Title: , Autonomous Helicopter Navigation System, System Level, Progress Report

*“A Project”*

Prepared by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

,

Checked by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

,

Approved by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

, Student Manager 2010

Authorised for use by Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr Luis Mejias, Project Coordinator

**QUT Avionics**

Queensland University of Technology

CRCSS-EESE, GPO Box 2434

Gardens Point Campus

Brisbane, Australia, 4001.

Telephone (+61 7) 3864 1772

Facsimile (+61 7) 3864 1517

e-mail luis.mejias@qut.edu.au

web <http://code.google.com/p/ahns10/>

This document is Copyright 2010 by the QUT. The content of this document, except that information which is in the public domain, is the proprietary property of the QUT and shall not be disclosed or reproduced in part or in whole other than for the purpose for which it has been prepared without the express permission of the QUT

**Revision Record**

|  |  |  |  |
| --- | --- | --- | --- |
| Document Issue/Revision Status | **Description of Change** | **Date** | **Approved** |
|  | Initial Issue |  |  |

**Distribution List**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Affiliation** | **Distribution Date** | **Approved** |
| Avionics Lab File Archive | QUT Avionics |  |  |

**Foreword**

The Autonomous Helicopter Navigation System (AHNS) project contains a ground control station (GCS) subsystem and a flight control system. Both of these have been placed in the charge of the author. These subsystems can trace their roles back to three of the project high level objectives (HLOs). The requirements governing the design of the subsystems are contained eight are directly pertinent system requirements (SR’s). Within the context of the entire project the first major milestones for these systems are GCS Testing, Station Keeping Testing and Augmented Flight Testing.

To present the statues of these systems and the likeliness of their milestones being delivered against the methodology being followed was examined. The methodology at system inception, preliminary design and design and implementation was systems engineering with an emphasis on using lessons and results attained from previous projects and research. Using this it was concluded that the subsystems will be able to deliver against the significant project milestones.

With limited initial milestones the ability of the subsystems to track timelines was gauged by considering the progress in three of the initial work packages. The first work package completed was the selection of a flight computer. The second and third work packages, one of development of the GCS and the other the design of the control system are seen to be on track for completion by the end of June and their results are already evident.

Risks present within the control and GCS subsystems include technical and scheduling however few have yet yielded consequences. Consequences have occurred at the cross over of other subsystems and were seen to be caused by networking, hardware procurement and code reuse. To mitigate most scheduling and technical risks a greater focus on testing was proposed to ensure technical compliance and to focus the investment of time. A considerable degree of technical flexibility was also realised as being key to delivering against the milestones.

**Table of Contents**

Paragraph Page No.

[1 Introduction 7](#_Toc264811173)

[1.1 Scope 7](#_Toc264811174)

[1.2 Background 7](#_Toc264811175)

[2 Reference Documents 8](#_Toc264811176)

[2.1 QUT Avionics Documents 8](#_Toc264811177)

[2.2 Non-QUT Documents 8](#_Toc264811178)

[3 06332064 Project Summary 9](#_Toc264811179)

[3.1 Methodology for Delivering Against Milestones 10](#_Toc264811180)

[3.1.1 Project Inception Systems Engineering Methodology 10](#_Toc264811181)

[3.1.2 Preliminary Design 11](#_Toc264811182)

[3.1.3 Design and Implementation 12](#_Toc264811183)

[3.2 Statement of Progress Against Milestones 16](#_Toc264811184)

[3.3 Risks 17](#_Toc264811185)

[4 Conclusions 18](#_Toc264811186)

[5 Lessons learnt and Recommendations 19](#_Toc264811187)

**List of Figures**

Figure Page No.

No Figures.

**List of Tables**

Table Page No.

No Tables.

**Definitions**

|  |  |
| --- | --- |
| AHNS | Autonomous Helicopter Navigation System |
| QUT | Queensland University of Technology |
| HLO | High Level Objective |
| GCS | Ground Control Station |
| GUI | Graphical User Interface |
| HMI | Human Machine Interface |
| IDE | Integrated Development Environment |
| UI | User Interface |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

The Autonomous Helicopter Navigation System (AHNS) project for 2010 entails the development of ground control station (GCS) software, onboard flight control logic and a data network to enable communication within the system. The status of these systems in their design and initial implementation stages can be reported by considering the scope of their system requirements, the methodology being followed to enable delivery of these requirements against the set milestones and the current milestone progress. To have a complete picture in the process both encountered and foreseeable risks will need to be considered.

## Scope

The following status report details the progress made so far in designing the overall software architecture. It also considers the progress of GCS implementation and network packet standardisation and usage. The flight control design is reported to the level of the current design.

## Background

The preliminary design stage included a trade study for a suitable for flight computer hardware selection. Trade studies for the GCS and networking were not deemed necessary due to initial and successful development in Qt and using UDP by AHNS 2009. The flight control methodology selection was researched as part of a library report and not best presented in a trade study due to the quantity of information and lack of clear alternatives.

# Reference Documents

## QUT Avionics Documents

|  |  |  |
| --- | --- | --- |
| RD/1 | AHNS-2010-SY-HL-001 | AHNS, High Level Objectives of |
| RD/2 | AHNS-2010-SY-SR-001 | AHNS, System Requirements of |
| RD/3 | AHNS-2010-SY-PM-001 | AHNS, Project Management Plan |

## Non-QUT Documents

|  |  |  |
| --- | --- | --- |
| RD/4 | OpenEmbedded | OpenEmbedded. 2009. Welcome to OpenEmbedded. Available: <http://wiki.openembedded.net/index.php/Main_Page> (accessed March 10 2010). |
| RD/5 | Overo Fire COM | Gumstix 2010. Overo Fire COM. Available: <http://www.gumstix.com/store/catalog/product_info.php?products_id=227>  (accessed March 7 2010) |
| RD/6 | PIXHAWK: Open Source Micro Air Vehicle | “PIXHAWK: Open Source Micro Air Vehicle Computer Vision at ETH Zurich”. 2010.  Available: <http://pixhawk.ethz.ch> (accessed June 18 2010) |
| RD/7 | Design and Control of an Indoor Micro Quadrotor | Bouabdallah, S, P Murrieri and R Siegwart. 2007. "Design and Control of an Indoor Micro Quadrotor." In Advances in unmanned aerial vehicles: state of the art and the road to autonomy, ed. K. P. Valavanis: Springer Verlag. |
| RD/8 | IEEE Design and control of an indoor micro quadrotor | Bouabdallah, S., P. Murrieri and R. Siegwart. 2004. "Design and control of an indoor micro quadrotor." In IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION: IEEE. |
|  |  |  |

In the event of any conflict between this document and any RD referenced herein, such conflict shall be notified to Dr Luis Mejias.

In the following text, RD/x identifies referenced documents, where "x" denotes the actual document.

# 06332064 Project Summary

The control and ground control station subsystems within the AHNS project are central to supporting and enabling autonomous flight. They are complemented by the state estimation and localisation subsystem and rely on the hardware and communications systems to enable flight testing.

The GCS aim is to provide a GUI for plotting sensor data and displaying flight information whilst enabling flight control parameter changes to be made in flight. The GCS role comes from HLO-5 Ground Control Station [RD/1] and the system requirements detailing it include SR-B-02, SR-B-08, SR-B-09, SR-D-07, SR-D-08, SR-D-09 and SR-D-10 [RD/2]. The nature of the GCS led to this role also including part of the implementation of HLO-6 Communications so as to enable GCS UDP communication.

Control was detailed in HLO-4 Autonomous Hovering Flight. The focus being controller development of an autopilot system to enable sustained indoor autonomous hovering flight. SR-B-03 led to the role including onboard flight control implementation in conjunction with the state estimation and localisation role. SR-B-10 establishes PID control as a baseline requirement so limited initial scoping such as trade studies is required. The prototype design, implement and test methodology, or doctrine of progressive refinement therefore plays a large part in the role of control.

The number of HLO’s and SR’s of interest to control and GCS development require the whole software architecture be a secondary responsibility of these two roles. This being the case the major milestones of the project, GCS Testing, Station Keeping Testing and Augmented Flight Testing are all also milestones of the control and GCS subsystems. These milestones are at the project integration and testing stage. At the integration stage the milestones of lesser significance to the control and GCS subsystems are onboard hardware and electronics integration. Nevertheless these are major project milestones. A complete listing of project milestones in the form of work packages is available in [RD/3].

Integration represents the major milestone in the project timeline but it is misleading to consider each subsystem decoupled through their initial design and implementation stages. It is particularly difficult to separate the requirement for the GCS to provide network functionality from the communications subsystem and its need to log and display all state data from the state subsystem. Likewise the control cannot be implemented or even designed in isolation. System engineering has therefore been applied as the dominant engineering methodology followed.

## Methodology for Delivering Against Milestones

To deliver against the milestones established in [RD/3] and created to meet the aims of the control and GCS subsystems a system engineering methodology is being followed. At each stage of project progress the engineering methodology remains focused on the delivery of the subsystem aims against identified project milestones. The following section details the use of this methodology at the stages of inception, design, implementation and testing for the GCS and control subsystems.

### Project Inception Systems Engineering Methodology

The methodology at project inception was of utmost importance to the 2010 AHNS team. The use of System Engineering dictated that HLO’s and SR’s be defined first to ensure the validity of the deliverables being completed and to establish agreeable acceptance tests for the final deliverables. The result of this methodology, focusing the aims of the GCS and control subsystems, has been made clear in the Project Summary.

In 2009 the AHNS team managed to only complete a fraction of their aims established at this stage. Within QUT Avionics following the Systems Engineering approach to projects often results in this level of project completion. In itself this is not the fault of System Engineering, but the over estimation of available time and underestimation of risk on the part of project groups. To reduce the amount of risk exposed work AHNS 2010 took considerable time to identify that work that was critically important to completing the HLO’s and compared this to the work left uncompleted by previous projects. A key outcome was the adoption of several baseline requirements.

Baseline requirements enable a component of the system design and development to be specified almost in its entirety so that less schedule risk is present due to considerable iteration. In terms of the control subsystem, input from the project supervisor and AHNS 2009 and 2008 sources led to a baseline requirement of PID control thus removing the inability of 2008 to deliver on a state space control architecture. The number of requirements were also removed to avoid the over specification of how HLO’s were to be met. A key example of this is the specification of the GCS development environment.

In 2008 AHNS specified a Linux base GCS however they later migrated to Windows. In terms of the HLO’s the move was inconsequential however the system requirements needed to change significantly. In contrast the AHNS 2009 proved the capabilities of the Qt C++ development framework for a Ubuntu based GCS. Both GCS solutions were capable, but in both cases arbitrary decisions were made at the system requirements stage. Decisions on the particular component or subsystem implementation decisions were therefore avoided with a particular focus on the ability to define meaningful acceptance tests for all requirements and clear statements of HLO’s instead.

The methodology followed at project inception therefore ensures that for the control and GCS subsystems there is enough risk reduced, technical flexibility to allow timely delivery against milestones. Definition of the testing of those requirements deemed absolutely critical to HLO completion augments this to ensure successful delivery.

### Preliminary Design

In the preliminary design stage research and planning was carried out to establish a suitable software system architecture and flight computer.

#### Software Architecture

In 2009 the requirement to provide a control update rate of 50Hz in proved unreachable using serial modems and basing the control on the GCS. To address this, control and state estimation have been moved to an onboard flight computer. To enable the use of priority scheduling between control, state estimation and communications tasks the 2009 reliance on threads was diversified. The threaded architecture of both the GCS and flight computer is shown in Figure 2. Given the previous successful software developments in 2009 a Linux based GCS operating system was chosen along with the Qt C++ GUI framework.

#### Flight Computer

Moving the control and state estimation onboard required the computing requirements to be considered. As part of the methodology to ensure an appropriate solution research was carried out into what other research organisations used for onboard processing power. Computer On Modules (COMs) proved to be some of the more common solutions especially in image processing. A flight computer trade study was carried out to consider several commercial off the shelf COMs. After weighting the specifications of these against decision criteria the OpenEmbedded based Gumstix Overo Fire [RD/4-RD/5] was selected.



Figure - Gumstix Overo COM

The Overo Fire COM provides Bluetooth and 802.11g WiFi connectivity for direct TCP/IP networking and support for digital image processing. The support for image processing has been documented and used in other undergraduate helicopter and quadrotor projects [RD/6]. It required the purchase of two additional expansion boards, the Summit and the Pinto-TM to enable interfacing with the full range of supported hardware.

Table - Gumstix Overo Fire COM Summary

|  |  |
| --- | --- |
| **Processor** | 600MHz OMAP 3530 with ARM Cortex-A8 CPU plus a C64x+ digital signal processor (DSP) and 2D/3D accelerator. |
| **Mass** | 6g |
| **OS Support** | Linux 2.6.31+ |
| **Memory** | RAM 256MB, Flash 256MB, MicroSD Card |
| **Hardware Interface** | *With aid of expansion board (purchased or made):*  802.11g WiFi Bluetooth UARTs I2C Bus SPI Bus GPIO 6 x PWM 6 x ADC |

#### Control Research

The methodology followed in the design stages focused on borrowing for already proven designs and improving where necessary. The selection of the control scheme was no different. At the preliminary design stage quadrotor control was researched given AHNS had previously used a conventional RC helicopter. The research focused on peer-review literature and that which documented already proven control methods. [RD/7] and [RD/8] are of particulate note given their development of a quadrotor rigid body dynamic model and the provision of a Simulink model implementation of the PID attitude controller. The availability of a proven design means there is little chance of the attitude control methodology being unsound at its design level.

### Design and Implementation

The design and implementation of the GCS and the design of the control system demonstrate a methodology focused on delivering against milestones.



Figure - Software Architecture

#### GCS Design, Implementation and Testing

2009 demonstrated the development of both a HMI and autopilot GUI. It was noted that the HMI was important in providing real time data visualisation whilst the autopilot GUI was needed to update and run the flight control. The methodology of splitting control and data display adds unnecessary interfaces. To meet the current HLO’s it was decided to merge the functionality of both in a GCS. A redesign of the GUI widgets was therefore undertaken with the more complex artificial horizon widget being ported to the new GCS design.

The approach of designing a static GUI layout in 2009 led to a considerable amount of time being dedicated to designing the layout to be operator efficient. This leads to any GUI layout modifications requiring code changes, a condition that increases the risk of untested code being used after system integration. The methodology adopted was therefore to break the GCS up into self contained dockable widgets that can be rearranged and even dragged into new windows. This has the added benefit of enhancing code reused and unit testing both considered industry best practise in software design.

Figure 3 shows the currently implemented GCS widgets undergoing UDP packet integration testing. Widgets can trace their design back to the system requirements of the GCS. Those widgets part of the design include:

* Artificial Horizon  
  *Provide an OpenGL visualisation of the current aircraft pitch and roll attitude*
* System Status  
  *Display onboard system health and status information including RC link statues, battery level, commanded engine outputs, flight computer CPU usage and uptime.*
* Wifi Communications  
  *Configure the UDP link between the flight computer and the GCS by setting the IP addresses and ports. Launch the GCS telemetry thread to send and receive UDP data. Monitor the communications thread for disconnection or failure*
* Received Data Console  
  *Display the packets received and the downlink telemetry data rate and duplicate/failed packet rate.*
* Transmitted Data Console  
  *Display the packets transmitted and the uplink data rate and duplicate/failed packet rate.*
* Data Plotter

*Provide real-time data plotting of data from the sensors, autopilot or flight computer.*

* Blackfin Camera Feed

*Connect to and display a live video feed from the onboard camera. Part of the State Estimation subsystem. Demonstrating the ability to develop widgets as components and later integrate them into the GCS.*

* Attitude Control Trims and Bounds  
  *Set the trims and bounds on the roll, pitch and yaw control loops.*
* Attitude Control Gains  
  *Set the PID control gains on the roll, pitch and yaw control loops.*
* Guidance Control Gains  
  *Set the PID control gains on the x, y and z position control loops.*
* Guidance Trims and Bounds  
  *Set the trims and bounds of the x, y and z position control loops.*
* Flight Control  
  *Set the active control loops and their set points. Enables command of the airborne control.*

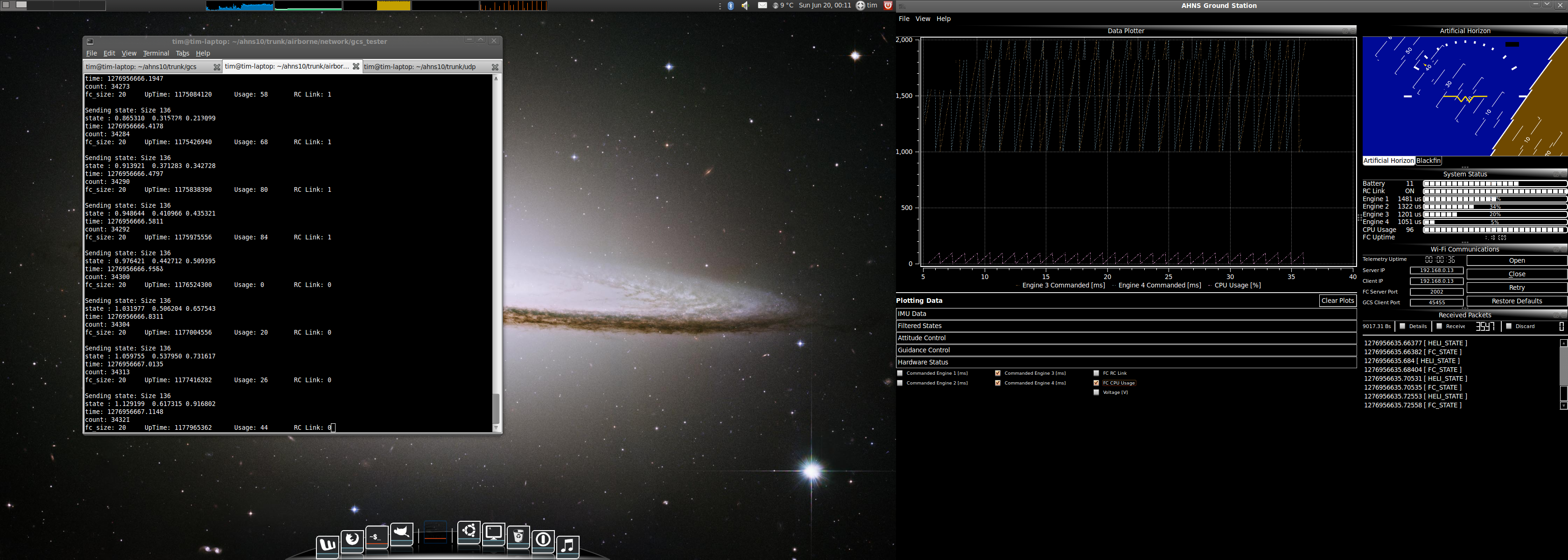
After the implementation of each widget a period of integration testing is undertaken. This involves the creation of GCS test programs designed to emulate the behaviour of the flight computer. They are used to connect to the GCS and receive and transmit data. This ensures that the GCS at all stages of development is tested to the level required for later milestones.

Figure - GCS GUI Implemented Widgets

#### Control Design

Control design is based on the flight tested work in [RD/7-8]. It abstracts the quadrotor’s reliance on differential control in terms of collective, roll, pitch and yaw commands, respectively. These can be written in terms of their incremental thrust increases on each engine.

To be converted to total individual engine thrust inputs the mixing matrix is applied:

Control can therefore be based on three attitude control PID loops; roll, pitch and yaw. For guidance the z or attitude input, can have an attached control loop. The x and y position control loops have been previously implemented as providing set points to pitch and roll loops. This, like all of the control will be confirmed with the available simulation tools then implemented, ground tested and flight tested.

The methodology followed in project inception, preliminary design and that being pursued in the current stage of implementation and testing demonstrates the GCS and control subsystems ability to deliver against milestones. The methodology followed has involved reflection on previous project work, research into other project work and a design and implementation stage focused on multilayer component, integration and system testing. It is certainly sound but more importantly it provides reduces the risk of HLO’s not being completed due to schedule and technical risks.

## Statement of Progress Against Milestones

Based on the documented work packages in [RD/3] the GCS and control subsystems are on schedule. Prior to the nearest major project milestone of GCS and Flight computer integration testing, three work packages need to be completed. These are WP-AP-01 Flight Computer Trade Study, WP-AP-03 Design Control System and WP-CG-01 Design of Ground Control Station.

WP-AP-01 was completed on schedule and resulted in the acquisition of the Gumstix Overo. The flight computer chosen has already proved capable of running flight control test programs, connecting to an 802.11g WiFi network and communicating via UDP to the GCS. Hardware and software complexity dictates that the control subsystem does not have the responsibility for on board software and the hardware connections. These are the responsibilities of the state estimation and hardware subsystems respectively.

WP-AP-03 is on track to be completed by its deadline of the end of June 2010. With the exception of guidance the controller has been completed to a level acceptable for attitude control. It has not been necessary to readjust the controller design since it was established in the baseline requirement. This is a testament to the soundness of the project inception methodology involving consideration of supervisor and past student experience. At the completion of this work package the design is of interest and not its software implementation. A delay in completion is not critical to the progress towards system integration.

Development of the GCS is due to be completed at the end of WP-CG-01. Of the widgets required only 6 remain to be implemented and tested. Conceptually four of these are the same whilst the final two are functionally similar to widgets already implemented. The GCS current state of implementation already fulfils the system level requirements for data plotting and UDP uplink and downlink capabilities. No change of proposed GCS capabilities is necessary.

## Risks

There have been minimal delays in the GCS and control subsystems. The exception to this is the interface of the hardware and control subsystems where acquisition of the flight computer took longer than expected due to supplier demand. The consequences of the resulting schedule risk were reduced by instead focusing on the software implementation and setting aside more time to complete the remaining hardware purchases.

The flight computer and its interface through the control logic to the hardware motor speed controllers remains a potential source of future technical risk. The approach has been discussed amongst all AHNS team members and widely researched leading to the conclusion that there are limited options and none which present less risk. The level of technical risk associated with this is moderate from a control implementation perspective. It may however be higher if the possibility of hardware failure is included, leading to severe consequences in scheduling and budgeting. To mitigate the risk replacement hardware funds have been sought and multiple AHNS members are involved in reviewing hardware activities.

GCS development has followed a methodology focused on reducing scheduling and technical risk. This included using previous year code where tested and proven. Given previous years have flow with the code used it was expected that the testing would not show component level failure. The reality however was that the artificial horizon would cause a program crash when displaying altitudes beyond 10m and the method of threading implemented in 2009 provide ineffective in keeping the GUI responsive during high telemetry data rates. Time therefore had to be taken to debug and then correct the errors. The impact on schedule and technical risk was not great but the approach to GCS testing was altered to be structured to test over a range of possible operating conditions.

A remaining source of foreseeable risk in both the control and GCS development is their unification during flight computer implementation along with the state estimation subsystem. This process will involve taking GCS data, transmitting it via UDP, receiving it at the flight computer and using it to configure the control. To reduce this risk packet structures are being developed and serialisation code has been developed and thoroughly tested after it was found the flight computer will need to communicate with both Little Endian and Big Endian systems. The utilisation of UDP presents the risk that flight critical packets will be lost or delivered out of sync. Flight critical packets need to have their receipt confirmed by the flight computer before the GCS can assume successful transmission. Like the major hardware risks the solution is in AHNS subsystem interface definition, peer review and testing.

# Conclusions

Within the AHNS 2010 project the ground control station subsystem and flight control subsystem are responsible for completing 2 HLO’s and contributing the success of another. In terms of systems requirements 8 are directly pertinent. The role of delivering these systems has been seen to be extremely important for project success.

By following a systems engineering methodology with emphasis on using lessons and results attained from previous projects and research it has proved possible for the subsystems to deliver against early project milestones. The stages specifically discussed in this methodology are project inception, preliminary design and design, development and implementation. Each stage has lead to the subsystems remaining on schedule to deliver against the significant project milestones of GCS Testing, Station Keeping Testing and Augmented Flight Testing.

There has been limited risk associated with the developments of the control and GCS subsystems. Risks occurring at the cross over of other subsystems include scheduling and budgetary risks for hardware, schedule and technical for communications and scheduling and technical for state estimation and localisation. Methods to reduce risk have included a greater focus on testing to ensure technical compliance and to focus the investment of time; peer-review and group discussion.

# Lessons learnt and Recommendations

Although the subsystems are on track for completion, their progress has not been without significant challenges. Lessons learnt from these challenges have guided the refinement of both design and methodology.

The first lesson learnt was the coupling between technical and scheduling risk, with the GCS development as a particular example. In 2009 an integrated development environment was not used. Instead a combination of text editors, manually created Makefiles and UI files were generated. It could be hypothesised that this slowed development time since the two group members responsible for the development had to juggle learning a new GUI framework with operating a diverse and cumbersome tool chain. Use of the QtCreator IDE enabled rapid development and testing of the UI, supporting code and efficient access to the Qt framework documentation. As a testament to its usefulness the key functions of the GCS, including improved threading, were implemented in one month.

Code reuse from previous years and other projects has been found to be efficient, but only if it is thoroughly tested. Although the level of testing is not more than that which would be required with custom written code, it was found it was possible to overlook the need. Oversights in testing have led to GCS crashes due to HMI string errors, little endian and big endian conversion errors in data packet creation and test program crashes. Although the GCS is highly customisable it was found that unfamiliar members of the AHNS project could provide useful feedback on how widgets could be laid out and how they are expecting them to operate.

These two lessons led to the ultimate realisation that specifying build frameworks and particular reuse of code would not have been sensible at the project inception. The delayed procurement process further illustrated the need for task flexibility. Indeed it is not possible to complete a whole subsystem in one attempt. Rescheduling, readjustment and iteration of the design and methodology needs to take place at all stages to ensure all subsystems are kept on schedule. At the very least teach subsystem’s progression needs to be such that other subsystems can be continued without foreseeable delays.

Finally research was found to be incredibly useful. Without this step it would not be trivial to simulate a new quadrotor control architecture, develop a GCS in a limited period of time or procure a flight computer capable of control, state estimation and 802.11g Wifi communication.