateral Autopilots and the Avionics Unit hardware designed in [1] must be used. Briefly, the Longitudinal Autopilot controls the speed and the altitude. The design of the Longitudinal Autopilot in [1] is such that the elevator is used to increase or decrease the altitude; changes in the speed that occurs due to the change in altitude is compensated for by changing the throttle output. The Longitudinal Autopilot is designed for what can be termed as cruise control, which is to maintain the speed to a default value despite changes in altitude or the bank angle. The Lateral Autopilot is used to bank the aircraft and execute a coordinated turn where the aircraft turns while minimizing its side acceleration and velocity.

2.2.2 Specific Thesis Contribution

The primary contribution of this thesis research is the Hardware in the Loop Simulator Setup consisting of a real time simulator and prototyping unit (dSpace), the embedded system and the Ground Station (hardware), and Flight Simulator software. For this thesis it was used to simulate 3D waypoint navigation and demonstrate "flexible mission" capability. However for later research, it might be used to test GPS/INS algorithms, and other flight algorithms. The current HILS setup can also be extended and enhanced by using a rate platform and a wind tunnel system so that the actual sensors can be brought into play.

Another important contribution of this thesis is the design of a 3D minimum path algorithm using an extension of the Feedback Guidance algorithm that provides for minimum path only in a 2D planar surface. Another contribution of this thesis is the mathematical derivation of a "bidirectional feedback guidance" that allows the UAV to reach a waypoint while facing a certain heading. This version is useful for loitering over a waypoint before moving onto the next waypoint. Another version of allows the UAV to accomplish the same, however taking the minimum possible path.

2.2.3 Significance of the Research

It is hoped that the thesis will give "practical" and "demonstrable" exposure to the AUS-UAV research that has been carried out for the past three years. So far the research bordered mainly on the following factors: gathering the required knowledge base, creating the required hardware and its integration, and designing testable hypothesis and models and verifying them through simulation. Completion of the thesis will extend the technology base of the AUS-UAV track, and bring this research into the spotlight.

This research is significant because it demonstrates the ability to use an existing hardware and at the same time extends its ability with minimum modifications. It also brings the research one step closer to the overall goal of the AUS-UAV research track. It eliminates a lot of the shortcomings of previous researches, using simple and yet intelligent redesign of the controller architecture and thereby allowing the use of existing UAV test bed.

2.3 Proposed Solution and Design Overview

The aim of the thesis is to design a UAV having "flexible mission" capability and one that can navigate a sequence of waypoints in 3D while remaining stable during flight. To allow this the embedded controller has been redesigned in order to allow more flexible mission capability, by creating a layered architecture hierarchy, with the Dynamic Flight Controller (DFC) at the bottom level of the hierarchy, the Longitudinal and Lateral Autopilot at the next level, and the Trajectory Tracker at the third level. This approach allows the embedded system to accept flexible mission with no extra coding between missions. Chapters 5 will explain in detail the design of the embedded system.

The algorithm chosen is based on its computational complexity and the length of the flight path generated. Technically, the algorithm should not be computationally intensive; enabling it to be executed at a higher frequency. This would lead to more accurate tracking of the path by the UAV. Also, the algorithm should give the smallest flight paths between two waypoints. Of all the strategies studied; Feedback Guidance give the smallest flight path since it consists of straight lines for most of its path, and follows a curve with minimum allowable radius when the heading needs to be changed, and it is the overall strategy followed in this thesis, with 3D navigation capability added to it. The strategy will be explained in detail in Chapter 3.

To address the shortcomings detailed in Section 1.1.3, a GUI based Ground Station has been designed. The new interface of the Ground Station is graphical and therefore intuitive and easy to use. The user is spared the need to know any low-level flight dynamics, and can instead spend his/her time completing high-level mission objectives such as selection of waypoints, and other mission parameters. The Ground Station allows flexible mission planning capabilities through the dynamic path-planning feature that generates for the user a smooth line that the UAV will approximately track for a given set of waypoints. New missions can be entered without the need to reprogram the embedded system onboard the UAV during every mission. Building the communication interface in the Ground Station also frees up the embedded system on board the UAV, allowing it to provide continuous flight control. The Ground Station allows the user to verify various flight paths for a selection of waypoints, before the waypoints are sent to the UAV. Although the Ground Station generates approximate paths, the user can be reasonably sure that approximation is a good one since they meet one crucial constraint, which is the maximum turn rate or the minimum turn radius for a given speed.

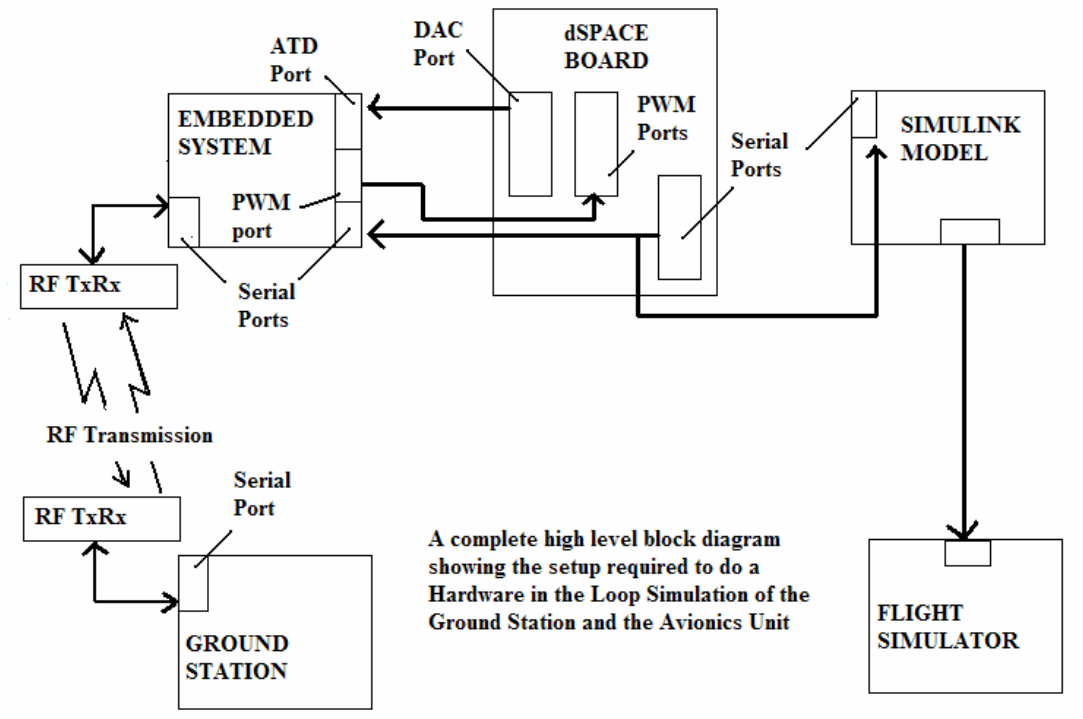


Figure 2-1: A High Level Overview of the HILS Setup

The Ground Station is one half of the two pronged attack used to tackle the problem of allowing the UAV to accept "flexible missions". It forms the mission

planning layer of the layered architecture. Parallel processing is used extensively since two different microcontrollers make up the embedded system on board the UAV, One microcontroller implements the DFC with the integrated autopilots, and the Trajectory Tracker, while the second microcontroller implements the Dead Reckoning Algorithm and can also be used to implement the supervisory fuzzy-gain scheduler. The communication occurs via the Controller Area Network (CAN) bus, which is a fail safe method of data communication between two devices in a rugged environment. Interrupt based programming is also used to acquire sensor data, and that adds to the parallel processing ability.

To test the overall Ground Station and embedded system hardware a HILS setup as shown in Figure 2-1 has been designed. The real time aspects of HILS allow the hardware to be tested in flight conditions, and to verify that the control ability is adequate to allow 3D waypoint navigation and "flexible" mission capability. The subsequent chapters are written based on each component shown in Figure 2-1, with the next chapter dealing with the mathematical (theory) aspect of path planning and constraints.

CHAPTER 3: 3D PATH ALGORITHM AND CONSTRAINTS

Path planning is required for any object going through guided motions, whether it is an autonomous car, an autonomous submarine, guided missiles, autonomous helicopter, and autonomous fixed wing aircraft. Since the dynamics of each of these vehicles are so different, and they have different constraints on there motion, therefore the algorithms used to generate these trajectories are different. For example on one hand, a car has the least amount of constraints of all the vehicle mentioned, and therefore there is a lot of flexibility in their trajectory, and on the other hand there is the guided missile, whose supersonic speeds make it very hard to control its trajectory. In the same trajectory for a helicopter is different from that of a fixed wing aircraft, since a helicopter is capable of hover flight, whereas a fixed wing aircraft needs to be in continuous flight to keep afloat. Therefore a fixed wing aircraft cannot make sharp turns and must follow a smooth path or trajectory as indicated in the previous chapter. In this chapter all the physical constraints that must be taken into consideration for the TRI-60 aircraft, as well as the constraints of the autopilots will be discussed, followed by a discussion of the path algorithm adopted.

3.1 3D Path Constraints

There are two aspects that constrain the path generated by the flight plan and they can be divided into physical and controller based constraints. The path-planning algorithm must take into account the following physical constraints of the UAV, and among them are: the top speed, the minimum flying radius (or maximum turn rate), the stall speed, and the maximum bank angle, the maximum rate of climb or descent to name a few. The aircraft must also fly at a minimum speed exceeding vehicle the stall speed, to generate enough upward lift to match its weight, so that they can remain afloat. The value of the stall speed increases as the aircraft banks, since due to the bank its wings do not support its whole weight and therefore more lift is needed to compensate for that. Other constraints come from the controller or autopilot design. The autopilots have been designed to effect changes from the current state to the reference state with a certain speed and order of response. Therefore the autopilot constraints can be thought of as physical constraints managed or offset by the controller surfaces of the aircraft including the rudder, elevator, ailerons, and throttle.

3.1.1 Turn Rate Constraints [24]

Equation 3.1 shows the minimum radius R, at which the UAV can fly at a given speed V, and bank angle ϕ.

$$R = \frac{V^2}{g * \tan\phi} \tag{3.1}$$

Equation 3.2 shows the minimum speed at which the UAV can fly when banked at an angle ϕ.

$$V_{STALL} = \sqrt{\frac{2 * W}{\rho * C_L * S * \cos\phi}} \tag{3.2}$$

Where W = total weight of the UAV, ρ = air density, C_L = lift factor, S = surface area of the wings.

The other constraint is that the UAV must fly at least 1.25 times its stall speed for safety reasons, and also the UAV should fly at 80% of its maximum speed to prevent the engine from overheating.

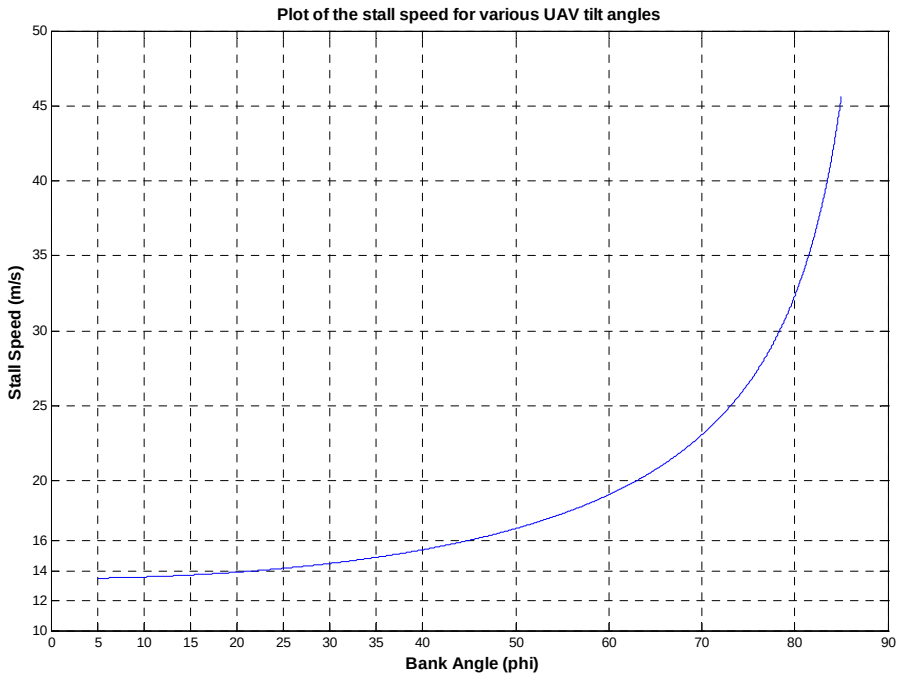


Figure 3-1: Plot of Theoretical Stall Speed Required for a Given Bank Angle

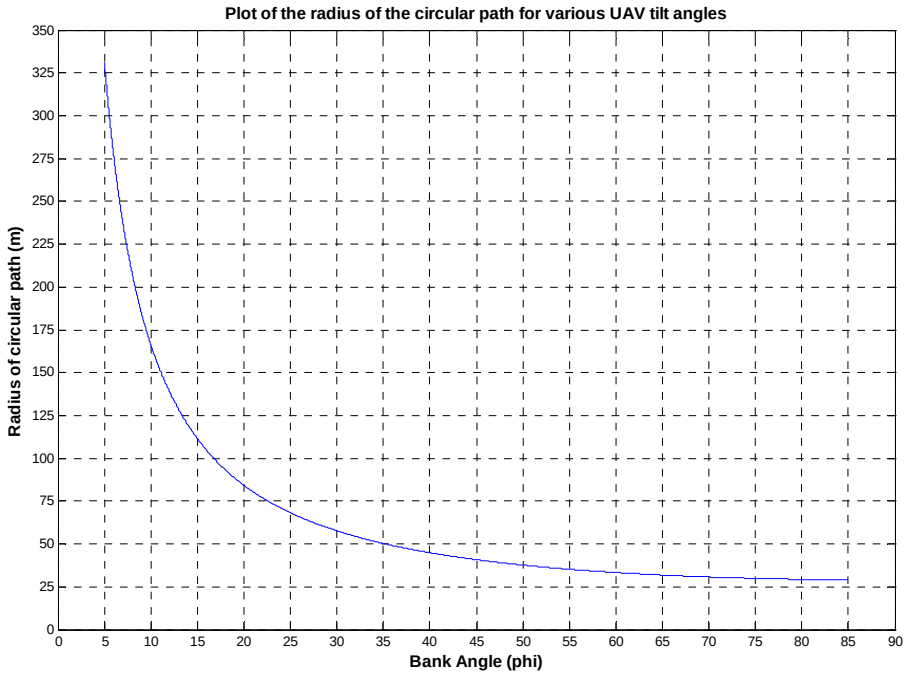


Figure 3-2: Plot of Flying Radii vs. Bank Angle Maintaining the Stall Speed

Figure 3-1 and 3-2 above shows the plots of stall speed and flying radius at various bank angles for our TRI-60 based on its total weight, surface area of the wings, and its lift factor. Since the maximum speed of the UAV is 25m/s to 30m/s, a speed of 20m/s is maintained during flight. At that speed, the maximum bank angle is 40 degrees, taking into account that the UAV is flying 1.25 times the stall speed (16m/s). Using that speed the minimum flying radius is 45m. The path algorithm must ensure that a path do not have a radius of less than 45 meters, else it will be physically impossible for the UAV to track that path. These are only simplified theoretical constraints assuming a rigid body in the general shape of an airplane. They do not take into account the control actions of elevator, aileron or rudder, and throttle which can be used to modify the minimum flying radius and the stall speed for a given bank angle.

3.1.2 Longitudinal Autopilot Constraints [25]

The Longitudinal Autopilot designed in [1] has been used in this thesis. Changes were made in the Q and R values of the LQR method to remove the steady state error in altitude, which plagued [1]. The K gains used in [1] resulted in a steady

state error of 30 meters! Figure 3-3 below shows block diagram of the Longitudinal Autopilot.

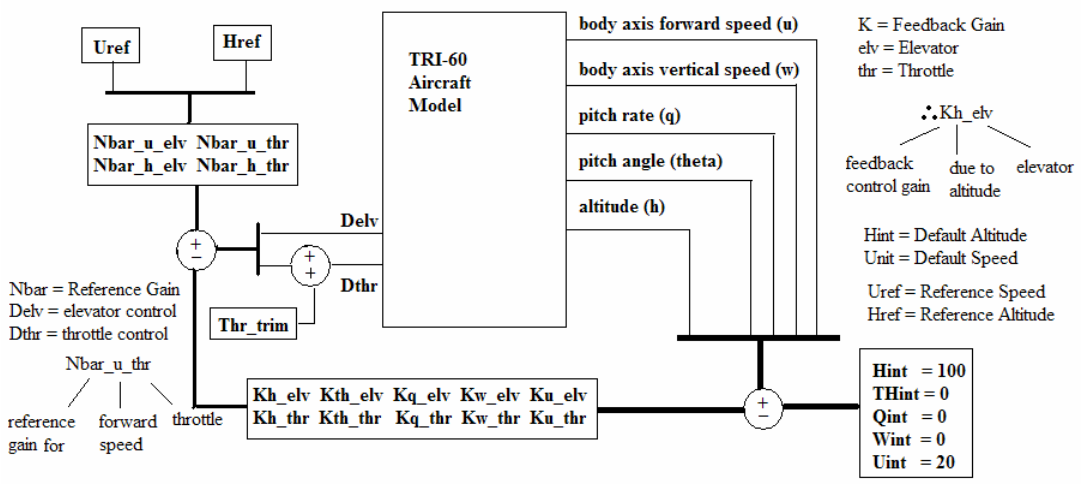


Figure 3-3: Block Diagram of the Longitudinal Autopilot

In this design of the Longitudinal Autopilot, the longitudinal states such as the forward speed (u), the vertical speed (w), the pitch angle (th), the pitch rate (q), and the altitude (h) are regulated about a default value. Deviations of any of the values from the default value (Hint, Uint) will cause appropriate negative feedback action (K Gain) from the throttle and the elevator according to the gain specified (e.g. Kth_elv = elevator feedback control due to pitch angle deviation, Kq_thr = throttle feedback control due to pitch rate deviation). Similarly the reference gains (Nbar) are used to track the given reference without in a one-to-one ratio (e.g. Nbar_h_elv = elevator control to track the altitude in a one-to-one ratio). The gains are found using LQR state feedback method that finds an option solution by minimizing the cost function:

$$J = \int \left(x'Qx + u'Ru + 2 * x'Nu \right) dt \qquad (3.3)$$

Subject to the state dynamics x = Ax + Bu. A and B specify the state space representation of the longitudinal motion of the aircraft as shown below:

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -0.20291 & 0.30323 & 0 & -9.81 & 0 \\ -1.6639 & -8.7996 & 20 & 0 & 0 \\ 0.92207 & -4.9047 & -28.523 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 20 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \end{bmatrix} + \begin{bmatrix} 0 & 51.5 \\ -13.342 & 0 \\ -162.75 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_E \\ \delta_T \end{bmatrix} \quad (3.4)$$

$$\begin{bmatrix} u \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_E \\ \delta_T \end{bmatrix}$$

The above state space model show that the action of the throttle affects only the forward speed, whereas the action of the elevator affects the vertical speed and the pitch rate. A more general model bas both the throttle and the elevator controlling the forward speed, the vertical speed and the pitch rate to varying degrees. The control input depends on the following equation:

$$\begin{bmatrix} \delta_E \\ \delta_T \end{bmatrix} = \begin{bmatrix} 0.0160 & -0.0289 \\ 0.0364 & 0.0127 \end{bmatrix} \begin{bmatrix} 0 \\ \Delta h \end{bmatrix} - \begin{bmatrix} -0.00633 & -0.0181 & -0.0540 & -0.646 & -0.0289 \\ 0.0341 & -0.00591 & 0.00249 & 0.202 & 0.0127 \end{bmatrix} \begin{bmatrix} u-20 \\ w \\ q \\ \theta \\ h-100 \end{bmatrix} \quad (3.5)$$

When the reference altitude (Href) and speed (Uref) are zero, the regulatory effect of the state feedback keeps the longitudinal states close to their default values. However, when tracking is required, the regulatory and the tracking portion of the state feedback compete for using of the control inputs, with the tracking portion of the state feedback trying to bring the required states to their new reference values, while the regulatory portion of the state feedback trying to keep the states close to their default values.

The second parameter which may constrain flight path planning is the rate of ascent or descent. To realize this parameter, the Longitudinal Autopilot of the aircraft was extensively simulated, since this autopilot is responsible for controlling the speed and altitude. Figure 3-4 shows the maximum simulated pitch angles and rates allowed when commanding the Longitudinal Autopilot to ascend or descend by a reference amount. Figure 3-5 shows the maximum simulated climb rate and the nature of the ascent or descent.

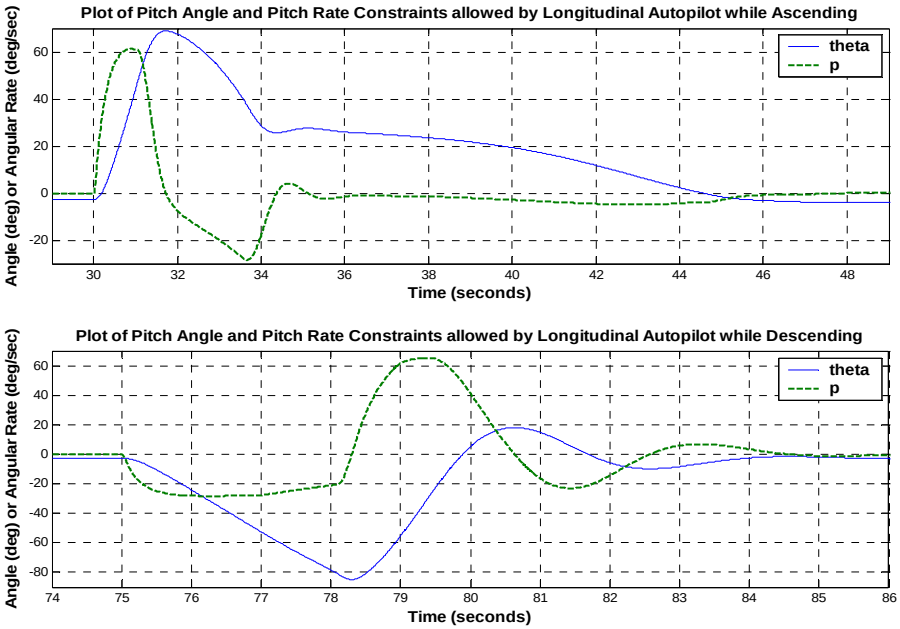


Figure 3-4: Autopilot Pitch Angle and Pitch Rate Constrains

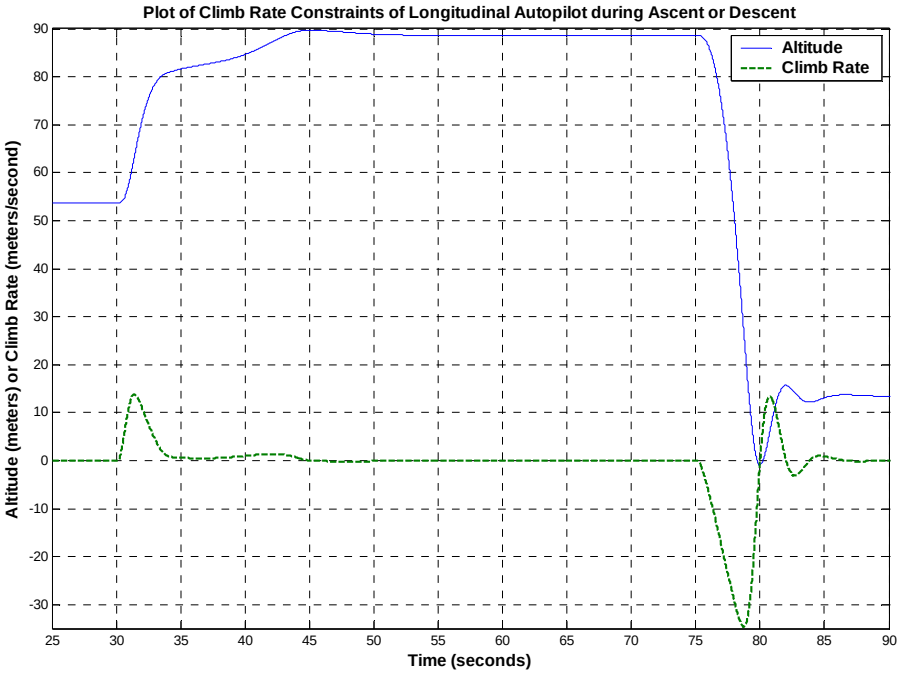


Figure 3-5: Autopilot Climb Rate Constraints and Behavior of Ascent and Descent

The LQR State Feedback Longitudinal Autopilot allows a maximum reference command of 35 meters up and 75 meters down and at a very reactive Q and R setting. Violating the upward and downward command reference command will destabilize the aircraft. Forcing the longitudinal autopilot to accept the more than 35 meter

ascend command will cause the autopilot to command an upward pitch of more than 60 degrees so that UAV can ascend as quickly as possible. When an aircraft is pitched upwards, the forward thrust has to support a greater portion of the weight of the aircraft, causing the forward speed of the aircraft to decrease significantly. The upward pitch and loss in speed are only transients; even so, if the transient effect does not disappear quickly enough, the aircraft will loose forward speed completely, and will stall and then undergo severe rolling following by yawing motion.

On the other hand forcing the Longitudinal Autopilot to accept more than 75 meters descent will cause it to command a downward pitch of 90 degrees (nosedive). The gravitational effect along with the thrust will cause the speed to increase significantly. There is a high chance that at that pitch attitude the aircraft may roll uncontrollably, with no other forces to counteract and stabilize the aircraft.

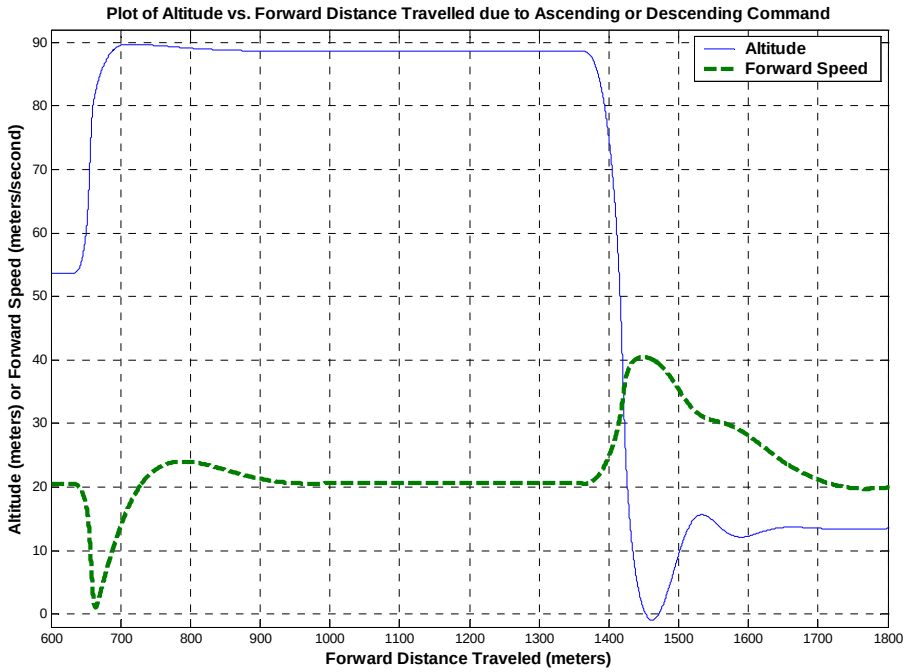


Figure 3-6: Distance Constraints to Reach Steady State after Ascent or Descent

The maximum climb rate allowed by the Longitudinal Autopilot is 15 m/s; where as the maximum rate of descent is as high as 35 m/s. The ascending motion of the aircraft is first order in nature with a time constant of ~10 seconds, and takes 15 seconds to settle, by which time the aircraft has moved a **forward** distance of 50m; where as the descending nature of the aircraft is a second order system with a fall time of 2.5 seconds, an overshoot of 20% and a settling time of ~15 seconds. Assuming that the forward speed is maintained at an average of 20 m/s, the aircraft travels some

300m before changes in altitude reaches *steady state* values as shown in Figure 3-6. It is important to keep in mind the overshoot while planning any descending motion, since even though the steady state altitude is above ground level, the overshoot might just crash the plane. The Longitudinal Autopilot acts to remove any changes in speed caused due to the changes in altitude or due to banking, so that except for transients the speed of the aircraft is constant.

3.1.3 Lateral Autopilot Constraints [25]

There were two different lateral autopilots tested for this thesis, one a PID control lateral autopilot that uses the roll (phi) as the proportional control and the roll rate (p) as the roll damper control for the ailerons, and the yaw rate (q) as the yaw damper control for the rudder. The PID Lateral Autopilot completely ignores the other states that form a part of the lateral motion of the aircraft such as the body frame side velocity (v), the body frame side acceleration (yb), and the track error. The other autopilot tested is the LQR state feedback autopilot that can control (or regulate) the roll (phi), roll rate (p), yaw rate (r), the body axis side velocity (v), the heading error (dPsi), the lateral track error, and the lateral acceleration (yb). However the model used for the state feedback in this thesis consists of only the roll angle, roll rate, yaw rate and the side velocity. Both autopilots can be commanded with a turn. Control of the heading error is handled using the Trajectory Tracker, while the concept of "track error" is not important for Feedback Guidance as implemented in this thesis, and the side acceleration is completely neglected since it is important only if the UAV must move through a narrow corridor. Figure 3-7 shows the Lateral Autopilot implementation using both methods.

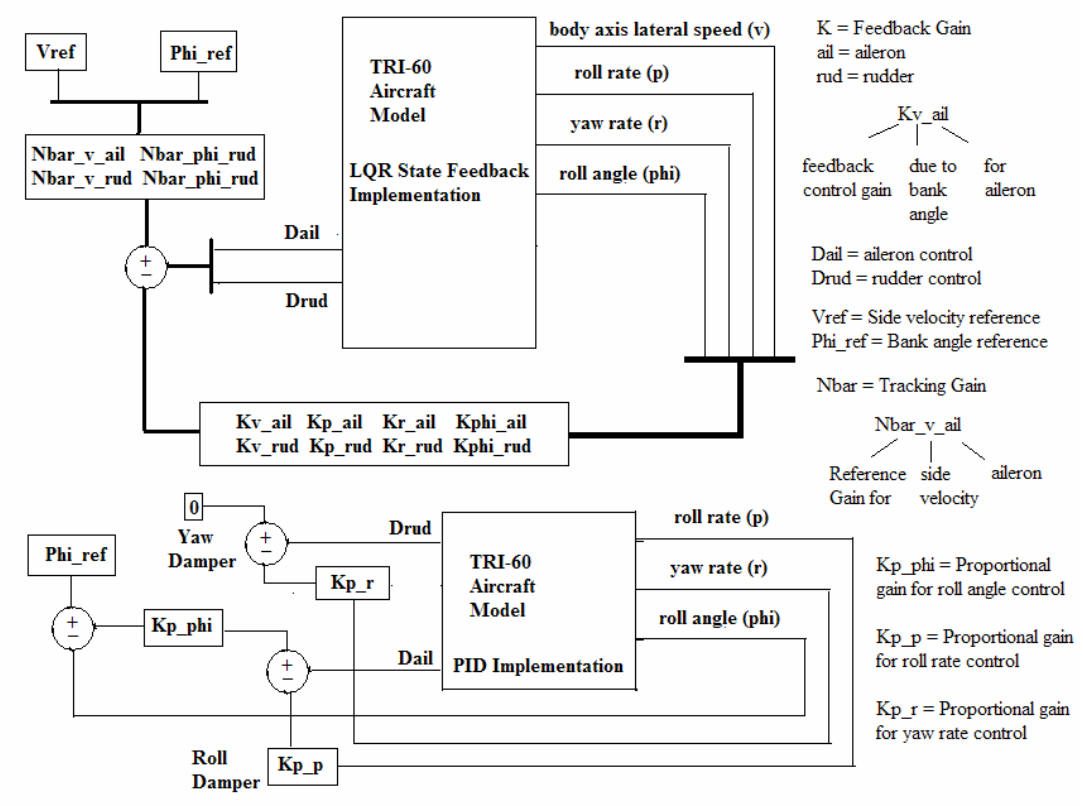


Figure 3-7: Block Diagram of the Lateral Autopilot Implementation using Two Methods

The PID controlled Lateral Autopilot is designed by creating a roll and yaw damper to minimize the roll and yaw rate respectively. The damper is a negative feedback mechanism that controls the aileron (Kp_p) and the rudder (Kp_r) using proportional control. The bank angle is controlled using a bank angle reference (Kp_phi) using proportional control as well. The following equations give the control input calculations using the PID method:

$$\begin{aligned} \delta_A &= K_\phi(R\phi - \phi) - K_P p \\ \delta_R &= -K_R r \end{aligned} \qquad (3.6)$$

Unlike in the longitudinal autopilot where at least two states were regulated around a default value (Hint and Unit), the lateral autopilot state feedback control tries to minimize all the states to zero. The sum of the feedback gains (K gain) of all four regulated states form the control action for both the aileron and the rudder (e.g. Kphi_rud = feedback control for the rudder due to deviation of the bank angle from zero). Similarly the reference gain (Nbar) is used to track the desired parameter in a one-to-one ratio (e.g. Nbar_phi_ail = aileron control to track the bank angle in a one-to-one ration).

The equation below describes the state space model for the lateral motion of the TRI-60 aircraft:

$$\begin{bmatrix} \dot{v} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -0.44275 & 0 & -20 & 19.56 \\ 0.67288 & -112.39 & 0.41395 & 0 \\ 14.457 & -0.37464 & -10.371 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & 2.8952 \\ 1178.4 & -0.2707 \\ -93.573 & -123.11 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_A \\ \delta_R \end{bmatrix} \quad (3.7)$$

$$\begin{bmatrix} v \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ p \\ r \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_A \\ \delta_R \end{bmatrix}$$

The state space model describes a lateral motion where the roll and the yaw rate depend on the aileron and the rudder inputs; where as the side velocity (v) depends only on the rudder. The roll angle is a derived or secondary state that depends solely on the roll rate. The outputs of the system to be tracked primarily are the roll angle and the side velocity. State feedback gains for the lateral state model are calculated using the LQR method. The regulatory portion of the state feedback controller attempts to minimize the magnitude of the side velocity, the roll and the yaw rate, and the roll angle (i.e. bring them to zero), whereas the tracking portion of the state feedback attempts to set the output to the commanded reference, so they compete against the other for the control inputs.

$$\begin{bmatrix} \delta_A \\ \delta_R \end{bmatrix} = \begin{bmatrix} 0.0051909 & 0.10008 \\ 0.12231 & -0.083085 \end{bmatrix} \begin{bmatrix} 0 \\ \Delta\phi \end{bmatrix} - \begin{bmatrix} 0.0055873 & 0.043025 & -0.029039 & 0.12849 \\ 0.0038113 & -0.00069341 & -0.046987 & 0.043444 \end{bmatrix} \begin{bmatrix} \mathrm{v} \\ \mathrm{p} \\ \mathrm{r} \\ \phi \end{bmatrix} \quad (3.8)$$

The following is a description of the constraints of the two autopilots during when the aircraft banks for a turn. Simulation shows that the aircraft can be banked at an angle of **60 degrees** and still maintain stability. However a 60 degrees is quite a severe banking action because the aircraft is traveling at 22.25 m/s, and an angular rate of 45 degrees/sec, in a circle of radius ~30 meters. This means the airframe must endure a crushing acceleration of 1.89G (approximately two times earth gravity). In this case stability does not necessary ensure safety. Therefore the constraint on the maximum bank angle is 40 degrees, since the aircraft looses only 2.5 meters of altitude, before stabilizing. At that bank angle, the aircraft travels at 21.25 m/s second at an angular rate of 22.5 degrees/sec around a circle or radius 57.5 meters, and thus has only 0.9G accelerating force (approximately earth gravity) acting on its body.

The aircraft also reaches just about steady state in 15 seconds, which is the time it takes the aircraft to fly one complete circle. Note that the figures described above merely state the maximum possible limits to each of the autopilots. Generally speaking during actual flight conditions, the autopilots will be commanded with more reasonable references such as +/- 10 degree roll or +/- 10m change in altitude. Figure 3-8, 9, and 10 shows simulated characteristics of the TRI-60 aircraft as it enters and exits a turn.

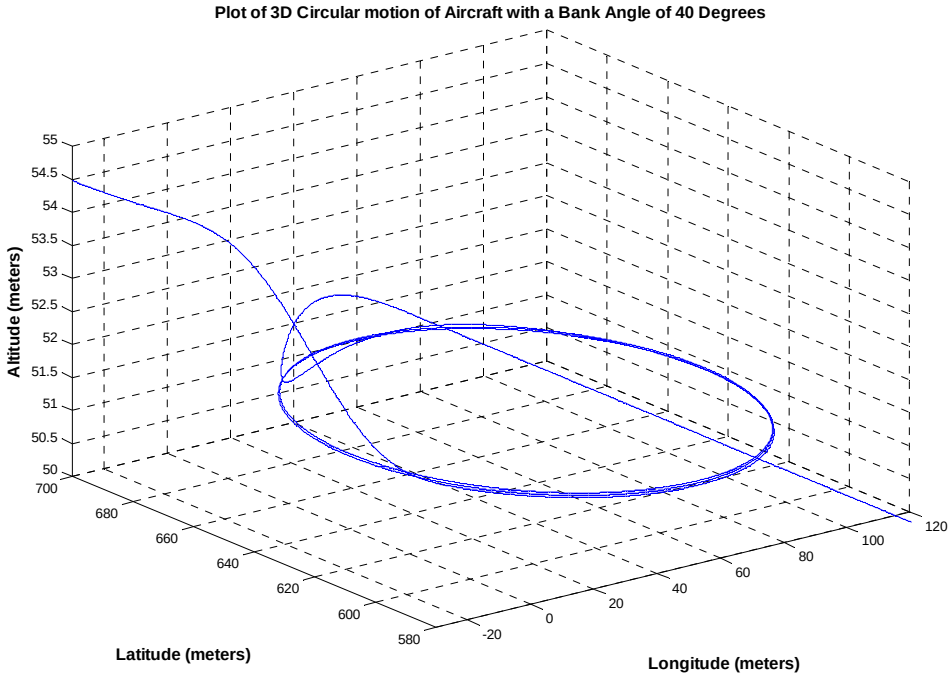


Figure 3-8: Plot of 3D Circular Path Taken by TRI-60 due to Roll of 40 Degrees

Simulations prove that the loss of altitude is the same regardless of the type of Lateral Autopilot being utilized. However as will be shown in later chapters, PID control Lateral Autopilot is more "reactive", in the sense that it tries to quickly attain the commanded reference with minimum steady state error, where as the LQR state feedback Lateral Autopilot provide a smoother transient response. However the lateral motion of the aircraft linearized around 20 m/s have a spiral mode frequency of 18 Hz, and therefore cannot be implemented in real time as state feedback control unless the controller offers a control loop frequency of at least 36 Hz.

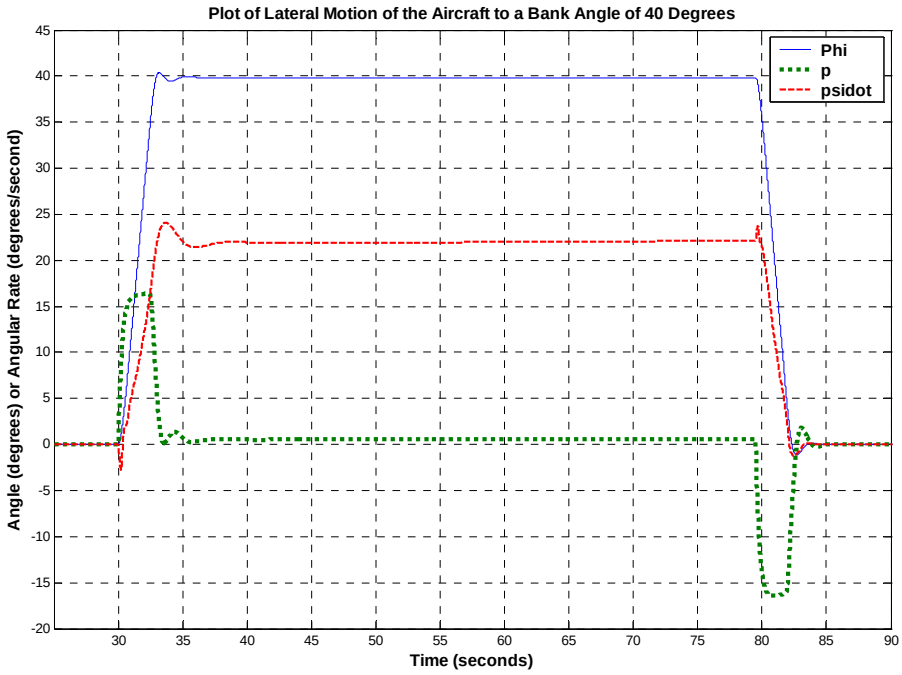


Figure 3-9: Plot of Angular Rate and Bank Angle during Entry and Exit of a Circular Path

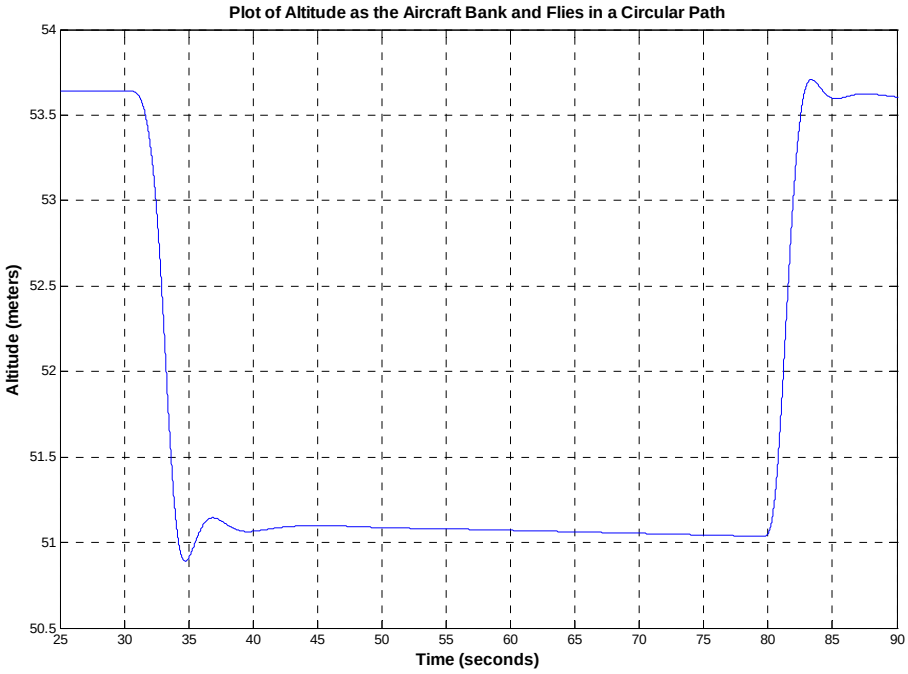


Figure 3-10: Plot of Altitude over Time as Aircraft Enters and Exits a Turn

Table 3-1: List of Constraints considered by Trajectory Tracker during Path Generation

Constraint	Range/Value	Effect of Constraint Violation
Maximum Pitch	+60 degrees to -85 degrees	Violating the upward pitch angle during an ascending motion will stall the aircraft since it takes too long to decrease the upward pitch, and will resulting in severe yawing motion which is hard to recover from
Maximum Turn Rate	20 degrees per second	At this turn rate at 21.9 m/s velocity, the minimum radius of the circular path is 61 meters. The aircraft experiences 0.73G force centripetal acceleration.
Maximum Roll	+40 degrees to -40 degrees	Higher roll will force aircraft into tight and potentially damaging turn involving high G acceleration
Maximum Climb Rate	+10 m/s to -35 m/s	The maximum climb rate is limited by the finite thrust that can be generated by the propeller for the given weight of the aircraft
Maximum Commanded Altitude Change	+35 meters to -75 meter	Commanding more than 35 meters at once will cause the aircraft to loose forward speed and stall since the transients of the longitudinal autopilot do not disappear fast enough
Ascending Motion	50 meters, 15 seconds	First order motion with time constant of 10 seconds. The UAV will travel a forward distance of 50 meters before it can reach the commanded altitude
Descending Motion	300 meters, 15 seconds	Second order motion with fall time of 5 seconds, overshoot of 20%. Should ensure the overshoot does not reach or go below ground level!! The UAV will travel a forward distance of 300 meters before it can reach the command altitude.
Minimum Distance Between Waypoints	300 meters	Must factor in the recovery distance after exiting a turn, plus the distance required for reaching steady state after descending or ascending, else the altitude requirements won't be met by the time the aircraft reaches a waypoint.

To summary Table 3-1 lists the following autopilot constraints which must be maintained during any flight maneuver but most likely during normal UAV operation, they should even reach these constraints.

3.2 Shortest Flight Algorithm in 3D

The path algorithm used is the natural extension of the Feedback Guidance Strategy using bang-bang control methodology [13] and shown in Figure 1-7. Feedback guidance strategy is a three-step process. It consists of straight lines and circular curves in each leg of the UAV mission. The UAV will approach a waypoint following a straight line, upon which it will execute one of the three maneuvers:

- If the next waypoint is within ± 10 degrees of the UAV current heading, it will bank proportional to the error in the heading and try to minimize the error before following a straight line to the next waypoint.
- If the next waypoint is more than 10 degrees of the UAV current heading, the UAV will fly a circular curve until it faces the next waypoint, and then fly straight to the waypoint.
- The UAV will continue in a circular path as long as the heading error remains more than +/- 10 degrees. Once the bank angle reaches to within +/-10 degrees, it will straighten. The time it takes for the UAV to straighten is enough to cover the 10 degrees.

Feedback Guidance Strategy will take the aircraft to its destination waypoint, but it cannot control the heading of the aircraft once it does reach its waypoint. The bearing of the aircraft depends on the angle at which the current heading of the aircraft matches the instantaneous destination heading. An extension of the Feedback Guidance Strategy, called the bidirectional Feedback Guidance Strategy, will be used to control its heading of the UAV when it reaches a waypoint as shown in the Figure 3-11 below. The strategy is called “bidirectional” because the path created is a combination of Feedback Guidance Strategy for the aircraft moving both ways, from point A to C and then from point C to A. Using this strategy, the aircraft moving from point A to C will face point C by the time it reaches point B, where as the aircraft moving from point C to A will face point A by the time it reaches point B. More details about this strategy will be given later in this section.

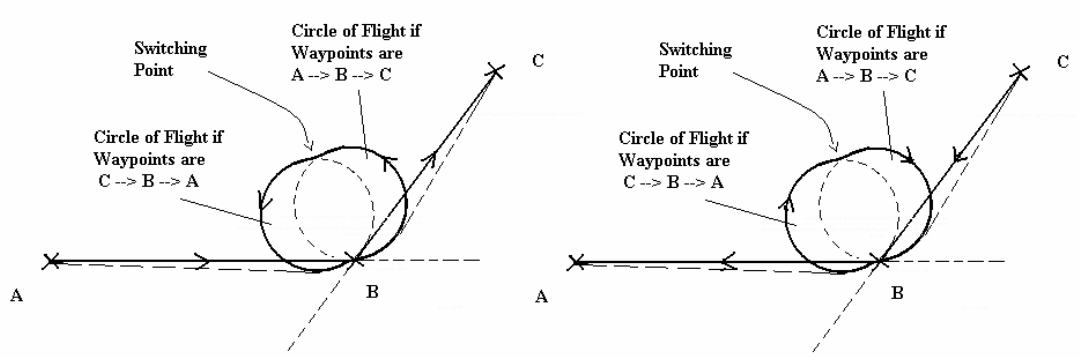


Figure 3-11: Extension of the Feedback Guidance Strategy – Bidirectional

Finally it must be noted, that in this strategy, there are four points where the control switches from straight line to circular path and back. The first area is when the aircraft enters the first circle, the second point is when the aircraft leaves the first circle, the third point is when the aircraft enters the second circle, and finally the last point is when the aircraft leaves the seconds circle. Due to this the path is not minimal in nature, since it involves flying over two circles at every waypoint. The final item to consider is altitude control, which is done when the UAV is flying in near straight lines.

3.2.1 Feedback Guidance Strategy using Bang-Bang Control Scheme [24]

Using the physical constraints described in the previous section, the kinematics model of the aircraft can be defined as:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix}=\begin{bmatrix}\cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi-\cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi+\sin\phi\sin\psi\\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi+\cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi-\sin\phi\cos\psi\\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}u\\ v\\ w\end{bmatrix} \quad (3.9)$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

Where $\dot{x},\dot{y},\dot{z}$ are the changes in the position in the inertial axes given the angular rate p, q, and r and the velocity u, v, and w of the aircraft in its body axes. The turn rate $\dot{\psi}$ is the turn planar 2D turn rate and must be smaller or equal to the maximum turn rate M, which is given by:

$$M=\frac{g\sqrt{n^2-1}}{u},\quad n=\frac{L}{W} \quad (3.10)$$

Where n = load factor of the UAV, L = total lift, W = weight of UAV, and u is the velocity. For the TRI-60 RC aircraft, M equals to 22.5 degrees per second at a speed of 20 m/s and a bank angle of 40 degrees. The climb rate $\dot{z}$ must be less than the maximum stable climb rate allowed by the longitudinal autopilot, which was found in the previous section to be +15 m/s to –35m/s.

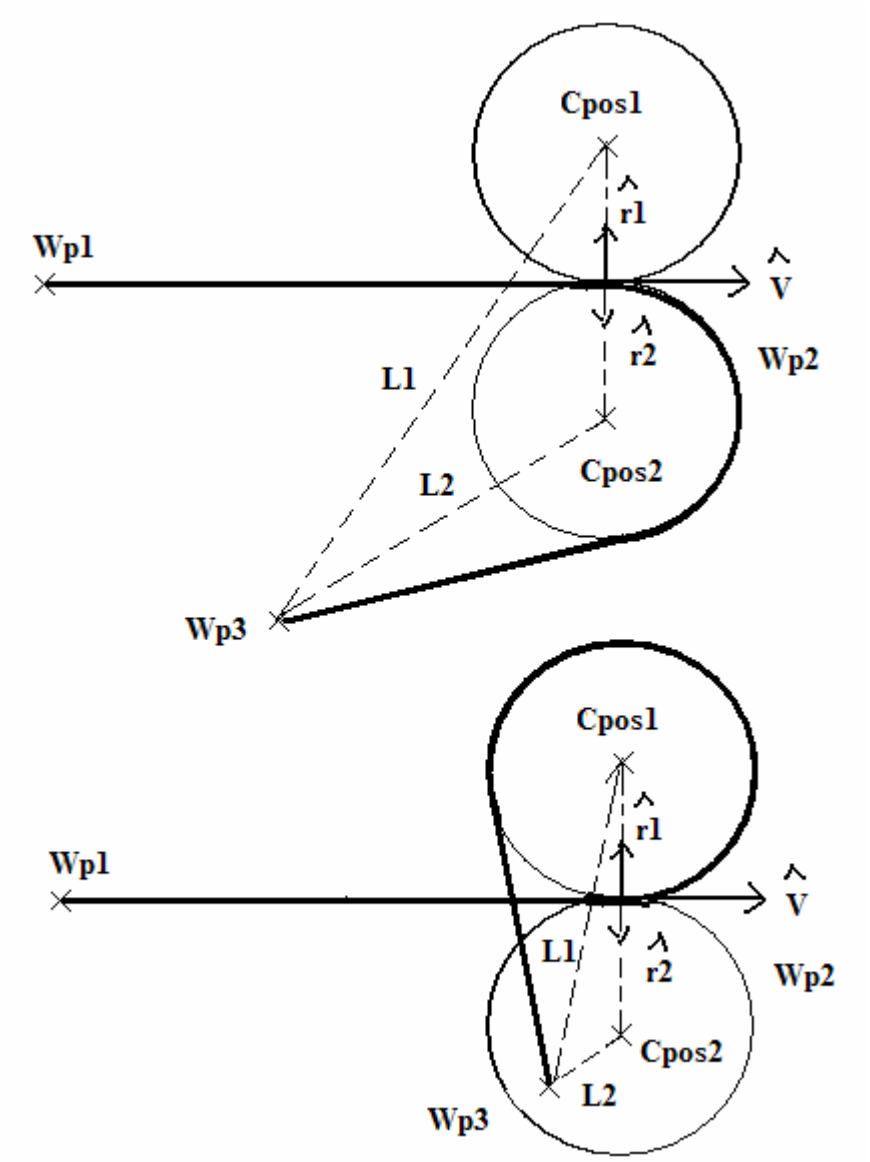


Figure 3-12: Path taken by the UAV When Entering a Curve using Feedback Guidance

Figure 3-12 above shows the decision taken by the path algorithm when turning. Recall that using this strategy, the aircraft once it reaches a waypoint, starts to turn until it faces the next waypoint, and then fly straight to the next waypoint. But as shown in Figure 3-12, there are potentially two circles the aircraft can fly over (i.e.

the aircraft can choose to turn in the clockwise or counterclockwise direction). This decision to turn in a particular direction is based on the distance from the center of the circular arc to the next waypoint. The center of the circular arc is given by:

$$C_{POS} = W_{P2} + R * \hat{r} \tag{3.11}$$

Where W_{P2} is the position of the waypoint where the turn is taken (waypoint 2), C_{POS} is the position of the center of the circular arc, R is the minimum radius of the circular arc as limited by the maximum turning rate of the aircraft, and $\hat{r}$ is the direction vector of the center of the circular arc from W_{POS}. If the direction vector of the current path (or the velocity vector) is given by $\hat{v} = a\hat{i} + b\hat{j}$ then:

$$\hat{r}_1 = b\hat{i} - a\hat{j} \text{ Or } \hat{r}_2 = -b\hat{i} + a\hat{j} \tag{3.12}$$

The distance between the centers of the two circular arcs and the next waypoint (waypoint3) is found (L1 and L2). Notice from Case 1, the UAV will fly over the circular arc whose center is nearer to the next waypoint. On the other hand, if the distance between the center of the circular arc and the next waypoint is smaller than R (Case 2), then the UAV will fly over the circular arc whose center is farther from the waypoint, since the maximum turn rate would prevent the UAV from reaching the next waypoint were it to turn towards the nearer circle. Whichever way the UAV turns, the direction of turn (clockwise or counterclockwise) depends on the cross product of the velocity vector of the current path $\hat{v}$ and the direction vector to the center of the circular path ($\hat{r}$) chosen to fly over. A negative cross product signifies a clockwise turn (CW), whereas a positive cross product signifies a counterclockwise turn (CCW).

Once the UAV is in a turn, it will iteratively compute the heading from its current position in the arc to the next waypoint, and compare this heading with its own current heading. The heading between the current position in the arc and the next waypoint is computed by:

$$H_P = \tan^{-1}\left(\frac{Y_N - Y_C}{X_N - X_C}\right) \tag{3.13}$$

Where X_P, Y_P are the coordinates of the next waypoint and X_C, Y_C are the instantaneous current coordinates while flying over the circular arc. Note: For navigational purpose, X axis is used for the Latitude, while the Y axis is used for the Longitude. The heading is computed from North in a clockwise direction. The

current heading of the UAV is taken from the compass. If the two headings are within ± 10 degree, then the turn is completed and the UAV flies straight to the next waypoint. This algorithm, although providing a minimum flying path, has the constraint that the UAV heading when it reaches the next waypoint cannot be controlled.

3.2.2 Bidirectional Feedback Guidance Strategy to Control Heading [25]

A secondary theme of this thesis is to control the heading of the aircraft once it reaches a waypoint. This algorithm does not produce time optimal paths since it forces the aircraft to take a circular path for a longer period of time. However, it is the most efficient way of controlling the heading of the aircraft at a given point. For example a command like go to lat 24 degrees 53 minutes 25.6 seconds, longitude 55 degrees 22 minutes 11.4 seconds and face a heading of 90 degrees (East) will be valid. The strategy described is an extension of the Feedback Guidance applied both ways. Figure 3-13 below introduces this concept; however the mathematical formulation is described below.

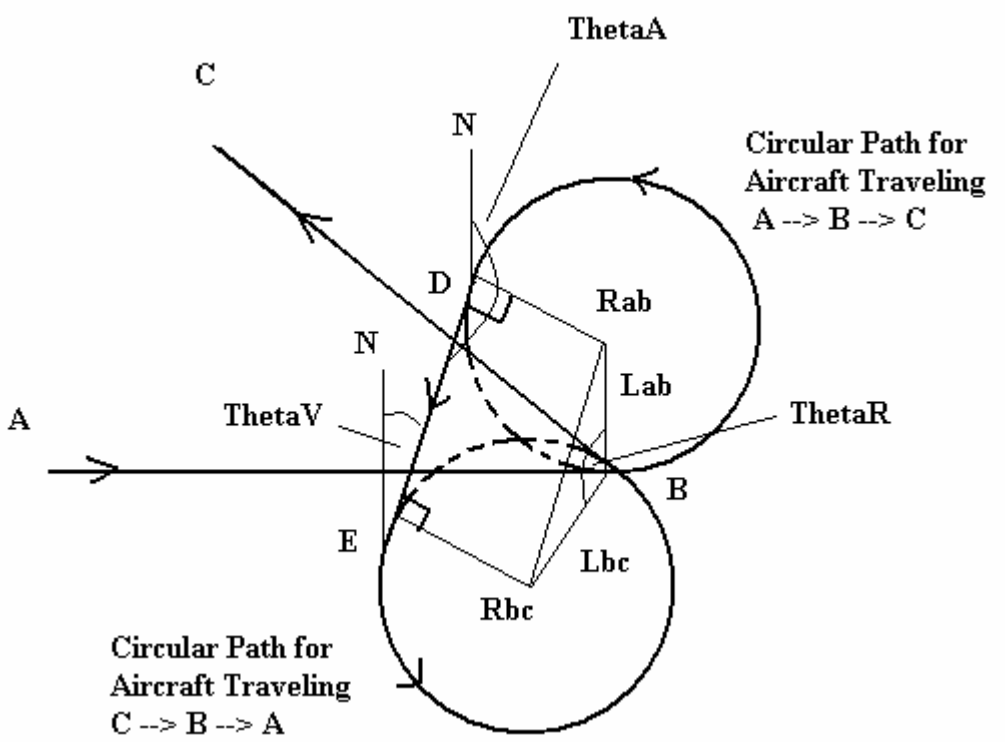


Figure 3-13: Detailed description of Bidirectional Feedback Guidance Strategy

Feedback Guidance Strategy is used to draw the familiar circle that the aircraft will enter when traveling from points A to B to C. But instead of exiting the circle

when the aircraft faces the next waypoint, it continues in the circle. At the same time another circle is drawn according to Feedback Guidance Strategy; this time as if the aircraft was traveling in the opposite direction from points C to B to A. The switching point is identified by that point when the current direction of the aircraft is exactly 180 degrees (opposite) from the theoretical current direction of the phantom aircraft were it traveling in the opposite direction. In the switching point the aircraft leaves the current circle, and fly straight until it intersects the other circle, whereupon it enters the other circle and continues on until it faces the direction of the next waypoint. That point also coincides with the moment when the aircraft reaches the current waypoint. The length of distance that aircraft travels in the straight lines between the circles is given by the distance between the centers of the two circles. The center is found according to the Feedback Guidance Strategy formulation. Equation 3.14 below gives the condition when the second switch occurs:

$$\begin{aligned} &\textit{if} \quad 0^0 <= \theta_A < 180^0 \quad \textit{and} \quad \theta_V = \theta_A + 180^0 \\ &\textit{elseif} \quad \theta_A = 180 \quad \textit{and} \quad \theta_V = 0^0 \\ &\textit{elseif} \quad 180^0 < \theta_A < 360^0 \quad \textit{and} \quad \theta_V = \theta_A - 180^0 \end{aligned} \tag{3.14}$$

If the above conditions are satisfied, then the second switching event takes place at Point D, and the aircraft enters a straight line. The aircraft continues until point E, and enters the previously virtual bottom circle. The distance between D and E is given by the following formula:

$$L_{AB}{}^2 + L_{BC}{}^2 - L_{AB}L_{BC}\cos(\theta_R) \tag{3.15}$$

Since the Radius of the circle is constant $L_{AB} = L_{BC}$ and the formula simplifies to:

$$L_{BC}{}^2[2 - \cos(\theta_R)] \quad or \quad L_{AB}{}^2[2 - \cos(\theta_R)] \tag{3.16}$$

The distance between D and E can be calculated by the above formula because the line DE is parallel to the line R_{AB} to R_{BC}. The GPS coordinate of Point E can be found from the GPS coordinate of Point D, the length of line DE and the heading of the real aircraft θ_A. Once the aircraft reaches point E, the third switch takes place and the aircraft enters the bottom virtual circle, and continues in a circle until the aircraft face the direction of the next waypoint whereupon the last switching takes place and the aircraft flies the straight line to Point C. Theoretically the last switching should

occur at Point B. Thus the aircraft can control its heading when it reaches a waypoint. The obvious disadvantage of this method is that the aircraft is forced to fly over two minimum radius circles (worst case scenario) every at waypoint, but that is the price paid to control heading. For practical results a tolerance of +/- 5 degrees will be used. This method is extremely useful if the UAV needs to loiter over a waypoint before heading to the next waypoint. Loitering is achieved due to the fact that the UAV passes over each waypoint twice before moving on to the next waypoint. The next section shows a more minimum time method to control the heading of the UAV.

3.2.3 Alternative method to Control Heading

An alternative method to bidirectional feedback guidance strategy to control heading when the UAV reaches a waypoint is the k-feedback guidance strategy. K-feedback guidance is a combination of k-trajectory and feedback guidance strategy as shown below in Figure 3-14 as compared with Figure 1-7 which depicts feedback guidance and Figure 1-9 which depicts k-trajectory. To fly this track, the UAV starts at Wp1 and flies in a straight line and then switches to a circular path at point C. The UAV then flies over circle A, until at point D, it switches over to circle B. The UAV flies in a circular path until it reaches Wp2 by which time it should be facing Wp3. At Wp2, the UAV switches from a circular path to a straight line.

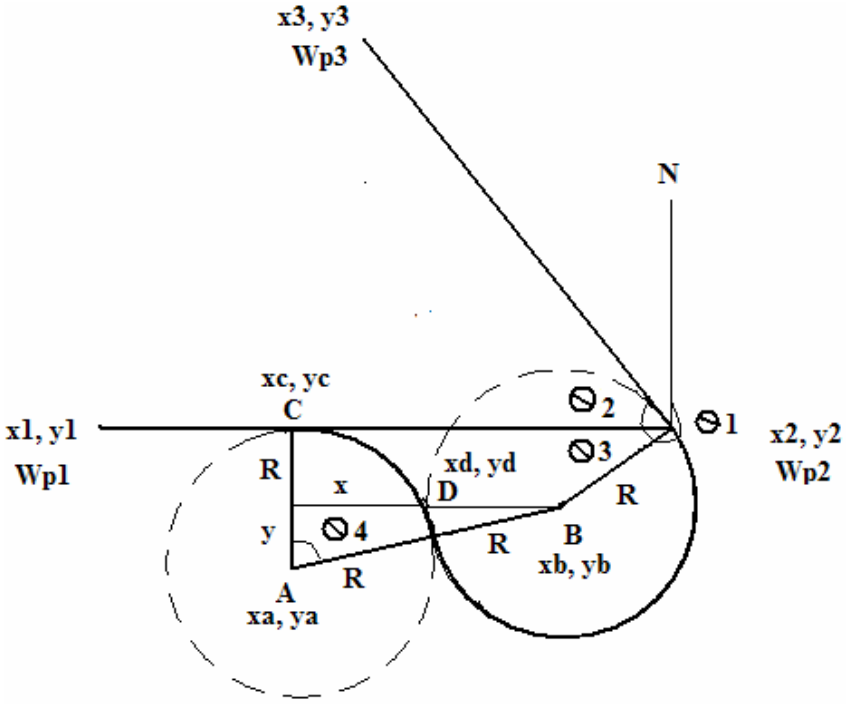


Figure 3-14: Detailed Sketch of k-Feedback Guidance

The above figure shows a method of navigation where there are three switching points (C, D, and Wp2), where only Wp2 is known. By applying trigonometry and geometry, points A, B, C and D can be ascertained. The bearing of Wp3 from Wp2 is θ_2, and $\theta_1 = \theta_2 - 90$ if the aircraft is to turn CCW or $\theta_1 = \theta_2 + 90$ if the aircraft is to turn CW. Using the coordinates of Wp2 (x2, y2), θ_1 and the radius of the circle R, the coordinates of B can be found:

$$\begin{aligned} x_B &= x_2 + R\cos\theta_1 \\ y_B &= y_2 + R\sin\theta_1 \end{aligned} \tag{3.17}$$

The centers circle A must be found in a two-step process. Using Equation 3.11 and 3.12, the centers of the two circles (xc1, yc1 and xc2, yc2) and are found from Wp2. [**Note**: The centers of the two circles are orthogonal to the direction vector $\hat{v} = a\hat{i} + b\hat{j}$ of Wp2 from Wp1]. The center of the circle closest to the point B is selected. The center (xn, yn) is then moved iteratively in the direction of $-\hat{v}$ in steps of $\frac{u_{MIN}}{f_{LOOP}}$ meters, where U_{MIN} is the minimum speed band of the aircraft and F_{LOOP} is the frequency of the control loop.

$$\begin{aligned} x_{N+1} &= x_N - \frac{u_{MIN}}{f_{LOOP}} a \\ y_{N+1} &= y_N - \frac{u_{MIN}}{f_{LOOP}} b \end{aligned} \tag{3.18}$$

Point A is located where the distance between the center of the circle (xn, yn) is more than or equal to 2R. Using a reverse form of equation 3.12, and the coordinates of Point A (xa, ya), the location of Point C (xc, yc) can be found. The geometric design is such that the location of Point D can always be found by averaging the coordinates of Point A and B. The maximum number of iteration to be carried is $\frac{2R * f_{LOOP}}{u_{MIN}}$ and occurs when Wp3 = Wp1 (i.e. when the aircraft must turn back full 180 degrees). For a radius of 250 meters and the minimum flying speed of 15 meters per second and a control loop frequency of 15Hz, the maximum number of iterations is 500.

Although this method is the best to control the heading of the UAV when it reaches a waypoint, it suffers from the fact that the waypoints must be separated by a

distance of C-Wp2 so the first switching point can be in front of the UAV when it reaches a waypoint.

3.2.4 Altitude Control for Paths in Three Dimensions

The final point of interest in this chapter is to decide whether the UAV should be commanded to change altitude when it is turning in a circle or when it is flying straight or both. Altitude needs to be changed due to the fact that the waypoints are given in three dimensions, so the UAV needs to ascend or descend to meet them. Ascending or descending during a turn will cause the UAV to follow a spiral path, whereas ascending or descending when the UAV flies straight will cause the UAV to follow a parabolic path. If the waypoint is far from the current waypoint (i.e. more than or equal to 300 meters from the point where the UAV exits a circle), the UAV will be commanded to change its altitude the moment it exits the turn and flies straight. This allows the UAV to track the change in altitude before reaching the next waypoint. However if the next waypoint is close by or less than 250 meters, then the UAV will be commanded to change its altitude the moment it enters a circle causing it to follow the spiral path.

The changes in altitude will be different in both cases. When the UAV is flying straight, the priority will be to track the change in altitude as fast as possible (Note: check Section 3.1.2 to find out the constraints of the Longitudinal Autopilot). However if the UAV is flying in a circle, the changes in altitude will be more gradual, with the UAV reaching the target altitude by the time it exits the circle. The reason for this is because the radius of the circular path is highly affected if the UAV is commanded to change altitude, and it is desired to keep these effects to a minimum. Figure 3-15 shows the effects in the radius of the circular path, when the UAV is commanded to change its altitude **drastically** during a turn. Circle A occurs as the UAV is increasing in altitude, and thus loosing airspeed in the process, thereby resulting in a small and tight circle. Circle B, on the other hand occurs when the UAV is decreasing in altitude, and thus gaining lots of airspeed, resulting in a large circle. Circle C occurs when the UAV remains at the same altitude, and does not experience much change in airspeed.

Figure 3-15 also shows that the UAV takes a very sharp turn when entering into a circular path when it is in the process of ascending or descending, as seen by the discontinuity in the paths of Circles A and to a lesser degree in Circle B. Such

discontinuity goes against the purpose of achieving a smooth trajectory. In Circle C the UAV faces no such discontinuity. Figure 3-16 below shows the 3D plots of the effects of the Longitudinal Autopilot increasing and decreasing the aircraft altitude while the Lateral Autopilot tries keeps the UAV flying in a circle.

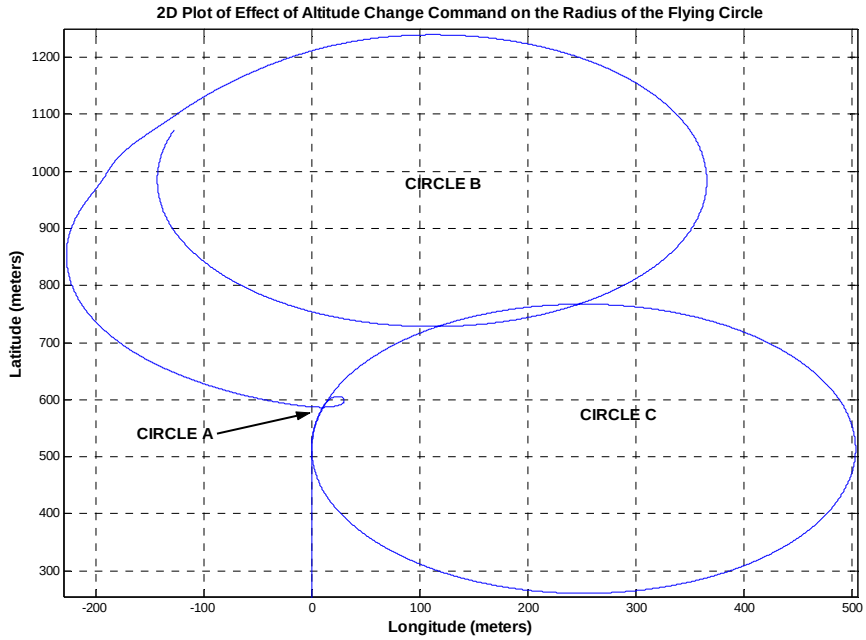


Figure 3-15: Effect of Altitude Change Command when UAV fly in a Circle

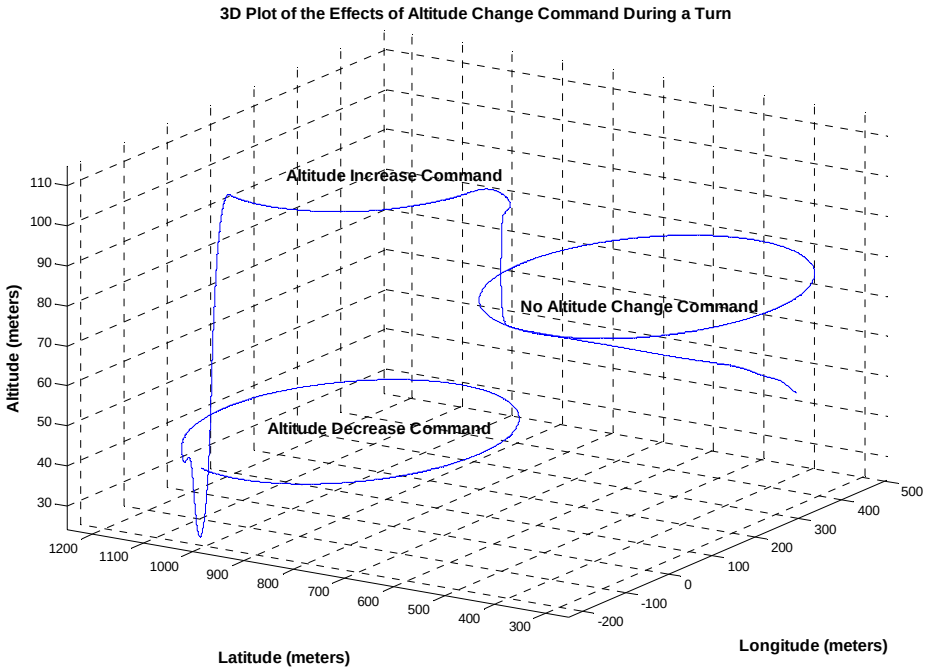


Figure 3-16: Effect of Altitude Change Command when UAV fly in a Circle

The final point to note is that while the UAV is ascending or descending, there is great variation in its velocity and its pitch, and therefore the UAV does not fly in a perfect circle, as evidenced in Circle A where the UAV is flying in a very small circle initially and then in a circle of increasing curvature during the time it takes the UAV to increase its altitude to the desired reference. On the other hand the UAV in Circle B is initially flying in a large circle and then in a circle of decreasing curvature during the time it takes the UAV to decrease its altitude. In Circle C, the curvature of the circle remains constant the entire time the UAV remains in the circle. To prove this point Figure 3-17 shows the changes in speed and pitch with time as the aircraft as it ascends and descends. The forward speed of the UAV decreases, while its pitch increases, when the UAV is ascending. This results in the overall forward ground speed to decrease significantly, thus the radius of the circle decrease. The reverse happens when the UAV descends.

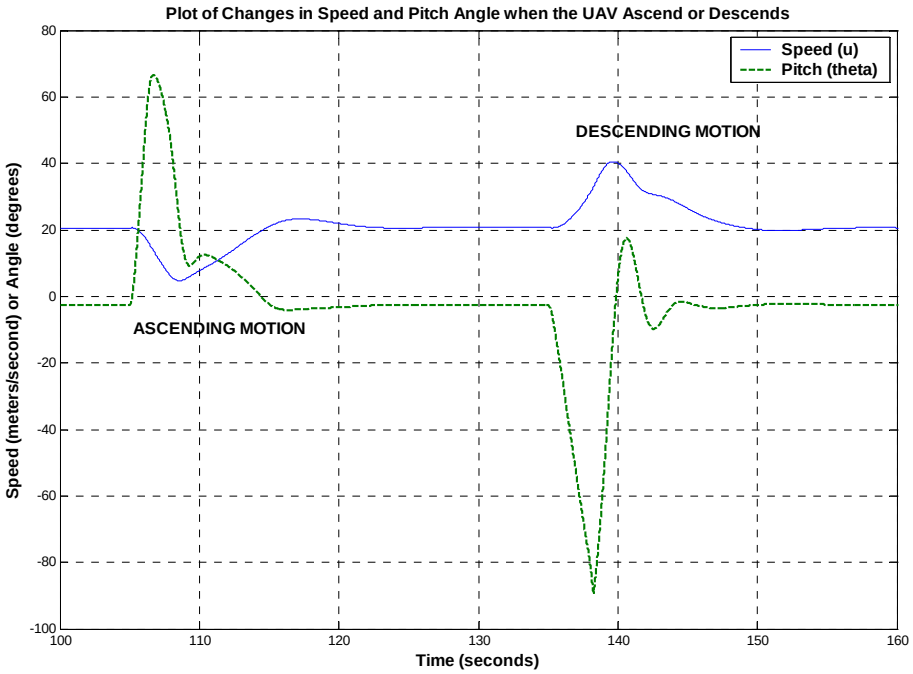


Figure 3-17: Changes in Speed and Pitch due to Ascending and Descending Motion

3.3 Dead Reckoning Algorithm via Inertial Navigation

For optimal performance the control loop, whether it be of the DFC or the trajectory tracker, needs to execute at a frequency of 12Hz or greater. This is

necessary for the DFC to maintain aerodynamic stability during flight despite continuously changing variables such as wind speed and direction, thermal activity and turbulences. This is due to the fact that the longitudinal state has a fast frequency of 3Hz. On the other hand, the control loop in the Trajectory Tracker also needs sufficient update rate, since the feedback guidance algorithm works ideally with continuous update rates. That is the main reason why a position update rate of 1 Hz as output from the GPS receiver is not enough. For a speed of 20 m/s and a flying radius of 50m, every second the UAV will traverse ~23 degrees within the circle. Therefore the controller is forced to decide on exiting the circle with an error of +/- 11.5 degrees. In this case if the next waypoint was 500m from the point the UAV exits the circle, then the UAV may miss the waypoint by more than 100m! However with a position update rate of 24 Hz, the error would be less than 5m. The following formula calculates this error:

$$D_E = \pm \frac{V * 360}{4 * \pi * R * F} \qquad (3.19)$$
$$error(m) = W_D * \tan(D_E)$$

Where De is the maximum error in degrees while exiting the circle, F is the frequency of position updates, V is the forward speed of the UAV, R is the radius of the circle, Wd is the distance to the next waypoint. Figure 3-15 below shows the effect of a discrete update rate.

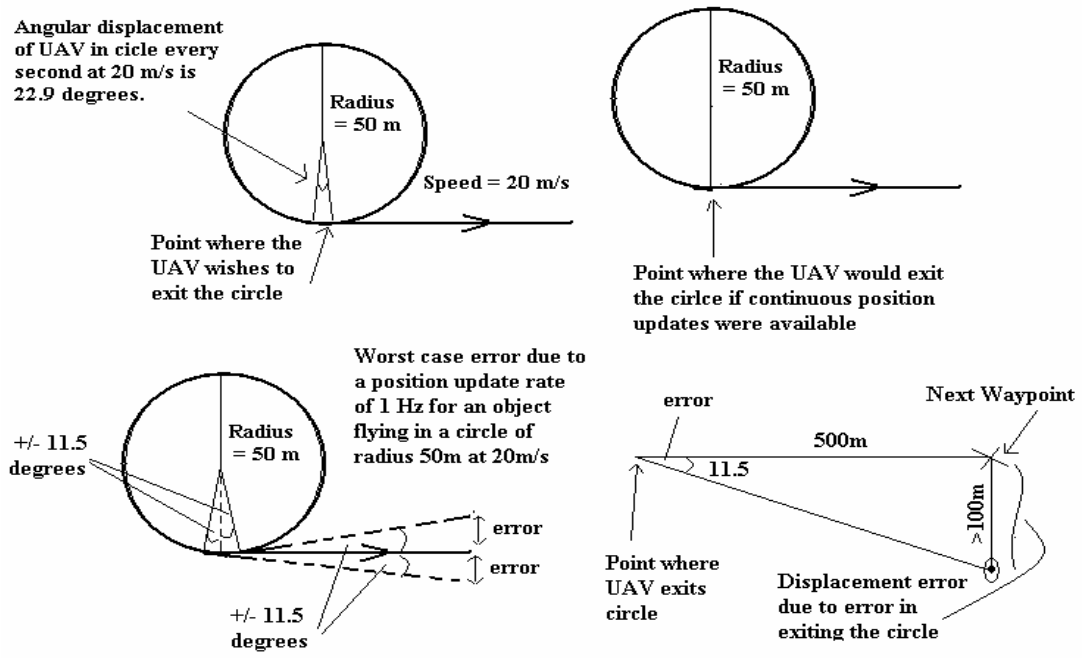


Figure 3-18: Schematic Representation due to Discrete Position Update of 1 Hz

Even when the aircraft is flying straight, it needs continuous position updates to increase the accuracy of waypoint navigation. With an update rate of only 1 Hz, the aircraft can track position only every 20 meters at nominal flying speed. This is why it is important to have a means of inertial navigation or dead reckoning to maintain continuous position update in between intermittent GPS updates. The dead reckoning algorithm utilized in this thesis is a simple one constructed out of the integration of velocities transformed from the body axis of the aircraft into the navigation axis utilizing the Euler angle (roll, pitch, and yaw) of the aircraft. Standard inertial navigation calculates both velocities and position of the aircraft by double integration of the acceleration of the aircraft transformed from the body axis into the navigation or the inertial axis. Inertial sensor such as the accelerometers and the gyroscopes are integral to this form of navigation. However due to lack of proper sensor orientation and packaging in the current Avionics Unit, it is impossible to utilize this form of inertial navigation. Therefore the velocities information is utilized for the dead reckoning algorithm; however this form of dead reckoning is not part of inertial navigation, since inertial sensors are not utilized. First a preview of dead reckoning through inertial means will be given, followed by the one used for this thesis.

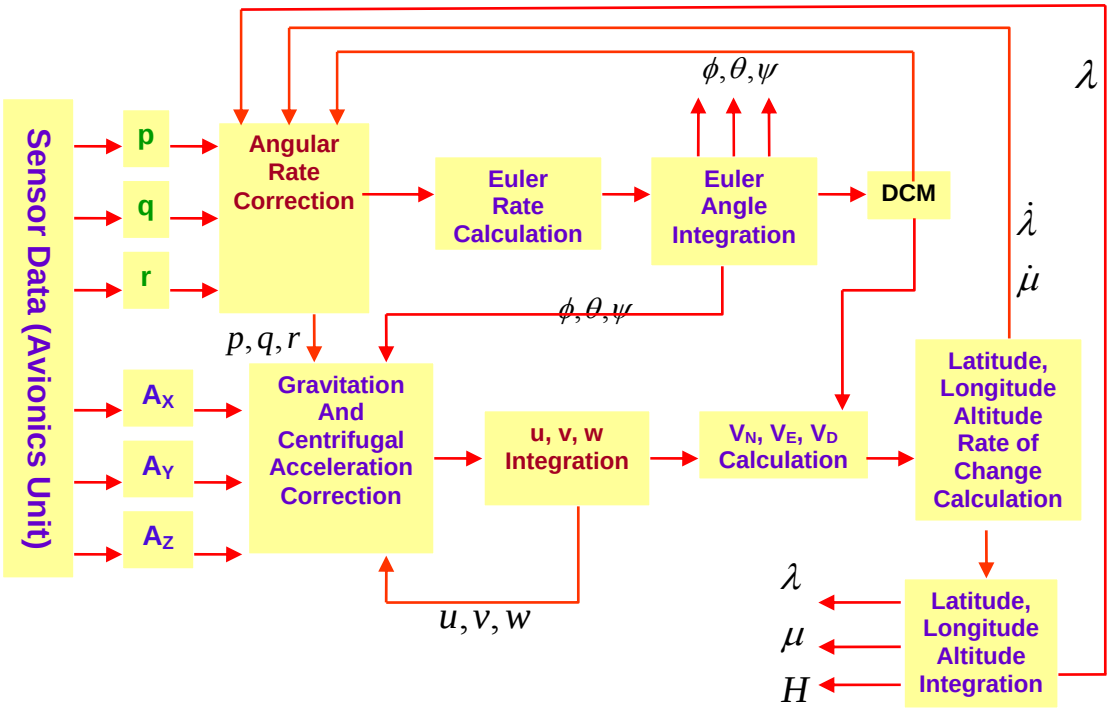


Figure 3-19: Inertial Navigation System High Level Overview [10]

Figure 3-19 and Figure 3-20 shows the high level overview of the Dead Reckoning system using strap-down inertial sensors (rate gyros and accelerometers). The basic idea in inertial navigation is to first resolve the acceleration (measured by the accelerometers) of the object in the inertial plane and then integrate the acceleration twice to give the position. To resolve the acceleration in the inertial plane, acceleration due to other sources such as the Earth's Gravity and the Coriolis forces due to the object's own motion must be removed. The Euler Angles play an important part in the resolution of the acceleration to the inertial axis, and they can be derived by integrating the "corrected" angular rate of the aircraft. The angular rate can be corrected by removing the effect of Earth's own rotation from the overall rate detected by the rate gyros.

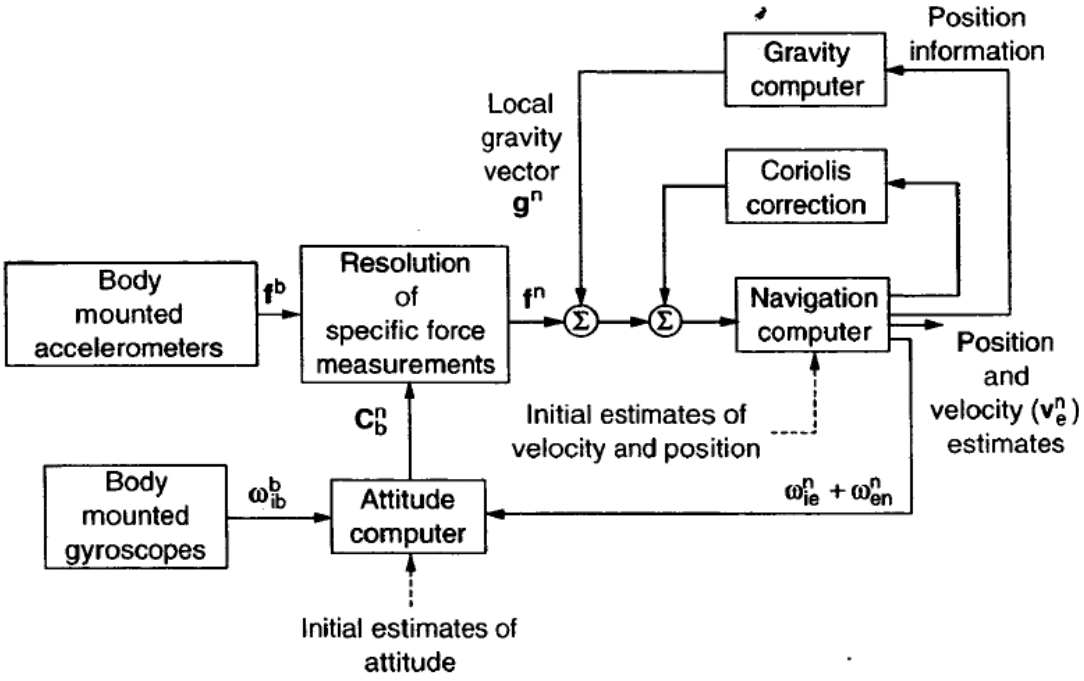


Figure 3-20: Inertial Navigation System Block Diagram in Earth Navigational Frame [27]

There are five basic steps involved in INS system:

- Measure angular rate using the rate gyro and remove the effects if Earth's rotation
- Calculate Euler Angles from the angular rates and create Direction Cosine Matrix (DCM)

- Measure acceleration using the accelerometer remove gravity and Coriolis acceleration
- Integrate the corrected acceleration to find velocity in the body frame
- Resolve the velocity in the inertial frame using the DCM and integrate to get position

Figure 3-21 below shows the terms involved in creating an INS system in the Navigation Frame of the Earth. The main points of interest are Ω, the Earths own angular velocity and λ, the current latitude.

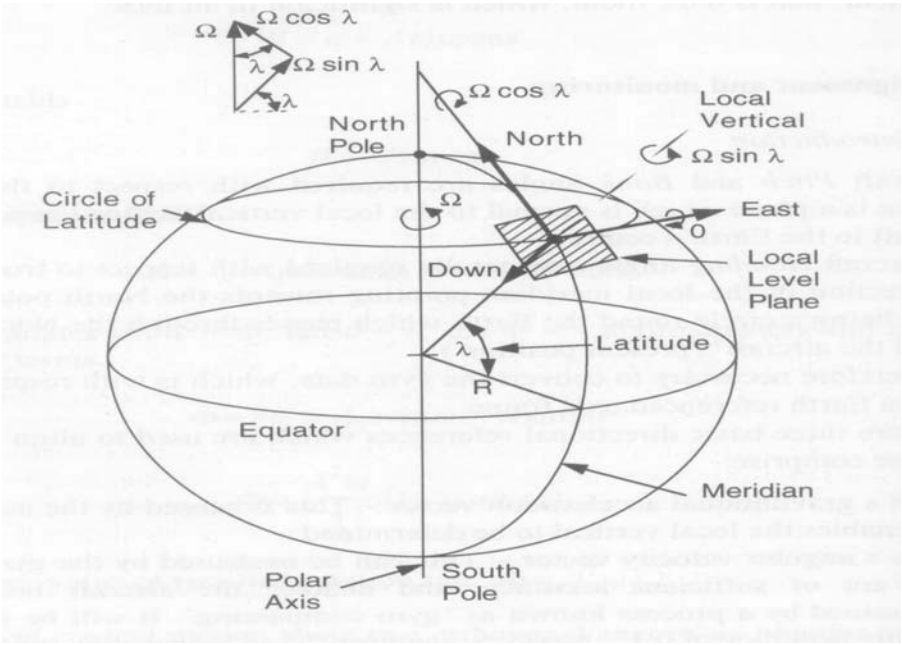


Figure 3-21: INS Navigation Frame terminology [27]

The equations below describe each step of the process of the INS algorithm. The equation to remove the effect of the Earth's angular rate from the measured body axis rate is given by:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}_M - DCM \left[\begin{pmatrix} \Omega\cos\lambda \\ 0 \\ -\Omega\sin\lambda \end{pmatrix} + \begin{pmatrix} \dot{\mu}\cos\lambda \\ -\dot{\lambda} \\ -\dot{\mu}\sin\lambda \end{pmatrix} \right] \quad (3.20)$$

Where Ω is the Earths angular rate, λ is the latitude and μ is the longitude. The corrected body axis angular rate (p, q, r) can be used to calculate the Euler rates using the following formula:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3.21)$$

Where p, q, r are the corrected body axis angular rate, and ϕ, θ, ψ are the Euler Angles. The Euler rate can be integrated to give the Euler Angle. The Euler Angles are used to calculate the DCM using the following matrix:

$$\begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \quad (3.22)$$

The DCM transforms the reference axis into the body axis of the object. The transpose of the DCM performs the opposite function. The equation to remove the effects of Earths gravitational and the Coriolis acceleration from the measured body axis acceleration is given by the following:

$$\begin{bmatrix} a_X \\ a_Y \\ a_Z \end{bmatrix} = \begin{bmatrix} a_X \\ a_Y \\ a_Z \end{bmatrix}_M + \begin{bmatrix} vr - wq \\ wp - ur \\ uq - vp \end{bmatrix} - \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix} \quad (3.23)$$

Where Ax, Ay, Az are the body axis acceleration, and p, q, r are the body axis angular rate, g is the gravitational acceleration, and ϕ, θ are the Euler Angles. The acceleration is integrated once to get the body axis velocities u, v, w. The body axis velocities are resolved into the Earth's navigational axis using the transpose of the DCM as shown below:

$$\begin{bmatrix} V_N \\ V_E \\ V_D \end{bmatrix} = DCM^T \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (3.24)$$

The position is found by integrating the velocities in the navigational frame, but to find the position in terms of latitude and longitude, the following equation is used:

$$\begin{aligned} \dot{\lambda} &= \frac{V_N}{R_{EARTH}} \\ \dot{\mu} &= \frac{V_E}{R_{EARTH}\cos\lambda} \\ \dot{H} &= -V_D \end{aligned} \quad (3.25)$$

Where R_{EARTH} is the radius of the earth, and λ, μ, H are the latitude, longitude, and altitude respectively. Integrating the above equation gives the position in terms of latitude and longitude.

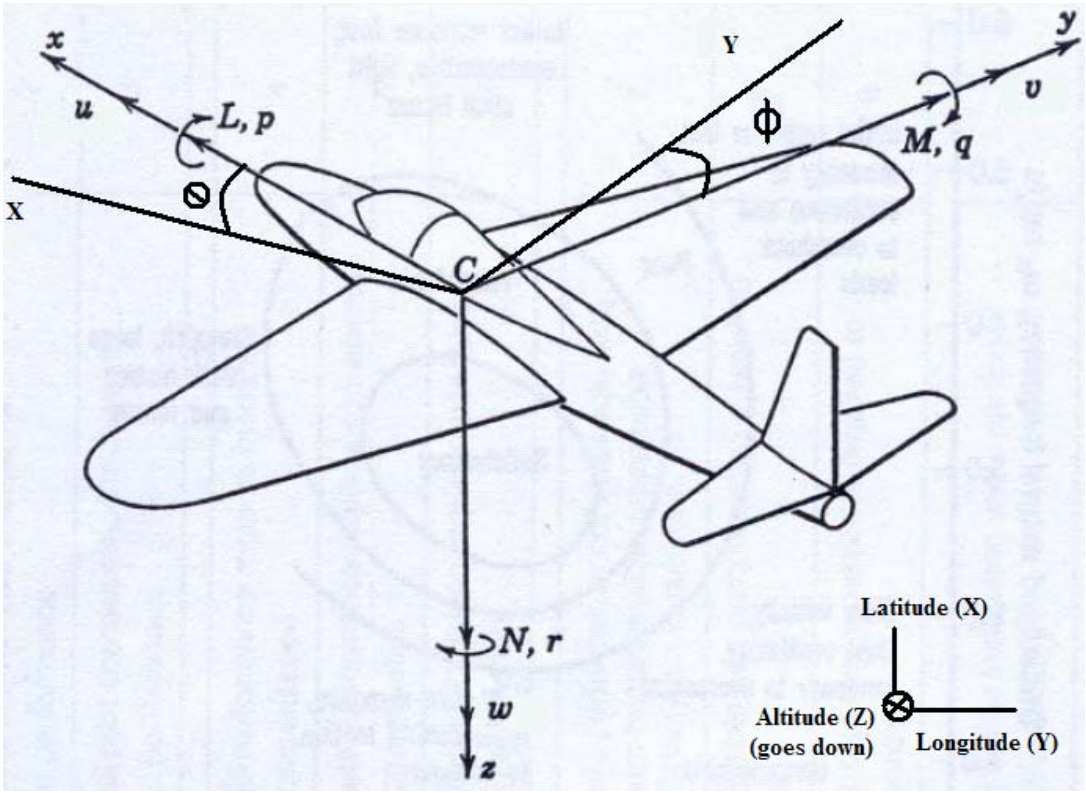


Figure 3-22: Sketch of Aircraft Body and Inertial Axis

Figure 3-22 above shows the body axes (x, y, and z) and the inertial axis (X, Y, Z). The dead reckoning algorithm followed in this thesis is not done utilizing the inertial method as explained above, but by integrating the velocity. A simplification of equation 3.9 using small angle simplification (sin x = x, and cos x = 1) and the fact that v and w ~ 0 gives the following simple equation whose integration will give the approximate position.

$$\begin{aligned} \dot{x} &= u\cos\psi \\ \dot{y} &= u\sin\psi \\ \dot{z} &= -\theta u \end{aligned} \qquad (3.26)$$

Where u is the body axis forward velocity and ψ is the current heading measured using compass.

CHAPTER 4: FLEXIBLE MISSION DESIGN – GROUND STATION

The aim of the research is to create a UAV with the following characteristics. Flexible mission capability is the top priority. In the context of this thesis, missions are taken to be a sequence of waypoints that the UAV must follow. The UAV should be able to track waypoints that are given in three dimensions using smooth and minimum path. At the end of mission, the UAV should be capable of receiving another mission without the need to reprogram the embedded system onboard the UAV. The user should be able to select mission quickly and with ease. This chapter describes the design process and methodology adopted to achieve this result.

After analysis the problem, the design was split into two major blocks or components. The first block consists of the Ground Station that will implement the dynamic path planner. The second block consists of the embedded system of the UAV's Avionics Unit, which implements the trajectory tracker, the autopilots, and the DFCS/SAS in a layered architecture discussed in Chapter 1. The Ground Station implements the mission layer and allows the user to select waypoints in a map and send them to the UAV. The rest of the layers are implemented in the embedded system onboard the UAV.

4.1 Ground Station Design

Ground stations are very important to monitor the state of flying objects such as satellites, missiles, and airplanes to name a few. For an UAV, the Ground Station provides important information such as pitch, roll, yaw, angle of slip, and angle of attack. The Ground Station also provides information regarding heading, location, altitude, and velocity. Very often, it is not possible to track an UAV in flight visually if it is far away; therefore, a Ground Station can be both valuable for the user and the UAV. A passive Ground Station merely monitors the UAV, but an active Ground Station should also be able to direct and guide the UAV through its mission, such as surveillance, path planning, calibration, and other functions.

The Ground Station designed hosts a Graphical User Interface (GUI) that provides the user with easy to understand visual data about the status of the UAV. The map on the top center of the display shows the location of the UAV on the map of a given area. The flight instrument panels to the lower right and left of the screen such as the odometer shows the UAV airspeed, the altimeter shows the UAV altitude,

the compass shows the UAV heading, and the artificial horizon shows the UAV orientation or attitude. The lower center panel gives information about the UAV angular rates, acceleration and rate of climb as textual information. There is also space kept for battery level indicator that can be enabled in future if needed.

The most important part of the Ground Station is the ability to plan missions. The user can select the waypoints in the map, which will be converted to GPS coordinates, and then transmitted to the Ground Station via the serial interface. Once the waypoints (a maximum of five for the purpose of this simulation) are selected, the user can have the Ground Station generate the approximate path the UAV will take during flight. During the actual mission the user can ascertain how well the UAV tracks the given path. Note that the Ground Station will only communicate the waypoints to the Avionics Unit, not the entire path. The Avionics Unit will run a similar algorithm as the Ground Station in its Trajectory Tracker to guide the UAV to the next waypoint. The path generated by the Ground Station is constrained by the physical limitations of the UAV. Therefore the user is not shown any path that will be impossible for the UAV to track. For this simulation the path will consist of straight lines and circles, which the UAV can execute properly. There will be no obstacles for the UAV to navigate through. Note that current design of the Ground Station allows the user to select waypoint in three dimensions, which are the requirement of this research and thesis. The latitude and longitude coordinates can be selected by clicking on the map, where as the altitude coordinate can be selected by moving the slider bar representing the altitude. The altitude chosen will be the position of the slider bar when the map is clicked.

Overall the Ground Station design does address a lot of shortcomings of the previous researches in AUS-UAV discussed in Chapter 1. With it the user will be able to give higher level commands to the UAV. These are called “missions” such as track a ground unit, surveillance or simply to follow a sequence of waypoints, loiter about a waypoint point and return back. The user will have access to maps to create a waypoint list by clicking on relevant positions in the Map to indicate the approximate (to within 50 meters) areas that the UAV must visit. The Ground Station will then show the user the approximate (smooth) path that will be taken by the UAV. Once the user is satisfied with the overall path, the waypoints are then sent to the Avionics Unit in the UAV via wireless link. This spares the user from having to actively trying

to control the flight of the UAV as in the case of a remote controlled plane, or as in the case of previous Ground Station design.

The functional coding and graphics design of the ground station was developed using Visual C++ 6.0 (VC6++). VC6++ was chosen above other software development kits (SDK) such as java because VC6++ is faster that java when it comes to manipulating and displaying graphics. VC6++ also has the added advantage of creating an executable file that can be run anywhere, without the need for the software application to be either open, or even installed. In [2] a ground station has been successfully implemented using Visual Basic, which allows the user more flexibility to create graphics, though not by a wide margin. VC6++ was chosen over Visual Basic due to the short learning curve required to understand and program it.

4.1.1 Path Generation and Waypoint Selection

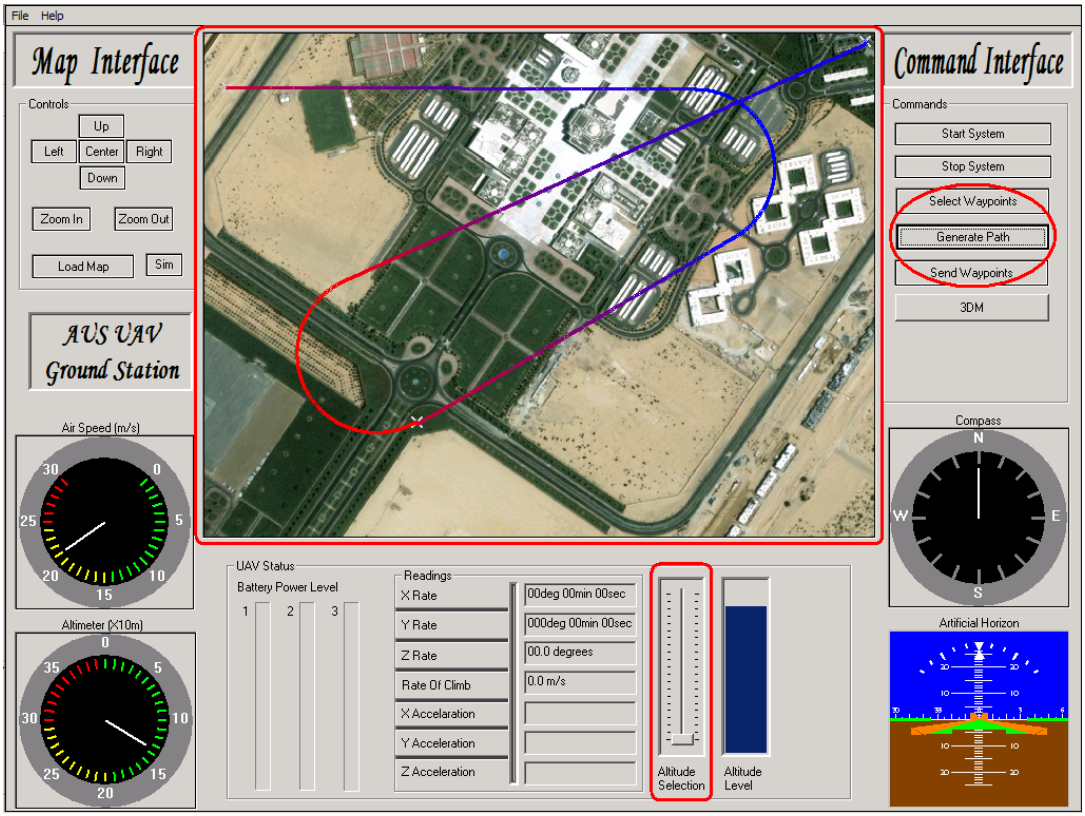


Figure 4-1: Pictorial Representation of the Ground Station Framework

Figure 4-1 above shows an outline layout of the ground station. The areas of interest are shown in red, and main among them is the "**map window"**, since it will display the location of the UAV on the map, and it can be used to select mission

waypoints for the UAV. The button "**Select Waypoints**" should allow the user to enter the waypoint by clicking on the map. Once the waypoints are selected, the button "**Generate Path**" should generate a smooth path, connecting the waypoints. Once the user is satisfied with the path, then the waypoint list can be sent to the UAV by clicking on "**Send Waypoints**".

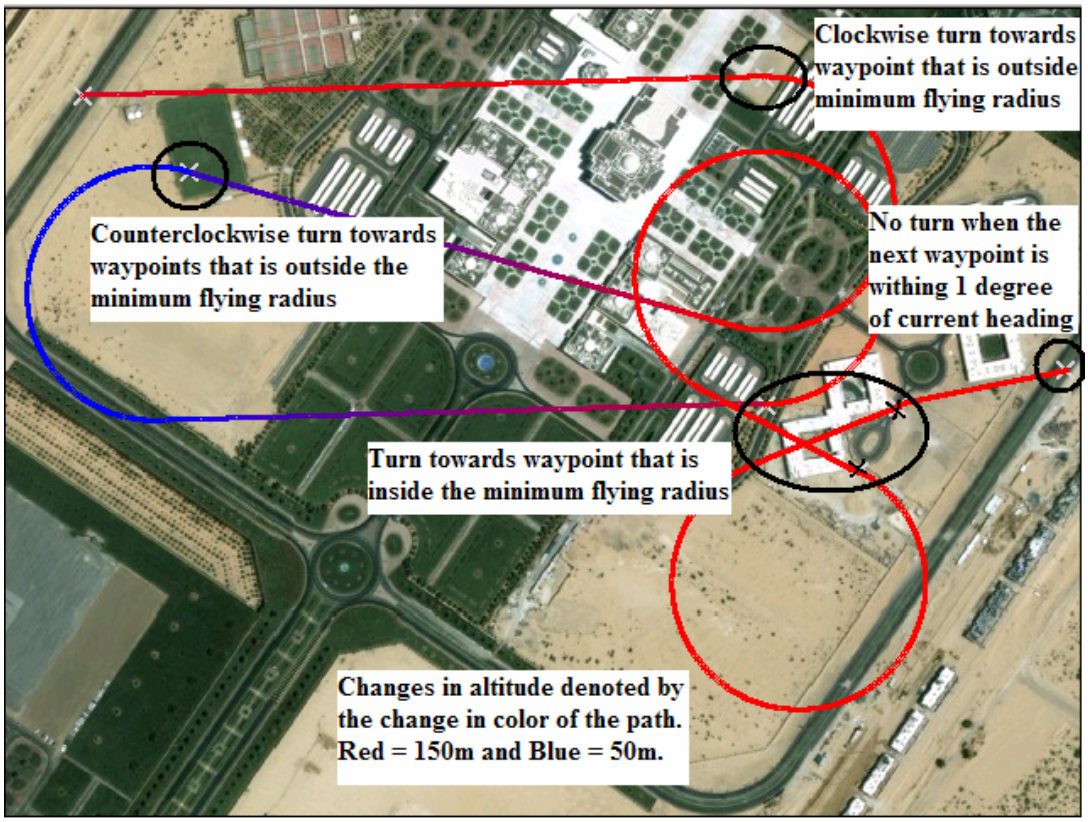


Figure 4-2: Map Area Showing the Waypoints Selected by the User

The paths shown in Figure 4-2 above incorporates the basic rules as shown below in Figure 4-3, namely that paths should not have sharp turns, nor force the UAV to bank more than its maximums bank angle. This is because a plane unlike a helicopter do not have hover mode, therefore unless the plane is moving it cannot stay aloft. But an object in motion, especially in the air cannot take turns unless, it is following a curve. Most planes have a limit to the sharpest curve they can attempt and remain stable. The radius of the circle chosen is 250m, which is well within the turn rate limit for the UAV traveling at ~21m/s at a bank angle of 10 degrees. If a waypoint fall within the minimum flying circle, then the UAV turns away from the waypoint in an attempt to reach it. On the other hand if the waypoint falls outside the

minimum flying circle, then the UAV will turn towards the waypoint in an attempt to reach it.

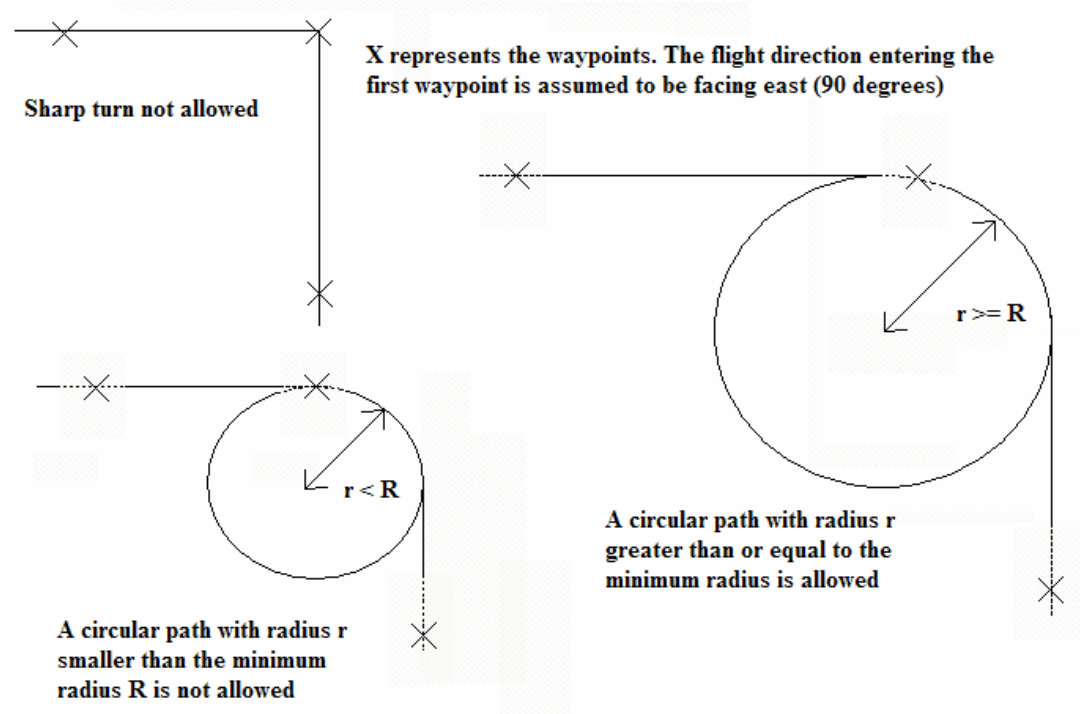


Figure 4-3: Schematic Representation of Allowed and Disallowed Paths

The final point of interest is the "**altitude selection slider bar**", which is shown in Figure 4-1. The "**altitude selection bar**" allows the user to select the altitude during waypoint selection. Before the user clicks on the map, they should adjust the slider bar representing the height of the waypoint. Once the user clicks on the map, the location as well as the altitude of the waypoint will be stored in the waypoint list. Figure 4-2 shows the altitude of the UAV as color coded paths generated by the path algorithm in the Ground Station. Altitude level of 50m is shown in blue, where as that of 150m is shown as red; colors in between represent intermediate altitude. The altitude changes as an exponential function of the distance traveled, with a time constant of 350m. This change in altitude is also approximate but well within the constraints of the UAV ascending and descending motion.

4.1.2 Ground Station Flight Panel Display Update Process

Aside from the allowing the user to select waypoints and generating the approximate path, the ground station is also responsible for transmitting these waypoints to the Avionics Unit and regularly receiving UAV status data from the

Avionics Unit and updating the instruments panel. This is achieved by creating a real time interrupt every quarter of a second, during which time the communication from the UAV is received and processed and the instruments display is updated. Figure 4-4 below describes the major blocks within the Ground Station.

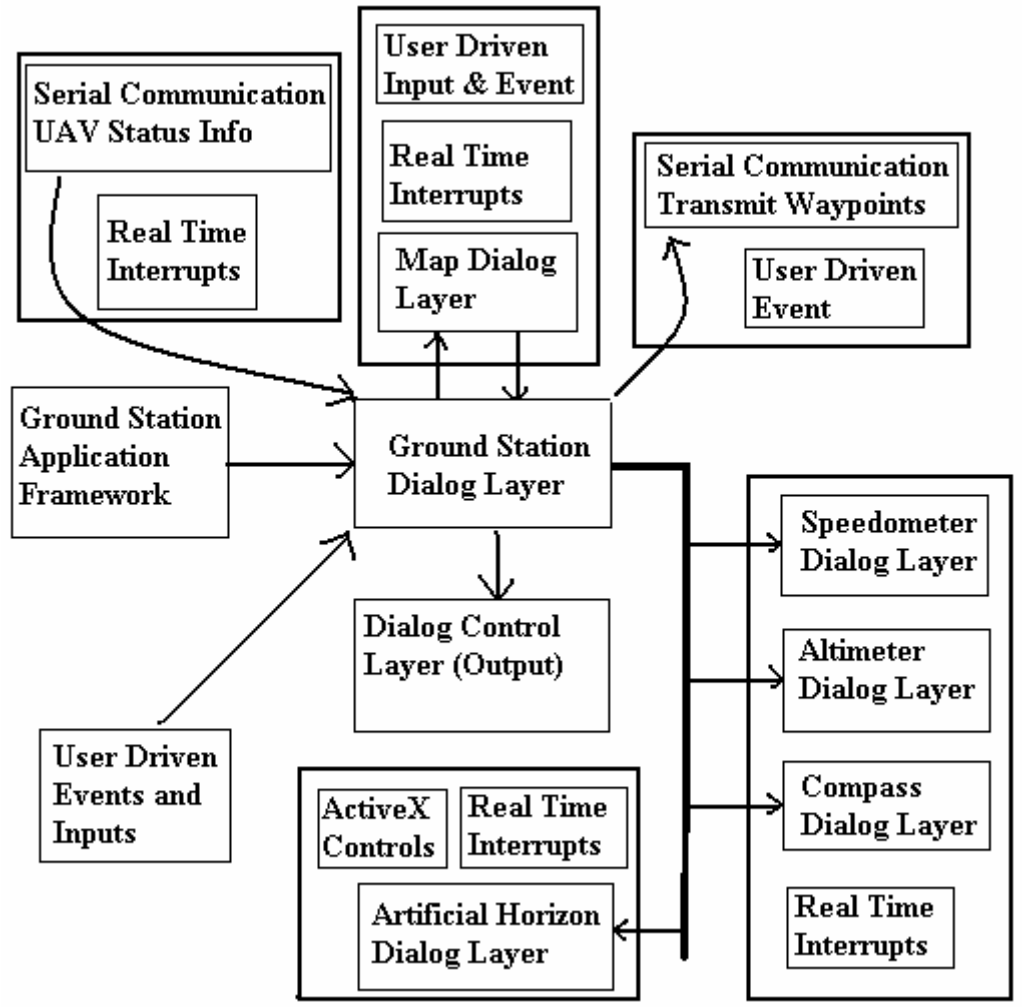


Figure 4-4: Block Diagram of Ground Station Components

The separate *dialog layers* are updated using the same data flow algorithm. During every real time interrupt, a flag is sent to the Avionics Unit indicating that Ground Station is ready to receive data. Then the Ground Station waits for the Avionics Unit to send the required data. The data once received is parsed into the speed, altitude and heading values and sent to these dialog layers. Once the dialog layers receive the required information, their displays are updated. Updating involves the redrawing of the entire dialog layer as is, with only the arm updated to point to the new speed, height or heading value. The flowchart in Figure 4-5 below illustrates this process.

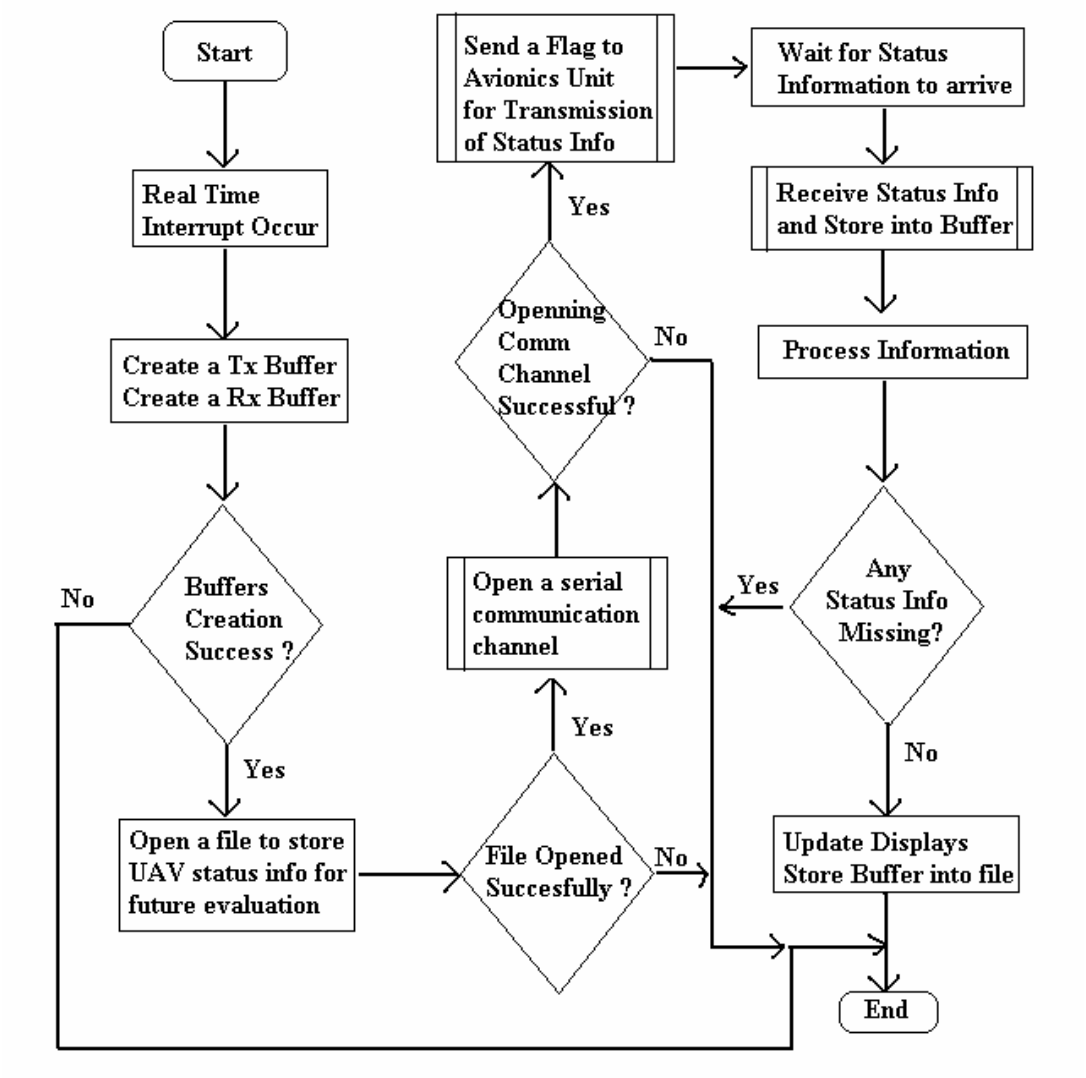


Figure 4-5: Flowchart of the Display Updating Process

Notice that the updating process involves several steps, and if any step fails the control returns immediately to the main program, and the displays are not updated. No effort is made to retry the failing step. The reason for this is that the real time interrupts occur periodically and continuously. Therefore if a display is not updated during one interrupt it is updated during the next. The second thing to note is that some of the steps, especially those involved with serial communication, are shown as actual processes. The details of these processes are illustrated in the Appendix. If the reader is interested in Serial Communication in VC6++, please check the appropriate appendix section.

The Compass, Altimeter, and Heading Dialog Layers are updated as explained above. The Ground Station (main) Dialog Layer sends the required information to the

sub dialog layers, and the layer update themselves by redrawing themselves, with the only change being the arm (pointer) pointing to a new value. The reason the instrument panels are used as separate layers is so that the main (big) dialog does not needs repainting, which speeds up the updating process. The Artificial Horizon is updated by the appropriate ActiveX control responsible for its display. Explaining ActiveX controls in VC6++ is outside the scope of this thesis report, and if the reader is interested, he/she is encouraged to review [21].

4.1.3 Path Algorithm Implemented in the Ground Station

The Map Dialog layer is updated differently from the other dialogs. Figure 4-4 shows that the Map Dialog layer is the only layer from which send information back to the main dialog layer. The updating process of the other dialog layers are driven by real time interrupts, but the Map Dialog layer is also updated using user driven inputs and events. This is because the map is the area where the user selects the waypoints. Clicking on the map is a user driven event, and the map must update itself to show an "X" in the exact location the user clicked. Also using the Map Interface Controls, such as zooms and map translations are also user events. The appropriate actions must occur when these buttons are clicked. The Map Dialog is also responsible for storing a buffer of waypoint sequence selected by the user. This is the information that the main dialog needs as input to its path algorithm and also to send to the Avionics Unit.

The final aspect of the Ground Station which needs elaboration is the implementation of the path algorithm. The inputs to the algorithm are the sequence of waypoints selected by the user. The final point to note is that the path algorithm is created for flying speed of approximately 21 m/s and a maximum bank angle of 10 degrees, therefore the minimum flying radius (R) is approximately 250m. This is the projected flight path, not the actual path taken by UAV, however a good UAV design should be able to follow this path. The calculations in the flowcharts were explained in Chapter 2. Figure 4-2 shows the paths plotted using the above algorithm, and Figure 4-6 below shows the flowchart pf the algorithm used.

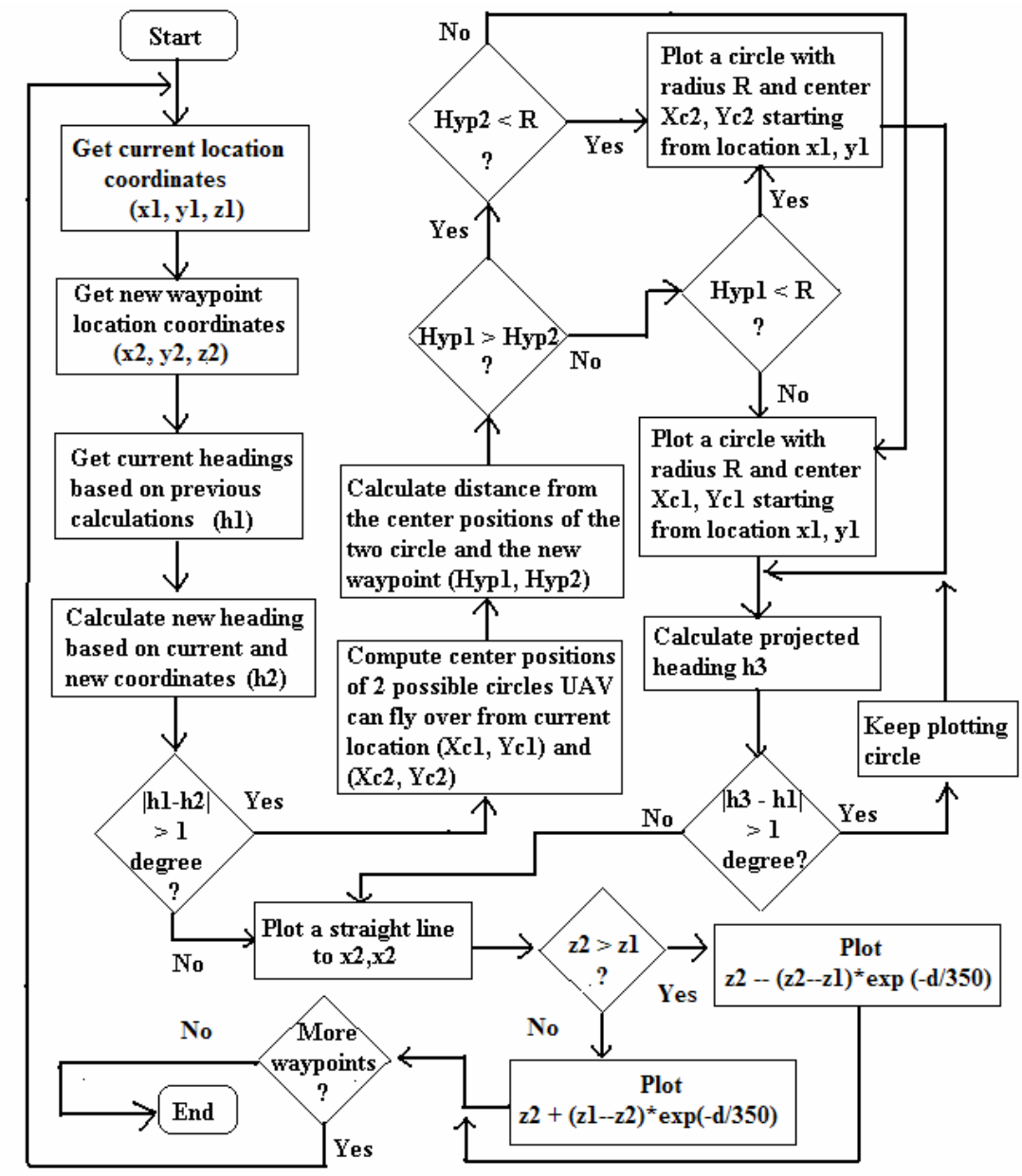


Figure 4-6: Flowcharting Illustrating Path Algorithm Implementation in Ground Station

4.1.4 Communication with the Embedded System

An important part of the Ground Station is the ability to communicate with the embedded system on board the Avionics Unit. The Ground Station sends the embedded system the waypoint coordinates in three dimensions in the beginning of the simulation, and the embedded system sends the flight status information during the remainder of the simulations. There must be protocol to ensure that the data sent is valid. The waypoints are sent in the following format:

@Wp1Lat | Wp1Lon | Wp1Alt | … | Wp5Lat | Wp5Lon | Wp5Alt#

Wp1Lat = Waypoint 1 Latitude, Wp1Lon = Waypoint 1 Longitude, Wp1Alt = Waypoint 1 Altitude. For the purpose of the simulation there will be five waypoints. Each of the waypoint is sent as ASCII version of integers (i.e. 56 is sent as '5' and '6'). An example of a typical frame sent to the embedded system is:

@0|0|100|1200|0|100|600|600|90|1200|1200|100|0|0|110#

The embedded system receives the frame from the Ground Station and sends it back character for character to the Ground Station. The Ground Station compares the received frame with the one it sent to ensure that there are absolutely no errors in the communication. This verification is important, since it directs the outcome of the entire simulation.

During the simulation the embedded system send flight status information to the Ground Station. Data integrity in this communication is not as important as the first one, since if a frame is corrupted, it is simply discarded. The following information is sent in a particular frame:

@Heading | Roll | Pitch | Velocity | Altitude | Latitude | Longitude#

The frame consists of floating point numbers in ASCII format, with one decimal point precision (-10.15 is sent as '-', '1', '0', '.', '1' – the 5 is not sent). A typical frame sent might look like:

@0.0|-2.4|115.2|20.4|356.2|500.0|300.0#

Validation is done in the Ground Station by splitting the frame using the '|' character as a delimiter. The Ground Station expects 7 different parts in any frame being sent. If the frame contains less than or more than 7 parts it is discarded. Another validation done is range checking, which ensures that each parameter received falls within their correct range (e.g. the roll angle should be between -15.0 and +15.0, else it is a mistake). Any frame that contains mistakes in the ranges of any of its parameters is discarded.

CHAPTER 5: FLEXIBLE MISSION DESIGN – EMBEDED SYSTEM

5.1 Embedded System Design

The Avionics Unit consists of various sensors, and actuators integrated together with the embedded system. The embedded system consists of two 16-bit Motorola 68HC12 microcontrollers running at 24 MHz. The sensors used are three (x, y, z) rate gyros to provide rate of turn (such as yaw) of UAV, three (x, y, z) accelerometer to provide information about the forces acting on the UAV, absolute pressure sensor to provide altitude of UAV, gauge pressure sensor to provide air forward speed of UAV, tilt sensors to provide pitch and roll angle of UAV, and angle of attack sensor. There are also three serial devices connected to the embedded system, including a compass providing heading of the UAV, GPS sensor providing location of UAV, and the RF transceiver used for communicating with the Ground Station. The embedded system is also responsible for actuating servos as shown in Figure 5-1 below.

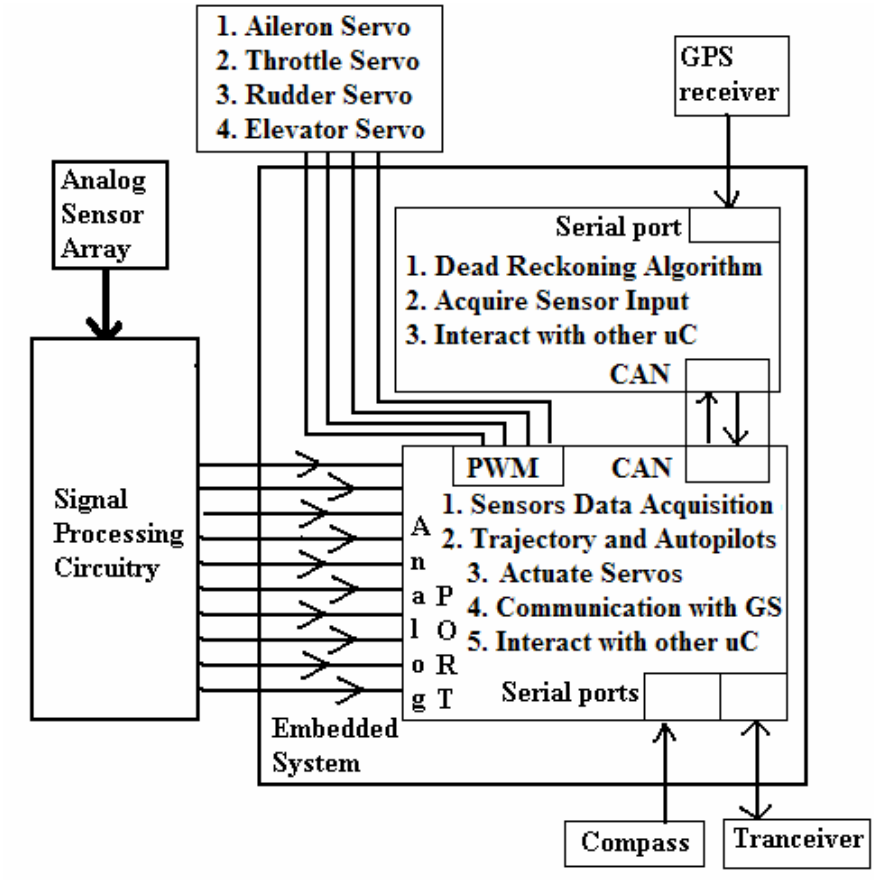


Figure 5-1: Block Diagram of Avionics Unit Embedded System Framework

The DFC in the embedded system is used to ensure smooth flight conditions. There are also two autopilots implemented in the embedded system: the lateral and longitudinal autopilot. The longitudinal autopilot stabilizes the UAV at a given altitude, or controls the rate of climb of the UAV. The lateral autopilot on the other hand stabilizes the UAV in a given heading or controls the rate of turn of the UAV. These autopilots run in parallel to achieve a three dimensional maneuver of the UAV. Therefore the trajectory controller implemented in the embedded system will utilize both the lateral and the longitudinal autopilot to have the UAV either move in a straight line, turn in a circular path, ascend or descend depending on the best method to reach the next waypoint. The final algorithm implemented in the embedded system is the Dead Reckoning or Inertial Navigation algorithm with a simple Kalman filter needed to calculate position in between the GPS updates. Figure 5-2 below shows the level at which each algorithm operates.

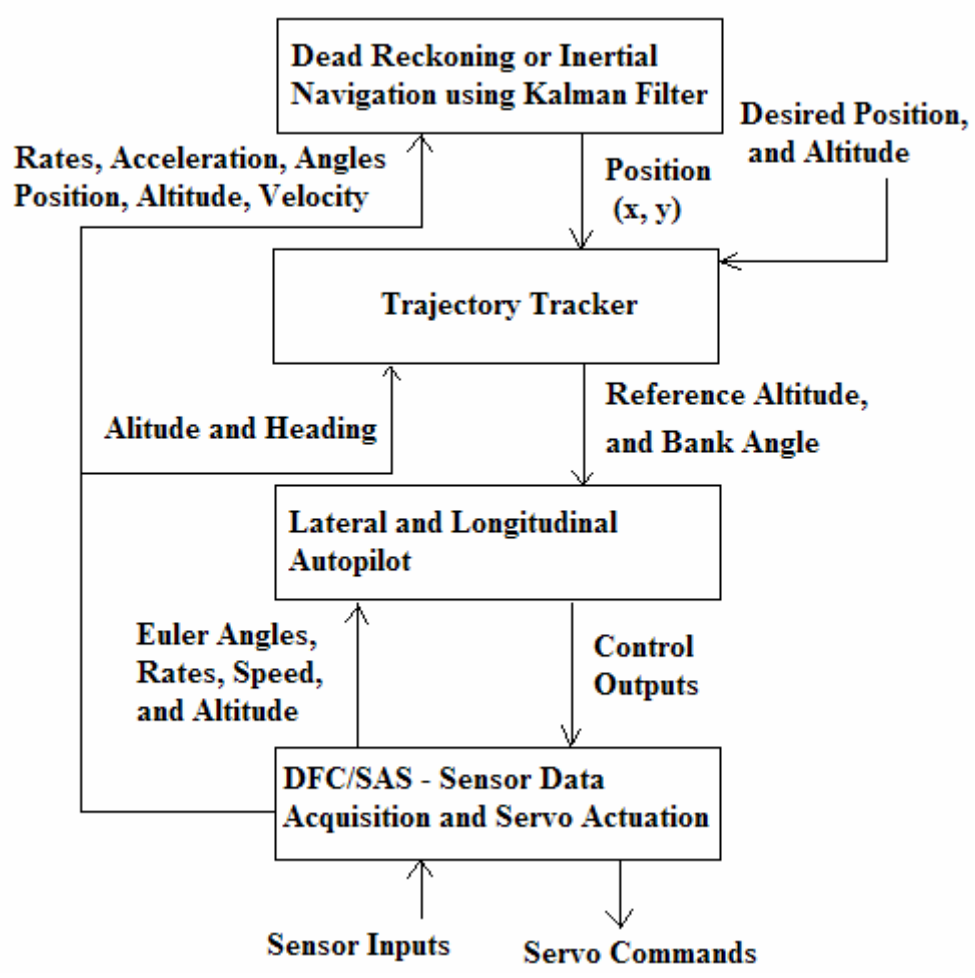


Figure 5-2: Flow Diagram of Embedded System Architecture

5.1.1 Real Time Implementation of Control Algorithm

Any control algorithm implemented inside the embedded system must satisfy the minimum system update rate. In the case of the TRI-60 UAV, the system consists of the aircraft and its control surfaces, and their respective servo actuators. The servo actuators have a time constant of 0.27 seconds [1], and therefore any control actuation signals to the servo needs to be updated at minimum 7.5 Hz to maintain Nyquist rate. Similarly, the aircraft have a fast natural frequency (or a short period mode) of 2.97 Hz [1], and therefore needs at least 6 Hz control update rate to maintain Nyquist rate. The controller updates must be distinguished from the sensor data updates, which are generally much higher, due to navigational requirements. For example, for an aircraft moving at 20m/s, the update rate must be at least 40 Hz. Figure 5-3 and Figure 5-4 show the complete list of processes inside both microcontrollers.

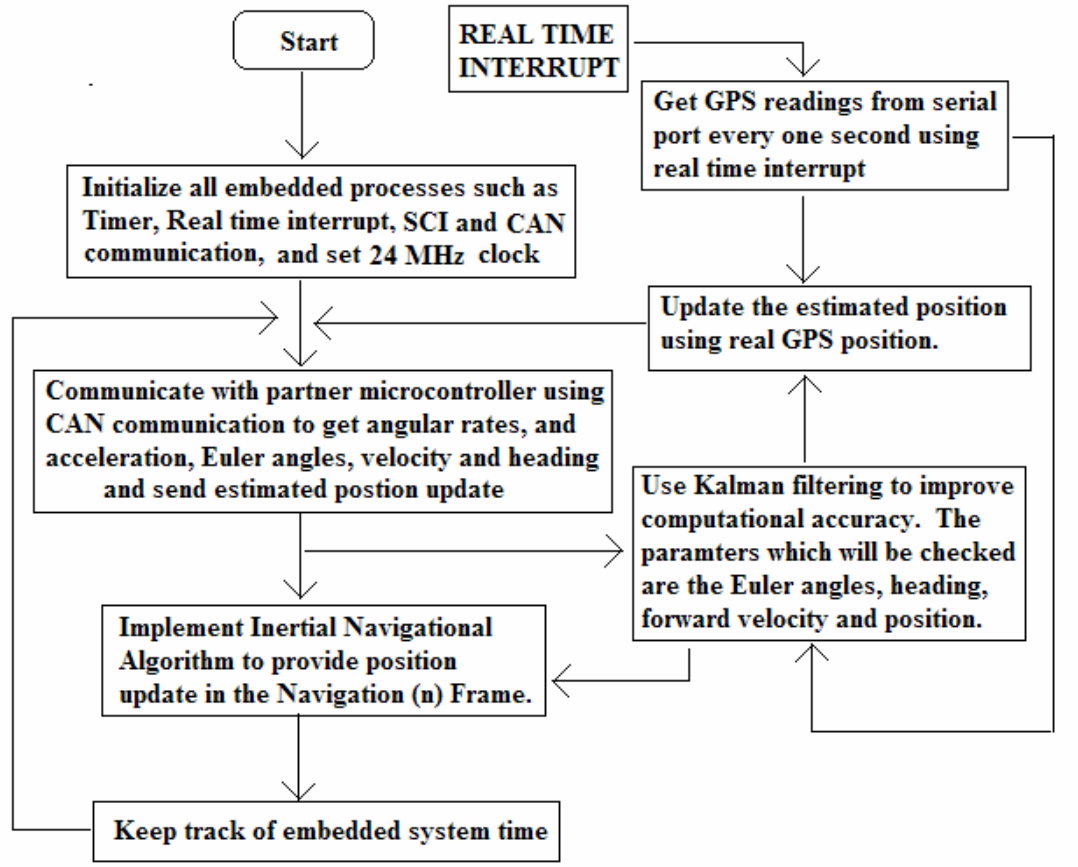


Figure 5-3: Flowchart of Processes inside the GNC microcontroller (Flight Computer)

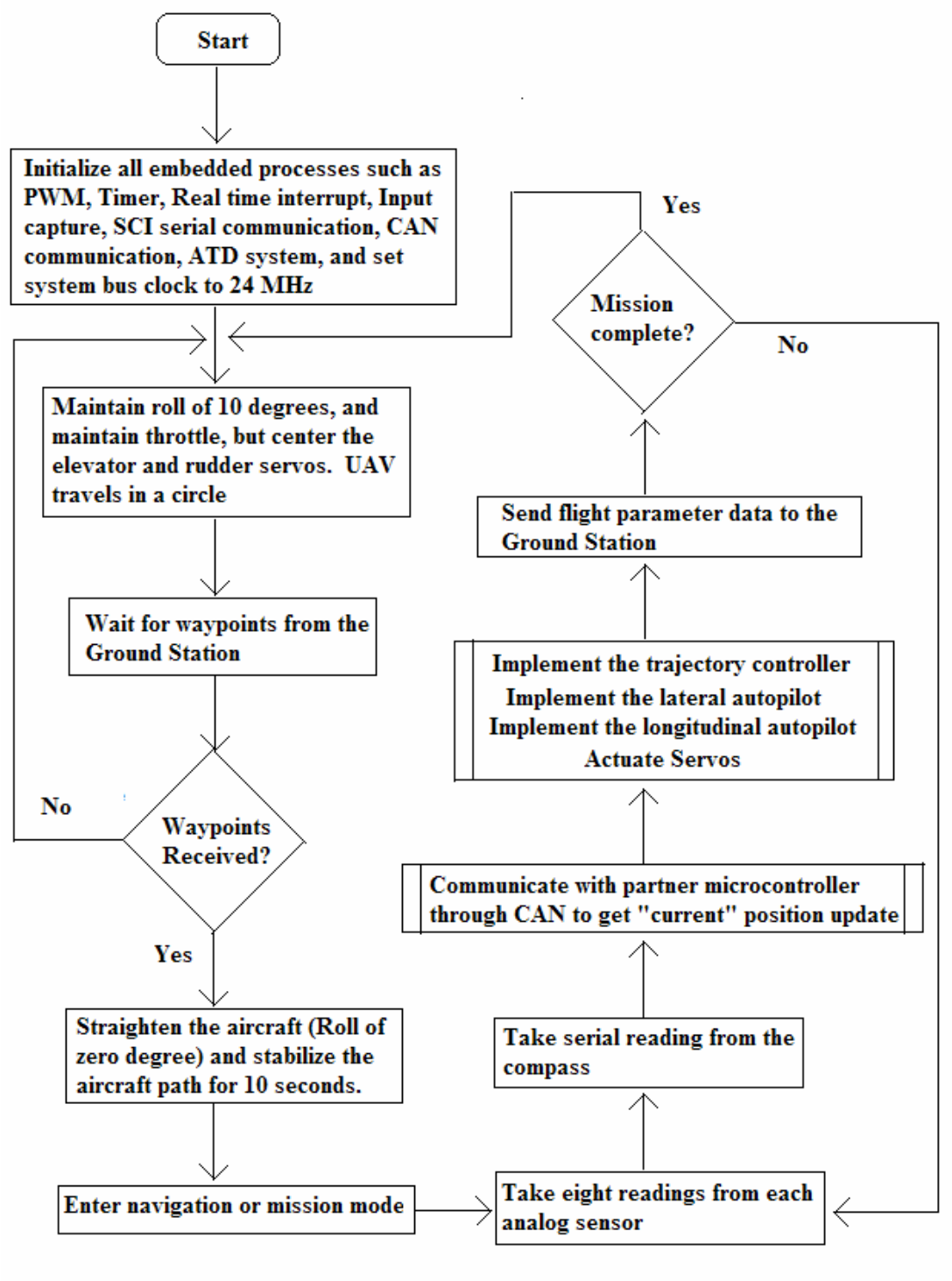


Figure 5-4: Flowchart of Processes inside the DFC/SAS and Trajectory Microcontroller

The embedded system should be able to accomplish sampling ten analog sensor data eight times each, receive serial data from the compass, communicate with partner microcontroller to receive the position update from the other microcontroller and to send flight parameter data to the other microcontroller, implement the control

law including the Longitudinal and Lateral Autopilots, and the Trajectory Tracker, and finally send all the flight parameter information to the Ground Station at a rate of 10 Hz or more to maintain smooth trajectory between waypoints and minimize sudden movements due to delayed control updates. The second microcontroller is used to implement the GPS/INS navigation system using Dead Reckoning and Simple Kalman Filtering. This GNC flight computer will run its algorithm at a faster rate of 50 Hz to ensure very accurate position updates are available

5.1.2 Implementation of the Trajectory Tracker Algorithm

The DFC/SAS and the Longitudinal Autopilots have been implemented in [1] and will not be discussed in this report. This thesis carries on the research from [1] and therefore the embedded code base remains the same. However the code have been slightly tweaked to allow more interrupt based processing than the earlier version, and a more flexible communication scheme with the Ground Station. The trajectory controller which implements the path algorithm is major theme of this project and therefore will be discussed in this report. The trajectory controller implemented in the microcontroller takes in the desired target coordinates of the next waypoints (xd, yd, zd), current position, velocity, and heading as input and outputs the desired roll angle and reference altitude as output.

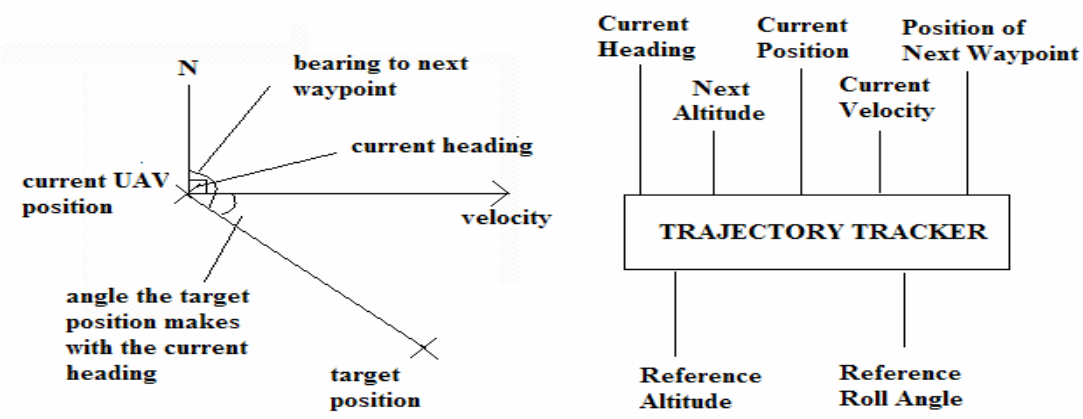


Figure 5-5: Information necessary for the trajectory controller

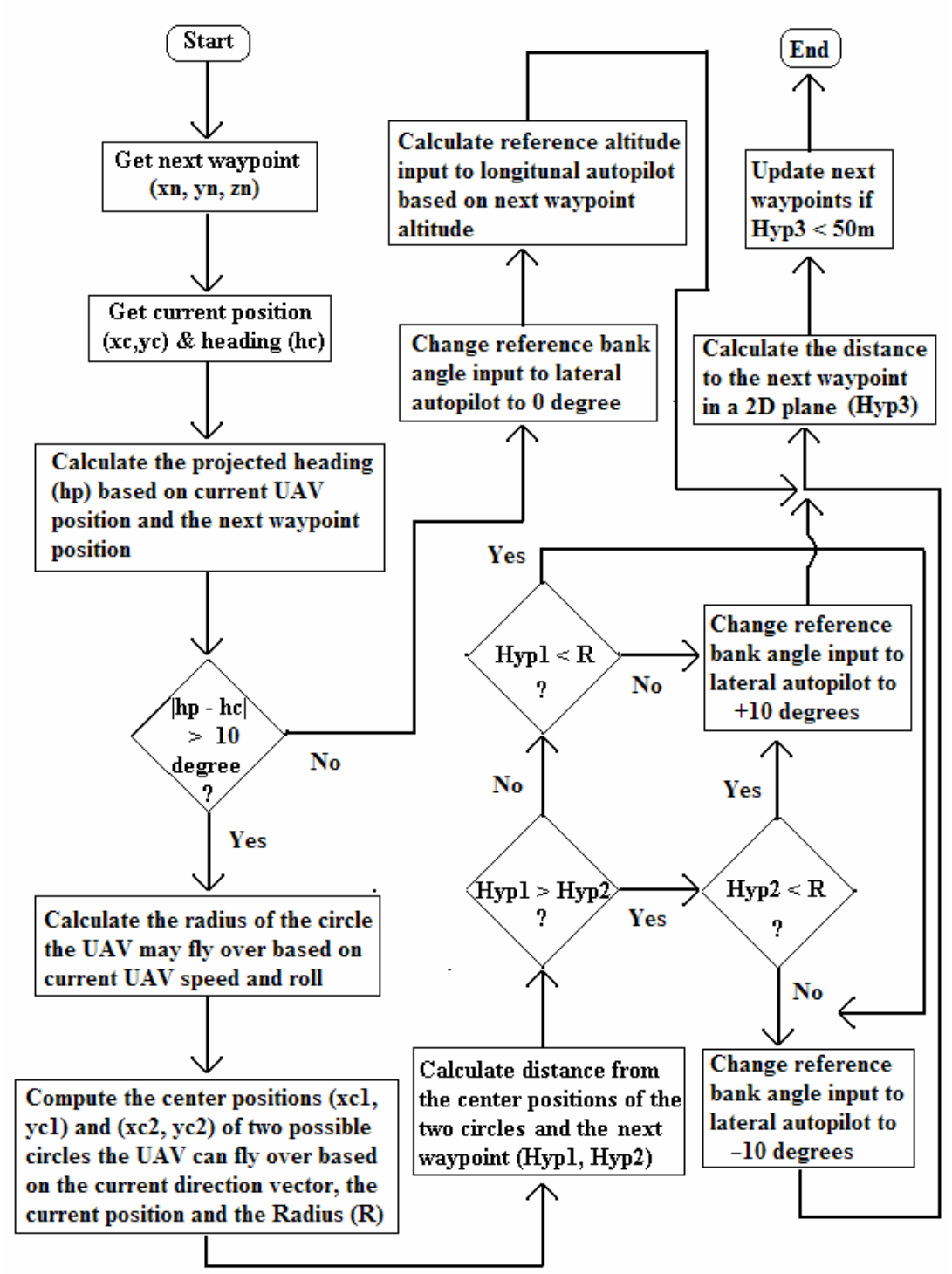


Figure 5-6: Flowchart showing Path Algorithm Implementation in Trajectory Controller

Figure 5-5 and Figure 5-6 illustrates the path algorithm as it is implemented in the Trajectory Tracker. The Trajectory Tracker takes the heading from the compass and then calculates the projected heading from the UAV's current location (supplied by the GNC Flight Computer) and the location of next waypoint. The decision to turn

is based on the difference between the projected and the current heading. If the heading error is more than 10 degrees, then the Trajectory Tracker will command the Lateral Autopilot to turn to the desired heading. The direction of turn is based on a "**two-step**" decision making process. The circle whose center is closer to the next waypoint is the circle chosen to fly over unless the next waypoint falls inside that circle. The Lateral Autopilot utilizes a PID controller to roll the UAV by +/-10 degrees, and keep it banked at that angle, causing the UAV to fly into a circle. The reason for the 10 degree cutoff for turning is due to the fact the UAV cannot exit the circle immediately. The UAV takes a finite amount of time to fly straight from a curvature, and it is within that time the UAV covers the 10 degree gap. Once the UAV reaches within 50m of a waypoint, the Trajectory Tracker tracks a new waypoint. An important fact about the Trajectory Tracker is that it does not keep any histories. The Trajectory Tracker is essentially an instantaneous controller, whose control outputs are based on only current flight parameter values. Future upgrades to the Trajectory Tracker should incorporate at least a three time step history to further smoother the control output of the UAV.

5.1.3 Communication between Two Microcontrollers

The microcontroller utilized for the embedded system is the Motorola HCS12Dx256. This is a powerful 16-bit microcontroller whose CPU is capable of running at 25 MHz, and has a lot of built in resources, including ATD ports, SCI/SPI ports, PWM output ports, Timer Capture ports, and CAN communication ports to name a few. One HCS12 is enough to integrate all the sensors, the controller and the actuators. However there are only two Serial Communication Interface (SCI) ports, when there is need for three; two to interface with two serial devices (Compass, and GPS) and one to communicate with the Ground Station. With an efficient multiplexer design it would have been possible to interface the two serial devices to one SCI port, and to leave the other SCI port for communication with the Ground Station, however this was not the design chosen in the previous research. Since this research is a follow up to the previous one, the hardware that has been utilized for the previous research is being used in this one as well.

As can be seen from Figure 5-1, the second microcontroller is used only to interface with one sensor. However this enables the second microcontroller to play a very essential role of the Guidance and Navigation Controller (GNC) computer that

implements a GPS/INS Dead Reckoning Algorithm using simple Kalman Filtering. This is a time consuming algorithm that would have slowed down the other microcontroller, which is already burdened with data acquisition, implementation of Trajectory Tracker and two Autopilot algorithms and at the same time also responsible for communicating with Ground Station.

The Inertial Navigation algorithm requires three angular rates information and three acceleration readings to compute the velocity and position estimates in three dimensions. However these estimates are based on initial conditions being known, including initial velocity, position, altitude and attitude. Therefore the algorithm requires a total of ten sensor readings, nine of which must be supplied from the first microcontroller. Most implementations of INS algorithms that provide velocity and position estimates are valid only for a few seconds [22], [23]. That is why the algorithm must be reset with accurate readings at regular intervals. The communication protocol is made so that, the first microcontroller supplies the second microcontroller with the rate and acceleration information at a rate of 12 Hz, and at the same time acquires the "position estimates" from the second microcontroller. However, every one second, the first microcontroller supplies the second microcontroller with all the sensor information and in return gets accurate position readings from the GPS sensor.

To enable such communication between the two microcontrollers various modes of communication were researched including Serial Peripheral Interface (SPI), Controller Area Network (CAN), and parallel communication protocol. Communication using SPI has been researched in [1], however this mode of communication is rejected because SPI is a communication between a master and a slave, but it is not possible to select any one microcontroller as master or slave. A second disadvantage is that SPI does not have a well defined communication protocol, and therefore there is no means of ensuring that data is transmitted without any error between the master and the slave (data integrity). Parallel mode of communication is rejected because it requires the use of 19 digital I/O pins, which is a waste of microcontroller resources. Parallel communication also requires data to be continuously transmitted, even when there is no data to be sent, thus requiring transmission of dummy data and this wastes microcontroller time.

This leaves the CAN communication scheme, which is a very secure mode of communication between "multiple peer nodes" using an internationally standardized

protocol (ISO 11898). CAN protocol ensure arbitration between multiple nodes connected to the CAN bus network ensuring no more than one node at a time can transfer data. Every data frame sent using CAN protocol has a specific identifier ensuring that only the addressed node can accepts the data. Every frame of data sent has its own 16-bit Cyclic Redundancy Code (CRC) to check for errors in the frame. The receiver computes the CRC based on the data content in the received frame and matches it against the CRC sent by the transmitter. The CRC algorithm is secure and can catch most framing errors. At the same time the CAN hardware monitors its own signals during transmission to further catch errors at the source. The CAN hardware allows retransmission of a data frame in case of error. The most important part in selecting CAN besides secure data transmission is the maximum of 1Mbaud data transmission rate allowed.

Figure 5-7: The CAN Data Frame Format

Figure 5-7 above shows the format of the CAN data frame, consisting of the Start of Frame (SOF) bit, the two to four bytes of Message Identifier, the Remote/Data frame bit, the Standard/Extended format bit, the Priority byte, the Data Length byte, one to eight bytes of data, two byte CRC, the acknowledgement bit, and the End of Frame bit. CAN protocol allow messages to be sent with an 11-bit standard identifier or a 28-bit extended identifier, with the Standard identifiers having more priority in case of transmission. CAN protocol also allow remote as well as data frames to be transmitted; a remote frame is a request to send data, where as data frame is the data transmitted.

A transmission rate of approximately 110kbaud has been implemented between the two microcontrollers. This ensures that complete transmission of data between the two microcontrollers (approximately 50 bytes) can occur within 5ms. Figure 5-8 below shows the CAN network architecture. There are two components to the CAN architecture, the CAN intelligence module (MSCAN) that is responsible for arbitration, error checking, encoding data into a frame format and sending data frames, and decoding received frame and extracting data. The CAN transceiver is responsible for converting TTL signals from the MSCAN module to and from the

CAN High (recessive) signal and the CAN Low (dominant) signal that required to send information through the CAN bus network.

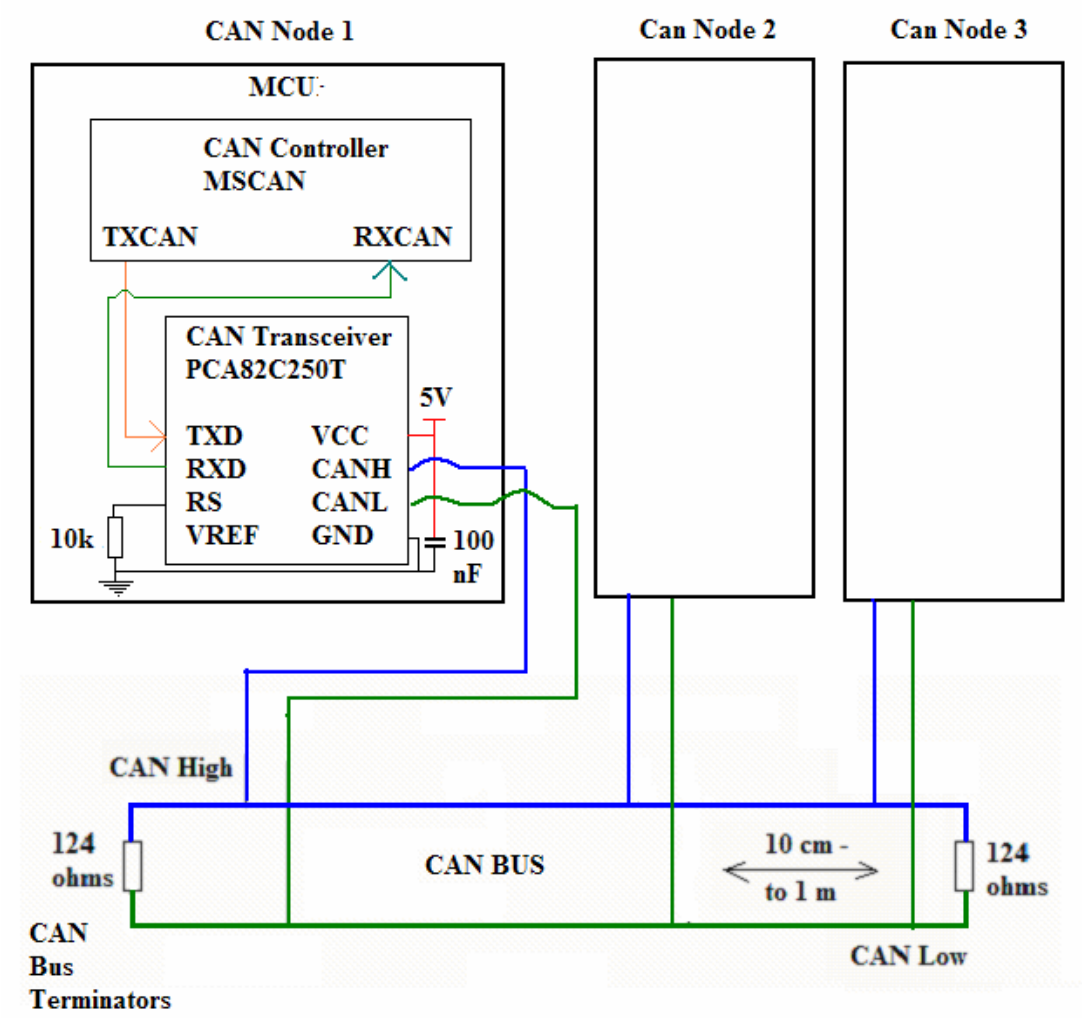


Figure 5-8: Schematic of CAN Network Architecture

Figure 5-9 below shows the CAN transmission and reception algorithm implemented in the embedded system to facilitate communication between the two microcontrollers. The algorithm defines the CAN initialization routine, and the specific algorithm to send and receive data. The algorithm should be followed closely to ensure proper communication. The algorithm shown below can be extended to use interrupts and well as multiple transmit buffers. For the HILS simulation conducted in this thesis, the **heading** (psi) and the body frame **forward velocity** (u) was sent using CAN from the DFC to the GNC microcontroller, while the **latitude** and **longitude** information was sent using CAN from GNC to the DFC microcontroller. For later research, more information can be sent using the CAN bus.

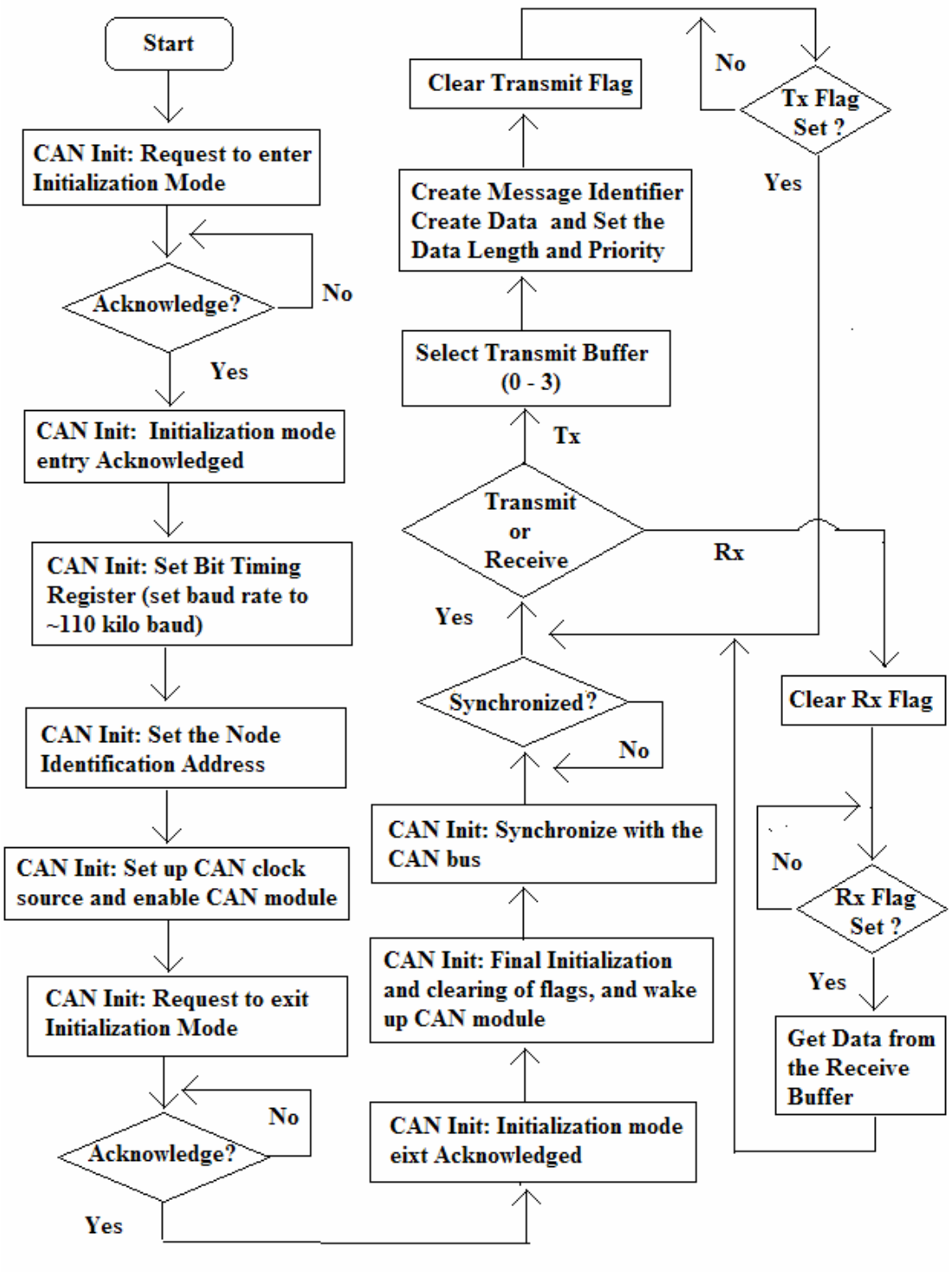


Figure 5-9: Detailed Flowchart of the CAN Initialization and Tx/Rx Algorithm

5.1.4 Communication with Ground Station

The ability to communicate between the Ground Station and the Embedded System on board the Avionics Unit is an essential feature of "flexible mission"

capability. The flowchart shown in Figure 5-4 in Section 5.1.1 illustrates how important this communication is. The Ground Station communicates with the embedded system in order to send the list of waypoints that the UAV must track, and the embedded system sends the same list back to the Ground Station so the user can be sure that the UAV has received the waypoint list properly. The Ground Station also signals the embedded system onboard the UAV to enter into "mission or autonomous" mode, where the UAV starts tracking the waypoint list sent to it. During the "mission mode" the UAV communicates its important flight parameter data back such as its speed, altitude, location, attitude back to the Ground Station. When the UAV is in "mission mode" the communication must convey all the flight parameter data in the "shortest" amount of time possible in order not to substantially affect the frequency of the control loop.

A primitive form of data encoding scheme has been devised, which allows all the flight parameter data to be encoded into 85 bytes of data regardless of what information the embedded system is sending to the Ground Station. Without the encoding scheme the data length would be variable depending on the condition of the UAV. For example, if the UAV was heading north in a straight line, the data communicated back to the Ground Station would probably be: "21.0 | 0.5 | 0.0 | 0.0 | 130" to indicate 21 m/s second velocity, 0.5 degree heading, zero degrees of roll and pitch, and 130 m altitude. Please note that not all data has been encoded for this example. Now on the other hand, if the UAV was flying in a circle the following data would probably be encoded: "21.0 | 100.5 | -10.4321242 | -2.453323223 | 120". Notice the huge problem that occurs when a floating point decimal value needs to be communicated. This problem can be solved by rounding the floating point value to just one decimal place and send the following data instead: "21.0 | 100.5 | -10.4 | -2.5 | 120". However, whereas the data length when the UAV was flying straight was only 20 bytes in size, the data length of the UAV when it is flying in a circle is 25 bytes in size.

With sixteen different sensor values to be communicated, the discrepancy between the data length increases and therefore there is need for a fixed length data encoding scheme. To overcome this difference in length between the larger numbers (such as 123.3) and the smaller numbers (such as 1.1), data is sent as a type instead of as a value. For example, all integers can be encoded into two bytes of data regardless of its value, therefore both 60000 and 6 takes up two bytes of data as an integer type,

whereas 60000 take up five bytes of data and 6 take up only one byte of data as a value. Similarly all float takes up only four bytes of data regardless of its value or precision. Sixteen sensor values take up 80 bytes of data (4 byte float + 1 byte separator). The other five data bytes are taken up by the beginning and end of frame markers. The following lines of code show how one float can be encoded into four bytes of data.

```
unsigned char *addr_GP1, *addr_GP2, *addr_GP3, *addr_GP4;
addr_GP1 = (unsigned char*)&GP2+0;  // NOTE:  The following lines
addr_GP2 = (unsigned char*)&GP2+1;  // of code is intended to access
addr_GP3 = (unsigned char*)&GP2+2;  // a float which is a four byte
addr_GP4 = (unsigned char*)&GP2+3;  // value one byte at a time
sci_output[2] = *addr_GP1;          // This is done by accessing the
sci_output[3] = *addr_GP2;          // memory location where the
sci_output[4] = *addr_GP3;          // float is stored using a
sci_output[5] = *addr_GP4;          // pointer to a character!!
```

GP1 is a float value that stores the current velocity of the aircraft. It is split up into four different bytes by first casting its address as a character (which is one byte length in C), and then using successive address values to get the value of the float.

5.1.5 Implementation of Dead Reckoning (GPS/INS) Algorithm

The Dead Reckoning algorithm is implemented in the GNC microcontroller which is interfaced with the GPS sensors. The GPS sensor gives a position fix every 1 second (for this thesis it is assumed that connection with the GPS satellite is never lost). This position is stored as the reference position, to which the integrated position is added at 12 Hz. The DFC microcontroller communicates the body axes velocities (u, v, and w) and the Euler angle ($\phi, \theta, and \psi$) to the GNC microcontroller, and these are used to calculate the position based on Equation 3.9. The position is communicated back to the DFC microcontroller. The code below shows the dead reckoning algorithm implemented in the GNC microcontroller.

```
the = YT2*m_pi/180.0;   // Get the current pitch in radians
psi = compass*m_pi/180.0;     // Get the current heading in radians
xvel = cos(psi)*GP2;    // Find the current x axis distance traveled
yvel = sin(psi)*GP2;    // Find the current y axis distance traveled
zvel = -the*GP2;        // Find the current z axis distance traveled
currtime = time_sec();  // Capture the current time
deltatime = currtime - prevtime;    // Find the time difference
prevtime = currtime;// Store the current time as the previous time
```

```
xdist = xdist + xvel*deltatime;     // Integrate the x axis distance
ydist = ydist + yvel*deltatime;     // Integrate the y axis distance
zdist = zdist + zvel*deltatime;     // Integrate the z axis distance
latcurr = latbase + xdist/(60*30); // convert x axis distance to lat
loncurr = lonbase + ydist/(60*27); // convert y axis distance to lon
```

Every one second the GPS unit is read, and the xdist, ydist, and zdist is reset with the new position given by the GPS reading. As explained before in the previous section, CAN communication is used to transfer the nine float values (36 bytes) between the two microcontrollers.

5.2 Calibration of the Avionics Unit [26]

The control algorithm of the Lateral and the Longitudinal Autopilot, the Trajectory Tracker, and Dead Reckoning Inertial Navigation requires input values from the sensors to compute the necessary control output. Correct values acquired from the sensor will result in smooth and effective control output, whereas defective and improper values from the sensors will lead to faulty control action, which can be disastrous. Defective values arise when the sensor stops working or the path from the sensor output to the acquisition unit gets damaged. During those situations, the only solution is to replace the sensor, or create redundancies, so that failure of one sensor input does not prove too disastrous. On the other hand wrong and improper sensor values are caused by any number of reasons, including orientation errors, noise and improper calibration of sensor output which is usually in voltage, to the actual parameter values such as degrees and degrees per second to name a few.

Calibration thus plays an important role in mapping a sensor voltage output to the exact parameter value. Some sensors such as the Compass and the GPS sensor does not need to be calibrated because they are pre-calibrated in the factory and because they do not output voltage values but rather the actual parameter value. On the other hand, the analog sensors such as the rate gyros, the accelerometers, the tilt sensors and the pressure sensors need to be calibrated. Some form of calibration was done in [1], but they mainly relied on the datasheet to estimate the equation relating the respective parameter value to the voltage output by the sensor. The form of calibration is not valid due to the fact that the datasheet usually gives an average value after thousands of sensors have been factory tested. Theses sensors are not necessarily the one that is being used in the Avionics Unit. For example, in [1] the

equation of the rate gyro has been given as V_{OUT} = 0.0167*rate + 2.657. The reasoning is that, since the Full Scale Range (FSR) of the voltage values from 0V-5V represents a full scale range of +/- 150^{O}/s, the gradient of the slope equals 5/300 or 0.0167. When the sensors are stationary, the zero offset voltage value has been recorded as 2.657V, which in this case is the y-intercept of the equation. The above reasoning is invalid because, the researcher clearly has not taken into account the rotation of the earth, which is approximately 15^{O}/s, and this manifests itself as a 2.657V zero offset instead of 2.5V! Reorient the sensor so that it is placed orthogonal to the rotational axis of the earth, and the zero offset should approach 2.5V. Continued use of the above equation will give invalid results.

Figure 5-10: 3DM-G Inertial Measurement Unit from MicroStrain

To resolve this problem, and to ensure that the equations relating the voltage output of the sensor to the respective parameter value are completely valid, a full scale calibration of the Avionics Unit has been carried out by measuring its output against a fully calibrated IMU called 3DM-G by MicroStrain shown above in Figure 5-10. 3DM-G is a nine degree of freedom sensor and outputs the instantaneous parameter value of the Euler Angles, angular rates, and acceleration at a rate of 50 Hz - 150 Hz. These outputs are available in raw voltage level, fully calibrated parameter values, and gyro drift compensated and gyro stabilized parameter values. Simultaneous analog readings were taken from the Avionics Unit and they were matched with the 3DM-G IMU readings using two separate methods as detailed in the next two sections.

3DM-G IMU has a Full Scale (FS) Orientation range of 360 degrees full scale (Matrix, Quaternion modes), Angular Velocity range of +/-300 degrees/second, Acceleration range of +/- 2 G's and Magnetometer range of +/-1 Gauss in all axes. It has an A/D Resolution of 12 bits, Orientation Angle resolution is less than 0.1 degrees (taken at most aggressive filter setting) The Temperature drift in a single axis is 0.025% per degree Celsius, Nonlinearity is 0.23% full scale (in static conditions) and the Repeatability is 0.1 degrees. Therefore 3DM-G is a suitably accurate IMU to calibrate the Avionics Unit.

5.2.1 Calibration of Inertial Sensors without Utilizing Microcontroller

The Avionics Unit sensors to be calibrated are the tilt sensor (measures pitch and roll), the three rate gyros (measures x, y and z angular rates), and the three accelerometers (measures x, y and z accelerations). These analog channels were read into a Pentium III 700 MHz laptop using the National Instruments Data Acquisition (NIDAQ) unit. LabView was the program used to capture these readings and save them as text files at a rate of 300 Hz for single sensor values, or at a rate of 90 Hz for three sensors sampled together, or at 30 Hz for all eight sensors taken together. The LabView was setup to take 500 to 1000 samples, and store them into a text file. LabView was also used to record the time when the sampling started and the time when the sampling stopped. The Front panel used to display the start and stop time. The sampling rate was calculated using the following formula:

$$SamplingRate = \frac{StopTime - StartTime}{NumberOfSamples} \qquad (5.1)$$

Attached to the avionics unit by a double-sided tape was the 9-axis Micro Strain 3DM-GX1 IMU, which has three magnetometers (measures Euler Angles in Degrees), three rate gyros (measures angular rates in Degrees/sec), and three accelerometers (measures acceleration in G). Care was taken to align the 3DM IMU properly with the Avionics Unit in order to minimize alignment errors. However since the 3DM cannot coexist in the exact same space as the Avionics Unit sensors, there will be a steady but small offset error introduced. The theoretical sampling rate of this unit is 150 Hz for raw analog sampling, but the rate decreases to 50 Hz when gyro compensation calculations are performed. The readings taken by this IMU were fed into the laptop by a serial RS232 connection. A C++ code was written in VC++

6.0 to read the serial port of the PC and acquire the IMU readings, and store them into another text file with a time stamp for each reading.

Care was taken to mount the IMU as close to the center of gravity (CG) of the Avionics Unit as possible, because this minimizes any "lever arm effect" If the IMU is not mounted at the center of gravity, then rotations around the center of gravity will cause the accelerometers to measure an acceleration proportional to the product of the angular rate squared and the distance between the IMU and the center of gravity. Care was also taken to conduct the experiment in an area free of magnetic interference (such as DC motors) and ferromagnetic sources. Since the IMU uses the earth's weak magnetic field to measure heading, even small amounts of magnetic material or man made magnetic fields near the sensor can have large effects on the heading measurement. The compactness of the Avionics Unit makes it is almost impossible to isolate the IMU completely from other electronic.

With the connection ready, the avionics unit was forced into several rotational and translational and random motions to measure the pitch, roll, x, y and z rates and accelerations. The voltage readings obtained from the Avionics Unit were then calibrated against near simultaneous pitch, roll, rate and acceleration readings obtained from the IMU. The 3DM-G readings were taken at 20Hz and the analogs readings from the Avionics Unit were taken at 300 Hz, and the closest analog value in time to 3DM-G value were matched. For example, for an analog reading at time t_1 = 10 11 12 13 14 15, and a 3DM-G readings at time t_2 = 12.25 14.75, then the analog value at time t_1 = 12 was matched with the 3DM-G value at time t_2 = 12.25 and the analog value at time t_1 = 15 was matched with 3DM-G value at time t_2 = 14.75. Since the means of acquiring data from the two sources is different, they cannot be completely matched in time, however if the analog data acquisition rate is done at ten to thirty times the rate at which the 3DM-G readings are taken, then this error can be minimized. Figure 5-11 below shows calibration setup using NIDAQ.

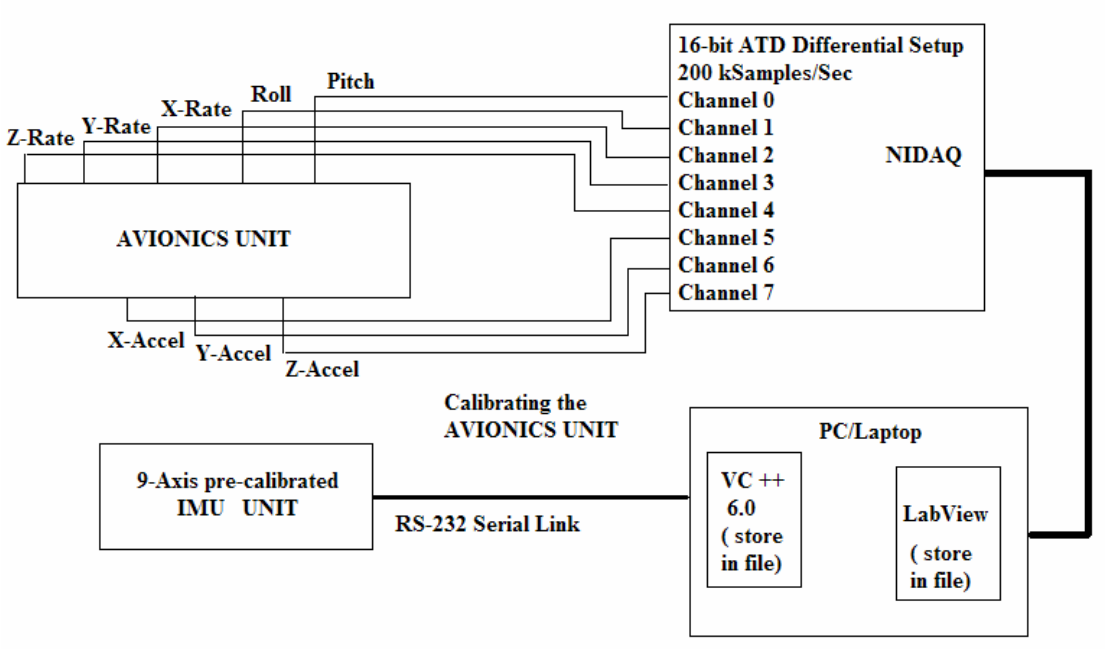


Figure 5-11: Calibration of the Avionics Unit using NIDAQ to Capture Analog Values

A filtering algorithm was written in C++ to match the readings of the IMU and the Avionics Unit. This process was coded using Visual C++ and the algorithm is shown below in Figure 5-12, and the NIDAQ unit used is shown in the Appendix D-3.

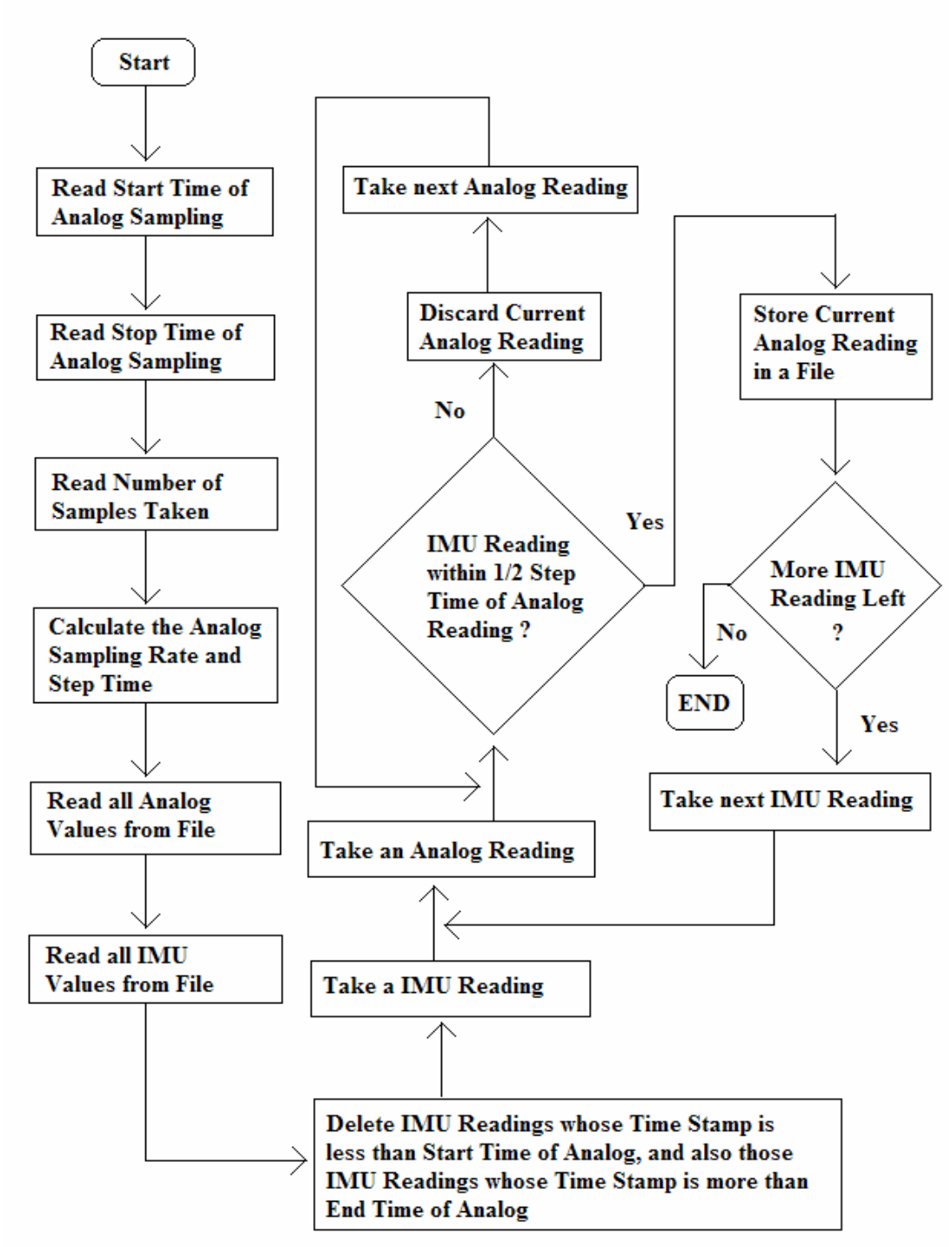
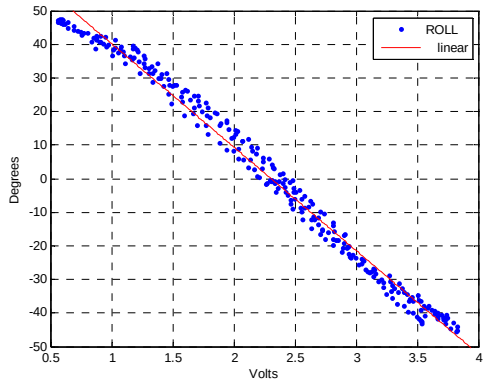


Figure 5-12: Flowchart of Filtering Operation to Match IMU and Avionic Unit Readings

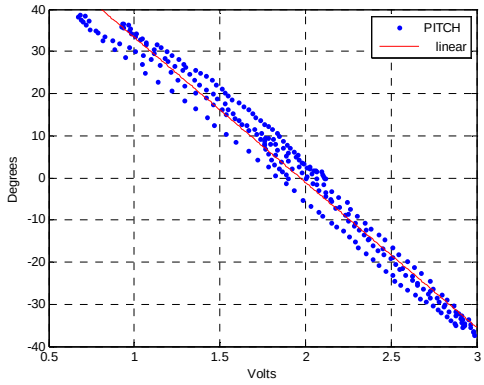
5.2.2 Calibration Results and Tuning of Inertial Sensors

The scatter plot of the filtered IMU readings and the filtered Avionics Unit readings are shown below in Figures 5-14 a-h. The Y-axis shows the IMU readings in

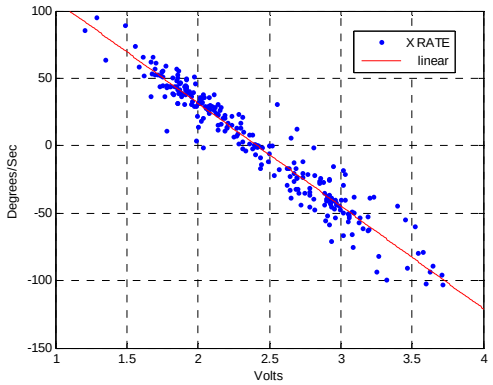
the respective parameter value, and the X-axis shows the Avionics Unit reading in voltage.



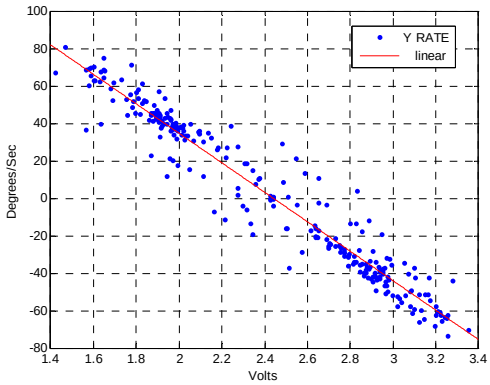
(a) Roll Relation



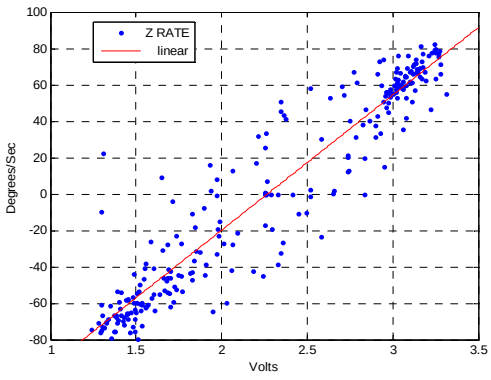
(b) Pitch Relation



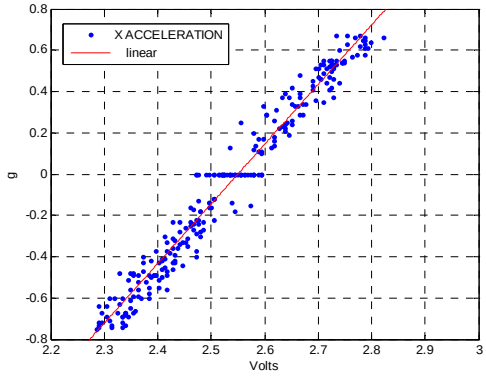
(c) Roll Rate Relation



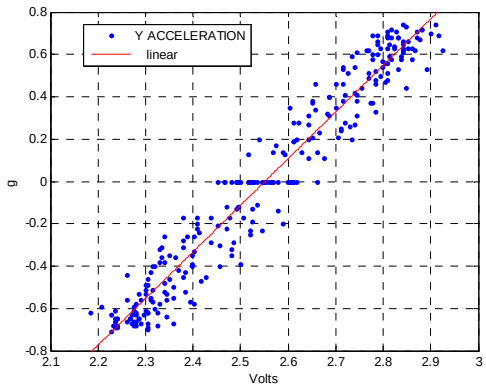
(d) Pitch Rate Relation

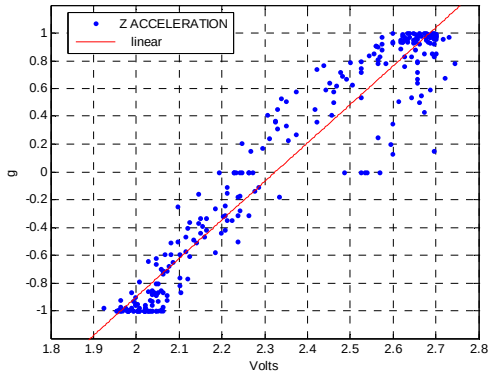


(e) Yaw Rate Relation



(f) X Acceleration Relation





(g) Y Acceleration Relation (h) Z Acceleration Relation

Figure 5-13 (a-h): Scatter plot of IMU Parameter Readings and Avionics Unit Voltages

Note from Figure 5-14 that a line of best fit (in red) has been drawn through the scatter plot. This line indicates the linear equation in y = mx + c relating the IMU reading (y) to the Avionics Unit voltage (x). The confidence of the line of best fit, which describes how closely the line fits most of the scatter points, has been auto calculated using regression analysis in Excel. Table 5-1 shows the initial linear formula relating the IMU parameter readings to the Avionics Unit voltages along with the respective confidences.

Table 5-1: Linear Equations Relating IMU Readings with Avionics Unit Voltages

Parameter	Equation	Confidence (%)
ROLL	y = -30.8x + 71.0	98.8
PITCH	y = -34.4x + 67.8	97.6
X RATE	y = -76.2x + 184.0	93.2
Y RATE	y = -78.5x + 192.0	94.7
Z RATE	y = 77.1x – 175.0	97.6
X ACCELERATION	y = 2.89x - 7.37	96.5
Y ACCELERATION	y = 2.25x - 5.74	96.7
Z ACCELERATION	y = 2.48x - 6.57	96.1

The filtered Avionics Unit voltages were plotted using the equations shown in Table 5-1, and those plots were superimposed over the plots of the filtered IMU reading plots. The two plots were further tuned by aligning the y-axis offsets, and matching the scale. The following steps are involved in the tuning process:

- Match the scale of both the plots: Scale defines the peak to peak range between the maximum and the minimum value of the plot. If the two scales do not match, then change the slope of the equation (M) by the ratio of the two scales

$$M_{NEW} = M_{OLD} * \frac{IMU_{SCALE}}{AU_{SCALE}} \qquad (5.2)$$

- Align the offsets of both plots: Redo the plot of the Avionics Unit with the new slope (M), and then superimpose the new plot over the IMU plot. Find the amount by which the two plots are shifted from each other in the Y-axis, by finding the difference in the average values of the two plots. Remove this shift from the y-intercept (C) of the equation using the formula shown below.

$$C_{NEW} = C_{OLD} + IMU_{AVG} - AU_{AVG} \qquad (5.3)$$

- Iterate the above process until satisfactory results are obtained.

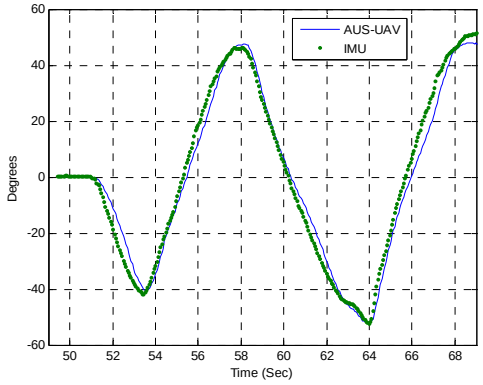
Once satisfactory tuning has been done, the calibration experiment was redone, and the new Avionics Unit voltages were compared to the new IMU readings using the tuned equations to ensure that the tuned equation holds. Figures 5-15 a-g and Table 5-2 shows the tuned equation and the superimposed plots of the both the IMU reading and the Avionics Unit readings calculated using the tuned equations. Notice the confidence level of the readings have increased, which indicates that the readings are more reliable and more accurately match the data available. Changes from previous equations are shown as M for slope change and C for y intercept change.

Table 5-2: Tuned Linear Equations Relating IMU Readings with Avionics Unit Voltages

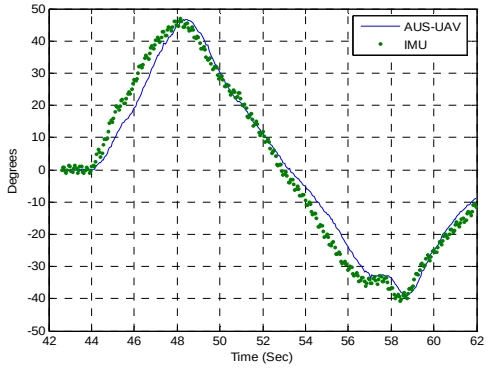
Parameter	Equation	Confidence (%)
ROLL	y = -29.1x + 68.5 (MC)	99.1
PITCH	y = -34.4x + 67.8 (U)	97.6
X RATE	y = -76.2x + 185.0 (C)	97.2
Y RATE	y = -78.5x + 192.0 (U)	97.7
Z RATE	y = 77.1x – 175.0 (U)	97.6
X ACCELERATION	y = 2.81x – 7.36 (U)	98.4
Y ACCELERATION	y = 2.77x – 6.65 (MC)	97.6
Z ACCELERATION	y = 2.78x – 6.83 (MC)	98.3

The plot for the pitch rate is not shown due to the fact that the 3DM-G IMU developed a slight oscillatory problem with pitch. These oscillations caused severe

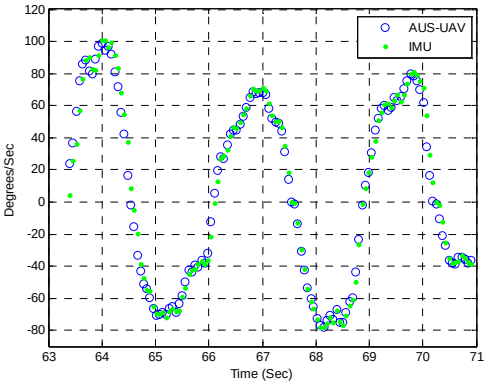
noise (i.e. noise with a large magnitude), which completely obscured the experimentally induced pitch rate of the IMU.



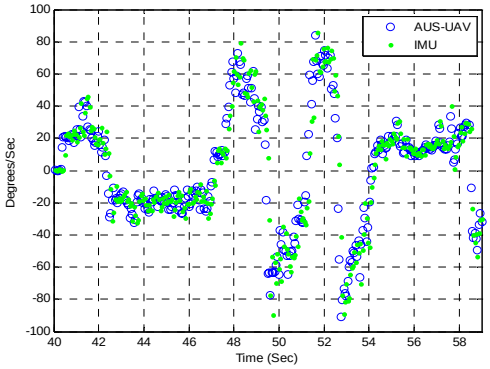
(a) Roll Comparison



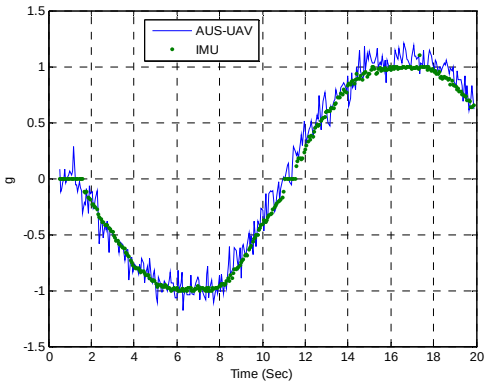
(b) Pitch Comparison



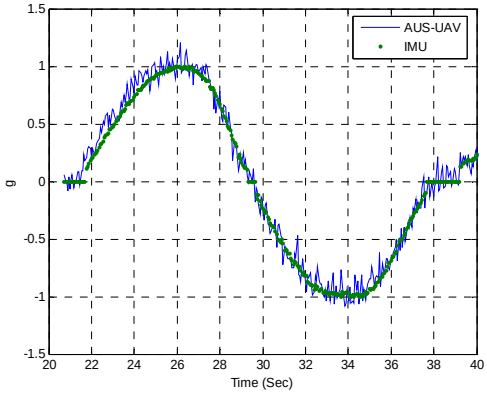
(c) Roll Rate Comparison



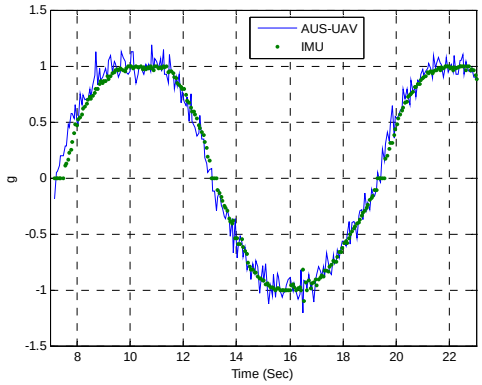
(d) Yaw Rate Comparison



(e) X Acceleration Comparison



(f) Y Acceleration Comparison



(g) Z Acceleration Comparison

Figure 5-14 (a-g): Plot comparing IMU readings with the Tuned Avionics Unit Readings

5.2.3 Calibration of Inertial Sensors using the Microcontroller

The final calibration was carried out using the microcontroller to read both the Avionics Unit sensors using its ATD converter, and to read the IMU unit simultaneously using its serial port. Since the same source was used to read both devices, the readings were automatically matched in time, and thus no extra filtering algorithm needed to be written. To ensure near simultaneous reading, request was first sent to the IMU using the serial port to send back gyro compensated Euler Angles, Rates, and Acceleration readings. At the same time the ATD port of the microcontroller was commanded to sample the Avionics Unit sensors at the same time.

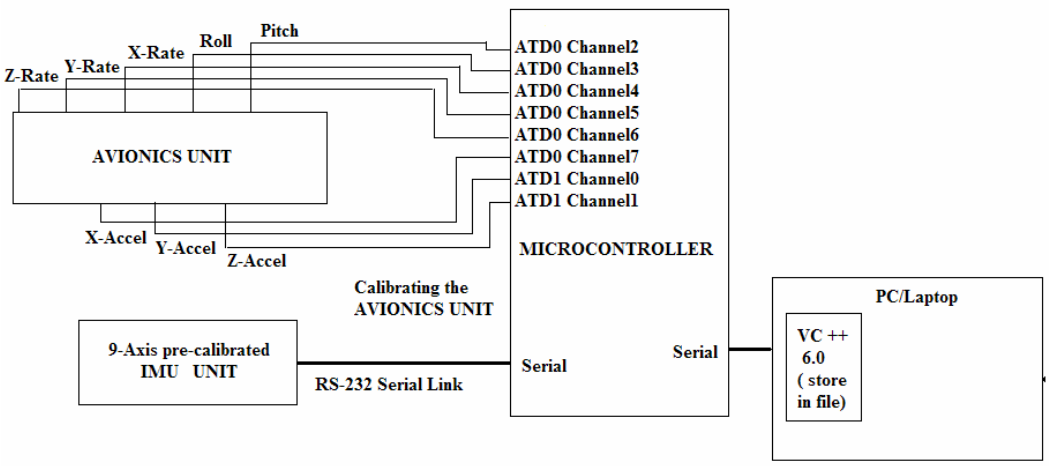


Figure 5-15: Block Diagram of the Calibration of Avionics Unit using Microcontroller

Figure 5-16 above shows the setup of the final calibration routine utilizing the microcontroller, and Figure 5-17 shows the picture of the setup with the Avionics unit in the top and the 3DM-G IMU in the bottom towards the center of the IMU. There is a battery that supplies both units with power. The serial cable of the IMU (the shiny silver one) is attached to the second serial ports of the microcontroller. The first serial port is used to communicate with the PC.

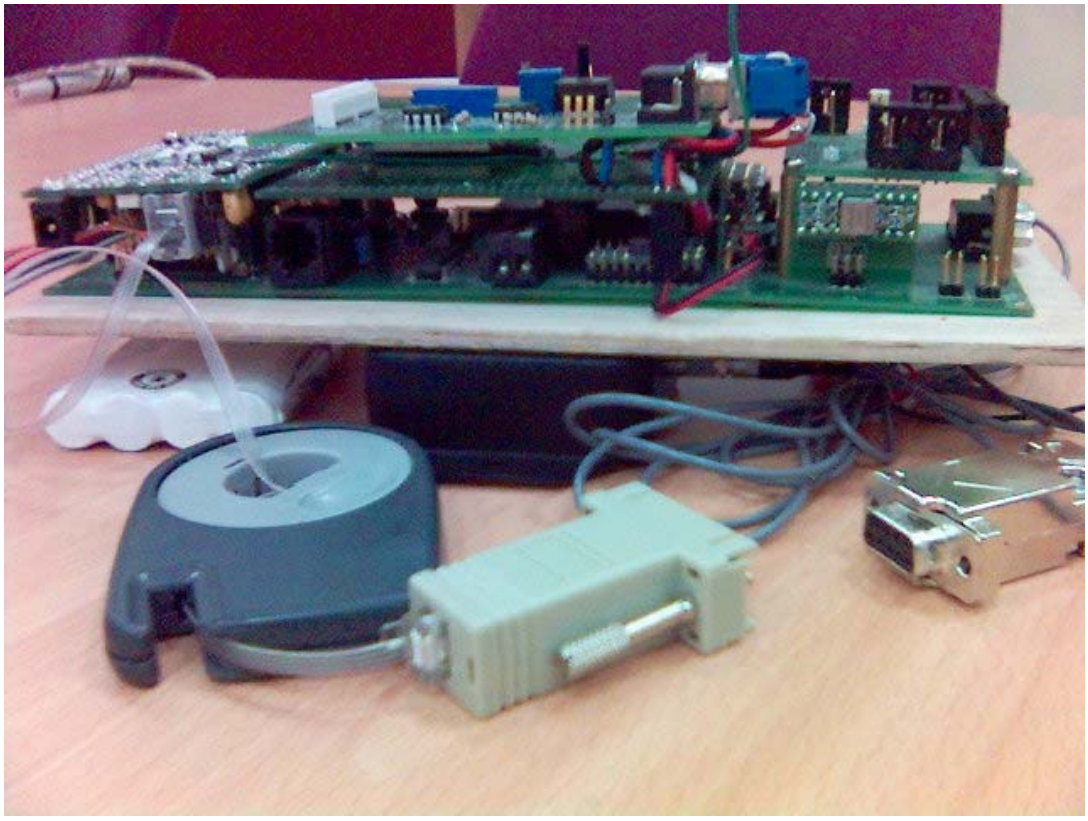


Figure 5-16: Avionics Unit Calibration using Microcontroller

This final experiment was important during the test flight the microcontroller will be the one that samples the sensor reading, so it is important to ensure that the formulas obtained from the previous calibration routine holds. The microcontroller also has a 10-bit ATD as opposed to a 16-bit ATD port of the DAQ, and therefore its sensor readings will not be as precise. On the other hand, filtering algorithms can be written for the microcontroller such as taking the average of 10 readings, or keeping a moving average of the sensor reading, to smoothen out the noise. The microcontroller unfortunately can only sample maximum at 100 kilo samples per second, which is half the speed of the DAQ unit. However since these extreme sampling rates are not necessary, the speed of sampling is not an issue.

For most part, the sample taken using the microcontroller matched the tuned formulas, given in the previous section. However there were slight changes and these are going to be listed in Table 5-3. These will be compared with formulas given in [1] to show the extent of which the previous calibration routine and the current one differ.

Table 5-3: Final Calibration Equation and Comparison with Equations given in [1]

Parameter	Current Calibration Equation	Previous Calibration Equation
ROLL	Y = -29.1x + 68.5 (U)	Y = 32.0x – 64.4
PITCH	Y = -34.4x + 73.5 (C)	Y = –32.0x + 66.9
ROLL RATE	Y = -76.2x + 182.0 (C)	Y = 60.0x – 159.0
PITCH RATE	Y = -78.5x + 190.0 (C)	Y = 60.0x – 144.0
YAW RATE	Y = 77.1x – 173.0 (C)	Y = 60.0x – 152.0
X ACCELERATION	Y = 2.81x – 7.36 (U)	Y = 2.67x – 6.24
Y ACCELERATION	Y = 2.77x – 6.65 (U)	Y = 2.69x – 6.98
Z ACCELERATION	Y = 2.78x – 7.13 (C)	Y = 2.80x – 6.80

Note from the above table that the microcontroller equations do not vary much from the tuned equations in terms of the slope of the equation. However, there are slight changes in the y-intercept for some of the equations, mostly the angular rates.

To get an idea of the accuracy of the above calibration equations, the 3DMG IMU is precise to within 0.1 degree, 0.1 degree/second, and 0.001G (based on the 12-bit ATD), and has repeatability to within 0.1 degree, 0.5 degree/second, and 0.005G. On the other hand the Avionics Unit measurements are precise to within only 2 degrees, 0.5 degree/second, and 0. 05 G (based on 10-bit ATD), and the repeatability falls within 3 broad categories:

Euler Angle: These sensors have the most repeatable with a variance of 0.1 degrees

Angular Rate: Angular rate sensors have a variance of 0.15 radians per second and are dependant on the frequency of the change in angular rate.

Acceleration: Accelerometers have the most noise and least repeatability with variance of 0.01G.

5.2.4 Calibration of the Air Data Unit

The air data component of the Avionics Unit consist of the Absolute pressure sensor to measure the altitude, the gauge pressure sensor to measure the speed, and the angle of attack sensor. The absolute and the gauge pressure sensor is connected to the Pitot tube that will measure the static atmospheric pressure and the dynamic

pressure. A servo potentiometer is used as the angle of attack sensor. The gauge pressure sensor was calibrated using the wind tunnel test. The absolute pressure sensor will be calibrated by measure voltage readings at various altitudes. There was no efficient means of testing the angle of attach sensor without using a room sized wind tunnel, therefore the calibration was made empirically using voltage output of the servo potentiometer at various angles.

A small scale wind tunnel was used to calibrate the gauge pressure sensor. The output voltage from the signal conditioning circuit of the gauge pressure sensor was compared with the airspeed given by a hand held airspeed measuring device. Care was taken to place the sensor of the hand airspeed measuring device orthogonal to the incoming air to get the most accurate readings. The device was pre-calibrated and is very accurate at low speeds. At low speeds up to 10 m/s, the measurement device has a resolution of 0.1 m/s. Beyond 10 m/s, the measurement device has a resolution of 1 m/s up to 20 m/s. After that the device has a resolution of 5 or 10 m/s. The maximum speed of the wind tunnel was 15 m/s where the Pitot tube was placed. Actually, the maximum speed is much higher if the Pitot tube can be placed in the optimal location. But, since this was not possible, the location where the Pitot tube was place, gave a max speed of on 15 m/s. Figure 5-18 shows the setup of the wind tunnel and the placement of the Pitot tube.

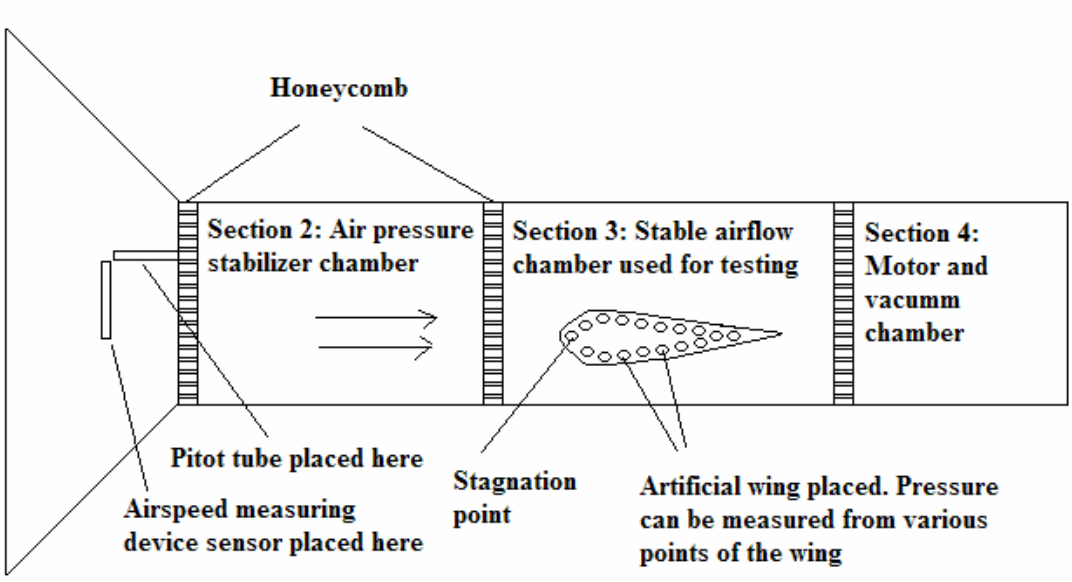


Figure 5-17: Wind Tunnel Calibration of Gauge Pressure Sensor

The Pitot tube operation in a wind tunnel is different from normal Pitot tube operation. Normally, the Pitot tube is rushing through the air and the air is nonmoving. Therefore the pressure measure at the stagnation point is higher than the air pressure at the sides of the tube. Therefore the edge of the Pitot tube measures the Dynamic air pressure. However, in a wind tunnel, the Pitot tube is nonmoving, and the air is rushing in to meet the Pitot tube. Air normally moves from one place to another due to a difference in pressure. The wind tunnel generates a vacuum, which the air rushes to fill. Therefore the pressure measured by the sides of the tube is the lower pressure, whereas the air pressure at the stagnation point is normal atmospheric pressure. In this case the side of the tube is responsible for measuring the dynamic pressure. Figure 5-19 illustrate the Pitot tube operation.

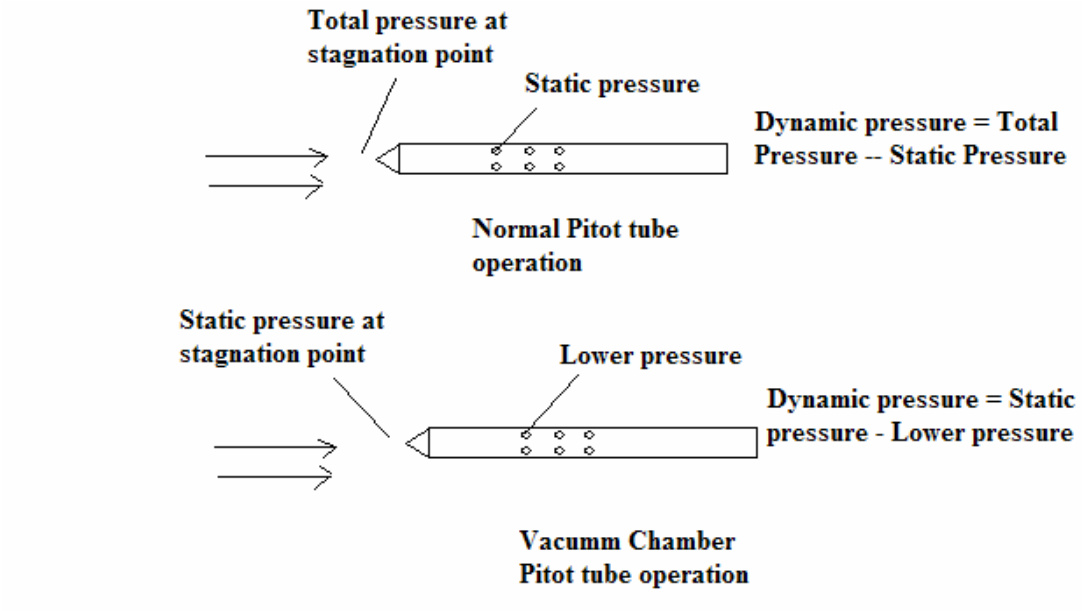


Figure 5-18: Illustration of Normal and Wind Tunnel Pitot tube operation

Table 3-4 gives the results and the analysis of the wind tunnel test data. The analysis done includes calculation of the actual voltage value from sensor equation, and comparison to the measured value, and the calculation of dP/dV, which is a constant.

Table 5-4: Tabulation of Wind Tunnel Results and Analysis

Wind Tunnel Speed (m/s)	dP (N/m^2)	Measure value (V) and dV calculated	Actual Value (dV) and measurement error (%)	dP/dV
0.0	00.00	1.72 (–0.00V)	---	---
2.5	02.44	1.70 (–0.02V)	0.024V (18.167%)	~122.0
4.0	06.25	1.66 (–0.06V)	0.062V (04.102%)	~104.0
6.0	14.06	1.58 (–0.14V)	0.140V (00.550%)	~100.0
10	39.06	1.33 (–0.39V)	0.391V (00.266%)	~100.0
15	87.89	0.84 (–0.88V)	0.890V (–0.019%)	~100.0

Accepting that the airspeed from the air measure devices is accurate, the pressure difference (dP) can be calculated, by assuming air density of 1.25 kg per meter cubed.

$$\Delta p = \frac{(\rho V)^2}{4} \qquad (5.4)$$

According to the datasheet for SX01, the sensitivity is 4mV/V/psi. That is with an input voltage of 5V, one psi of pressure difference will output 20mV. One psi is 6892.7 Pascal; therefore, the following table gives the output of the sensor and the output of the amplifier for an amplification of X3.45, the following formula can be used to calculate the actual dV.

$$dV = \frac{dPascal * 4\frac{mV}{V * Psi} * 5V * 3.45}{6892.7\frac{Pascal}{psi}} \qquad (5.5)$$

From Table 5-4 it can be noted that there is a constant measurement error is less than a percent for speeds greater than 5m/s. This is ok since the autopilots will operate at around 20 m/s, so the error is within acceptable limits. On the other hand the relation between dP and dV is constant at approximately 100 Pascal per volts, which correlates to 16 m/s speed per output voltage. Since the ATD can detect 5mV changes, the precision of the pressure sensor and it signal conditioning circuitry is 0.08 m/s or 8cm/s changes in speed.

Table 5-5 shows the calibration data of the absolute pressure sensor giving the voltage output at various altitudes. This has been accomplished by taking the voltage readings at several levels of a tall building. Note, that these readings are approximate because they do not take into account the humidity and temperature, both of which are low due to indoor air conditioning. Finally Table 5-6 shows the calibration equation

of the air data unit for speed altitude and angle of attack. These equations are compared to those found in [1] by theoretical means.

Table 5-6: Result of the Air Data Unit

Air Data Parameter	Current Calibration Equation	Previous Calibration Equation
FORWARD SPEED	Y=SQRT(225*(x–1.72)/0.88)	Y = 16.4x – 29.6
ALTITUDE	---	Y = –481.0x + 1444.0
ANGLE OF ATTACK	---	Y = 72.0x – 180.0

CHAPTER 6: HILS SETUP – SIMULINK MODEL DESIGN

Simulation is an important phase that must be completed prior to any real world practical testing. Unlike previously when mathematical models did not exist, excess money was needed to test actual prototypes of real systems. A classic example is the auto industry. In fact, there was no was of knowing before making the prototype or the real system if it would work as specified. If system were high powered machinery that performed dangerous operation, failure of the equipment would often be catastrophic and lead to losses in life. Currently modern systems are always by default simulated in software before the actual prototype is tested. With the accumulation of knowledge, it is possible to acquire accurate mathematical models of complete systems, and test them quickly due to advances in computing power. Thus simulations reduce costs, and increases efficiency.

Pure simulation is often used to understand the behavior of a system, or to predict an outcome under different internal and external influences. Unfortunately although a pure simulation alone is not enough to predict all possible outcomes due to the fact that not all practical considerations may be taken into account. For most real systems, there are characteristics that are unknown or too complex to model by pure simulation. That is why nowadays Hardware in the Loop Simulation (HILS) testing is done. HILS is basically a specific form of simulation, where a part of the mathematical model is replaced by the actual hardware. Good system engineering practice would begin with a pure simulation and as components become better defined with the aid of simulation, they can be fabricated and replaced in the control loop. Once physical components are added to the loop, un-modeled characteristics can be investigated, and controls can be further refined. The use of HILS eliminates expensive and lengthy iterations in machining and fabrication of parts, and speeds development towards a more efficient design [1].

6.1 Simulation Only Design

The RC plane that is going to be used as the UAV is the TRI-60 model. This model is created in Simulink in [1] to simulate the aerodynamic property of the UAV during flight. The TRI-60 model is implemented using the Aerosim Blockset version 2.8. Figure 6-1 shows the implementation below. The model takes as input control signals of the aileron, rudder, elevator and throttle specified in degree radians. The

model outputs the various sensor data that are simulated, including analog data from gyros, accelerometers, tilt sensors, compass, and GPS sensor. These sensor values are simulated using aerodynamic the model of the TRI-60 UAV, the propulsion model of the engine, the atmospheric and earth model, and a model that solves the equations of motion, and sum of forces and moments.

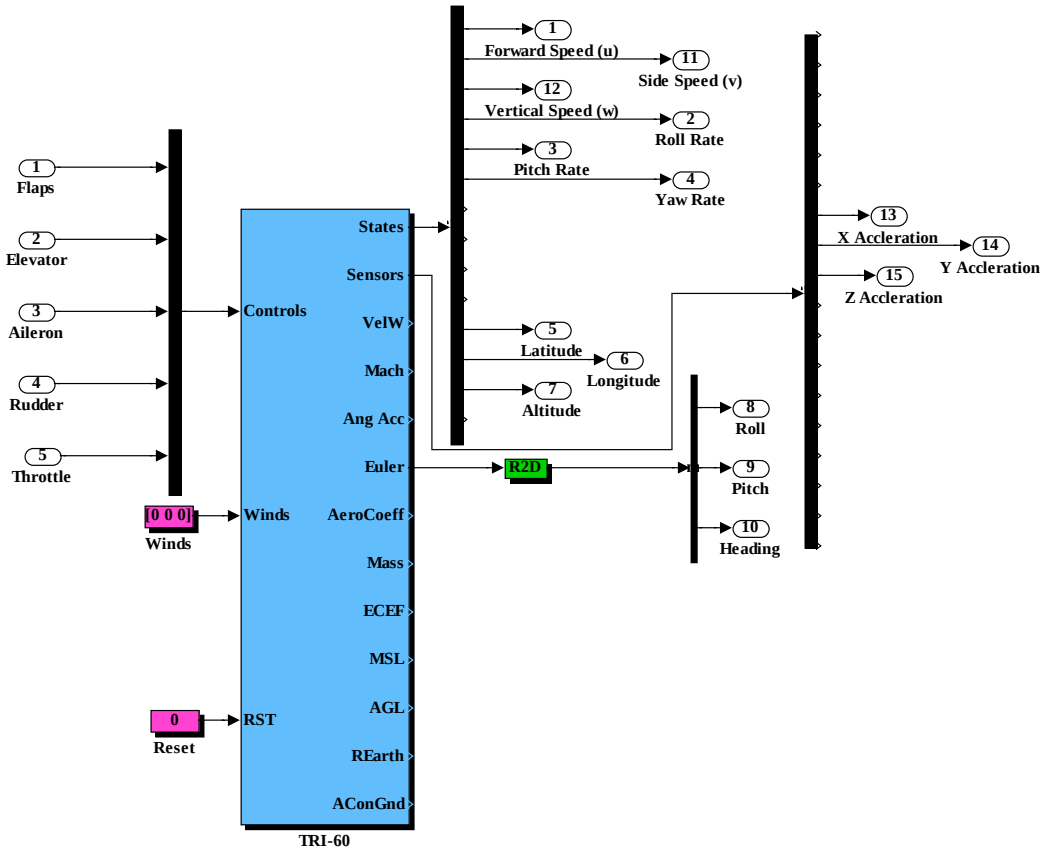


Figure 6-1: Simulink model of TRI-60 UAV Implemented using Aerosim Blockset 2.8

6.1.1 Design of Trajectory Tracker and Autopilot Algorithm

The TRI-60 model was first used to iteratively simulate the effects of the autopilots to various reference commands. This was essential to know the dynamics limits and constraints of the autopilot. The results of these investigations are discussed in Chapter 3 under the section 3D Path constraints. To summarize, it was discovered that the Longitudinal Autopilot can accept a maximum altitude increase command of 35 meters and a maximum altitude decrease command of 95 meters, without destabilizing the aircraft. To maintain aircraft speed and attitude to within

their respective sensor limits (tilt sensor limit at +/- 70 degrees, and gauge pressure sensor limit at maximum 30 m/s), it is necessary that the commanded reference does not force the aircraft to go beyond these limits, even for short periods of time. To ensure that is the case the maximum reference altitude increase command should be limited to 20 meters, and the maximum altitude reference decrease command should be limited to 25 meters. Since the autopilots were completely tested for stability in [1], stability analysis was not does for this thesis.

The next step was to test the design of the Trajectory Tracker algorithm in Simulink using the TRI-60 model and the available autopilots. The Trajectory Tracker is built around the Autopilots using the same algorithm discussed in Chapter 2 and implemented in Chapter 3 in the embedded system. The Simulink Model was connected to a Flight Simulator program called Flight Gear using an Ethernet link, to display the motion of the aircraft during the simulation. Flight Gear gives a visual of the aircraft attitude during flight, which is more intuitive to the user than just numbers and graphs available from the simulation. Figure 6-2 shows the design of the Trajectory Tracker algorithm in Simulink. All the necessary inputs and outputs are shown for the various components such as the Autopilots, Trajectory Tracker, TRI-60 model and Flight Gear Flight Simulator (FGFS).

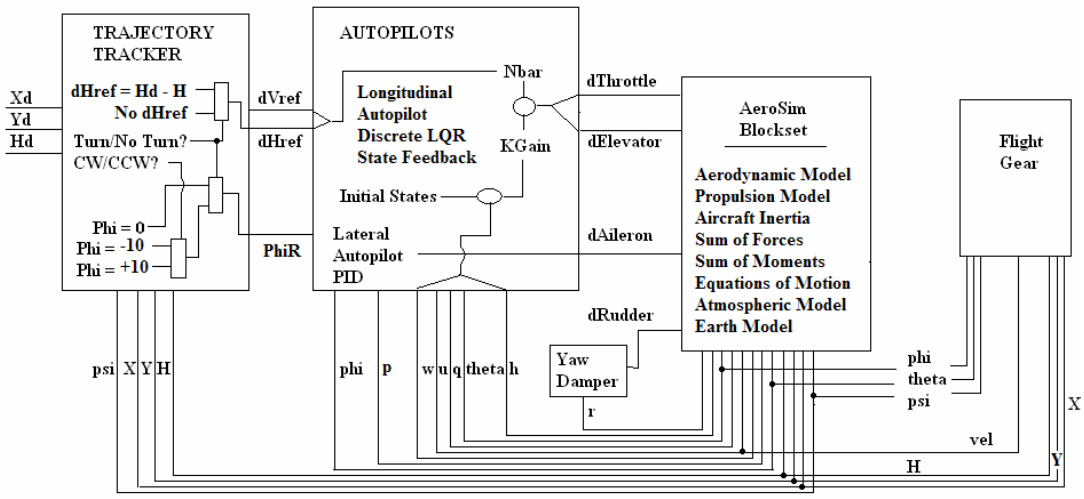


Figure 6-2: Trajectory Tracker Algorithm implementation in Simulink

The Trajectory Tracker shown above is implemented using a PIID controlled Lateral Autopilot using the Roll as the P control and the Roll rate as the D control for the aileron, while the yaw rate is used as the Proportional control for the rudder as yaw

damper. In a later section, an LQR state feedback Lateral Autopilot will be discussed as an enhancement.

6.1.2 Design of Waypoint Navigation System

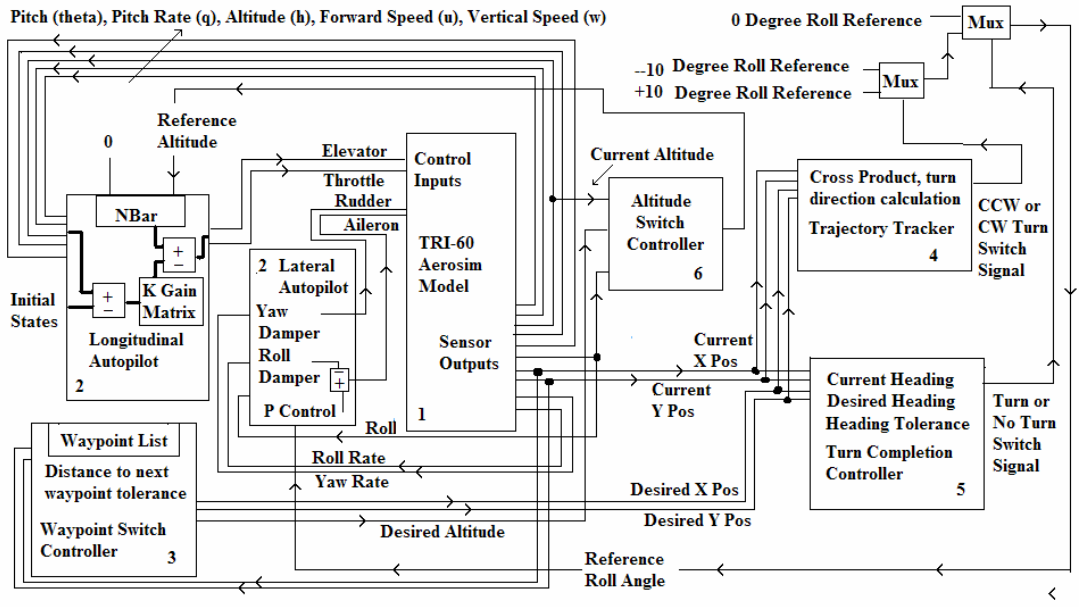


Figure 6-3: Simplified Block Diagram of Waypoint Navigation System in Simulink using PID Lateral Autopilot

Figure 6-3 above shows the complete "simulation only" Simulink model that implements the waypoint navigation system. This model consists of six main components. The first and the main component is the TRI-60 Aerosim model, which implements the aerodynamic model, the earth and atmospheric model, the propulsion model, and the equations of motion. It takes in as input aerodynamic controls values such as aileron, rudder, throttle, and elevator and wind direction and speed. It outputs the state and location of the aircraft, which can be used as "sensor" values. The second component is the longitudinal and lateral autopilot. The longitudinal autopilot is responsible for speed and altitude control. The lateral autopilot is responsible for heading control. The third component is the waypoint switch, which is responsible for switching from one waypoint to the next when the UAV reaches to within 50 meters of a waypoint in a 2D horizontal plane. Waypoint consists of the desired x, y, and z locations. The next three components together are part of the Trajectory Tracker. The fourth component is the "turn entry" layer which determines if the UAV should turn and if so, the direction of turn. The decision to turn is taken if the heading

error is more than two degrees. The fifth layer is the turn completion layer, which determines when the UAV is ready to exit from a turn and follow a straight line to the next waypoint. The decision to exit a turn is taken when the current UAV heading is within six degrees of the desired heading. The reason for exiting the turn quite early is that the UAV take a finite amount of time to fly straight from a curve, and during this time the UAV covers the six degrees. The last component is the "altitude tracking" layer. This layer is responsible for commanding the longitudinal autopilot with a reference altitude, which is the difference between the current and desired altitude. The altitude tracker only becomes active when the UAV is not turning or more specifically when the UAV has just completed a turn. The decision to change altitude while flying in a circle or flying straight has been discussed in Chapter 3. A sample result of the test run is show in Figure 6-4.

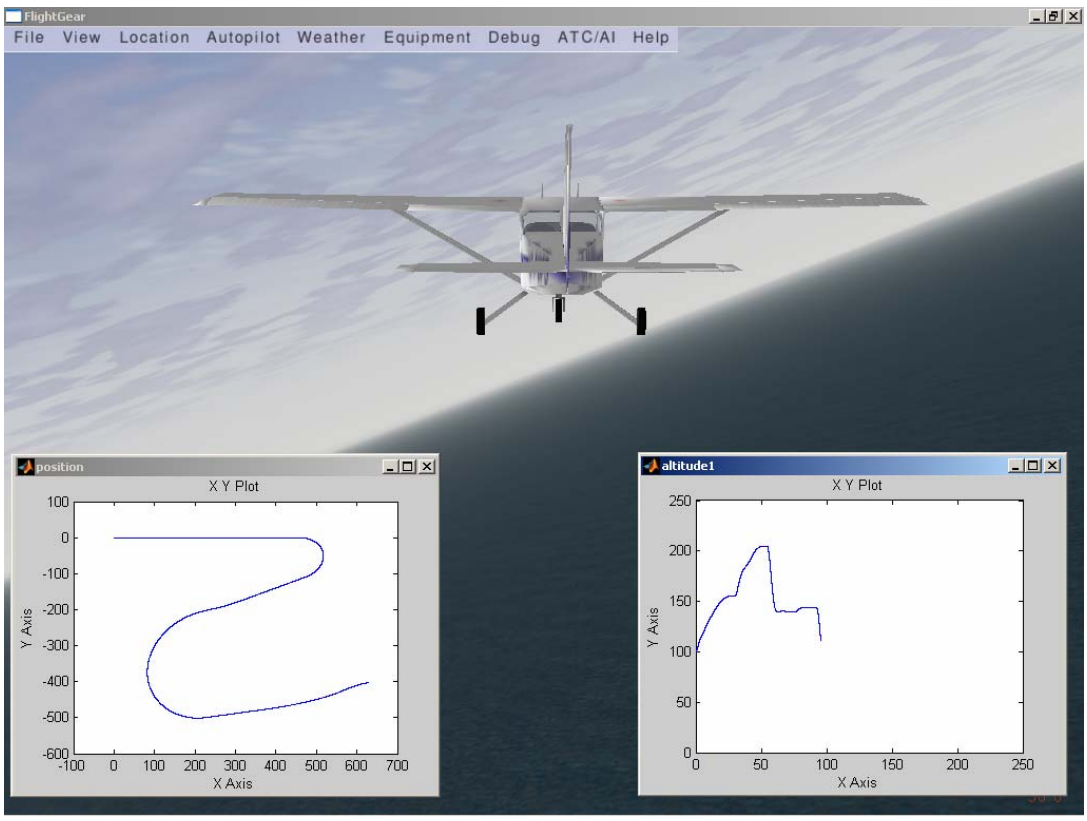


Figure 6-4: Sample Waypoint Navigation Result in FlightGear and Matlab X-Y plot

The above figure shows a snapshot of the simulation during run time. Flight gear shows the aircraft attitude and motion in three dimensions, where as the X-Y plot shows the position of the aircraft in three dimensions. The next chapter will analyze the results and examine the effectiveness of the trajectory tracker algorithm in detail.

6.1.3 Alternative to Bang-Bang Trajectory using PID Lateral Autopilot

LQR State Feedback Lateral Autopilot has been utilized as an alternative to the PID controlled Lateral Autopilot and the performance of both autopilots will be tested to ascertain the best autopilot to utilize in the embedded system. To recap PID Lateral Autopilot controls the output, which is the roll angle (phi), using the aileron. The roll angle forms the proportional control for the aileron, whereas the roll rate forms the roll damper control for the aileron. The rudder is used to control the yaw rate (yaw damping). The State feedback approach on the other hand creates the feedback gains for all the states involved in the lateral motion. A complete state feedback approach would control the roll angle (phi), roll rate (p), the yaw rate (r), the side velocity (v), the side acceleration (yb), the heading error, and the track error. To have a more detailed understanding of the implementation of the Lateral Autopilot, see Section 3.1.3. The following figure shows how a LQR state feedback Lateral Autopilot can be integrated into the Waypoint Navigation architecture using Bang-Bang control. Figures 6-5 shows the State Feedback Lateral Autopilot and it is waypoint navigation implementation, and the equation below shows the control input generation.

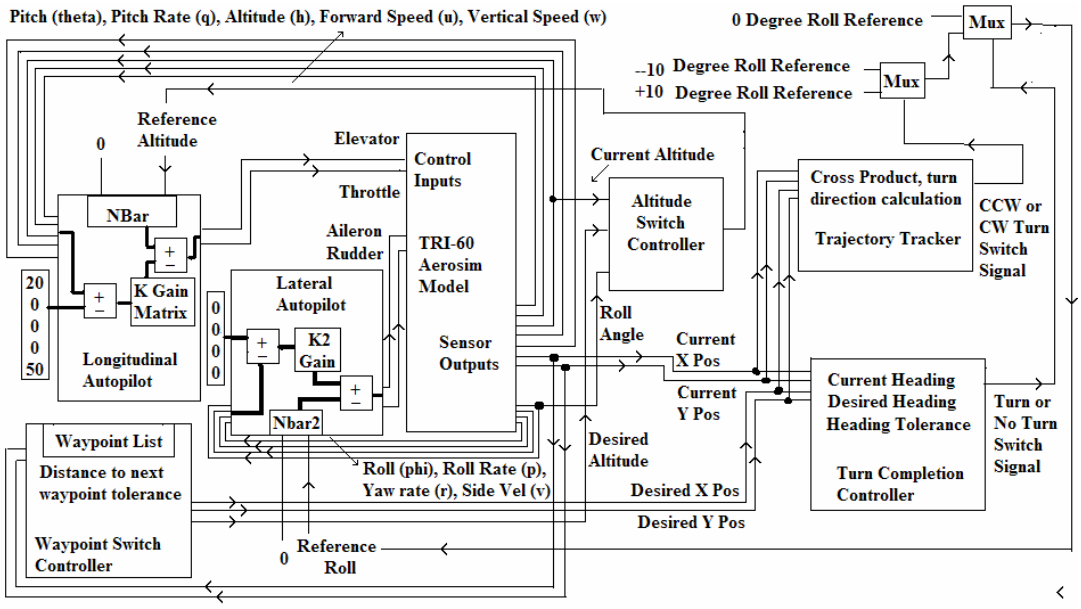


Figure 6-5: Simplified Block Diagram of Waypoint Navigation System in Simulink using LQR State Feedback Lateral Autopilot

Another alternative method of waypoint navigation is the replace the bang-bang Trajectory Tracking algorithm with a heuristic algorithm. In the bang-bang

control scheme, the rolls command is 10 degrees, -10 degrees, or 0 degrees as shown in Figure 6-2, 3 and 5. However with a heuristic design, the roll command is proportional to the heading error, with the roll command saturating at +/- 10 degrees. This offers a smoother transition from a circle to a straight line and vice versa. With a heuristic approach, the heading error is minimized to zero, by PID action soon after the UAV exits a turn. This saves the effort to make corrective turns as the UAV approaches a waypoint. With bang-bang approach, the UAV is forced to a Dutch roll about the zero heading error line. The Dutch rolling effect can be minimized by increasing the tolerance of the heading error by up to 5 degrees, but this method has the disadvantage that corrective turns occurs quite late. Figure 6-6 shows the Trajectory Control made using the heuristic design.

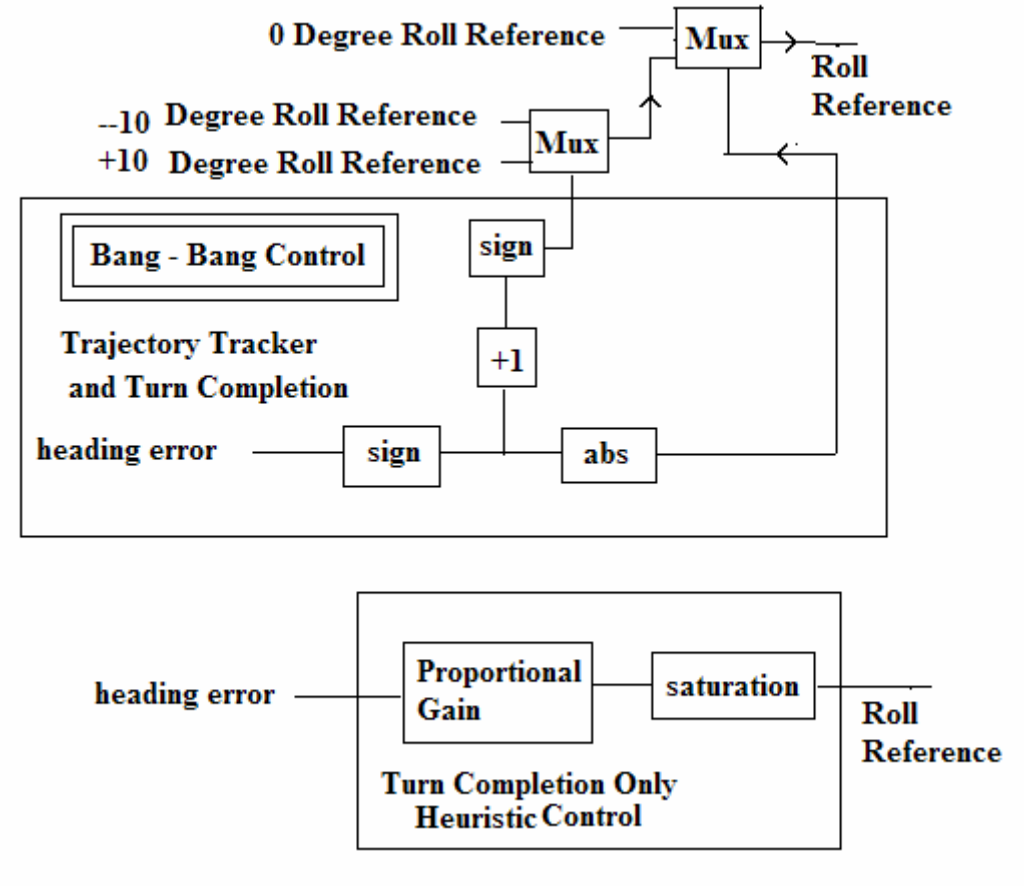


Figure 6-6: Difference between the Heuristic and Bang-Bang approach

CHAPTER 7: HILS SETUP – DSPACE AND FLIGHT GEAR

7.1 Hardware in the Loop Simulator Design

The next step is to design the actual Hardware in the Loop Simulator Setup. Where as the "Simulation Only" Simulink model was completely "closed loop" containing the autopilots, trajectory tracker, waypoint navigation, TRI-60 model in one model and can run on it own. The Simulink HILS model on the other hand is an "open loop" model containing only PWM inputs, the PWM duty cycle to degree radians scaling function, the TRI-60 Aircraft model, the sensor outputs, the sensor parameter to voltage scaling function, and the sensor parameter to NMEA sentence creation function. The model is made using RTI Matlab toolbox supplied by dSpace. Since it is an open loop model it cannot run on its own for long without the aircraft becoming unstable. The closed loop control is provided by the autopilots, trajectory tracker and dead reckoning algorithm implemented in the embedded controller in the Avionics Unit (Chapter 3). The Avionics Unit along with the Ground Station forms the "hardware" component. The Simulink HILS model is shown below in Figure 7-1.

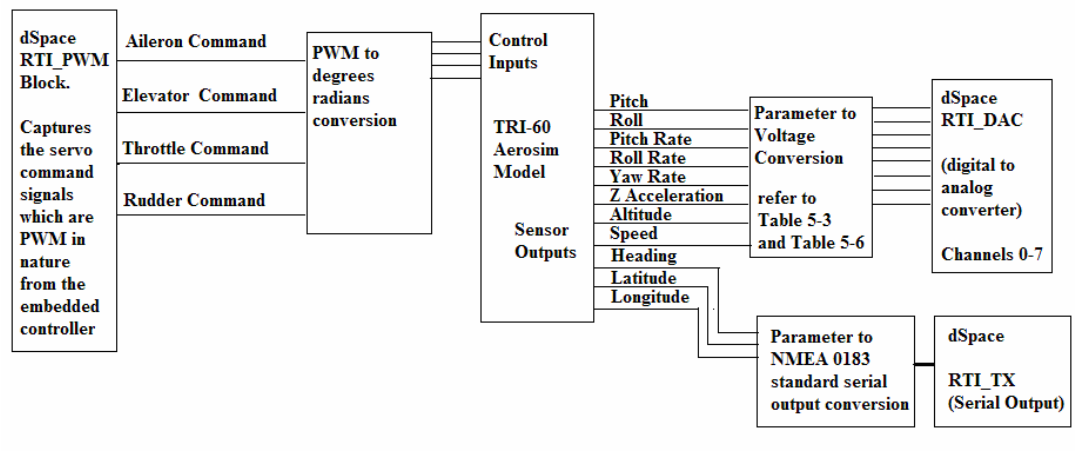


Figure 7-1: Simplified Block Diagram of the Simulink Model used in the HILS Setup

Figure 7-2 below shows the overall setup of the Hardware in the Loop Simulator Setup. It consists of the "hardware" part, which is a combination of the Ground Station and the Embedded System, the Simulator portion which consists of the dSpace Unit loaded with the model shown in Figure 7-1, and runs a virtual aircraft that can be controlled by the Embedded Controller, and the Flight Simulator part for visual effects consisting of another Simulink model that takes in information from

dSpace and sends them to Flight Gear. The figure also list all the data sent between each of the components.

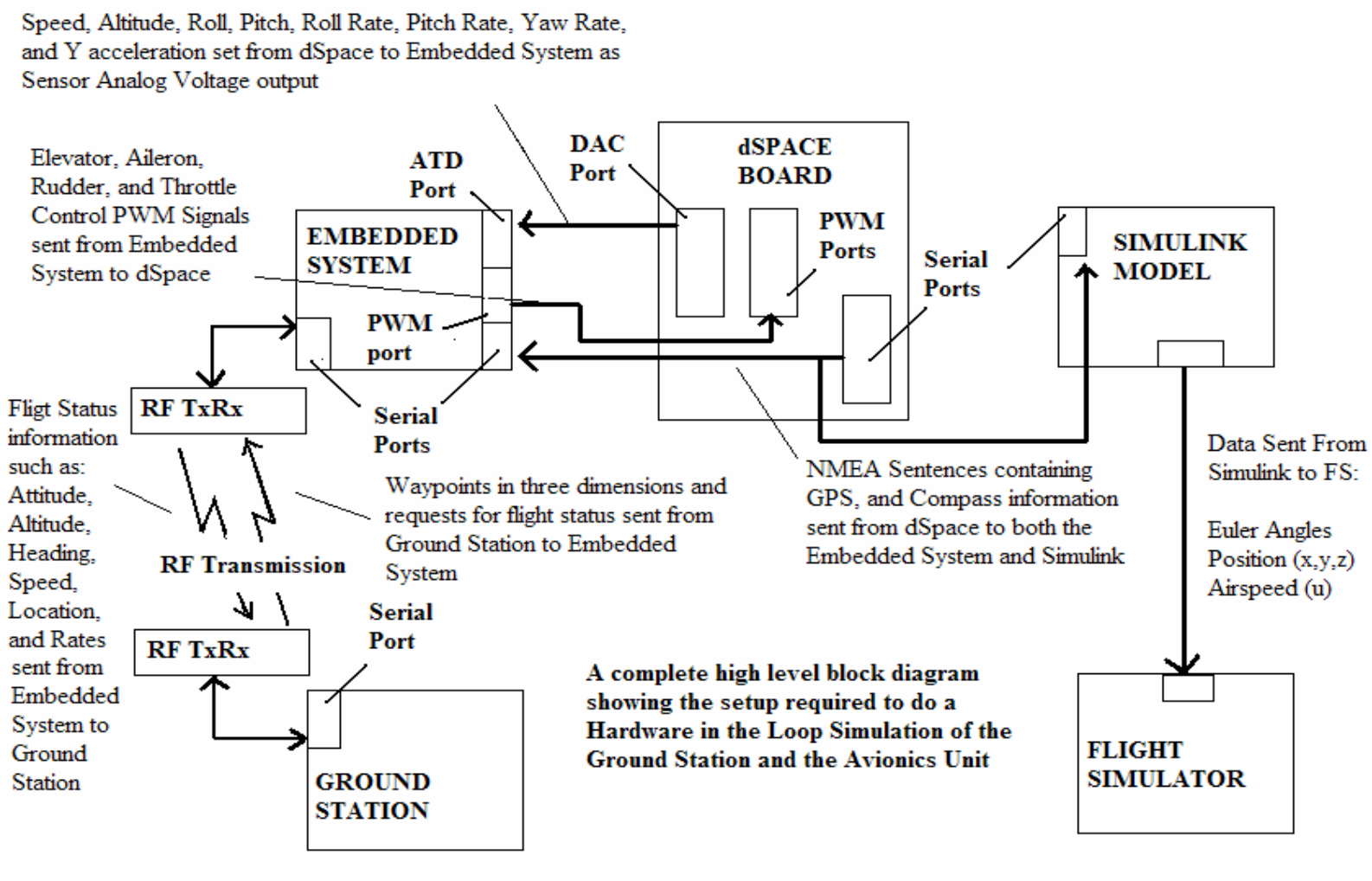


Figure 7-2: Overview of the HILS Setup for the Embedded System and Ground Station

7.1.1 High Level Overview of the HILS Setup

Hardware in the loop simulator is used to test the functionality of the Ground Station, and the Trajectory Tracker, the two autopilots, and dead reckoning algorithm implemented in Avionics Unit's embedded system. This simulator models an actual UAV in flight performing a mission. Mission consists of a set of waypoints that the UAV must fly over in sequence. There are four components to the setup:

- **Ground Station**: This unit will display the status and the location of the UAV to the user using various graphics and text based objects. The Ground Station will also allow the user to select waypoints and enter other mission commands. The Ground Station allows the user to plan mission, by displaying the path taken by the UAV for a given set of waypoints. Once the user is satisfied with the mission, the Ground Station communicates allows the user to send the waypoints to the UAV. The Ground Station was created using Visual C++ 6.0.
- **Avionics Unit**: This unit is placed onboard the UAV and is responsible for taking various flight parameter measurements. These include inertial measurements such as Euler angles, angular rates, acceleration forces, location, altitude and speed. The embedded system in it implements a dynamic flight controller, longitudinal and lateral autopilots, a Trajectory Tracker, and dead reckoning algorithm to guide UAV on its mission. The embedded system is also responsible for accepting missions from the Ground Station and periodically sending UAV status and location to the Ground Station. The embedded controller together with the Ground Station will be the "hardware" that will be tested. The embedded controller consists of the "Mini-Dragon" and the "S12 compact" boards based on Motorola HCS12DP256 microcontroller family.
- **Simulink Model**: This unit contains the mathematical model of all the sensors, actuators, and engines onboard the UAV. Simulink also contains the atmospheric, earth, and flight dynamic models and equations of motions model. Basically the Simulink model will simulate the actual UAV in flight, and will respond based on the actuator commands (PWM signals) from the Avionics Unit [1], pass it through the linear aerodynamic model and non-linear overall model, and then calculate appropriate sensor outputs. Some sensor information are output as voltage values through the analog port, while others are output through the serial port.

- **Flight Simulator**: This unit is there to show the user a visual representation of the UAV in flight [2]. Flight Gear receives the information it needs to update its display from the LAN network. Aerosim Blockset for Simulink has a FlightGear interface that facilitates such communication.

The user will select waypoints in the Ground Station, generate a path, verify that the path is suitable, and then send the waypoints to the Embedded System via the RF transceiver. The embedded system in the mean time will be waiting for the waypoints from the Ground Station. Once the embedded system receives the waypoints, it sends it back to the Ground Station to let the user verify that the correct waypoints have been received. The mission starts with the user acknowledging the receipt by the embedded system of the correct waypoints. The trajectory controller will utilize these waypoints to guide the 'UAV' that is modeled by Simulink.

During the simulation, the Ground Station will request periodically the flight status information from the embedded system via an RF transceiver. Once the embedded system receives a request from the Ground Station, it will briefly halts its Trajectory Tracker and Autopilot algorithm and sent the required information to the Ground Station via the RF transceiver. Once the transmission is complete, the Trajectory Tracker and Autopilot algorithm will continue. The Ground Station will utilize these data to display the position of the UAV on the map, the heading and orientation in the compass and artificial horizon respectively. The user can verify if the UAV tracks the generated path on the map, and can utilize this information to tune the path algorithm.

The trajectory controller programmed in the embedded system will use the heading, altitude, speed, current location and the next location and altitude to guide the UAV to its next waypoint by generating reference commands to the autopilots. The autopilots take these reference commands and generate control signals for the ailerons, elevator, rudder and throttle. The autopilot needs the body axes velocities including the forward (u), side (v) and vertical (w) velocities as well as the angular rates (p, q, r) and Euler angles (phi, theta, psi) to generate control signals. The control signals are such that the aircraft is always stable during flight. The control signals are servomotor commands generated using Pulse Width Modulation (PWM). These PWM signals are input to Simulink via the PWM input/output port of the dSpace kit.

The dSpace unit runs the real time model of the TRI-60 aircraft during flight condition. The TRI-60 model is "downloaded" into the dSpace 1104 controller board

previous to the running of the simulation. The TRI-60 model in the dSpace unit will communicate the simulated sensor data requiring voltage output to the Avionics Unit via the digital to analog converter. These analog outputs include the speed, altitude, y acceleration, the angular rates and the tilt angles. Heading (Compass) and position (GPS) information are input back to the embedded system via the serial port in the dSpace kit using RS-232. The TRI-60 model receives from the embedded system PWM control signals for the elevator, ailerons, rudder and throttle. The PWM signals are converted to control signals in degree radians and then sent through the TRI-60 model to simulate the next sets of sensor outputs.

During the simulation, outputs from the dSpace unit are input back into another Simulink model via the serial port. The Simulink model parses this information and extracts the Euler angles, position, altitude, and speed and sends this information to the Flight Gear Flight Simulator program via the Ethernet LAN. Flight Gear shows visually the aircraft during flight. The visual is most prominent in displaying the attitude of the aircraft that can be deduced by looking at the horizon. The motion of the aircraft can be deduced by looking at the "ground" over which the aircraft moves.

Chapter 3 describes in detail the implementation of the Ground Station and the embedded system design. The simulator of the HILS setup is a simple open loop design described in the previous section. The next section will describe in detail the design of the Flight Gear Flight Simulator component.

7.1.2 Flight Gear Flight Simulator Design

The HILS setup can be considered complete without the necessity of a Flight Simulator program. The objective of the HILS is to test the hardware performance of the embedded system and Ground Station combination. Performance can be reviewed by verifying from simulation results the accuracy of the algorithms. Performance benchmark includes arriving to within 50 meters of a waypoint, making sure the path is smooth, and does not have to many bumps or oscillations, and making sure that the aircraft is not subject to frequent motions. All these can be verified without the need of any visual display of in-flight conditions. However, there is an appeal to visually display the aircraft in flight conditions, and this is where Flight Gear Flight Simulator (FGFS) comes in.

FGFS in its most basic form takes the Euler angles such as Roll, Pitch and Yaw (Heading), the position (x, y, z), and the airspeed. Roll and Pitch allows FGFS to display the attitude or the tilt of the aircraft. Roll attitude is displayed through changes in the horizon, which gets titled to the left or to the right with respect to the aircraft. Pitch attitude is displayed as the aircraft pointing up towards the sky or pointing down towards the ground. Small angle pitch attitude is displayed through the horizon which rises and falls with respect to the aircraft. The "relative" altitude of the aircraft is displayed by changing the distance between the aircraft and the ground. Distance from the ground is visually represented by the amount of detail that can be picked up by looking at the ground. Finally the aircraft is displayed by the motion of the ground with respect to the aircraft. Through out it is by checking how the surrounding changes with respect to the aircraft that the user can get an idea of the aircraft during flight condition.

Since Flight Gear accepts input only through the LAN network using the TCP/IP protocol, it is necessary to format the data accordingly. The Flight Gear interface of Aerosim allows this task to be accomplished, but this cannot be implemented as part the TRI-60 model that is run from the dSpace unit. This is because the dSpace controller board does not have an Ethernet port that allows LAN communication, and therefore dSpace cannot directly send information to Flight Gear. A separate Simulink model runs from the PC whose task is to acquire data from dSpace and supply it to Flight Gear.

An initial design envisioned the use of using National Instruments Data Acquisition Unit (NIDAQ) to sample the analog output of dSpace and pass them to Flight Gear. However, it was not possible to integrate the data acquisition unit in Matlab. This is due to the fact that it is necessary to "build" any the Simulink model for real time execution, and the Flight Gear interface did not build properly. Therefore an alternative serial method of communication was developed to acquire data from dSpace. The "CommStr" Blockset allows the user to acquire serial RS232 data as complete strings, which can then be parsed to extract specific information. Position (x, y), altitude (z) and velocity information can be encoded into a GPS NMEA statement, where as the Euler angles can be encoded into the Compass NMEA statement. Thus all the data that Flight Gear needs is available in these two NMEA lines. These lines are output to the embedded systems as Compass and GPS input, and at the same time also output to the Simulink model. An S-function written using

C code was used to encode seven separate floating point values into one string consisting of the GPS and Compass NMEA sentence combined.

The Compass output is encoded as follows:

$PTNTHPR, <heading>, N or A, <pitch>, N or A, <roll>, N or A*<checksum>[CR] [LF]
$PTNTHPR, 305.4, N, -25.6, N, -12.3, N*34[CR] [LF]

The GPS output is encoded as follows:

$GPRMC, <time>, V, <latitude>, N or S, <longitude>, E or W, <airspeed>, <altitude>, <date>, <heading>, W*<checksum>[CR] [LF]
$GPRMC, 154232, A, 2425.612, N, 05210.515, E, 020.4, 100.4, 230394, 003.1, W*43[CR] [LF]

The code to encode the information is shown below:

```
char NEMA[127],temp1[127];
double a, b, c, d, e, f, g;

a=2548.5155+Lat[0];      //ADD REFERENCE LATTITUDE
b=5448.5125+Lon[0];      //ADD REFERENCE LONGITUDE
c = Vel[0];
d = Alt[0];
e = Yaw[0];
f = Pit[0];
g = Rol[0];

//Copy formated lat, lon and heading

sprintf(temp1,"%0.4f,N,0%0.4f,E,0%0.1f,A,141202,D,0%0.1f,W*FI\n\r
        $PTNTHPR,%0.1f,J,%0.1f,K,%0.1f,L*OQ\n\r",a,b,c,d,e,f,g);

strcpy(NEMA, "$GPRMC,154232,V,");
strcat(NEMA, temp1);

strcpy(y0,NEMA);
y1[0] = strlen(NEMA);
```

For the purpose of the simulation, the GPS coordinates are referenced to a fixed base of 25 degrees 48.5155 minutes latitude and 54 degrees 48.5125 minutes longitude, which is approximately the position of the United Arab Emirates. Figure 7-3 shows the Simulink Model that acquires data from the Serial port, parses them and then sends the data to Flight Gear Flight Simulator program.

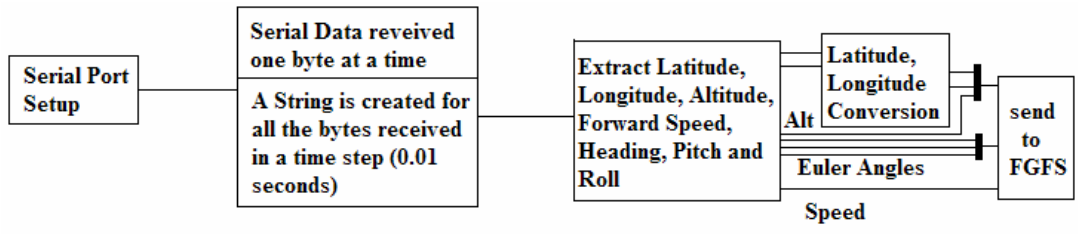


Figure 7-3: Simulink Model of Serial to Flight Gear Data Management System

7.1.3 Stages in HILS Design

In order to ensure that the HILS works just as smoothly as the "Simulation Only" part, the transition from pure simulation to hardware integrated simulation has been done in stages. The HILS Setup along with the dSpace-Simulink model that has been shown in Figures 7-1 and 7-2 describes the complete HILS setup. However the entire setup cannot be made to work if constructed at once, since there are too many variables that can lead to error. Thus the HILS was setup in stages

In the first stage, the overall goal was to ensure that the Trajectory Tracker algorithm worked. Therefore the Trajectory Tracker blocks were removed from the "Simulation Only" model, and in its place were put two dSpace digital input blocks. These blocks represented the "Turn/No turn" signal, and the "CW turn/CCW turn" signal respectively. The two signals were generated by the embedded control which now ran the Trajectory Tracker algorithm. Figure 7-4 below shows the software setup, while Figure 7-5 shows the Hardware setup. As can be seen the autopilots remained in the model, and therefore the controls for the elevator, throttle and aileron was not subject to discrete control.

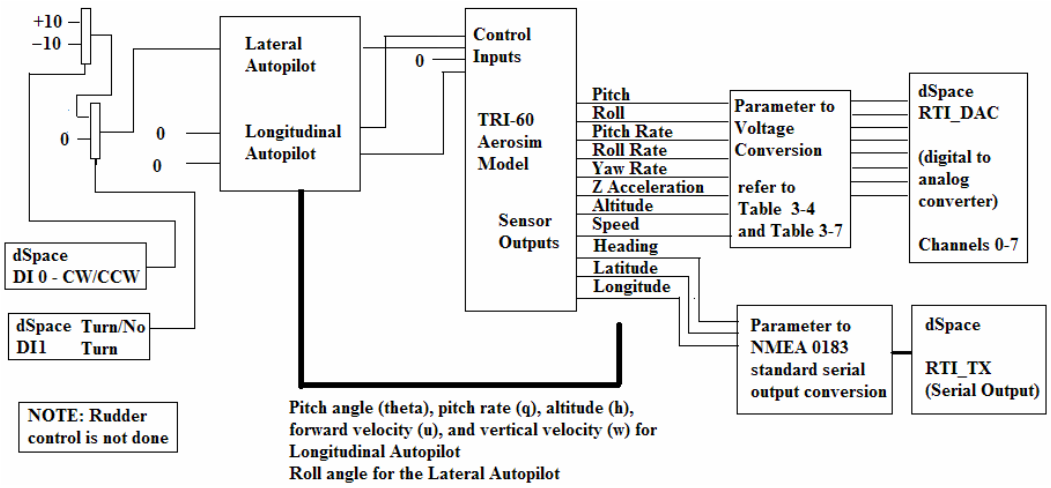


Figure 7-4: First stage of HILS setup – Simulink Model

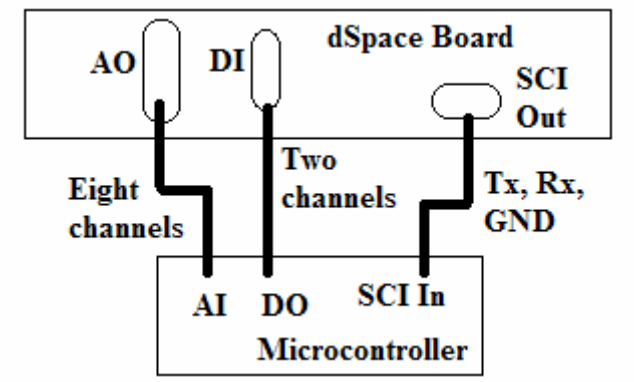


Figure 7-5: First stage of HILS setup – Hardware Setup

In the first stage only 2D waypoint navigation was done, so the altitude remained constant through out the simulation. The Flight Gear Flight Simulator was not integrated during the first stage of the HILS setup; therefore the user had no visual of the aircraft during flight.

In the second stage, the lateral and longitudinal autopilots were shifted to the hardware (microcontroller), and the Flight Gear flight simulator was integrated. Thus the second stage tested both the Trajectory Tracker algorithms and the autopilot from the hardware. Thus the TRI-60 model of the aircraft was now subject to discrete control at only 15Hz which is the speed at which the hardware can execute a control loop consisting of analog data acquisition, serial data acquisition, Trajectory Tracker algorithm, Autopilot algorithm and the serial data output. The Simulink model downloaded into dSpace now looks like the one shown in Figure 7-1, where as the Simulink model that captures serial data and transfers them to FGFS looks like the one shown in Figure 7-3. The hardware setup is for stage two is shown below in Figure 7-6.

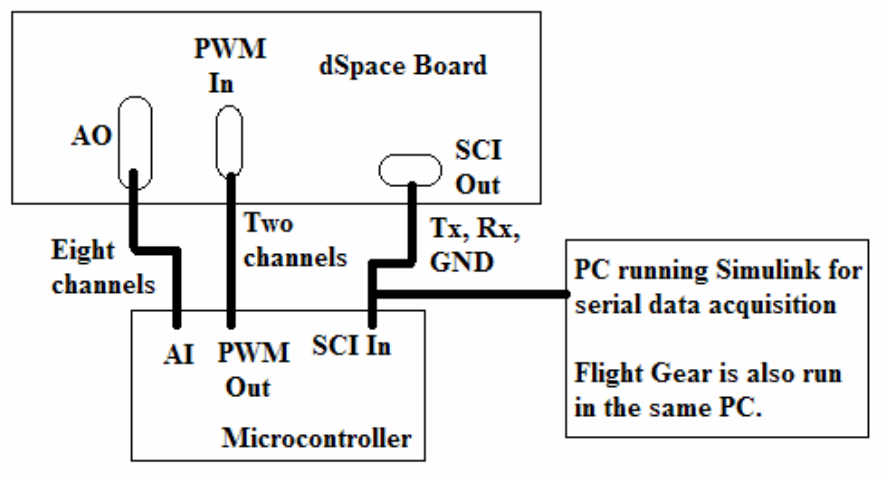


Figure 7-6: Second stage of HILS setup – Hardware Setup

In the second stage waypoint navigation was tested in both 2D and 3D for both PID and LQR state feedback lateral autopilots. Since the FGFS software was integrated, the user could also see the motion of the aircraft during flight. The motion of the aircraft is not as smooth at "Simulation Only" due to the fact that the controls are discrete. In this

In the third stage, the microcontrollers were split into two. The first microcontroller implementing the control loop with analog data acquisition, serial compass data acquisition, the Trajectory Tracker, the Longitudinal and Lateral Autopilots, and the serial data output to Ground Station. The second microcontroller is used mainly for serial GPS data acquisition and dead reckoning. The two microcontrollers communicate via the CAN bus. This is the final stage since all the components necessary for the HILS setup is implemented. In this stage the Heuristic and the bang-bang Trajectory control is tested with both the PID and LQR State feedback Lateral Autopilots. Waypoint navigation is done in both 2D and 3D. The dSpace-Simulink setup will look like the one shown in Figure 7-1, where as the complete hardware setup will look like the one shown in Figure 7-2, and the Simulink Serial setup to transfer data to FGFS will look like the one shown in Figure 7-3. Figure 7-7 below shows the electrical wiring diagram for reference, so that the HILS can be setup in future by another researcher. Figure 7-8 shows the picture of the HILS Setup

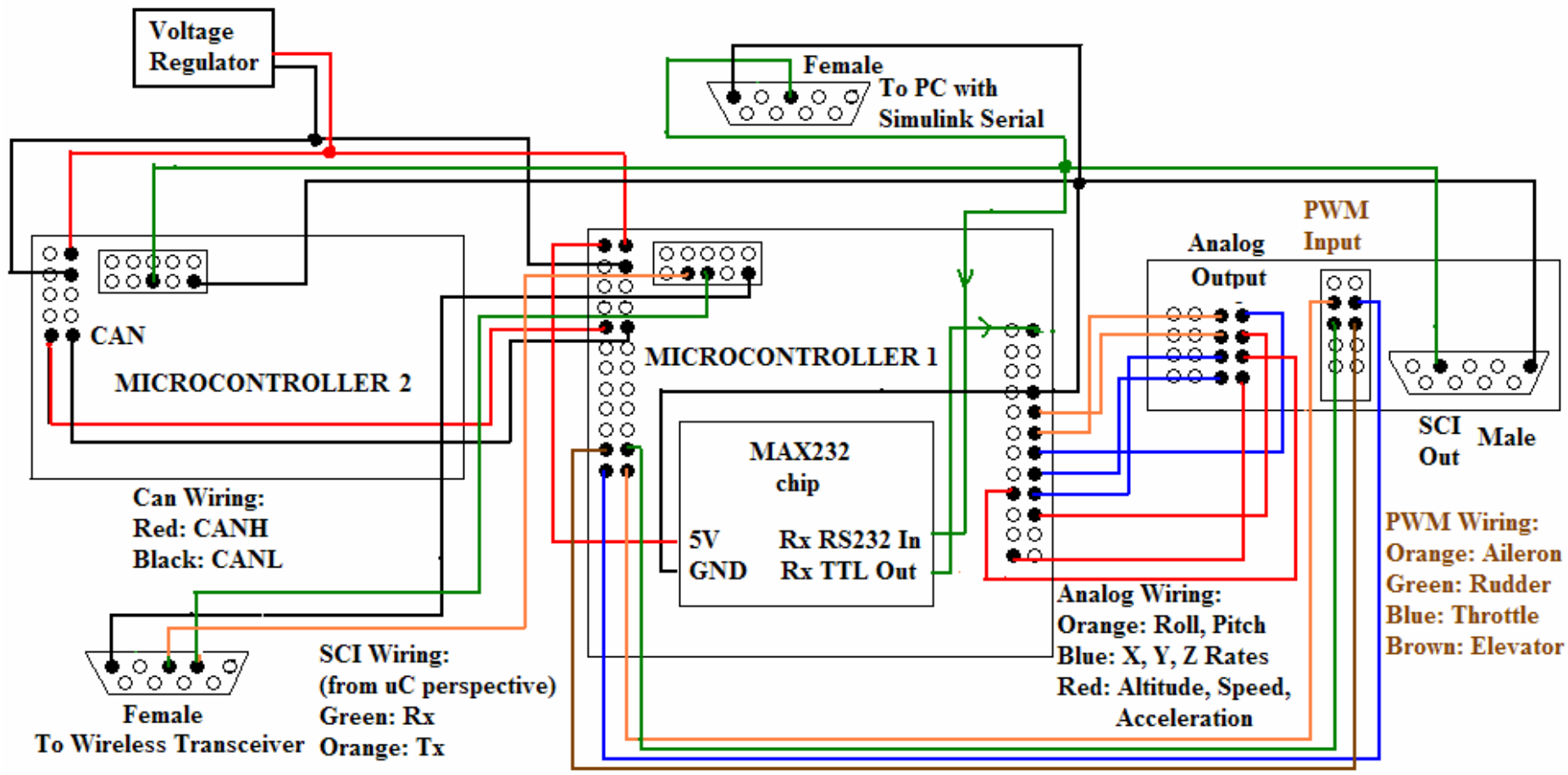


Figure 7-7: Complete electrical circuit diagram of the HILS Setup

NOTE: The above figure shows how to connect two S12 Compact microcontrollers to the dSpace Board for the HILS Setup.
NOTE: Please refer to Appendix to check Pin Numbers and Names for S12 Compact Board, dSpace Board, and Male/Female DB9 connectors

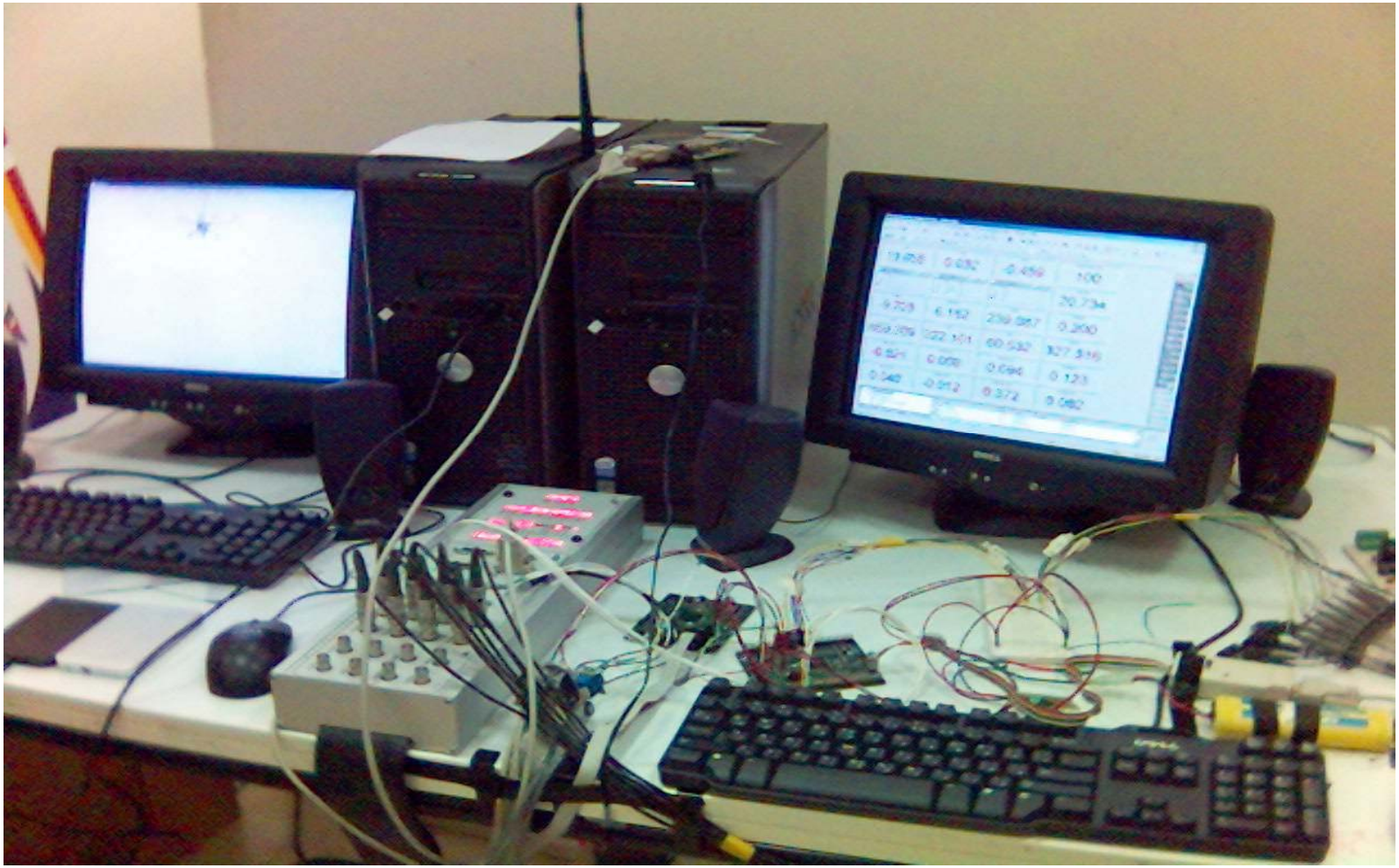


Figure 7-8: Photo of the HILS Setup

CHAPTER 8: RESULTS AND ANALYSIS

This chapter will provide in depth analysis of the "Simulation Only" and HILS results obtained. All aspects of the flight will be considered for discussion such as the control inputs, the performance of the lateral and longitudinal state variables, and most importantly the nature of the path followed during the flight. The objective is to reach the desired waypoints, and the fact that the UAV has passed through each waypoint is not open for discussion because it is a non-negotiable requirement. However discussion of the performance criteria such as aircraft stability and the smoothness of the path followed is valid, and can even be compared to previous research carried out in this area. First to reiterate, the following have been accomplished:

- Software simulation and HILS of waypoint navigation system in 3D utilizing bang-bang control methodology using a PID controlled lateral autopilot
- Software simulation and HILS of waypoint navigation system in 3D utilizing modified bang-bang control methodology using a heuristic PID controlled lateral autopilot
- Software simulation and HILS of waypoint navigation system in 3D utilizing enhanced bang-bang control methodology using heuristic LQR state feedback lateral autopilot

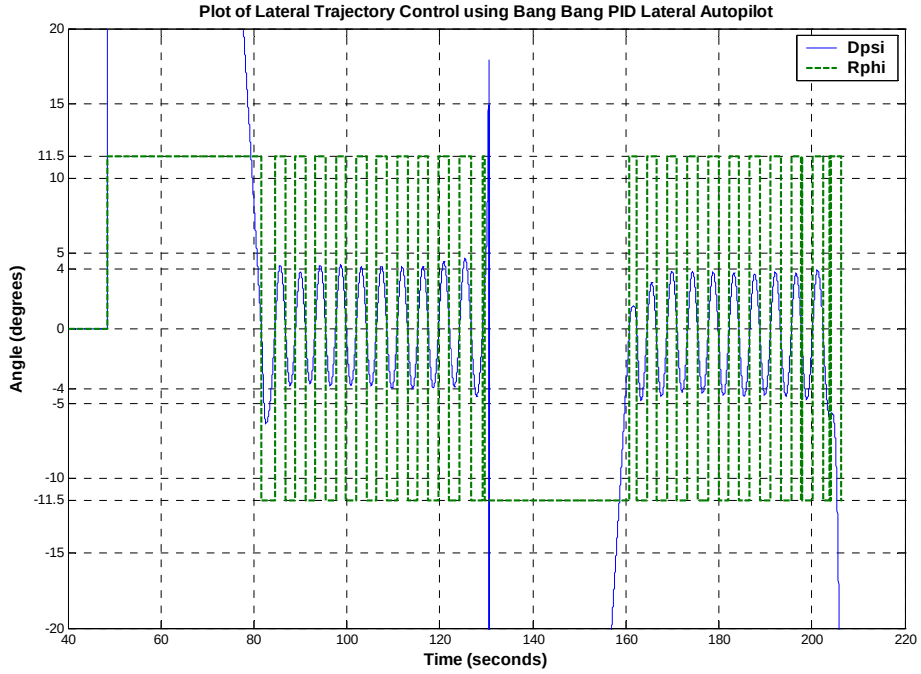
8.1 "Simulation Only" Waypoint Navigation Results and Analysis

The "Simulation Only" Results will be discussed prior to the discussion of the HILS results. Analysis of the "Simulation Only" results will include analysis of the path taken by the aircraft, and the control inputs during each maneuver and the resulting state of the aircraft. Analysis of the HILS results will deal more with timing analysis and the overall degradation of the aircraft performance due to discrete control inputs.

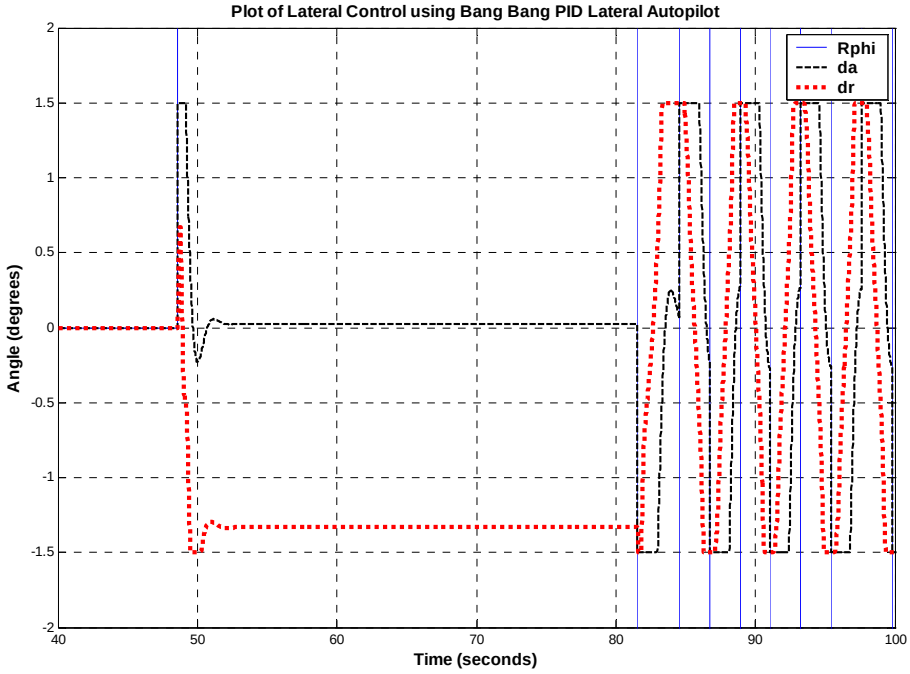
8.1.1 Bang-Bang control using PID Lateral Autopilot

Bang-Bang approach is a pure on-off control of the lateral autopilot. The aircraft either rolls by 10 degrees or -10 degrees whenever it needs to turn, or flies straight when it does not need to turn. The decision to turn is made when the aircraft has any heading error more than a certain tolerance level. For this thesis the tolerance

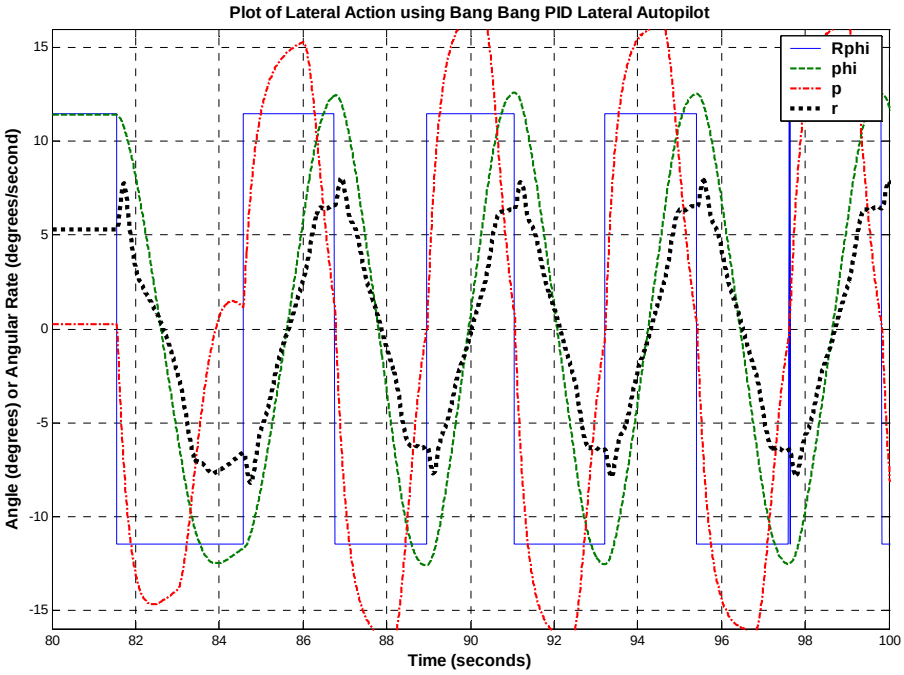
is set to zero, but better performance will be achieved if the tolerance is increased to a few degrees. The overall disadvantage of this method is chattering, because the aircraft keeps oscillating about the zero heading error point and executing what could be termed as a Dutch roll maneuver. The PID control also suffers from a lack of damping of its oscillation about the zero heading error line. The reason for the chattering is due to the fact that the aircraft cannot instantaneously execute a reference control command, causing it to overshoots its target heading. Thus the aircraft is always subject to continuous but maximum and opposite reference command on after the other. Figures 8-1 (a – c) shows the result of the Bang-Bang PID control of the Lateral Autopilot. Bang-Bang PID control leads to an overall oscillation of +/- 4 degrees about the reference heading ψ as can be seen from Figure 8-1a. On the other hand PID control results in very less steady state error (less than 0.5%) tracking the commanded bank angle ϕ as can be seen in Figure 8-1c. The response is a second order response with an overshoot of 3.7% and a time constant of 1.55 seconds, and steady state is achieved much faster than the LQR state feedback method. However due to this the control inputs during the bang-bang portion of the flight are often saturated as shown in Figure 8-1b. Control inputs have been saturated to prevent destabilization of the aircraft.



(a) Lateral Trajectory Control



(b) Lateral Control

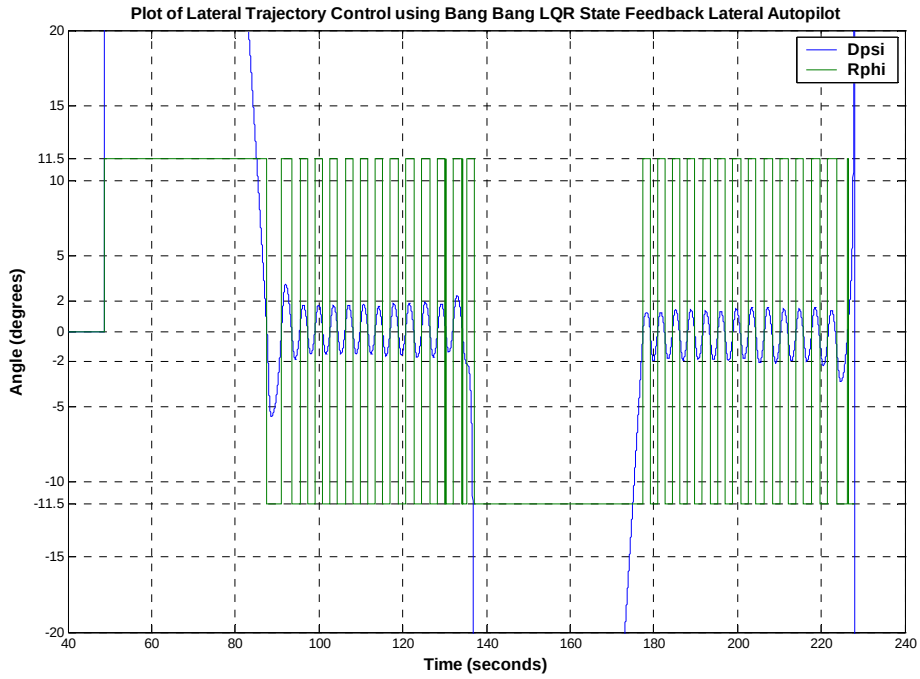


(c) Lateral Response

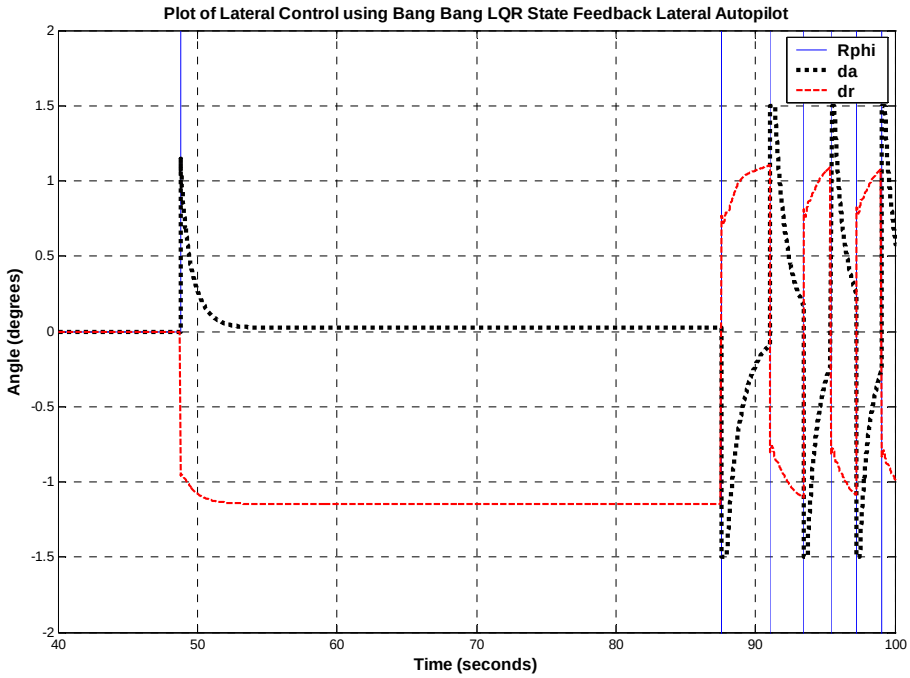
Figure 8-1 (a – c): Bang-Bang PID Lateral Autopilot Control Response

8.1.2 Bang-Bang Control using LQR State Feedback Lateral Autopilot

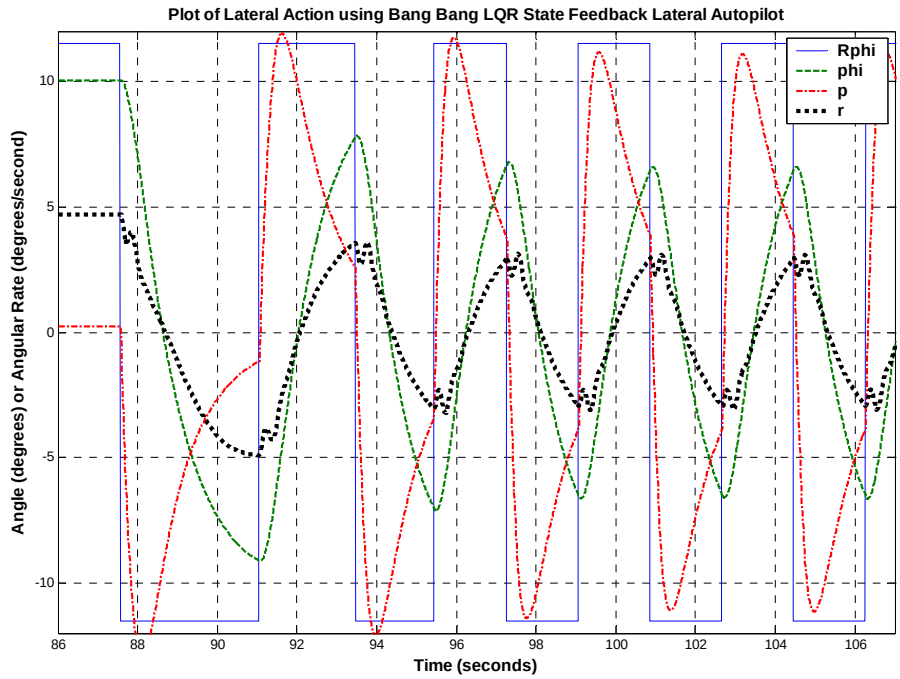
One advantage of state feedback control is the ability to control all the states of a system, unlike PID control, which only controls the response of the output. The lateral state of a system consists of the roll angle, the roll rate, the yaw rate, and the body frame side velocity. When lateral action is absent, such as during the time the aircraft flies straight, the regulatory effects of the state feedback keeps removes the roll and yaw rate, and the side velocity and keeps the roll angle at zero. During a turn, the above states are minimized as best as possible, thus executing a coordinated turn. Overall this regulatory effect ultimately has a damping effect on the control output resulting in more relaxed and smoother turn. Unfortunately this results in large (13.64%) steady state error while tracking the roll angle, which causes the aircraft to take a wider turn at a given speed. The bang-bang effects of LQR state feedback is less pronounced due to this damping effect, and thus the overall oscillation about the reference heading is only +/- 2 degrees as seen in Figure 8-2a. Despite the rapid switching of the reference command, the lateral control does not saturate as seen in Figure 8-2b. The tracking response of the roll angle is first order response with time constant of 1.8 seconds, and reaches steady state in 5 sec.



(a) Lateral Trajectory Control



(b) Lateral Control

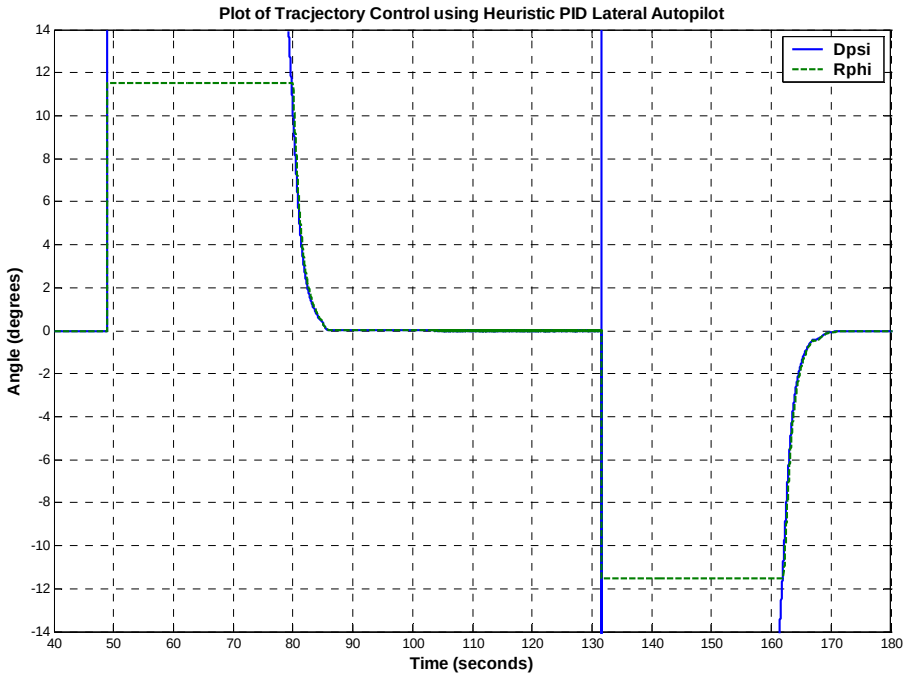


(c) Lateral Response

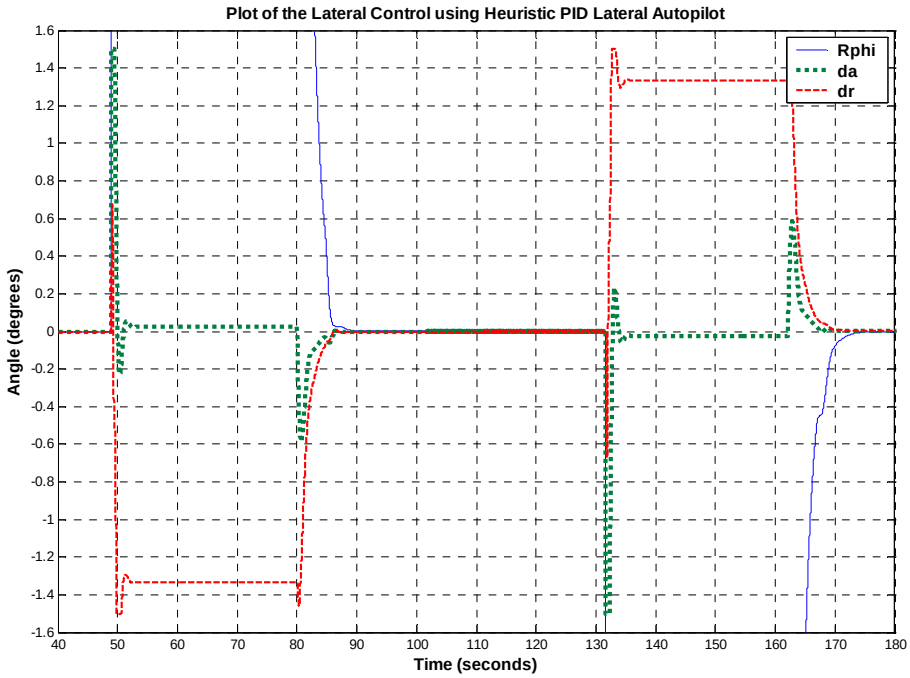
Figure 8-2 (a – c): Bang-Bang LQR State Feedback Lateral Autopilot Control Response

8.1.3 Heuristic Control using PID Lateral Autopilot

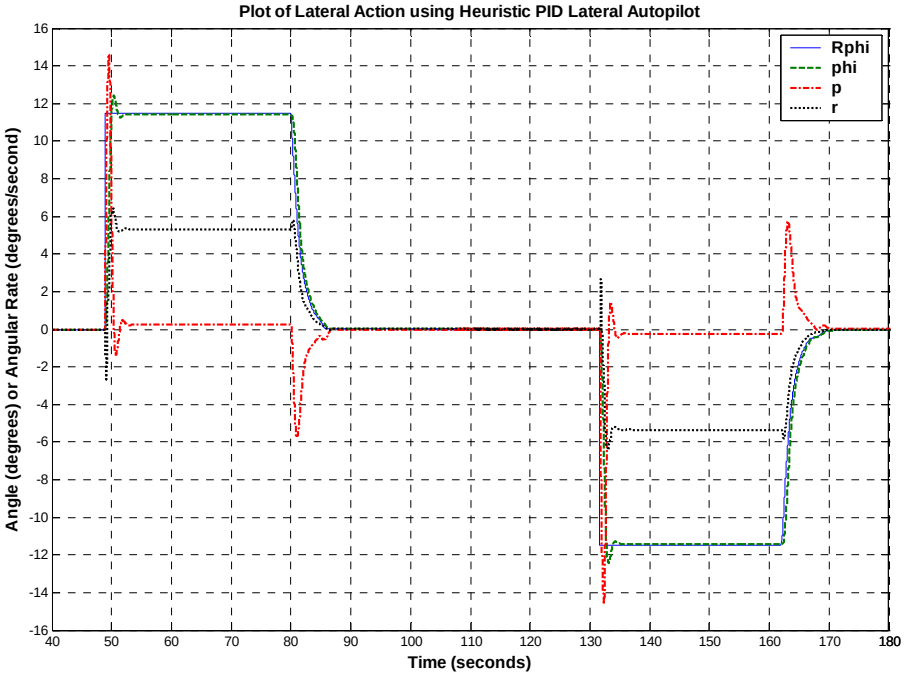
Heuristic Trajectory control, unlike Bang-Bang Trajectory control results in a smoother exit from a turn. Heuristic control can also perfectly track a reference heading as shown in Figure 8-3a. Unlike bang-bang control which emphasizes full control input of +/-10 degree of roll angle ϕ to correct the slightest heading error $d\psi$; heuristic control adjusts the roll of the aircraft according to the magnitude of the heading error using a PID approach, saturating at 10 degrees. PID controlled Lateral Autopilot shows better performance than the LQR state feedback controlled Lateral Autopilot due the very reactive nature that proved disastrous during bang-bang control. From the previous section, it is determined that a PID controlled Lateral Autopilot can track a steady reference roll with almost zero steady state error, and also shows very fast response time. Similarly from Figure 8-3c, it can be seen that a PID controlled Lateral Autopilot can exactly track a changing reference roll command without any time-lag. This allows the aircraft to straighten by the time the heading error is zero as shown in Figure 8-3a. The lateral control inputs are also more relaxed due to the lack of bang-bang control, except for entry into a turn when the autopilot is commanded a full roll of 10 degrees.



(a) Lateral Trajectory Control



(b) Lateral Control

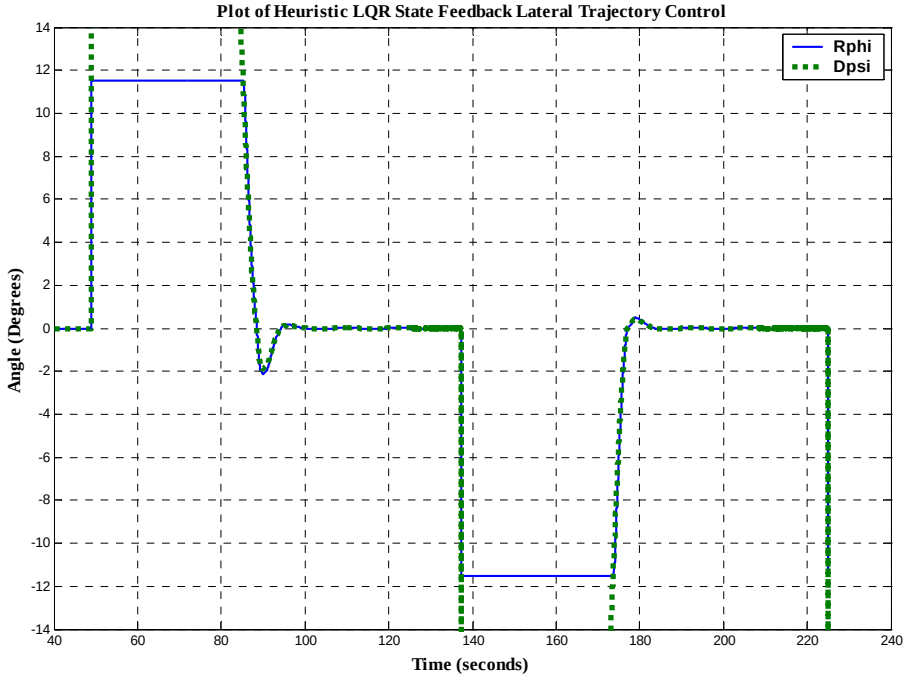


(c) Lateral Response

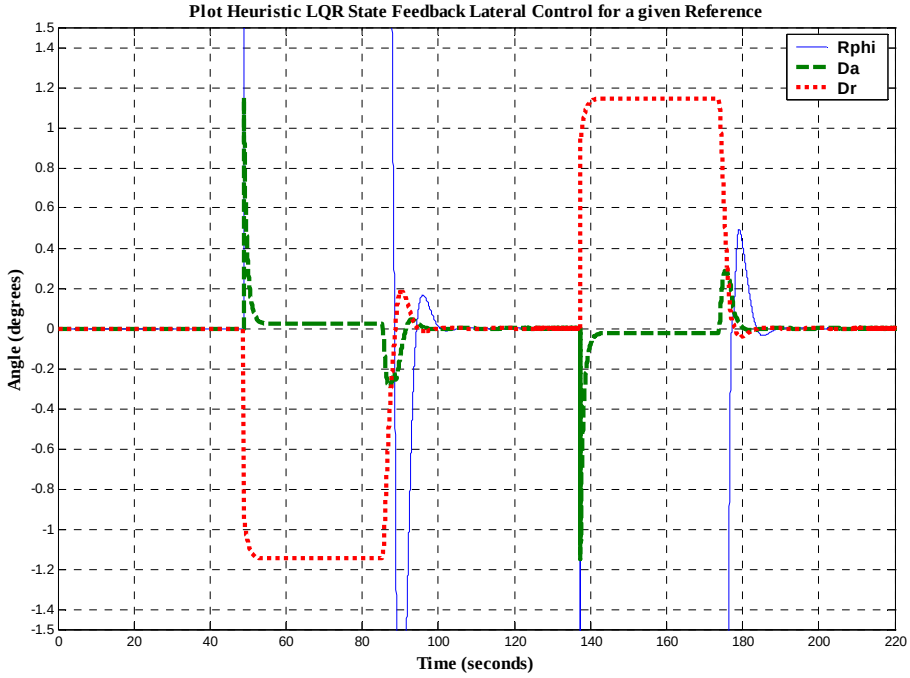
Figure 8-3 (a – c): Heuristic PID Lateral Autopilot Control Response

8.1.4 Heuristic Control using LQR State Feedback Lateral Autopilot

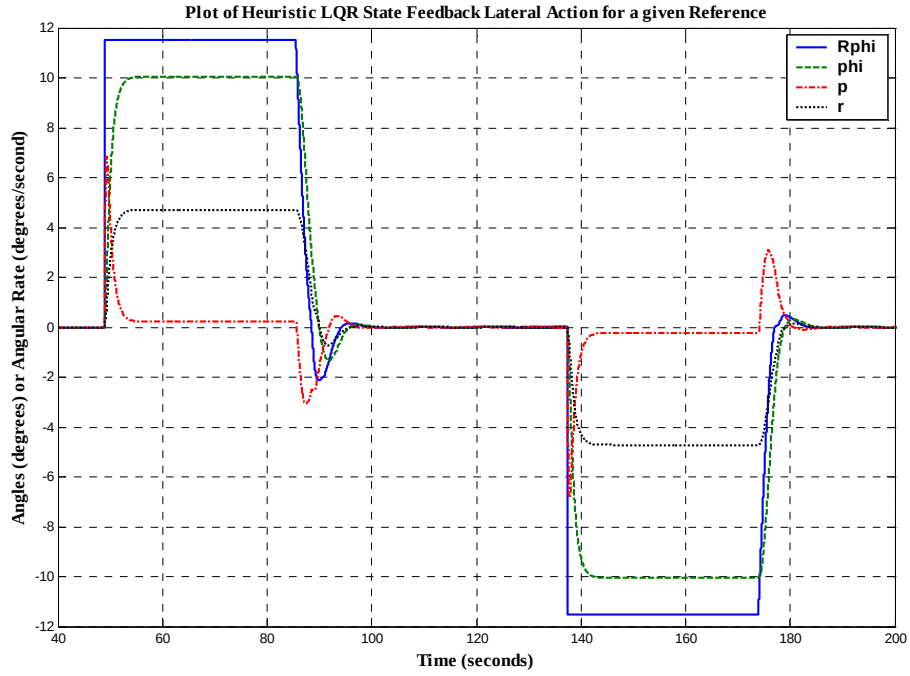
Heuristic Trajectory Controlled using LQR state feedback Lateral Autopilot results in a smoother flight than its bang-bang counterpart. However as can be seen from Figure 8-4a, the Trajectory Control is not as efficient as its PID controlled counterpart. LQR state feedback controlled Lateral Autopilot cannot track a steady reference roll angle without showing large steady state error, and a slow response time. This causes the Lateral Autopilot response to always lag behind the command lateral roll angle input as shown in Figure 8-4c. Thus the aircraft overshoots the desired heading before it can fully correct itself as shown in Figure 8-4a. However careful inspection if Figure 8-4a shows that there are small perturbations around the desired heading, thus the lateral action does not reach steady state. Also the aircraft takes 15 seconds to fully exit a turn, where as the Heuristic PID controlled Lateral Autopilot took 5 seconds to exit a turn. However despite all the disadvantages listed, the LQR state feedback Lateral Autopilot is a better choice than the PID alternative since it does not force the aircraft to change its state as rapidly, and thus the aircraft experience smoother motion. However, this method requires control input a 100 Hz or better to operate efficiently.



(a) Lateral Trajectory Control



(b) Lateral Control

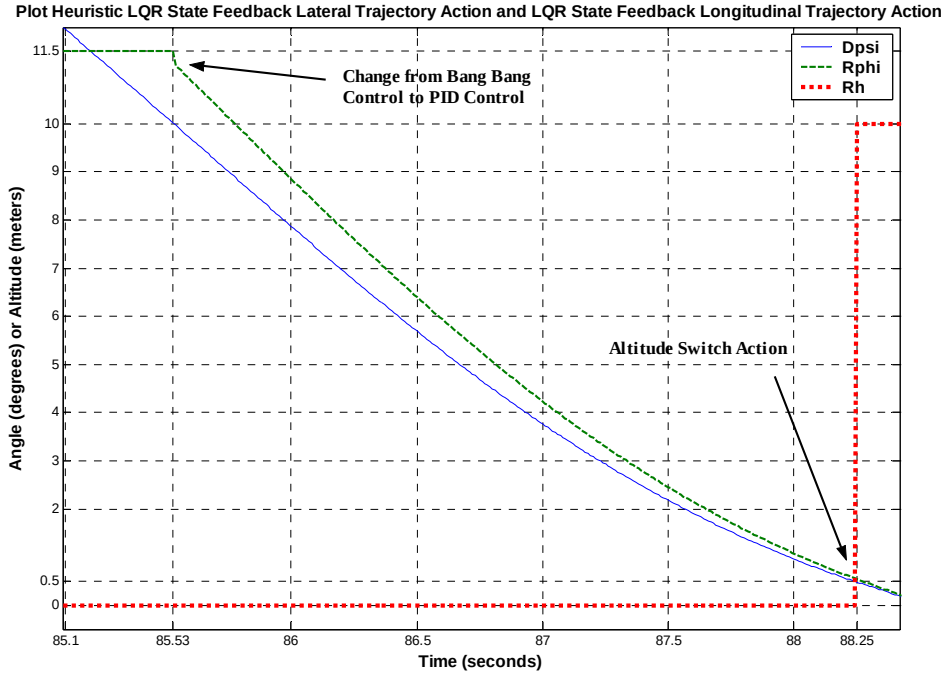


(c) Lateral Response

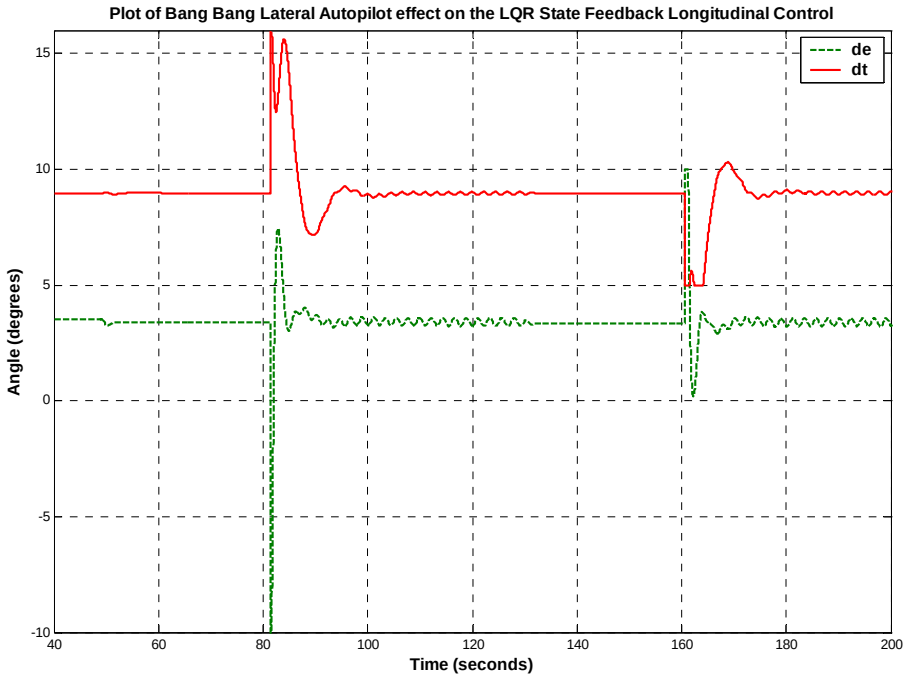
Figure 8-4 (a – c): Heuristic LQR State Feedback Lateral Autopilot Control Response

8.1.5 LQR State feedback Longitudinal Autopilot

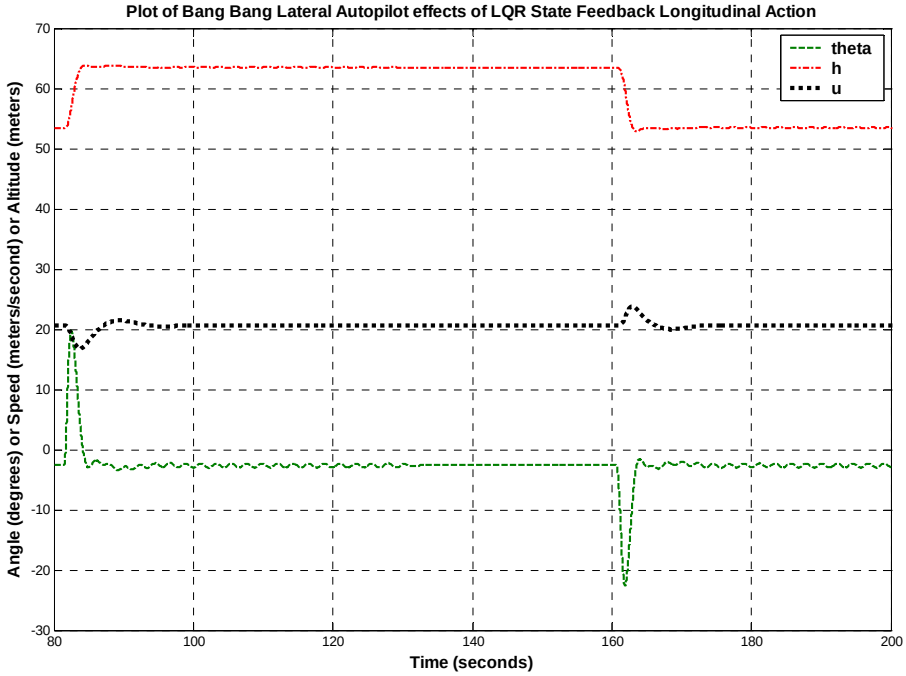
The Longitudinal Autopilot is created using a LQR State feedback in [1]. The longitudinal autopilot is used by the Trajectory Tracker to control the altitude of the aircraft. The Trajectory Tracker only tracks the altitude when the aircraft is flying straight after exiting a turn. Therefore from Figure 8-5a it can be seen, the altitude switch command is given when the heading error is less than 0.5 degrees. The Longitudinal autopilot has a steady state error of 3.5 meters. The ascending motion has a first order response, while the descending motion has a second order response. The forward velocity of the aircraft is regulated at 20.5 meters per seconds, unless the aircraft is ascending or descending, in which cases the forward speed changes by maximum of 4 meters per second from the steady state value. The action of the longitudinal autopilot has been discussed extensively in Chapter 4. Figure 8-5b and Figure 8-5c shows the longitudinal control and response to an altitude switch command of +/- 10 meters. However, notice the slight ripples in both the control input and the longitudinal response. These ripples are side effect of the bang-bang control action using PID controlled Lateral Autopilot. The extensive Dutch roll effect affects the longitudinal motion of the aircraft.



(a) Longitudinal Trajectory Control



(b) Longitudinal Control

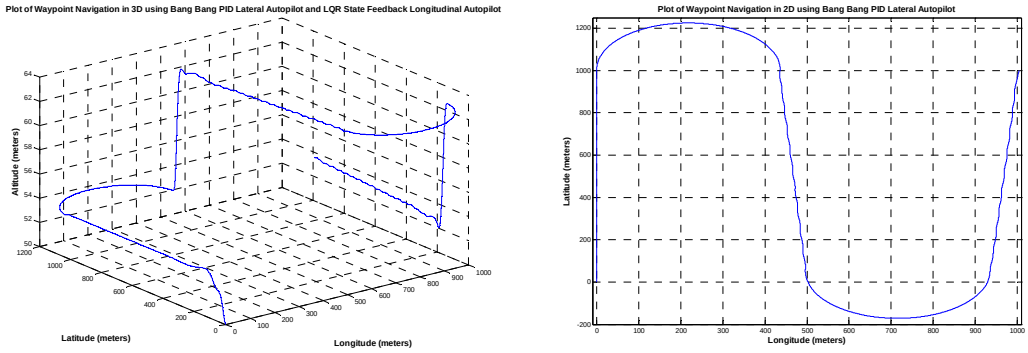


(c) Longitudinal Response

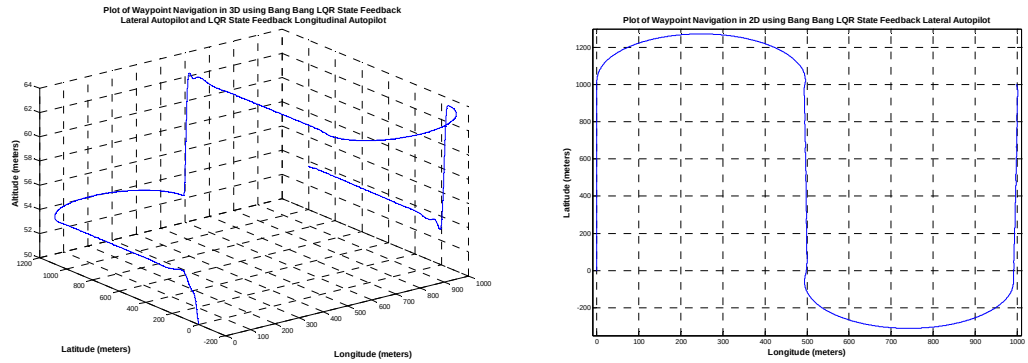
Figure 8-5 (a – c): LQR State Feedback Longitudinal Response to Dutch Roll Effect

8.1.6 Summary of "Simulation Only" Waypoint Navigation

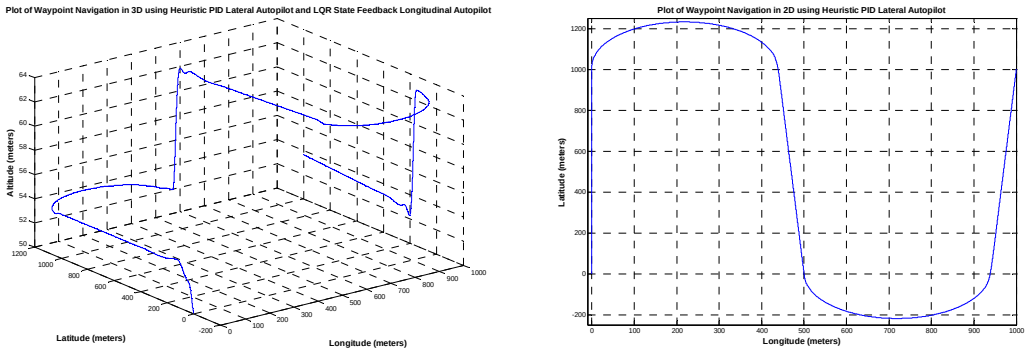
Figure 8-6 (a-d) shows the plots of waypoint navigation using the four "simulation only" methods described so far. The best results were obtained using the Heuristic Trajectory Tracking approach. In both cases, the aircraft reach the desired waypoint "dead on target". The reader must be given to understand that this is only a simulation result where continuous updates of all the states necessary were available such as position, heading, altitude, and all the other states necessary for the autopilots. These parameters were available continuously, and were very precise, and they did not suffer from noise associated with sensing devices. Thus the results achieved can be described as ideal. The shortest path was taken by the Heuristic Trajectory Tracker with a PID controlled Lateral Autopilot, which completed the mission in 206 seconds. The Bang-Bang Trajectory Tracker with the LQR State Feedback Lateral Autopilot took the longest to complete the mission taking 228 seconds. Overall the PID controlled Lateral Autopilot performed better because the plane was able to bank more for a given bank reference command, and thus take sharper and shorter turns.



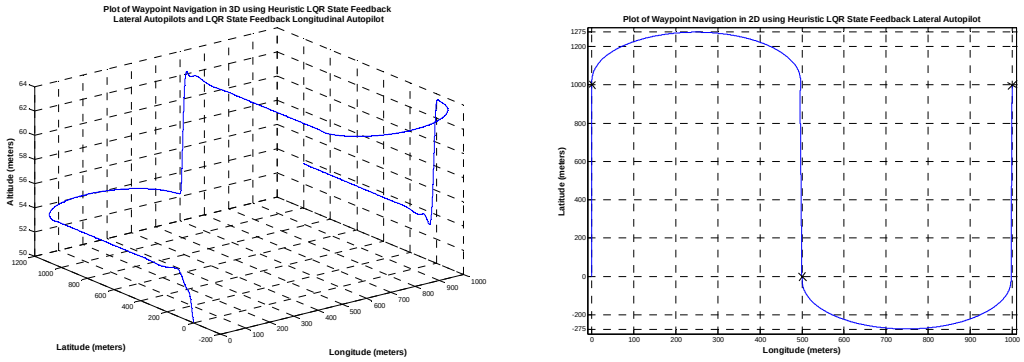
(a) Bang-Bang Trajectory and PID Lateral Autopilot



(b) Bang-Bang Trajectory and LQR State Feedback Lateral Autopilot



(c) Heuristic Trajectory and PID Lateral Autopilot



(d) Heuristic Trajectory and LQR State Feedback Lateral Autopilot

Figure 8-6 (a-d): Plots of 3D and 2D Waypoint Navigation using Various Approaches

The next section will describe the Hardware in the Loop Simulation results, which will be more down to earth and practical. Control and analog sensor updates will only occur at a rate of 15 Hz, where as GPS sensor updates will occurs at 1 Hz. The sensors are not as accurate and has limit due to the ATD as to the smallest change in parameter that can be sensed. For example, the compass is accurate only to 0.5 Hz. However the analog sensors also have associate noise, which will mask small changes in the parameter from the controller. Therefore the results obtained from the HILS will be more relevant during the actual flight test.

8.2 HILS Waypoint Navigation Results and Analysis

The following figure show the HIL simulation result. Figure 8-7 shows the result for the first stage, where only the trajectory tracker algorithm in 2D is being tested. The result shows very smooth lines, and perfect turns, which is no different from the result of "pure simulation". This proves that the Trajectory Tracker

algorithm is not affected by discretization, because the decision to turn clockwise, counterclockwise, or not turn changes only around waypoints, and when the UAV exits the circle. The rest of the time the decision remains the same. So a very low frequency control loop can work just as well as a control loop that is continuous.

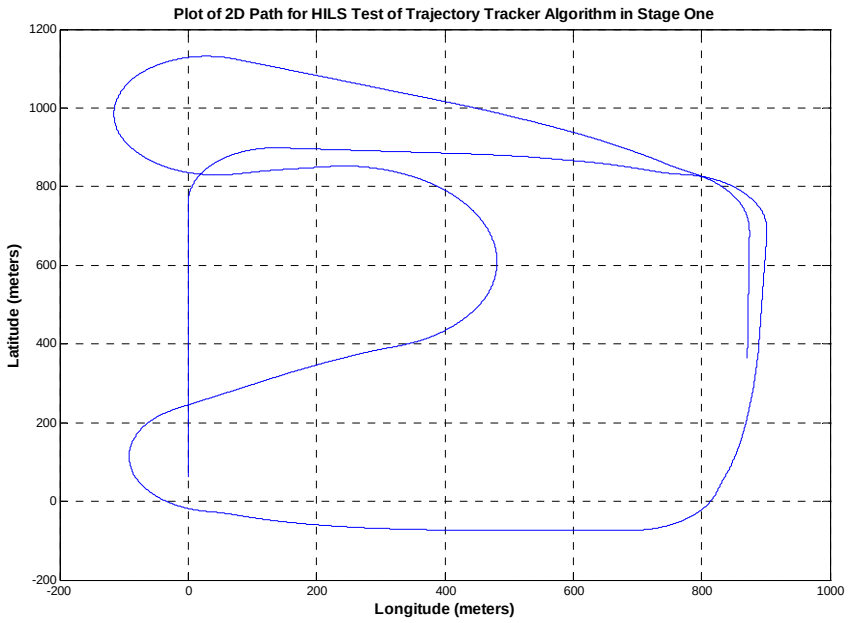


Figure 8-7: Result of HILS Stage One Run Testing Trajectory Tracker algorithm in 2D

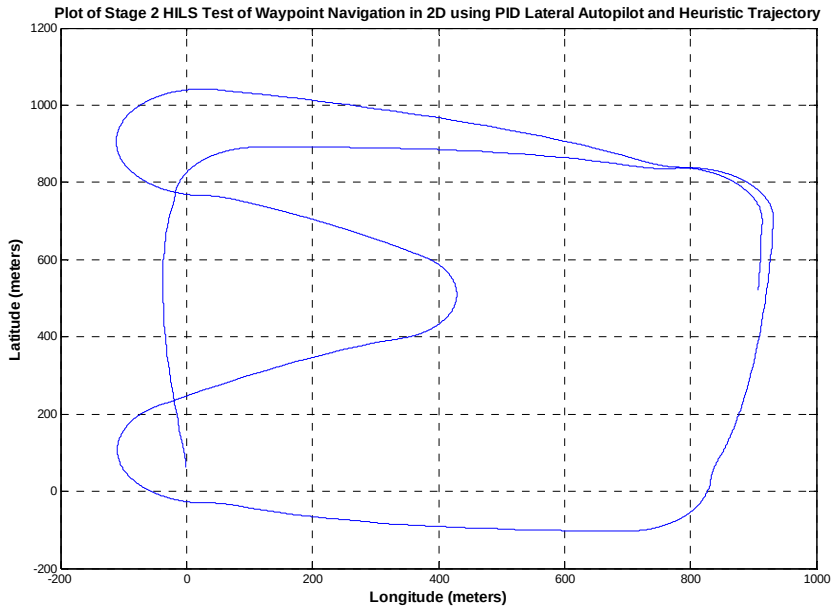
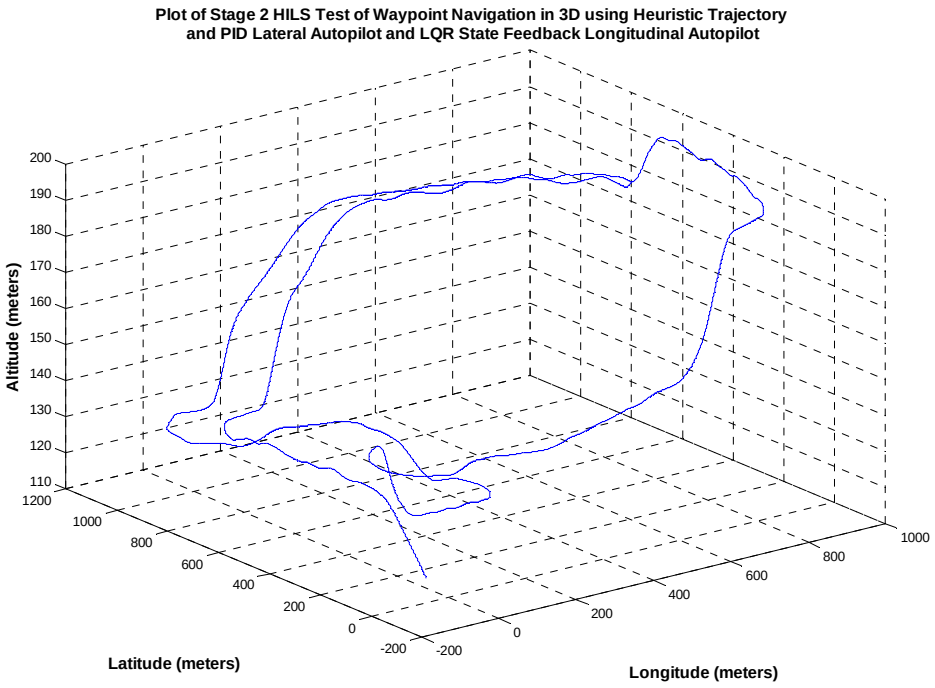
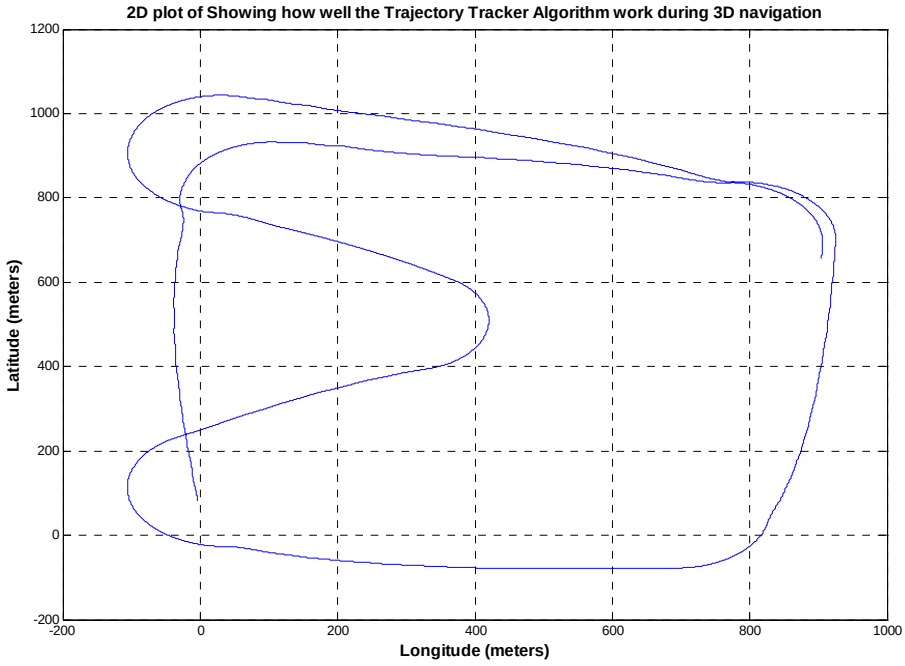


Figure 8-8: Plot of HILS Stage Two Run Testing Heuristic Trajectory, Autopilots in 2D

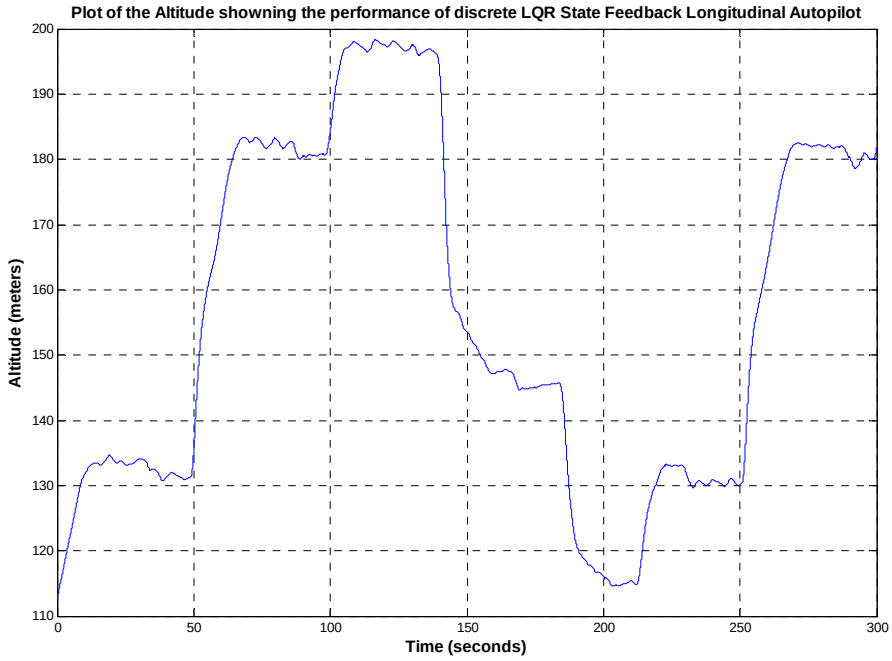
Figure 8-8 above shows the result of adding the autopilots into the hardware (i.e. discretizing the autopilots. The path is still smooth, although they are not straight as the stage one result. This is possible because the longitudinal motion of the TRI-60 RC aircraft has a short period frequency of 3Hz, while the longitudinal control is applied at 15 Hz. Therefore the controller performs as well as the continuous version. Figures 8-9 (a-c) shows the Stage two test of the Trajectory Tracker Algorithm using heuristic trajectory control and utilizing the PID controlled lateral autopilot and LQR state feedback controlled longitudinal autopilot



(a) 3D plot



(b) 2D Plot



(c) Altitude Plot

Figure 8-9: Plot of Stage Two Run Testing Heuristic Trajectory, Autopilots in 3D

Figure 8-9a and Figure 8-9c shows that the LQR state feedback longitudinal autopilot oscillates around the reference altitude if it is discretized. This is the first difference

that appears in the HILS simulation from that of pure simulation. These oscillations can be minimized but not removed completely, and is one of the shortcomings of discrete control.

CHAPTER 9: CONCLUSION AND FUTURE WORK

The current system has limitations in that the Avionics Unit is not portable to different aircraft models. It takes too much space, and therefore it can be used only in the TRI-60 RC aircraft model. Another limitation is the low sampling rate of the sensors, and a low control frequency due to lack in processing speed of the embedded system used. With a control rate of 15 Hz, the longitudinal state of the UAV can easily be controlled since the short period mode is only 3 Hz at 20 m/s. However, the lateral state cannot be easily controlled because the roll mode has a frequency of 18 Hz, which is more than the current control rate. To control the lateral motion of the aircraft smoothly, a control frequency of 90 Hz is required. This is not possible using the current microcontroller. With proper interrupt design, the control frequency can be bumped up to 35 Hz, but that is all. In future either a more powerful microcontroller may need to be used, or there should be a separate data acquisition microcontroller whose purpose is only to acquire data by sampling ATD channel and reading the serial port. The control microcontroller must be separate. It is due to the uncontrollability of the LQR state feedback Lateral Autopilot at 15 Hz, that the PID controlled Lateral Autopilot will be implemented in the Hardware.

9.1 Real-Time Timing Analysis in Hardware and Cost Analysis

The real time control loop could be executed at 16 Hz frequency, which although was sufficient to control the longitudinal motion of the aircraft was not able to control the lateral motion of the aircraft using state feedback method. The following gives the timing analysis for the each section of the code execution:

- ATD clock rate: 0.5 MHz
- ATD individual channel sample rate: 0.5 MHz/32 = 15.625 kHz
- ATD individual sensor sample rate = 15.625 kHz/8 = 1.953 kHz
- ATD sampling interval with 12 sensors = 1.953 kHz/12 = 162.76 Hz
- Therefore it takes 6.14 ms to sample all the analog sensors
- SCI input time: 850 bits / 115200 bits per second = 7.38 ms
- SCI output time: 550 bits / 115200 bits per second = 4.77 ms
- Total Input/Output time: 18.3 ms (i.e. I/O rate by itself is 54.7 Hz)
- Interrupt rate (RTI + CAN): 22.888 Hz
- Each interrupt take 1 ms to process, therefore total interrupt time: 22.888 ms

- Interrupt and I/O Time: 41.2 ms (i.e. Interrupt and I/O rate = 24.27 Hz)
- Therefore the Trajectory Tracker, Longitudinal Autopilot, Lateral Autopilot, and servo actuation code time: 21.3 ms
- Total Control Loop time: 62.5 ms (i.e. Control loop rate of ~ 16 Hz).

The cost of the overall thesis can be broken down into the following hardware and software platforms:

- 2 x PCs (One to run the Flight Gear Flight Simulation and one to interface with dSpace)
- 1 x Laptop (Ground Station – part of the hardware)
- 1 x dSpace 1104 Controller Board (DSP processor runs TRI-60 real time model)
- 2 x S12COMPACT Microcontroller (the hardware being tested)
- 2 x XTREME RF Transceivers (part of the hardware)
- 1 x NIDAQ Board and PCMCIA card (for calibration)
- A hardware debugger BDM (Kevin Ross) model
- Visual Development Studio 6.0
- Matlab 7.1 and Simulink
- Image Craft Embedded C development platform
- NOICE serial downloader program
- Flight Gear Flight Simulator program and Aerosim Simulink Library
- Probes, wiring, DB9 cables

9.2 Summary of Contributions

A full scale Hardware in the Loop Simulator has been design to test Waypoint Navigation in three dimensions. The Hardware in the Loop Simulator Setup is design to simulate the effects of the UAV once it is released from manual control. The Trajectory Tracker algorithm has been created for 3D navigation using an extension of the Feedback Guidance algorithm in 3D. A heuristic PID approach has been employed to control heading. The altitude is controlled immediately after the UAV exits a turn if the next waypoint is far way from the current waypoint. If the next waypoint is near to the current waypoint, then the altitude is controlled as soon as the UAV enters into a turn. Furthermore, the mathematics of 3D navigation has been described in Chapter 3. Although not tested, two alternative trajectory algorithms called the Bidirectional Feedback Guidance and k-Feedback Guidance has been

developed to control the heading of the UAV so that the UAV is already be facing the next waypoint by the time it reaches the current waypoint. Ground Station design and the redesign of the embedded system allow the UAV to have flexible mission capability whereas in the previous research, the waypoints had to be hard-coded into the UAV.

9.3 Future Work

The limitations of this thesis have already been described, and any future research should attempt to solve these limitations. More important thought, there is need to make a more accurate Inertial Measurement Unit (IMU), so that Inertial Navigation or Dead Reckoning can be performed. Current Avionics Unit configuration has a shoddy design that only allows the inertial sensors to be used for auto piloting and stability. Research must be conducted to create a AUS GPS/INS system that can be utilized in any future test bed. Another limitation that existed from previous research to this one is that the feedback gains that are linearized around 20 m/s forward speed and zero degree pitch. These gains work well if the changes from the default speed, roll and pitch do not deviate by much. Therefore a method of gain scheduling must be performed.

The trajectory algorithm must be changed from the instantaneous operation implemented in this thesis to one that tracks a predefined path. This requires histories to be kept of previous points, and also knowledge of future points, which will be used to control the track of the UAV. Another algorithm that needs to be implemented is the obstacle avoidance algorithm and the algorithm for take off and landing. Once these are done, the UAV can become truly “autonomous”. Future research should attempt to use a more powerful processor such as the PC-104 system so that more features can be added to the UAV functionality.