DEVELOPMENT OF AN EXPERIMENTAL TILT-WING VTOL UNMANNED AERIAL VEHICLE

JACOB VENTER

THESIS PRESENTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING AT THE UNIVERSITY OF STELLENBOSCH

SUPERVISOR: DR. THOMAS JONES

APRIL 2006
Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, unless mentioned otherwise, and that I have not previously, in its entirety or in part, submitted it at any university for a degree.

Jacob Venter: ...................... Date: ......................
Abstract

This project was initiated by ARMSCOR to research the feasibility of an unmanned aerial vehicle launching from and landing on a South African Navy ship. The objective of this project was to develop a strong basis for other similar projects.

A new experimental airframe in a tilt-wing configuration was built. Simulation models were developed for this airframe and show that it should be capable of controlled takeoff and landing, hovering and normal flight. A key aspect to make this possible is suitable avionics. The avionics suite was developed to be flexible, modular and lightweight and was flight tested extensively in cooperation with another project and proven to work reliably. The development of the airframe and avionics are first outlined and then described in detail.

Work towards an autonomous flight demonstration will continue in two Masters Projects using this project as basis.
Uittreksel

Hierdie projek is van stapel gestuur deur ARMS COR om die doenbaarheid van onbemande vliegtuie wat van skepe af opstyg en land, te ondersoek. Die doel van hierdie projek is om 'n soliede fondasie te skep vir ander, soortgelyke projekte.

'n Nuwe eksperimentele lugraam met 'n kantelvlerk opstelling is gebou. Daar is ook simulasiemodelle van hierdie lugraam ontwikkel en die simulatsies wys dat beheerde opstyg, landing en vlug moontlik is. Die sleutel is deur die gebruik van geskikte vlieg-elektronika. Hierdie elektronika is ontwikkel om buigsaam, modulêr en liggewig te wees en dit is deeglik getoets in samewerking met 'n ander projek en werk betroubaar. 'n Oorsig van die ontwikkeling van die lugraam en vlieg-elektronika word gegee en word later ook in detail bespreek.

Hierdie projek word opgevolg deur twee ander Meesters projekte waarin onbemande vlugte gepoog sal word.
Acknowledgements

I would like to extend my gratitude to the following people for their assistance. Without your help, this project would not have come as far as it did.

- ARMSCOR for funding the project.
- My supervisor, Dr. Thomas Jones, for guiding the project into the right direction, but still giving me enough room to make it feel like my project.
- My colleague, Fanus Groenewald, for all the planning sessions, assistance, interesting ideas and RC flying. You made working in the lab a lot of fun. Also, thank you for the permission to use photographs and information from your thesis.
- Willie van Rooyen for finishing the airframe which I started. It is a work of art.
- All my friends at ESL, especially Eckhard Kühn and Pieter Theron, for all the laughs and the interesting discussions.
- My parents for teaching me the values of life and your continued love and support.
- My girlfriend, Alida Coetzee, for your love and patience.
# Contents

Declaration ii  

Abstract iii  

Uittreksel iv  

Acknowledgements v  

Contents vi  

List of Figures xii  

List of Tables xvi  

Nomenclature xvii  

Chapter 1 – Introduction 20  

1.1 Background ................................................................. 20  
1.2 Project Goals ............................................................... 21  
1.3 Thesis Outline ............................................................. 22  

Chapter 2 – Flight System Overview 24  

2.1 Requirements .............................................................. 24  
2.1.1 Airframe ............................................................... 24  
2.1.2 Avionics ............................................................... 24  
2.2 Implementation of the Flight System ............................ 25  
2.2.1 Airframe ............................................................. 25
2.2.2 Avionics ................................................................................................................... 26
2.3 Summary ........................................................................................................................... 29

Chapter 3 – Airframe Development and Modelling .............................................................. 31

3.1 Airframe Requirements ................................................................................................. 31
3.2 Initial Ideas ...................................................................................................................... 31
3.3 Development of the Final Airframe ............................................................................... 34
  3.3.1 Choice of Model Aircraft ......................................................................................... 34
  3.3.2 Choice of Power Plant ............................................................................................. 35
  3.3.3 Modifications to the wings ...................................................................................... 37
  3.3.4 Mounting the wing to the fuselage ......................................................................... 39
3.4 Modelling of the Airframe ............................................................................................. 40
  3.4.1 Thrust Vectoring ..................................................................................................... 41
  3.4.2 Moments of Inertia .................................................................................................. 41
3.5 Summary .......................................................................................................................... 43

Chapter 4 – Control Law Design and Simulation .................................................................. 44

4.1 Building the Simulink Model ......................................................................................... 44
  4.1.1 Modelling of Thrust Vectoring ................................................................................ 45
  4.1.2 Coordinate transformation system ......................................................................... 47
  4.1.3 Aircraft Dynamics .................................................................................................... 48
4.2 Linear Model ................................................................................................................... 49
  4.2.1 Acquiring the Linear Model ..................................................................................... 49
  4.2.2 Verifying the Linear Model ..................................................................................... 50
4.3 Control Law Design ....................................................................................................... 54
  4.3.1 Control Strategy ....................................................................................................... 54
  4.3.2 Decoupling the model .............................................................................................. 56
Chapter 5 – Avionics Development

5.1 PC/104CAN Controller ................................................................. 61
  5.1.1 Hardware .................................................................................. 62
    5.1.1.1 Power Supplies ................................................................. 63
    5.1.1.2 Complex Programmable Logic Device (CPLD) Hardware ........ 63
    5.1.1.3 Microcontroller Hardware .................................................. 64
  5.1.2 Software .................................................................................. 65
    5.1.2.1 Complex Programmable Logic Device (CPLD) Software .......... 65
    5.1.2.2 Microcontroller Software ................................................. 70
  5.1.3 CAN over ISA Protocol ........................................................... 74
5.2 Servo Board .................................................................................. 78
  5.2.1 Hardware .................................................................................. 79
    5.2.1.1 UPS Circuit ...................................................................... 79
    5.2.1.2 Microcontroller Hardware ................................................. 81
    5.2.1.3 Power Supply .................................................................... 82
  5.2.2 Software .................................................................................. 82
    5.2.2.1 Generating Servo Outputs ............................................... 82
    5.2.2.2 Measuring RC receiver servo positions ................................ 85
    5.2.2.3 Other Interrupts Service Routines ..................................... 86
    5.2.2.4 Mixing Algorithm ............................................................ 89
5.3 CANsens board .............................................................................. 90
5.4 CANsensIMU Board ...................................................................... 91
List of Figures

Figure 1.1 – Global Hawk (left) and Predator (right) ................................................................. 20
Figure 1.2 – Seeker II .................................................................................................................. 20
Figure 1.3 – Vulture .................................................................................................................. 21
Figure 1.4 – Thesis Outline ....................................................................................................... 23
Figure 2.1 – The completed Airframe ...................................................................................... 25
Figure 2.2 – Block Diagram of Implemented Avionics ............................................................ 27
Figure 2.3 – The fully assembled avionics pack ...................................................................... 28
Figure 3.1 – Initial Concept of Airframe .................................................................................. 31
Figure 3.2 – Configuration of vertical outlets with control surfaces .................................... 32
Figure 3.3 – First variation of ducted fan concept ................................................................. 33
Figure 3.4 – Second variation of ducted fan concept ............................................................. 33
Figure 3.5 – The unmodified Ultra Stick (before) ................................................................. 35
Figure 3.6 – OS91FX Methanol-powered engine .................................................................... 36
Figure 3.7 – AXI5330/18 Brushless Motor ............................................................................ 37
Figure 3.8 – Position of the paddles behind the engines ....................................................... 38
Figure 3.9 – Underside of wing ............................................................................................... 39
Figure 3.10 – Standard symbols and axes definition for equations of motion [6] ............... 40
Figure 4.1 – Simulink model of Thrust Vectoring ................................................................. 45
Figure 4.2 – Plots of the thrust vectoring simulation .............................................................. 46
Figure 4.3 – Simulink block to transform Cartesian Coordinates to Geodetic ....................... 47
Figure 4.4 – Simulink block describing full aircraft dynamics ............................................. 49
Figure 4.5 – Simulink block used to verify the linear model ................................................ 50
Figure 4.6 – Plots of Position and Orientation for Test 1 ...................................................... 51
Figure 4.7 – Plots of Position and Orientation for Test 2 ..................................................... 52
Figure 4.8 – Plots of Position and Orientation for Test 3 ...................................................... 52
Figure 4.9 – Plots of Position and Orientation for Test 4 ...................................................... 53
Figure 4.10 – Plots of Position and Orientation for Test 5 ..................................................... 53
Figure 4.11 – Block diagram of Control Strategy .................................................................. 55
Figure 4.12 – Simulink block containing the controllers ....................................................... 57
Figure 4.13 – Entire Simulink Simulation ................................................................. 57
Figure 4.14 – Position of the aircraft during the test ........................................... 58
Figure 4.15 – 3D Velocity of the aircraft during the test ........................................ 59
Figure 4.16 – Euler angles of the aircraft during the test ........................................ 59
Figure 5.1 – Block Diagram of PC/104CAN Controller ......................................... 61
Figure 5.2 – The fully assembled PC/104CAN Controller ...................................... 63
Figure 5.3 – PIC hardware and port assignments .................................................... 65
Figure 5.4 – Block diagram of the logic on the CPLD .......................................... 66
Figure 5.5 – Registers used on the CPLD ................................................................. 67
Figure 5.6 – Protocol Handshake Logic ................................................................. 68
Figure 5.7 – MAX+plus II simulation of Read/Write from ISA and PIC .................... 69
Figure 5.8 – Flowchart of main program loop ......................................................... 70
Figure 5.9 – Flowchart of High Priority ISR .......................................................... 72
Figure 5.10 – Flowchart of Low Priority ISR ......................................................... 74
Figure 5.11 – CAN over ISA Write Cycle ............................................................... 75
Figure 5.12 – CAN over ISA read process ............................................................ 75
Figure 5.13 – Block Diagram of Servo Board ......................................................... 78
Figure 5.14 – Fully assembled Servo Board .......................................................... 79
Figure 5.15 – UPS Circuit of the Servo Board ....................................................... 80
Figure 5.16 – Alternative UPS circuit ................................................................. 80
Figure 5.17 – dsPIC hardware and port configuration ........................................... 81
Figure 5.18 – Flowchart of Timer2 ISR State Machine .......................................... 83
Figure 5.19 – Flowchart of Timer4 ISR State Machine .......................................... 84
Figure 5.20 – Flowchart of Input Capture ISR ...................................................... 85
Figure 5.21 – Flowchart of Timer5 ISR ................................................................. 87
Figure 5.22 – Flowchart of CAN ISR .................................................................... 88
Figure 5.23 – Block Diagram of CANsens Board ................................................... 90
Figure 5.24 – The assembled CANsens Board ..................................................... 91
Figure 5.25 – Fully assembled IMU ....................................................................... 92
Figure 5.26 – Honeywell Magnetometer ............................................................... 93
Figure 5.27 – Fully assembled GPS/Aerocomm Motherboard ............................... 94
Figure 5.28 – Power Distribution Diagram................................................................. 95

Figure D.1 – A typical RC servo .................................................................................. 113
Figure D.2 – Servo connectors for JR (left) and Futaba (right)................................. 115
Figure D.3 – Example of a +20° angle command signal ........................................... 115
Figure E.1 – Pin assignments of CAN connector (Top View) .................................... 117
Figure E.2 – Pin assignments of RS232 connector (Top View)................................. 117
Figure E.3 – Altera JTAG Programming Header [27] (Top View)............................... 118
Figure E.4 – Microchip ICD programming header (Top View).................................. 118
Figure F.1 – PC/104CAN Schematic Overview......................................................... 120
Figure F.2 – Schematic of PC/104CAN Microcontroller ............................................ 121
Figure F.3 – Schematic of PC/104CAN Altera CPLD ............................................... 121
Figure F.4 – Schematic of PC/104CAN Power supply............................................... 122
Figure F.5 – Schematic of PC/104CAN Fan driver..................................................... 123
Figure F.6 – Top side of PC/104CAN PCB................................................................. 123
Figure F.7 – Bottom side of PC/104CAN PCB........................................................... 123
Figure F.8 – Top Overlay of PC/104CAN PCB........................................................... 124
Figure F.9 – Bottom Overlay of PC/104CAN PCB..................................................... 124
Figure F.10 – Schematic of Servo Board..................................................................... 126
Figure F.11 – Top side of Servo Board PCB............................................................... 126
Figure F.12 – Bottom Side of Servo Board............................................................... 127
Figure F.13 – Top Overlay of Servo Board............................................................... 127
Figure F.14 – Bottom Overlay of Servo Board.......................................................... 127
Figure F.15 – Schematic of CANsens Digital Circuits............................................... 129
Figure F.16 – Schematic of CANsens Analogue Circuit............................................ 129
Figure F.17 – Top side of CANsens Board............................................................... 130
Figure F.18 – Bottom side of CANsens Board.......................................................... 130
Figure F.19 – Top Overlay of CANsens Board.......................................................... 130
Figure F.20 – Bottom Overlay of CANsens Board.................................................... 131
Figure F.21 – Schematic of GPS/Aerocomm Motherboard........................................ 133
Figure F.22 – Top side of GPS/Aerocomm Motherboard........................................... 134
Figure F.23 – Bottom side of GPS/Aerocomm Motherboard ........................................ 134
Figure F.24 – Top Overlay of GPS/Aerocomm Motherboard .................................... 135
Figure H.1 – Position of aircraft during Test A ......................................................... 145
Figure H.2 – Velocity of aircraft during Test A ....................................................... 146
Figure H.3 – Orientation of aircraft during Test A .................................................. 146
Figure H.4 – Position of aircraft during Test B ....................................................... 147
Figure H.5 – Velocity of aircraft during Test B ....................................................... 147
Figure H.6 – Orientation of aircraft during Test B .................................................. 148
List of Tables

Table 2.1 – Estimated cost of the avionics and airframe .................................................. 30
Table 3.1 – Calculation of Aircraft Moments of Inertia and CG ........................................ 42
Table 3.2 – Weight budget of the aircraft ......................................................................... 43
Table 5.1 – Legend of the symbols used in Figure 5.7 ...................................................... 70
Table 5.2 – Summary of CAN over ISA commands ........................................................ 76
Table 5.3 – Format of the Status packet ........................................................................... 77
Table 5.4 – Format of a CAN packet ................................................................................ 77

Table B.1 – CAN packet Layout..................................................................................... 107
Table B.2 – Broad outline of CAN protocol .................................................................... 108
Table E.1 – Pin out of PC104 Header [28] .....................................................................119
Table F.1 – Bill of Materials of PC/104 CAN PC-Board ............................................... 125
Table F.2 – Bill of Materials of Servo Board ................................................................. 128
Table F.3 – Bill of materials of CANsens Board ............................................................ 132
Nomenclature

Acronyms and Abbreviations

3D Three Dimensional
A/D Analogue to Digital
AC Alternating Current
ATE Advanced Technologies and Engineering
CAA Civil Aviation Authority
CAN Controller Area Network
CG Centre of Gravity
CPLD Complex Programmable Logic Device
CPU Central Processing Unit
DC Direct Current
DCM Direction Cosine Matrix
dsPIC Digital Signal Peripheral Interface Controller
ECEF Earth Centred Earth Fixed (Cartesian Coordinate System)
EID Extended Identifier
EIRP Effective Isotropic Radiated Power
ESL Electronic Systems Laboratory
ESR Equivalent Series Resistance
GPS Global Positioning System
HDD Hard Disk Drive
HIL Hardware In The Loop
ICD In-Circuit Debugger
IDE Intelligent Drive Electronics
IMU Inertial Measurement Unit
ISA Industry Standard Architecture
ISR Interrupt Service Routine
JTAG Joined Test Action Group
LCD Liquid Crystal Display
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiPo</td>
<td>Lithium Polymer</td>
</tr>
<tr>
<td>LLA</td>
<td>Latitude Longitude Altitude</td>
</tr>
<tr>
<td>MOPS</td>
<td>Minimized Open PC System</td>
</tr>
<tr>
<td>MOSFET</td>
<td>Metal Oxide Semiconductor Field Effect Transistor</td>
</tr>
<tr>
<td>NiCad</td>
<td>Nickel Cadmium</td>
</tr>
<tr>
<td>NiMH</td>
<td>Nickel Metal Hydrate</td>
</tr>
<tr>
<td>OBC</td>
<td>Onboard Computer</td>
</tr>
<tr>
<td>PC</td>
<td>Personal/Portable Computer</td>
</tr>
<tr>
<td>PIC</td>
<td>Peripheral Interface Controller</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse-Width Modulation</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RPM</td>
<td>Revolutions Per Minute</td>
</tr>
<tr>
<td>RTCM</td>
<td>Radio Technical Commission for Maritime Services Commission</td>
</tr>
<tr>
<td>S/VTOL</td>
<td>Short/Vertical Take-off Or Landing</td>
</tr>
<tr>
<td>SID</td>
<td>Standard Identifier</td>
</tr>
<tr>
<td>SISO</td>
<td>Single Input Single Output</td>
</tr>
<tr>
<td>TTL</td>
<td>Transistor Transistor Logic</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>UPS</td>
<td>Uninterruptible Power Supply</td>
</tr>
<tr>
<td>US</td>
<td>University of Stellenbosch</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VHSIC</td>
<td>Very High Speed Integrated Circuit</td>
</tr>
</tbody>
</table>
Symbols used in Simulation

Φ Roll Angle
Θ Pitch Angle
Ψ Yaw Angle
p Roll Rate
q Pitch Rate
r Yaw Rate
X Position in the north direction
Y Position in the east direction
Z Position in the down direction
U Velocity in the north direction
V Velocity in the east direction
W Velocity in the down direction
ϕ_{x1} Port Side Horizontal Paddle angle
ϕ_{y1} Port Side Vertical Paddle angle
F_1 Port Side Engine Thrust
ϕ_{x2} Starboard Side Horizontal Paddle angle
ϕ_{y2} Starboard Side Vertical Paddle angle
F_2 Starboard Side Engine Thrust
Chapter 1 – Introduction

1.1 Background

The interest in and development of UAVs increased dramatically over the past few decades. Although the most well known UAVs, like "Global Hawk" and "Predator", were developed and manufactured in the USA, the whole world takes part in the development of this new technology, including South Africa.

South Africa joined the UAV race in the mid 1980s with the introduction of the "Seeker" UAV developed by Denel Aerospace Systems. In the early 1990s, "Seeker II" was developed and it has been approved by the CAA for use in Civil Aerospace. Another proudly South African UAV is the "Vulture", developed by ATE.
Although UAVs have a strong military application, they can also be used in a commercial environment. A new military application is Unmanned Combat Aerial Vehicles (UCAVs), but few are operational at this stage. The most common military applications include surveillance, fire-control and coastal patrol. Commercial applications can range from aerial photography to vegetation analysis and crop dusting.

The UAV programme at the University of Stellenbosch was started in 2001. The first project was to automate a fixed-wing RC aeroplane [11] and was soon followed by the automation of a small electrically powered helicopter [9]. Current projects include this one, autonomous flight of a methanol-powered helicopter, autonomous takeoff and landing and acrobatic flight.

1.2 Project Goals

The broader project was initiated by the South African Navy to research the feasibility of an unmanned aircraft launching from and landing on a ship. There will be two follow-up projects to try and achieve this.

This project had the following goals:

- To develop an airframe capable of normal and vertical flight as no such airframes are available on the open market.
- To develop sensors and avionics compatible with this airframe that can also be used in other projects.
• To model this airframe and simulate simple manoeuvres to prove that the chosen actuators are sufficient to control the airframe.

• To extend the simulation to include hardware-in-the-loop.

• To control this airframe using the developed avionics.

• To finish the project within the two years allocated for a Masters project.

• To provide a solid basis for similar future projects.

1.3 Thesis Outline

The following section gives an outline of this thesis and a flow diagram summarising this thesis can be seen in Figure 1.4.

Chapter 2 gives an overview of the flight system, consisting of the avionics and airframe. The identified requirements for both of abovementioned parts are identified and listed. The detail of the development is in the two subsequent chapters.

Chapter 3 explains the process of developing the airframe. Several concepts are mentioned as well as the final airframe that is modelled in preparation for the simulations to follow.

Chapter 4 shows how the simulation is put together. After this, a controller, capable of stabilizing the position and heading of the aircraft, is developed, tested and explained.

Chapter 5 not only contains the detailed design of the hardware developed by the Author, but also contains an overview of all other hardware needed to complete the avionics.

Chapter 6 lists the results and the recommendations for future projects.
This thesis's structure is summarised in the following flowchart.

![Thesis Outline Flowchart](image_url)
Chapter 2 – Flight System Overview

The flight system consists of an airframe and avionics. In the next sections, the requirements are listed followed by brief descriptions of the implementation. Details of the airframe and avionics implementation are given in Chapter 3 and Chapter 5 respectively. A cost estimate is given at the end of the chapter.

2.1 Requirements

2.1.1 Airframe

The requirements for the airframe are the following:

- Shall be capable of S/VTOL and long endurance/range
- Shall be controllable by onboard avionics
- Shall be possible to be built in a reasonable time at a reasonable cost
- Shall make maximum use of commonly available components
- Shall be repairable and maintainable

2.1.2 Avionics

As the avionics will be used in an aircraft, it needs to adhere to some specifications to reduce the risk of failure. The following requirements were set:

- Shall be modular
- Shall be easy to use, reconfigure and upgrade
- Shall be robust enough to withstand the shocks and vibrations associated with an aircraft
- Shall make maximum use of readily available components
- Shall be able to be used in a variety of UAVs
- Shall make provision for hardware-in-the-loop simulation
- Should have minimum mass and power consumption
- Shall have enough processing power to be able to run the required control algorithms on the aircraft, and shall have enough spare capacity for future developments
• Should be compatible with existing systems
• Should make use of knowledge and experience in the ESL

It is quite a difficult task to adhere to all of the requirements mentioned above. A lot of planning and research was done to get to the final solution and it will be discussed in the next sections.

2.2 Implementation of the Flight System

2.2.1 Airframe

There are no "off the shelf" airframes available that are VTOL capable and therefore one had to be designed and built. Chapter 3 provides the detail of the development of the airframe. In short, it was decided to buy a commercial RC aircraft ("Ultra Stick"), modify it to accommodate two engines and add the required actuators behind the engines to vector the thrust in any direction to be able to control the aircraft. This was the most practical option available at the time. The final product can be seen in Figure 2.1.

![Figure 2.1 – The completed Airframe](image)
2.2.2 Avionics

Background
The need for a proper avionics set arose from the increasing number of UAV projects in the laboratory. The Author and Fanus Groenewald [10] were assigned the job of developing the new avionics. The development process involved the following steps:

- Identifying the requirements for the system and its components
- Studying existing systems in both the laboratory and the rest of the world
- Designing a practical and easy-to-use flight system architecture
- Choosing the main computer
- Identifying the different sensors and systems needed
- Designing, developing and testing the components
- Testing the system as a whole and making the necessary changes

Earlier avionics developed in the laboratory made use of a number of Atmel Microcontrollers to do the main processing [11] and a high speed ground link to do all of the processing on the ground station [9]. Although these are avionics solutions, they are tailor-made for the specific task and with limited processing power. These earlier avionics helped to identify the requirements of this avionics system.

Implementation
Figure 2.2 shows the developed flight system architecture in the form of a block diagram and Figure 2.3 shows the fully assembled avionics. Note the key at the bottom of Figure 2.2 denoting the hardware that was developed during the span of this project and the hardware that was bought "off-the-shelf" or developed earlier. Commands for the avionics are given through the RF modem connected to the ground station. The receiving RF modem is connected to the heart of the avionics, namely the main computer. Other hardware connected to the main computer is the GPS receiver and the PC/104CAN controller. The PC/104CAN controller is the link between the CAN bus, to which the sensors and Servo Board are connected, and the main computer. Manual control of the aircraft is through a conventional RC system which is connected to the Servo Board. The
Servo Board determines whether the main computer or the backup pilot controls the servos according to a set of criteria.

Figure 2.2 – Block Diagram of Implemented Avionics

There are several options for the main computer. The PC/104 form factor computer was selected because it is quite small and there are a variety of processors to choose from. There are PC/104 expansion cards readily available, should the need arise, for instance, to have a frame grabber for vision based navigation. The final selection was the Kontron MOPS LCD7 [36] that has the following basic specifications:

- Fanless Intel® PIII Celeron® 300MHz CPU
- 2xRS232 Serial Ports
- 2xUSB Controllers
- 64MByte RAM
- 32MByte Solid-state IDE HDD
Figure 2.3 – The fully assembled avionics pack

The Kontron PC/104 computer represents a good balance between price, processing power, power consumption and weight. There are also faster versions of the same computer available if the need arises for more processing power.

The decision to use a CAN bus was motivated by the following:

- CAN is faster than RS232 communication.
- The modules are easy to use.
- The protocol is already in place to use multiple nodes. If RS485 was used, the protocol needed to be developed.
- Most competitive microcontrollers have an integrated CAN module.
- The CAN protocol already implements error correction and was developed to be used in high-noise environments.
- The ESL supports the required expertise and instrumentation.

To minimize the effect of noise, it was decided to have microcontrollers close to the sensors to digitise the measurements and transmit it via the CAN bus, instead of having
long analogue wires running from the sensor to a central measurement unit. The microcontroller samples the signals at a high rate (1kHz) and then down samples it to reduce measurement noise and therefore increasing the effective resolution.

It was decided to use commercial RC equipment together with the Servo Board to reduce cost and also to be able to have an experienced RC pilot as a backup in the system. The Servo Board is the link through which the backup pilot accesses the system. There is a backup battery connected to the system to keep the Servo Board, servos and RC receiver alive in case of main battery failure.

All RC equipment was standardized on JR (Japan Radio) products.

The electronic hardware developed for the flight system will be discussed in detail in Chapter 5.

2.3 Summary

This Chapter described the process of developing the flight system. Firstly, it was shown what the requirements of the flight system are. An overview was given of the airframe and avionics developed to meet the requirements.

Table 2.1 shows the estimate cost of the flight system.
### Avionics

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Computer</td>
<td>R 5,000.00</td>
</tr>
<tr>
<td>CANsens</td>
<td>R 300.00</td>
</tr>
<tr>
<td>CANsensIMU</td>
<td>R 2,500.00</td>
</tr>
<tr>
<td>Servo Board</td>
<td>R 600.00</td>
</tr>
<tr>
<td>GPS/Aerocomm Motherboard</td>
<td>R 1,500.00</td>
</tr>
<tr>
<td>RC Sundries</td>
<td>R 800.00</td>
</tr>
<tr>
<td>Main Battery</td>
<td>R 600.00</td>
</tr>
</tbody>
</table>

### Airframe

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC aircraft</td>
<td>R 1,500.00</td>
</tr>
<tr>
<td>Flight Batteries</td>
<td>R 8,000.00</td>
</tr>
<tr>
<td>Speed Controllers</td>
<td>R 4,500.00</td>
</tr>
<tr>
<td>Brushless Motors</td>
<td>R 4,400.00</td>
</tr>
<tr>
<td>Servos</td>
<td>R 2,000.00</td>
</tr>
<tr>
<td>Sundries</td>
<td>R 750.00</td>
</tr>
</tbody>
</table>

| Total Cost                 | R 32,450.00 |

*Table 2.1 – Estimated cost of the avionics and airframe*

Chapter 3 and Chapter 5 detail the development of the airframe and the avionics respectively.
Chapter 3 – Airframe Development and Modelling

This chapter first describes the development of the airframe concepts, choice of airframe, choice of power plant and construction of the final airframe. After this, the basic airframe model is developed.

3.1 Airframe Requirements

As mentioned in Chapter 2, the airframe has the following requirements:

- Shall be capable of S/VTOL and long endurance/range
- Shall be controllable by onboard avionics
- Shall be possible to be built in a reasonable time at a reasonable cost
- Shall make maximum use of commonly available components
- Shall be repairable and maintainable

3.2 Initial Ideas

During the initial phase of the project, the idea was have a central power source and then redirect the power with pipes and have flaps in the pipes to direct the airflow. Figure 3.1 shows the initial concept.

![Initial Concept of Airframe](image)

The power source is situated in the centre of the four vertical outlets. A multitude of valves would allow the controller to determine the amount of airflow that goes to each of
the four vertical outlets as well as the horizontal outlet. The vertical outlets will be used to stabilize the aircraft in a hover and the horizontal outlet to enable normal flight.

![Configuration of vertical outlets with control surfaces](image)

*Figure 3.2 – Configuration of vertical outlets with control surfaces*

The vertical outlets each has a single flap, as shown in Figure 3.2, to allow the air to be directed. With this configuration, the sum of all the forces applied can be in any direction and a moment can also be generated to control yaw. The amount of airflow out of any outlet can also be varied to generate pitch and roll moments. The problem with this setup is that for pure lateral and longitudinal movement there are forces working against each other, resulting in a loss of lift.

In theory, this all seemed viable up to the point where the valve configuration had to be designed in detail. The available power plant (OS91 Ducted Fan) can generate about 6kg of thrust, but because the air can have a total direction change of up to 270°, the loss will be too much to be able to lift a reasonable load.

To solve this problem, the concept of having multiple power plants and have the aircraft take off like a rocket, was developed. There were two basic types as shown by Figure 3.3 and Figure 3.4.
The first variation would make use of two outlets, each with full thrust vectoring. The advantage of this is that the airflow does not go through any bends and thus has the least amount of loss. The problem with this setup is to get the airframe strong enough to support the wings and the complexity of the thrust vectoring. Another problem is that there cannot be a vertical and horizontal stabilizer behind the engines, because it has to take off by standing with engine outlets facing downwards.

![Figure 3.3 – First variation of ducted fan concept](image)

The second variation has two power sources, but with four outlets. The outlets are arranged in a similar way to those in Figure 3.2. There is a lot of loss of thrust due to the bends in the pipes, but the two power sources should have enough thrust to lift a reasonable load.

![Figure 3.4 – Second variation of ducted fan concept](image)
The main reasons why none of the rocket concepts were used are the following:

- The complexity of building the airframe and to achieve effective thrust vectoring
- The cost of two ducted fan units
- The problem of constructing the wings accurately

### 3.3 Development of the Final Airframe

It was decided to use a model aircraft and modify it to enable it to take off vertically. The reason for this is to save development time and cost. It was decided to tilt the whole wing with both engines attached to the wing. This would not only make the mechanics of the transition easier, but also enable the use of paddles fitted to the wing behind the engines to vector the thrust. Another reason for tilting the whole wing in stead of only the engines was because the wing would obstruct the airflow.

The development of the airframe can be divided into the following parts:

- Choosing a Model Aircraft
- Choosing a Power Plant
- Modifying the wings and mounting the power plants
- Mounting the wings to the fuselage

### 3.3.1 Choice of Model Aircraft

Several Remote Control Aircraft were considered, but the final choice was the Ultra Stick, as shown in Figure 3.5.
The main reasons for choosing an Ultra Stick is that it is quite light and has a split aileron (not shown in Figure 3.5) as an option. The whole aircraft is made of balsa wood and plywood, and that would make it easier to modify.

### 3.3.2 Choice of Power Plant

The initial plan was to use two methanol-powered engines, but there were several problems with this choice. Any motor swinging a propeller generates a moment and therefore it is necessary to swing the two propellers in opposite directions. The methanol engine used (OS91FX - Figure 3.6) was only manufactured to turn in one direction. It was necessary to modify the crankshaft of the engine to be able to turn in the opposite direction. The engine was tested after this modification, and it was noticed that it did not run reliably and had a lot of vibration.
This led to the decision to use an electric brushless motor. The advantages of using an electric brushless motor in stead of a methanol-powered engine are the following:

- It has less vibration
- It causes less noise
- It is clean (no emissions) and can therefore be used indoors
- It does not use methanol which is flammable
- The weight and balance of the aircraft does not change due to fuel usage

There are also a few disadvantages of using an electric brushless motor:

- It requires heavy batteries
- It has a much shorter flight time
- It has a long turnaround time due to the batteries which take a long time to charge
- It may cause electromagnetic interference due to the high currents

There are not many light and small electrical motors capable of generating about 6kg of thrust. The motor that was chosen is an AXI 5330/18 Brushless Outrunner Motor [19]. It has 2kW shaft output power and draws about 70A@40V at full power. A picture of the motor is shown in Figure 3.7.
Each motor is powered by supplied from a ten-cell LiPo battery pack with a capacity of 3200mAh. These were the only locally available LiPo batteries capable of supplying the required current.

### 3.3.3 Modifications to the wings

A standard Ultra Stick is not capable of having two strong engines mounted on the wings. A slot had to be cut in the wing for the motor to fit and the wing had to be strengthened to be able to handle the extra weight of the batteries and the motor. A carbon-fibre rod was used to connect the two wings to each other and also served as an attachment point for all the heavy additions like the battery and the motor. This carbon-fibre rod will also be used as the axle about which the wing can be tilted.

Thrust vectoring is required to control the aircraft during a hover. Figure 3.8 shows the vertical and horizontal paddles fitted to the wing.
The size of the servos that will move the paddles is calculated by determining the maximum torque that can be exerted by the paddle on the servo at full thrust. The following parameters are used:

- A maximum deflection paddle of \( \theta_{\text{max}} = \pm 45\degree \)
- A moment arm from servo to paddle of \( R = \frac{1}{2} \)
- Maximum thrust of \( F_{\text{max}} = 10\text{kg} \)
- Maximum efficiency of \( \eta = 25\% \)
- Total length of paddle \( L = 5\text{cm} \)
- Force applied on the paddle is in the centre of the paddle

Maximum torque exerted on the servo:

\[
T_{\text{max}} = \frac{F_{\text{max}} \times \sin \theta_{\text{max}} \times L \times \eta}{2} \times R
= \frac{10 \times \sin(\pm 45\degree) \times 5 \times 0.25}{2} \times 0.5
= \pm 2.210\text{kg.cm}
\]  (3.1)
The servo chosen for the job is a JR NES-331, capable of a maximum torque of 3.2\,kg\,cm, which is about twice the calculated absolute maximum rating. The calculated rating is a very optimistic figure and the actual torque will be much less.

The other servos used are according to the aircraft's specification.

The paddles behind the engines were mounted perpendicular to each other to decouple the lateral and longitudinal control of the aircraft, or in other words to have the horizontal paddles control forward movement and vertical paddles control sideways movement.

Figure 3.9 shows the underside of the wing with the brushless motor and battery pack clearly visible. The battery pack is the metal grey rectangle to the bottom left of the motor.

![Figure 3.9 – Underside of wing](image)

### 3.3.4 Mounting the wing to the fuselage

The wing pivots around the carbon-fibre rod and an aluminium gear was attached in the middle. A steel-reinforced belt is used to connect this gear to the gear on the servo which will rotate the wing. A box mounted to the fuselage was used to house all abovementioned components and to raise the wing to get sufficient ground clearance.
There are still some problems with the mechanics to rotate the wing, but it will only start to play a role during normal flight and not during a hover:

- The servo is not powerful enough to fully rotate the wing.
- There is no locking mechanism to keep the wing in position; therefore the servo will constantly draw current to keep the wing in place.
- There is play in the gears and belt.

See Figure 2.1 on page 27 for the finished airframe.

### 3.4 Modelling of the Airframe

This section describes the way the basic model of the airframe was obtained. This model does not include aerodynamics and is therefore only applicable while the aircraft is hovering. The model is obtained by using the standard six-degrees-of-freedom dynamic equations of motion, with the motors and paddle angles translated to forces and moments acting on a mass with inertia. Figure 3.10 shows the standard axes definition used in the modelling and simulation.

![Standard symbols and axes definition for equations of motion](image)

*Figure 3.10 – Standard symbols and axes definition for equations of motion [6]*
3.4.1 Thrust Vectoring

The aircraft has two engines with horizontal and vertical paddles behind it. The equations of motion use forces in the X, Y and Z directions. The mathematics described in Appendix A was developed by the Author and calculates the specified forces from given total thrust and the two paddle angles.

Given

\[
\begin{align*}
F & \quad \text{Total Force of Engine} \\
\phi_x & \quad \text{Deflection angle of horizontal paddle} \\
\phi_y & \quad \text{Deflection angle of vertical paddle} \\
\eta_x & \quad \text{Efficiency of horizontal paddle} \\
\eta_y & \quad \text{Efficiency of vertical paddle}
\end{align*}
\]

then the forces in X, Y and Z are

\[
\begin{align*}
F_x &= \frac{F \tan \phi_x}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}} \quad (3.2) \\
F_y &= \frac{F \tan \phi_y}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}} \quad (3.3) \\
F_z &= \frac{F}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}} \quad (3.4)
\end{align*}
\]

3.4.2 Moments of Inertia

The moments of inertia as seen in Table 3.1 were calculated with the torsion pendulum set up as used in [11] and [12]. All coordinates are referenced from the 3D middle of the carbon-fibre rod.
All measurements are with aircraft in VTOL configuration

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>d</th>
<th>T</th>
<th>m</th>
<th>g</th>
<th>Total Time</th>
<th>Total Osc</th>
<th>J</th>
<th>J from CG</th>
<th>J around CG</th>
<th>CG (0;0;0) is middle of rod</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll Inertia</td>
<td>3.1</td>
<td>0.5</td>
<td>2.9</td>
<td>1.474</td>
<td>9.8</td>
<td>47.1</td>
<td>16.0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.2</td>
<td>X</td>
</tr>
<tr>
<td>Pitch Inertia</td>
<td>1.4</td>
<td>0.2</td>
<td>1.5</td>
<td>1.474</td>
<td>9.8</td>
<td>46.3</td>
<td>30.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>Y</td>
</tr>
<tr>
<td>Yaw Inertia</td>
<td>3.1</td>
<td>0.5</td>
<td>3.3</td>
<td>1.474</td>
<td>9.8</td>
<td>42.6</td>
<td>13.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.3</td>
<td>Z</td>
</tr>
<tr>
<td>Roll Inertia</td>
<td>1.054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.3</td>
<td>0.1</td>
<td>X</td>
</tr>
<tr>
<td>Pitch Inertia</td>
<td>1.054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>Y</td>
</tr>
<tr>
<td>Yaw Inertia</td>
<td>1.054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.3</td>
<td>0.1</td>
<td>Z</td>
</tr>
<tr>
<td>Roll Inertia</td>
<td>1.054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.3</td>
<td>0.1</td>
<td>X</td>
</tr>
<tr>
<td>Pitch Inertia</td>
<td>1.054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>Y</td>
</tr>
<tr>
<td>Yaw Inertia</td>
<td>1.054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.3</td>
<td>0.1</td>
<td>Z</td>
</tr>
<tr>
<td>Roll Inertia</td>
<td>1.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.2</td>
<td>0.1</td>
<td>X</td>
</tr>
<tr>
<td>Pitch Inertia</td>
<td>1.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>Y</td>
</tr>
<tr>
<td>Yaw Inertia</td>
<td>1.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.2</td>
<td>0.1</td>
<td>Z</td>
</tr>
<tr>
<td>Roll Inertia</td>
<td>1.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.2</td>
<td>0.1</td>
<td>X</td>
</tr>
<tr>
<td>Pitch Inertia</td>
<td>1.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>Y</td>
</tr>
<tr>
<td>Yaw Inertia</td>
<td>1.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0</td>
<td>0.2</td>
<td>0.1</td>
<td>Z</td>
</tr>
<tr>
<td>Roll Inertia</td>
<td>2.2</td>
<td>0.3</td>
<td>1.5</td>
<td>1.385</td>
<td>9.8</td>
<td>44.6</td>
<td>30.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>X</td>
</tr>
<tr>
<td>Pitch Inertia</td>
<td>2.3</td>
<td>0.4</td>
<td>2.7</td>
<td>1.385</td>
<td>9.8</td>
<td>53.3</td>
<td>20.0</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>Y</td>
</tr>
<tr>
<td>Yaw Inertia</td>
<td>2.3</td>
<td>0.4</td>
<td>2.7</td>
<td>1.385</td>
<td>9.8</td>
<td>63.8</td>
<td>24.0</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>Z</td>
</tr>
</tbody>
</table>

**Total Aircraft Moment of Inertia about CG**

<table>
<thead>
<tr>
<th>Wing</th>
<th>Motor</th>
<th>Battery</th>
<th>Fuselage</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll Inertia</td>
<td>0.21</td>
<td>0.23</td>
<td>0.11</td>
<td>0.034</td>
</tr>
<tr>
<td>Pitch Inertia</td>
<td>0.02</td>
<td>0.03</td>
<td>0.01</td>
<td>0.208</td>
</tr>
<tr>
<td>Yaw Inertia</td>
<td>0.26</td>
<td>0.20</td>
<td>0.11</td>
<td>0.194</td>
</tr>
</tbody>
</table>

**Aircraft CG**

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0305</td>
<td>0.0000</td>
<td>0.0433</td>
</tr>
</tbody>
</table>

*Table 3.1 – Calculation of Aircraft Moments of Inertia and CG*
The parts listed in Table 3.1 (wing, motor, battery and fuselage) were weighed and measured before the whole aircraft was assembled.

3.5 Summary

The process of developing and building the airframe was discussed in this chapter. The basic equations and values for a basic model of the airframe were derived and calculated. The derivation for equations (3.2), (3.3) and (3.4) can be found in Appendix A.

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wing with CF rod</td>
<td>1.47 kg</td>
</tr>
<tr>
<td>Two motors and mounts</td>
<td>2.11 kg</td>
</tr>
<tr>
<td>Two Battery Packs</td>
<td>2.28 kg</td>
</tr>
<tr>
<td>Fuselage</td>
<td>1.39 kg</td>
</tr>
<tr>
<td>Avionics (est.)</td>
<td>1.00 kg</td>
</tr>
<tr>
<td><strong>Total Weight</strong></td>
<td><strong>8.25 kg</strong></td>
</tr>
</tbody>
</table>

Table 3.2 – Weight budget of the aircraft

The weight budget of the aircraft is shown in Table 3.2. The total thrust was measured to be around 11 kg, which gives a thrust to weight ratio of $1\frac{1}{3}:1$. 
Chapter 4 – Control Law Design and Simulation

This chapter describes how the simulation was developed and the control law derived. The control law was combined with the dynamic model of the airframe and the test results show that the airframe is controllable.

4.1 Building the Simulink Model

Simulink uses block diagrams to describe the simulation and enables a good systematic approach to building a model. For this project, the following steps were followed to build the model:

- The thrust vector equations, (3.2), (3.3) and (3.4) were implemented in a block diagram and simulated for verification.
- The moments that were generated by the abovementioned forces, were calculated and also put into a block diagram.
- The six-degrees-of-freedom block contained in the Aerospace Blockset was added and the forces, moments and moments of inertia (from Table 3.1) were implemented. Up to this point the simulation is complete and could be used for open-loop simulations.
- A block containing the control algorithms was added and the simulation completed.

Fanus Groenewald [10] investigated methods of having a visual output of the simulation in combination with the plots generated. The result of the investigation was to use a blockset called Aerosim. This blockset enables the user to use a normal flight simulator (like FlightGear or Microsoft Flight Simulator) to do all the 3D rendering. A block in Simulink communicates through a TCP/IP network with the flight simulator, informing it of the orientation and position of the aircraft. This is a very useful feature, because it

1 Please note that there are defined body and earth axes. The text in this section will assume the reader knows when to use body and earth axes. Also, this section only describes the Simulink model of the aircraft in a hover. For more information about these axis systems, see [8].
enables the user to observe the behaviour of the aircraft, instead of having to analyze plots.

The rest of this section will show how the different components were developed and simulated.

4.1.1 Modelling of Thrust Vectoring

Figure 4.1 shows the Simulink model used to test the Thrust Vectoring equations. This model is a direct representation of the equations calculated in section 3.4.1.

![Simulink model of Thrust Vectoring](image)

The simulation parameters for generating the plots in Figure 4.2 are as follows:

- Engine total thrust of 8kg
- Paddle efficiency of 25%
- Maximum paddle angle of ±45°

A sine wave is applied to both the vertical ($\phi_y$) and the horizontal ($\phi_x$) paddles, while the thrust of the engine is kept constant. A simple verification of the thrust vectoring
equations is to read the thrust from the plots and use Pythagoras's theorem to calculate the total force, which should be the same as the engine's thrust (8kg).

At T=1.5s:

- \( F_x = 1.52 \) kg
- \( F_y = 1.52 \) kg
- \( F_z = 7.71 \) kg

Therefore, the total force is

\[
F = \sqrt{F_x^2 + F_y^2 + F_z^2} = \sqrt{(1.52)^2 + (1.52)^2 + (7.71)^2} = 8 \text{ kg}
\] (4.1)

Figure 4.2 – Plots of the thrust vectoring simulation
4.1.2 Coordinate transformation system

The six-degrees-of-freedom dynamic equations coordinates are ECEF (North, East, Down) and the Aerosim blockset uses geodetic coordinates (Latitude, Longitude, Altitude). It is therefore necessary to transform the Cartesian coordinate system to the geodetic coordinate system. Since this is only for display purposes, the coordinate transform is only an approximation. The block that converts the coordinates is shown in Figure 4.3.

Referring to Figure 4.3:

- A starting point is specified in Geodetic Coordinates and the simulation will add an offset calculated from the X, Y and Z positions.
- The Latitude and Longitude angles are calculated by using small angle approximations.
- Earth Radius \( R = 6378137m \) [8].
- The division and multiplication are done according to equations (4.2) and (4.3).

![Figure 4.3 – Simulink block to transform Cartesian Coordinates to Geodetic](image)

The latitude approximation is done by using the small angle approximation of a sine function. This produces (4.2).

\[
\mu = \frac{X}{R}
\]  

(4.2)

where:
\( \lambda \)  
Latitude in degrees

\( X \)  
Position North in metres

\( R \)  
Equatorial earth radius in metres

Longitude is dependant on the latitude because the effective radius of the earth decreases as you approach the poles. (4.2) and (4.3) are therefore basically the same, except that the earth radius is scaled according to the latitude.

\[
\lambda = \frac{Y}{R \cos \mu} \tag{4.3}
\]

where:

\( \lambda \)  
Longitude in degrees

\( \mu \)  
Latitude in degrees

\( Y \)  
Position East in metres

\( R \)  
Equatorial earth radius in metres

Altitude is just the negative of the Z position, because an increase in height causes an increase in altitude, but a decrease in Z (see Figure 3.10).

### 4.1.3 Aircraft Dynamics

Referring to Figure 4.4:

- The blue block contains the equations to calculate all the forces and moments generated by the two engines and four paddles.
- The gravity force is added to the forces calculated in the blue block, and fed into the six-degrees-of-freedom dynamic equations block (orange).
- A big state vector contains all possible states.
- A test was added to stop the simulation as soon as the altitude reaches zero (aircraft hits the ground).
The gravity vector is calculated by multiplying the total aircraft mass by the Direction Cosine Matrix supplied by the six-degrees-of-freedom block.

### 4.2 Linear Model

#### 4.2.1 Acquiring the Linear Model

The limited amount of time forced the Author to use MATLAB to acquire the linear model of the airframe instead of calculating it by hand. The \texttt{linmod()} command in MATLAB can calculate the linear state-space model of a Simulink block. The block in Figure 4.4 describes the plant in control systems terms and was used to calculate the linear model, except that only the following states were used: 

\[ (X, Y, Z) (\Phi, \Theta, \Psi) (u, v, w) (p, q, r) \]

which are the position, orientation, velocity and angular rate vectors of the aircraft.

The linear model was acquired about the following trim condition:

- \( \phi_{x1} = 0^\circ \)
- \( \phi_{y1} = 0^\circ \)
- \( F_1 = \frac{\text{Aircraft Weight}}{2} \times (-g) \)
- \( \phi_{x2} = 0^\circ \)
- \( \phi_{y2} = 0^\circ \)
• \( F_2 = \frac{\text{Aircraft Weight}}{2} \times (-g) \)

Subscript 1 is the engine and paddles on the port side of the aircraft and subscript 2 is the starboard side.

The linear state-space model obtained by MATLAB can be found in Appendix C.

### 4.2.2 Verifying the Linear Model

The linear model was verified by comparing it with the non-linear model with various step commands on the different inputs as mentioned above. The Simulink block shown in Figure 4.5 was used to compare the linear model with its non-linear equivalent. The block named "Linmod" is only used to get the correct 12 states from the full 27 state vector.

![Simulink block used to verify the linear model](image)

Figure 4.5 – Simulink block used to verify the linear model

A series of different scenarios was set up to verify the linear model:

- **Test 1**: Put a common mode deflection on the horizontal paddles (\( \phi_1 \) and \( \phi_2 \))
- **Test 2**: Put a differential mode deflection on the horizontal paddles
- **Test 3**: Reduce the thrust of both the engines (decrease magnitude of \( F_1 \) and \( F_2 \))
• Test 4: Put a common mode deflection on the vertical paddles ($\phi_1$ and $\phi_2$)

• Test 5: Put a differential mode deflection on the vertical paddles – thrust dumping

The results for the five tests are shown in Figure 4.6, Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10 and are discussed later on. For the figures to follow, the results of the linear model are shown in the top two graphs and the results of the non-linear model below them.

![Figure 4.6 – Plots of Position and Orientation for Test 1](image-url)
Figure 4.7 – Plots of Position and Orientation for Test 2

Figure 4.8 – Plots of Position and Orientation for Test 3
Figure 4.9 – Plots of Position and Orientation for Test 4

Figure 4.10 – Plots of Position and Orientation for Test 5
**Test 1 – Common mode deflection of the horizontal paddles (Figure 4.6):**
There is a small difference between the linear Z-position and the non-linear Z-position. This originates from the linearization process of equation (A.9). When \( \cos \phi \) is linearized, it just becomes 1, and therefore the thrust lost due to vectoring is not modelled in the linear version. The other positions and angles match up.

**Test 2 – Differential mode deflection of the horizontal paddles (Figure 4.7):**
Again, like in the previous test, the Z-position is not the same for the linear and the non-linear simulations for the same reason as above and the other positions and angles match up.

**Test 3 – Reduction of thrust of both the engines (Figure 4.8):**
The linear and non-linear simulations compare well because thrust increase and decrease is linearized and is taken into account.

**Test 4 – Common mode deflection of the vertical paddles (Figure 4.9):**
Once again, the linear and non-linear simulations compare well except for the Z-position.

**Test 5 – Differential mode deflection of the vertical paddles (Figure 4.10):**
The Z-position does not compare for the same reason as in Test 1. The rest of the positions and angles are matched closely.

From all above Figures it can be seen that the linear model is very close to the non-linear model. These different tests also form the base from which the controllers will be designed.

**4.3 Control Law Design**

**4.3.1 Control Strategy**
The aircraft was designed in such a way that the actuators are coupled to certain motions of the aircraft (also see Figure 3.10):
• Differential mode use of horizontal paddles causes the aircraft to yaw (heading).
• Common mode use of horizontal paddles causes the aircraft to pitch and move forwards and backwards (longitudinal movement).
• Differential mode use of horizontal paddles causes thrust to be dumped, and in combination with common mode use of the engines, causes the aircraft to move up and down (vertical movement).
• Common mode use of horizontal paddles causes the aircraft to roll and move sideways (lateral movement).

The control strategy is based on the four kinds of movement listed above. Four separate controllers will be used to control the four kinds of movement of the aircraft and their control outputs added to each other in the correct way.

The block diagram of the proposed control strategy can be seen in Figure 4.11.

Figure 4.11 – Block diagram of Control Strategy
4.3.2 Decoupling the model

The model discussed in the previous section is a full state (contains all of the states) model and needs to be broken (decoupled) into smaller pieces, relevant to each one of the controllers. The smaller models are acquired by manipulating the complete model until the smaller models are arranged diagonally with zeros or small numbers on all sides.

The following states are needed in the controllers:

- For the lateral controller: $Y, V, \Phi, p$
- For the longitudinal controller: $X, U, \Theta, q$
- For the vertical controller: $Z, W$
- For the heading controller: $\Psi, r$

The decoupled models used to develop the controllers can be seen in Appendix C.

4.3.3 Development of Controllers

Due to the limited time that was available to develop the controllers, they were all done in MATLAB. As the aim of the controllers is only to show that it is possible to control the aircraft, it was assumed that all states are available for feedback. This meant that all the controllers are multi input, single output controllers. Conventional SISO tools can therefore not be used and it was decided to use LQR controllers. All controllers were developed with the `lqr()` function in MATLAB and the source code can be found in Appendix G.

The process of developing the controllers was similar for all four. The input and output states were weighed until an acceptable response was obtained. The controllers take a position and heading reference and a certain velocity can be commanded by applying a ramp to the position input.

The Simulink block used to do the control is shown in Figure 4.12. The differential and common mode signals are added together by the column of sum blocks.
4.4 Simulation

Figure 4.13 shows the complete Simulink block used for simulation.

A series of test were run to test the controllers:

- At T=0s, a slew rate limited 50m step command on X
- At T=15s, a slew rate limited 50m step command on Y
- At T=30s, a slew rate limited -20m step command on Z
• At T=40s, a slew rate limited 360° step command on Ψ

Figure 4.14 shows the position, Figure 4.15 shows the velocity and Figure 4.16 shows the orientation of the aircraft during the test. The four commands were issued at different times to be able to observe the cross-coupling. The controllers were designed for a north-facing aircraft, thus a change in direction can be catastrophic (see Appendix H).

As can be seen from the abovementioned figures, the controllers work quite well, but it must be remembered that the simulation is executed in perfect conditions: there is no sensor noise and all the states are available. The purpose of the simulation is to show that the actuators are able to control the aircraft.

![3D Position of the Aircraft](image)

*Figure 4.14 – Position of the aircraft during the test*
Figure 4.15 – 3D Velocity of the aircraft during the test

Figure 4.16 – Euler angles of the aircraft during the test
The cross-coupling between the paddle angle and vertical thrust can clearly be seen in Figure 4.14 and Figure 4.15. The aircraft loses a little altitude, but the vertical controller manoeuvres the aircraft back to the commanded altitude. The cross-coupling is mainly as a result of the total downward thrust that is reduced as the paddles deflect the airflow.

There are no axes transformations currently in the controllers to compensate for the situation when the aircraft does not face north. The lateral and longitudinal controllers will only work correctly if the aircraft faces north.

4.5 Summary

This chapter showed how the model of the aircraft was developed and verified. It was shown that the four controllers are able to stabilize the aircraft in 3D position and heading.

The specific controller gains can be found by executing the M-file in Appendix G.

Additional tests on the controllers are shown in Appendix H.
Chapter 5 – Avionics Development

This chapter describes the flexible, modular, lightweight avionics system and its sub units in detail. Two subunits were developed by the Author for this project and are described in detail. For the other subunits, only high-level descriptions are given. Refer to Figure 2.2 for the avionics system block diagram.

5.1 PC/104CAN Controller

The PC/104CAN Controller is the interface between the main computer and the CAN bus. It also controls the timing of all processes in the avionics. The power supplies for all the electronics are located on the same board. A block diagram of the PC/104CAN controller is shown in Figure 5.1. The large arrows indicate external interfaces.

![Figure 5.1 – Block Diagram of PC/104CAN Controller](image)

The main components of this board are:

- A 6.5V, 1A switch-mode regulator (PT5105) used for powering the sensors on the CAN bus.
• A 5V, 4A switch-mode regulator (TRACO TEN 20-1211) used for powering the PC/104CAN Controller, the main computer and the RF modem and GPS.

• A Microchip PIC18F6585 Microcontroller handling all intelligent functions, such as CAN bus and RS232 communication, temperature and voltage monitoring and control timing.

• An Altera EPM7064S CPLD handling the communication with the main computer via the ISA bus.

There are "off the shelf" PC/104CAN controllers available (like RTD ECAN527HR-1 [13]), but none are able to generate the precise timing needed for control purposes. Therefore it was decided to develop a relatively simple CAN Bus controller and add the necessary power supplies to power the main computer.

The schematics and PC-Board diagrams of the PC/104CAN Controller can be found in Appendix F. The pin out for the different connectors is described in Appendix E.

**5.1.1 Hardware**

The assembled PC/104CAN Controller is shown in Figure 5.2. Note the main components as mentioned above.
5.1.1.1 Power Supplies

Both power supplies are switch-mode. The main computer draws about 3A maximum at 5V, which amounts to 15W at 5V. If linear regulators were used, the power drawn from the battery would have been 3A at 12V, or 36W. The switch-mode regulator has an efficiency of 84% typical [22], therefore the main computer will draw about 18W from the battery, which is half that of a linear regulator. The Traco Power regulators are supported in the ESL and have a good track record.

The 6.5V regulator [23] is, as mentioned above, also a switch-mode regulator. This is one of the few 6.5V switch-mode regulators on the market and samples are readily available. The unusual voltage was chosen to enable the different nodes on the CAN bus to have local regulation from 6.5V down to 5V with a low dropout linear regulator. This ensures better filtered power supplies on the sensor boards.

5.1.1.2 Complex Programmable Logic Device (CPLD) Hardware

The purpose of the CPLD is to act as an interface between the microcontroller and the ISA bus, because the microcontroller is not fast enough to handle the ISA protocol.
The CPLD must be fast enough to handle the ISA protocol and have enough input/output pins for address lines, data lines, etc. The only manufacturer considered for the CPLD was Altera, because it is readily available and there is good support in the ESL. MAX+plus II was used for the development and simulation and it was found that an Altera EPM7064STC-10 has sufficient IO pins and enough programmable logic units to solve the problem. See section 5.1.2.1 for the implementation and simulation results.

See Figure F.3 for the specific pins used on the CPLD.

5.1.1.3 Microcontroller Hardware

At the heart of the CAN controller is a Microchip PIC18F6585 [20] running at 38.4MHz. The reason for using this PIC is that it has an integrated CAN module and only needs an external CAN interface to work. Furthermore, the ESL already has the needed compiler and hardware to program the PIC microcontroller. This specific PIC was chosen for its large number of IO ports, which allowed for a dedicated eight bit wide port to connect to the CPLD, even though there are analogue channels and a UART implemented. No bit-shifting of the data is therefore required. Before the CPLD was used, a prototype was designed and built to test the feasibility of connecting the Microcontroller directly to the ISA bus. It was found that the microcontroller took too much time to decode the ISA address and it was therefore not possible to use it.

The hardware and port assignments of the microcontroller can be seen in Figure 5.3. The lines on the right hand side go to the CPLD, except for the interrupt lines that go directly to the PC/104 (ISA) connector. All lines are one bit wide, except for the 8-bit data bus.
A 2.5V reference voltage [25] and simple resistor divisor network is used for monitoring the voltage sources. The reason for using an external voltage reference is to enable the microcontroller to measure the 5V bus voltage. If the 5V source was used as a reference, the microcontroller would not be able to monitor it.

The temperature sensor [26] together with the CPU fan is used to keep the temperature of the main computer within allowable limits. To save battery power, the CPU fan is turned off when the measured temperature is less than approximately 36°C, and turned on again when the temperature goes beyond 38°C. To simplify the design, the temperature sensor is connected directly to the analogue input of the microcontroller.

The main purpose of the microcontroller is to synchronize the sample time of all the sensors and actuators on the CAN bus, collect the data of the different CAN sensors, upload it to the PC via the ISA bus and send the calculated servo positions to the Servo Board. This whole process is then repeated every 20 milliseconds.

5.1.2 Software

5.1.2.1 Complex Programmable Logic Device (CPLD) Software

Figure 5.4 shows an overview of the logic on the Altera CPLD. See Table 5.1 for a description of the abbreviations used.
The CPLD can be divided into the following main parts:

- Storage Registers
- Protocol Handshake logic
- ISA address and Read/Write decoding logic
- PIC address and Read/Write decoding logic

Storage Registers

The CPLD is used as a dual-port RAM device to enable communication between the PC and the PIC. There are 4 bytes of RAM used for this purpose.
Referring to Figure 5.5:

- 0x3BC is used as data register from PC to PIC
- 0x3BD is used as a command register from PC to PIC and contains the read/write status bits
- 0x3E8 is used as data register from PIC to PC
- 0x3E9 is used as command register from PIC to PC

The four registers were implemented using VHDL and are detailed in Appendix G.1.1 on page 136.

The specific addresses of the registers can be changed in the address decode logic. Care must be taken not to use reserved addresses. Bit 1 of the address bus is not wired to the CPLD (see Table E.1 and Figure F.3) and therefore the four registers are aliased. The registers mentioned in Figure 5.5 are aliased at 0x3BE, 0x3BF, 0x3EA and 0x3EB.

**Protocol Handshake Logic**

The Protocol Handshake Logic can be seen in Figure 5.6. The handshake logic determines whether the data written by the PC was read by the PIC, and if the data written by the PIC was read by the PC. This is done by using two Read/Write bits in register 0x3BD. The bit will only be set after a completed write and only be reset after a completed read.
If register 0x3BD bit 7:

= 1 \rightarrow \text{PC wrote data into 0x3BC}
= 0 \rightarrow \text{PIC read data from 0x3BC}

If register 0x3BD bit 6:

= 1 \rightarrow \text{PIC wrote data into 0x3E8}
= 0 \rightarrow \text{PC read data from 0x3E8}

These two status bits are used to ensure the data byte that was written into a register (either 0x3BC or 0x3E8) is read before another byte is written into the same register.

The last bit of register 0x3BD is connected to an external interrupt pin on the Microcontroller to generate an interrupt every time a byte is written into register 0x3BC.

**ISA and PIC address and Read/Write decoding logic**

This is the part of the CPLD which determines which one of the registers will store the value on the data bus in a Write Cycle, and which of the registers will output to the data bus in a Read Cycle. See Appendix G.1.2 for the VHDL code. The specific values in the VHDL code was obtained by using the required address (for example 0x3BC), dropping
the 2\textsuperscript{nd} bit (0x3BC becomes 0x1DE) and adding the required values for AEN (address enable = 0 always), IOR (Read Cycle = 0 for read) and IOW (Write Cycle = 0 for write).

Figure 5.7 shows a typical ISA read/write and what happens when the PIC reads/writes. Note the value of 10 written by the PIC being read by ISA later on, and the value of 13 being written by ISA and read out by the PIC as well as ISA. See Table 5.1 for a legend of the symbols in Figure 5.7.

![Figure 5.7 – MAX+plus II simulation of Read/Write from ISA and PIC](image)

Multiplexers had to be used to prevent logic contingency on the outputs of the different registers. An 8x4 multiplexer was used for ISA reads to enable the PC to read from all four registers, while an 8x2 multiplexer was used for PIC reads. See Appendix G.1.4 and G.1.5 respectively for the VHDL code of the multiplexers.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Non-abbreviated Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEN</td>
<td>Address Enable</td>
<td>ISA line indicating a valid address is on the address bus</td>
</tr>
<tr>
<td>IOW</td>
<td>IO Write Cycle</td>
<td>ISA line that goes low with a write cycle</td>
</tr>
<tr>
<td>IOR</td>
<td>IO Read Cycle</td>
<td>ISA line that goes low with a read cycle</td>
</tr>
<tr>
<td>SA[8..0]</td>
<td>System Address</td>
<td>ISA address bus Note: Since SA1 is not used, this indicates the values of SA[9..2, 0]</td>
</tr>
<tr>
<td>SD[7..0]</td>
<td>System Data</td>
<td>8-bit ISA data bus</td>
</tr>
<tr>
<td>picwrite</td>
<td>PIC write</td>
<td>Write line connected to the microcontroller</td>
</tr>
<tr>
<td>picread</td>
<td>PIC read</td>
<td>Read line connected to the microcontroller</td>
</tr>
<tr>
<td>picaddr</td>
<td>PIC Address</td>
<td>Address line connected to the microcontroller</td>
</tr>
<tr>
<td>picdata[7..0]</td>
<td>PIC data</td>
<td>8-bit data bus connected to the microcontroller</td>
</tr>
</tbody>
</table>

Table 5.1 – Legend of the symbols used in Figure 5.7

5.1.2.2 Microcontroller Software

The flowchart for the main program loop is shown in Figure 5.8. Complete detail of the whole program can be found in Appendix G.

![Flowchart of main program loop](image-url)
The main program loop is an infinite loop. Most events are handled by interrupt service routines, not the main program loop.

The 18 series PIC microcontroller has two interrupt priorities, namely High Priority and Low Priority. The low priority interrupt can interrupt normal program functions, while the high priority interrupt can interrupt the low priority interrupt as well as normal program functions. This makes the high priority interrupt ideal for use with functions that need critical timing. Only two interrupts were chosen as high priority:

- Timer0 interrupt
- External Interrupt

The low priority interrupt was used for non-timing-critical housekeeping functions. The only housekeeping functions needed are voltage and temperature monitoring. The A/D Interrupt together with Timer2 interrupt was used for this and was set to the slowest possible interrupt rate, to keep housekeeping processing to a minimum.

See Figure 5.9 for the flowchart of the high priority ISR and Figure 5.10 for the flowchart of the low priority ISR.
Figure 5.9 – Flowchart of High Priority ISR

**Timer0 Interrupt**

Timer0 is used for generating the 50Hz synchronization pulse on which all of the control code is based. The only functions handled by Timer0 ISR are to reset the timer value for a 20ms period and send out the CAN sync packet. The Timer0 Interrupt was placed before the external interrupt to have a predictable delay, even if both interrupts happen at the same time. The latency from Timer0 overflow interrupt to updating Timer0 registers was measured in MPLAB with the simulator and taken into account when updating the Timer0 register.
**External Interrupt**

The external interrupt is connected to the protocol handshake logic in the CPLD. It handles the communication to and from the PC via the ISA bus. The CAN over ISA protocol is discussed in section 5.1.3. The command sent by the PC is read as detailed in Figure 5.7 and the appropriate action is taken for the command. By using the external interrupt as a high priority interrupt, a quick response from the microcontroller is guaranteed after the PC sent a command. The specific external interrupt source used (INT0) can only be used as a high priority interrupt.

**Timer2 Interrupt**

The only purpose of the Timer2 interrupt is to lower the rate at which the analogue channels are sampled. The timer itself is set up to generate interrupts at the slowest rate possible, about 120Hz. A simple counter system then lowers the rate at which the timer interrupt actually enables the A/D process. The A/D interrupt will be generated when the measurement is complete. With a counter value of 30, the rate is about 4Hz. This amounts to every channel being measured once every second, since there are 4 channels. It is not necessary to measure the analogue channels faster than 1Hz, since it is only used for informational purposes.
Figure 5.10 – Flowchart of Low Priority ISR

A/D Interrupt
The A/D ISR stores the most recent measurement and then sets the module up to measure the next channel. It also controls whether the fan is on or off, depending on the temperature measurement.

5.1.3 CAN over ISA Protocol
This protocol was developed to enable CAN packets to be sent via the ISA bus from the PC/104CAN controller to the PC. A protocol was needed, because there is a large difference in processing speed between the PC and the microcontroller, and the data
transfer speed needs to be regulated. This is done by using the protocol handshake bits to ensure a byte is read before a new byte is written in its place. The processes are exactly the same in both directions, except for the handshake bit being used. Figure 5.11 shows the write process and Figure 5.12 shows the read process.

Note the order in which data and commands are read and written. The command must always be read/written before the data, because the Protocol handshake Logic reacts to read/write cycles on data and not on commands. The Data Written and Data Read bits mentioned in Figure 5.11 and Figure 5.12 are the same bit, but are inverted (Data Read = 1, Data Written = 0).
The value in the Data Register when issuing a command is the number of data bytes to follow if there is data to be transferred. See Table 5.2 for the list of commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>PIC will echo with 0x01 in data</td>
<td>Loop-back test</td>
</tr>
<tr>
<td>0x2A</td>
<td>N/A</td>
<td>Request CAN data</td>
</tr>
<tr>
<td>0x24</td>
<td>40(+13\times\text{(Number of CAN packets to transmit)})</td>
<td>CAN data ready to Transmit</td>
</tr>
<tr>
<td>0x0F</td>
<td>None</td>
<td>Request Status</td>
</tr>
</tbody>
</table>

**Table 5.2 – Summary of CAN over ISA commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x42</td>
<td>Bytes of CAN data</td>
<td>CAN data ready to upload</td>
</tr>
<tr>
<td>0x41</td>
<td>None</td>
<td>CAN data transferring</td>
</tr>
</tbody>
</table>

0x01 This is only to check from the PC side if the PIC communication channel is working or not. The PC issues the command and the PIC replies in register 0x3E8 (see Figure 5.5) with 0x01.

0x42 The PIC issues this command when all CAN packets have been received and are ready to be transferred to the PC. The data register (0x3E8) will contain the total number of bytes that need to be transferred to the PC. No CAN data is transferred with this command.

0x41 It is not actually a command, but it tells the PC that CAN data is being transmitted. It is mainly used to verify that the PC and PIC are synchronized.

0x2A After the PIC issued the 0x42 command, the PC will reply with this command when ready. This will tell the PIC to transfer all the collected CAN data.

0x24 The PC will issue this command as soon as the data collected with the 0x2A command was processed and new actuator positions calculated. As seen in Table 5.2, the minimum amount of bytes being transferred is 40. This can be divided into five groups of 8 bytes each. The first group contains housekeeping data, and
the other four groups contain the sixteen 2-byte servo channels that go to the Servo Board. If there are any other CAN packets that need to be sent, they are appended to the 40 bytes.

0x0F The PC issues this command to get the status values from the PIC. The PIC will reply with 7 bytes. See Table 5.3 for the description of the packet. Byte 0 is sent first.

<table>
<thead>
<tr>
<th>Byte No</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>6.5V Measurement</td>
<td>Battery Voltage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte No</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>5V Measurement</td>
<td>Temperature (8bit)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte No</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Fan Status</td>
<td>10bit Temperature Measurement</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 – Format of the Status packet

CAN packets (for commands 0x24 and 0x2A) are always assembled as shown in Table 5.4, regardless of how many data bytes there actually are. The reason for doing this is to have less processing on the microcontroller, as it always transfers all its CAN registers directly. The PC program determines which data is useful and which is not by using the data length byte. The Extended Identifier (EID) of the CAN packets is not sent in sequence. Make sure to look at the PIC datasheet [20] for the order of the EID bytes. The EID registers are transferred directly.

<table>
<thead>
<tr>
<th>Byte</th>
<th>12 - 9</th>
<th>8</th>
<th>7 - 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>EID[28..0]</td>
<td>Data Length</td>
<td>Data Bytes [0..7]</td>
</tr>
</tbody>
</table>

Table 5.4 – Format of a CAN packet
5.2 Servo Board

The Servo Board is used to provide power to and control the servos. It also makes provision for manual control through a normal RC receiver. The block diagram of the Servo Board is shown in Figure 5.13.

The main components of the Servo Board are:

- A Power Trends PT6361 5V 10A [24] switch mode regulator to supply the servos with power from the main Lithium Polymer battery

- A Microchip dsPIC30F5011 [21] microcontroller to drive the servos, to measure RC servo positions and to switch the backup battery on in case of LiPo battery failure. The microcontroller also has to determine, depending on certain criteria, whether either the RC servo positions or PC servo positions will be sent to the servos.

- A number of discrete components to automatically power the microcontroller from the backup battery in case of CAN bus power failure.
• A relay controlled by the microcontroller to determine whether the servos are powered by the regulator or the backup RC battery.

5.2.1 Hardware

The assembled Servo Board is shown in Figure 5.14.

![Fully assembled Servo Board](image)

*Figure 5.14 – Fully assembled Servo Board*

5.2.1.1 UPS Circuit

It is critical that the microcontroller on the Servo Board does not stop working. A Backup Battery in conjunction with the UPS Circuit ensures that the processor always has power.

---

1 Photo taken from Fanus Groenewald’s thesis [10] with permission.
As can be seen from Figure 5.15, the UPS circuit is quite simple and consists of a few capacitors, two diodes and a low dropout linear regulator. The two diodes are Schottky diodes because they have a typical voltage drop of 0.2V instead of 0.7V for normal diodes. The diodes' function is to isolate the two power supplies (backup battery and regulator) from each other. There is unfortunately a problem with the current setup, namely that the microcontroller will be powered from the backup battery until the battery voltage drops to below 5V (the output voltage of the regulator). Since this is a prototype, it would not be a problem. A possible solution to this problem (see Figure 5.16) is to move the diodes to the input of the linear regulator, and so the battery voltage must be more than 6.5V for the battery to supply any current.

Figure 5.15 – UPS Circuit of the Servo Board

Figure 5.16 – Alternative UPS circuit
A more elegant solution would be to make use of a voltage monitor circuit which uses a MOSFET to connect and disconnect the battery, but it is more complicated and uses more components.

5.2.1.2 Microcontroller Hardware

The main functions of the microcontroller are to handle CAN communication, measure Remote Control Pilot servo commands and drive the servos connected to the system. Initially, two separate microcontrollers were used, but after trying to implement it, it was found that it is not a very stable solution. The problem is that the microcontroller measuring positions cannot be interrupted while the commands are being measured, and the other microcontroller cannot be interrupted while the servo positions are being updated. This proved to be a difficult problem to solve, and finally it was decided to use another microcontroller that can measure servo positions in hardware. The only microcontrollers found to have enough input capture units was the dsPIC series manufactured by Microchip. The smallest dsPIC with eight input capture units, dsPIC30F5011, was used. Figure 5.17 shows the implemented microcontroller setup.

![Figure 5.17 – dsPIC hardware and port configuration](image)

All eight RC receiver servo positions are connected to input capture hardware. This ensures that the microcontroller does not spend a lot of processing power on timing. It also ensures good repeatable measurements, because interrupt latencies do not play a role.
Six of the sixteen servo outputs are connected to PWM hardware on the microcontroller. The other ten PWM outputs are generated with two hardware timer interrupts. Although the latency on some of the output channels can be as much as 10ms (half a sample time), using interrupts has the big advantage of using very little processing power and has a very good resolution. Mission critical servos must therefore be connected to the output capture unit, because the latency of those six servo channels is zero.

See Appendix D for the detail of the PWM signal that goes to a servo.

5.2.1.3 Power Supply

The choice of a power supply was mainly influenced by the printed circuit board footprint size and amount of current it can supply. The PT6361C was chosen for its relative small footprint and large current capability (10A @ 5V). Also, the component is available as a sample. The power supply is adjustable, and was adjusted to have an output of 5.5V to increase the torque and speed of the servos, without the danger of damaging the servos.

The datasheet [24] recommends using low ESR capacitors on the input and output. It was not possible to get low ESR capacitors, so normal capacitors were placed in parallel to get a lower series resistance. Three capacitors were used on the input to get a total of 660μF and two on the output to get a total of 440μF. A ceramic capacitor (100nF) was also added to reduce the effect of the inductance of the electrolytic capacitors.

5.2.2 Software

The dsPIC has an excellent interrupt configuration (see [21]), and therefore most of the servo output functions are done by interrupt service routines.

5.2.2.1 Generating Servo Outputs

The following resources are used to generate servo outputs:

- Output Capture Module (6 channels)
- Timer2 Interrupt Service Routine (2 channels)
Timer4 Interrupt Service Routine (8 channels)

The Output Capture Module is used to generate 6 of the 16 servo outputs. There are 6 Output Compare registers that are compared with the value of Timer2. If the value of Timer2 is less than the value of the Output Compare Register, the associated output pin is set high. If it is more, the value of the pin is reset. The module is configured to shut down automatically after a cycle is completed, to ensure that all the output pins stay low while Timer2 is used to generate the PWM signal for the other two channels.

Figure 5.18 – Flowchart of Timer2 ISR State Machine
Timer2 generates two more PWM signals by having the timer generate an interrupt after all the output capture modules are shut down to start a basic state machine. This state machine will set and reset the appropriate output pins. See Figure 5.18 for the flow diagram of the state machine of Timer2.

Timer4 is used in the same way as Timer2, except that Timer4 is dedicated to drive a similar state machine as Timer2 and can therefore run in parallel to the output capture module and Timer2. See Figure 5.19 for the flow diagram of the state machine of Timer4.
The whole servo output process is started by doing the following:

- Configure and enable the Output Capture Modules
- Reset Timer2 and Timer2 state machine
- Reset Timer4 and Timer4 state machine
- Set up Timer4 period register for Servo6
- Set Servo6 output high
- Start Timer2 and Timer4

5.2.2.2 Measuring RC receiver servo positions

RC receiver servo positions are measured using Input Capture Modules. Figure 5.20 shows the flow diagram of the Input Capture ISR.

![Figure 5.20 – Flowchart of Input Capture ISR](image)

This ISR is the same for all 8 Input Capture channels, except for the specific registers used to access the data. The Input Capture process works in the following manner:

- It is configured to use a specific timer (in this case Timer3).
• It is configured to store the value of Timer3 and generate an interrupt on every edge (rising and falling).
• It will then store the value of Timer3 in hardware the instant an edge is detected. After that, the interrupt is generated and the service routine determines whether it was a rising or falling edge (by looking at the value of the input pin) and executes the required code.

It does not matter if there is a slight latency from the time the edge occurs until the ISR is executed, because the value of the Timer3 is stored as soon as the edge is detected. This results in a good repeatable measurement.

5.2.2.3 Other Interrupts Service Routines

Timer1 Interrupt
Timer1 is used to start the A/D sample process. Timer1 is set up to be quite slow, because it is not necessary to sample it at a high rate.

A/D Interrupt
This A/D Interrupt Service Routine is where all the housekeeping is done. The A/D module was configured to sample all the analogue channels (Backup Battery voltage and PT6361 voltage) in sequence and if the PT6361 voltage is too low, use the relay to switch to Backup Battery.

Timer5 Interrupt
The flow diagram for Timer5 ISR is shown in Figure 5.21.
Timer5 is used as a 50Hz time base if the CAN sync packet (see Appendix B.1) is absent. The actual period of Timer5 is a little more than 20ms to allow for normal latencies without timing out. It also serves as a test to see whether the CAN bus is working properly or not. Computer control of the servos is not possible when Timer5 is the time base for servo output. A successful CAN sync packet reception will reset Timer5 counter in the CAN Interrupt Service Routine.

**CAN Interrupt**

Figure 5.22 shows the flow diagram of the CAN Interrupt Service Routine.
The CAN interrupt service routine can be divided into two parts. The first part receives the CAN sync packet and the other part handles all other CAN communication (see Appendix B.1). The reason for this split is that the Microcontroller's CAN module has two CAN receive buffers which was set up appropriately.

The requirement for Autopilot Armed (see Figure 5.22) is a preset receiver channel within certain boundaries (tested by the Servo Board) and the autopilot armed status from the PC side (see Appendix B.1). Only if both are true the Servo Board will allow PC servo commands to go to the connected servos.
5.2.2.4 Mixing Algorithm

From Figure 5.17 it can be seen that there are eight servo channels coming into the system and 16 channels going out. This gives rise to the problem of what to do with the other eight channels. This was solved with a mixing algorithm which enables the user to specify exactly how the eight servo input channels from the receiver should be mapped to the sixteen servo outputs. The mixing algorithm can be described using (5.1). It is always executed except when the PC is in control of the position of the servos.

\[
\begin{bmatrix}
S_0 \\
S_1 \\
S_2 \\
\vdots \\
S_{15}
\end{bmatrix}
= \begin{bmatrix}
M_{0,0} & M_{0,1} & M_{0,2} & \cdots & M_{0,7} \\
M_{1,0} & M_{1,1} & M_{1,2} & \cdots & M_{1,7} \\
M_{2,0} & M_{2,1} & M_{2,2} & \cdots & M_{2,7} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
M_{15,0} & M_{15,1} & M_{15,2} & \cdots & M_{15,7}
\end{bmatrix}
\times
\begin{bmatrix}
R_0 \\
R_1 \\
R_2 \\
\vdots \\
R_{16}
\end{bmatrix}
\]

(5.1)

Where:

- **S**  Servo Position Output (16-element vector)
- **M**  Mixing Matrix values (16×8 Matrix)
- **R**  Receiver Servo Position (8-element vector)
5.3 CANsens board

The main purpose of the CANsens board is to convert sensor analogue values to digital CAN packets which can be received by the PC/104CAN Controller. It samples a total of 12 analogue channels of which eight are 16bit and four are 10bit. The eight 16bit channels are put through an analogue anti-aliasing filter before going to the A/D converter. All channels are sampled at a rate of 1kHz and is then sub sampled at a rate of 50Hz. A digital anti-aliasing filter is implemented on all channels to prevent aliasing when sub sampling. Figure 5.23 shows the block diagram of the CANsens Board.

![Block Diagram of CANsens Board](image)

The main components of the CANsens Board are:

- A PIC 18F458 Microcontroller
- ADS8344 eight channel, 16bit A/D converter
- Several analogue Anti Aliasing filters

Like the Servo Board, the CANsens Board also gets a supply voltage from the CAN bus connector.

---

1 This piece of hardware was developed by Fanus Groenewald. All information and pictures in this section were taken from his thesis [10] with permission.
Schematics and PCB layout diagrams can be found in Appendix F.3.

5.4 CANsensIMU Board\(^1\)

CANsensIMU is an analogue extension module for the CANsens Board. It consists of the following parts:

- 3×Rate Gyroscopes (ADXRS150) [30]
- 2×Two-Axis Accelerometers (ADXL202) [31]
- 1×Three-Axis Magnetometer (Honeywell HMC2003) [29]

Figure 5.25 shows the assembled CANsens Board with the IMU attached to it. The gyroscopes and accelerometers can be seen on the small boards mounted perpendicular on the top right hand side.

\(^1\) This board was developed by Johan Bijker, MSc. Eng. Student, ESL, to graduate in 2006.
5.4.1 Gyroscopes

The gyroscopes are used to measure angular rate. The ADXRS150 is capable of measuring a maximum range of 150º/s, which is sufficient considering the successful autonomous flight [11] with an effective range of 60º/s. Resolution is not lost as a result of not scaling the output of the gyroscopes according to [10].

![Fully assembled IMU](image)

Figure 5.25 – Fully assembled IMU

5.4.2 Accelerometers

Accelerometers are used to measure lateral accelerations. The ones used are capable of measuring a maximum of ±2g (±19.2m/s²) and with the 16bit A/D converter, no scaling is needed and no resolution is lost [10]. The choice of accelerometer can be motivated by the track record in the laboratory and the fact that it is affordable. It was used in a variety of other projects ([9], [10], [11]) in the laboratory.

---

1 Photo taken from Fanus Groenewald’s Thesis [10] with permission.
5.4.3 Magnetometer

The magnetometer is the single most expensive unit in the IMU. The Honeywell HMC2003 Magnetometer was chosen for its good track record in previous projects [9]. The magnetometer is required for absolute heading control, since GPS heading information is noisy at low speed and hover. The magnetometer is able to measure magnetic field strengths of up to ±2 gauss.

![Honeywell Magnetometer](image)

*Figure 5.26 – Honeywell Magnetometer*

5.5 GPS/Aerocomm Motherboard

The GPS/Aerocomm Motherboard houses the u-Blox RCB-LJ GPS receiver [32] and Aerocomm AC44861 RF modem [33] and has the following main components:

- MAX3232 drivers to translate the TTL levels of the RF modem and GPS receiver to RS232 levels.
- A 3.3V Linear Regulator to supply power.
- 2 connectors for the RF modem and GPS receiver.

The GPS receiver and Aerocomm RF modem connect to the PC/104 PC’s serial ports and the headers on the side of the board (see Figure 5.27) are wired so that a 10-wire ribbon cable connects them directly. A third 10-pin header connects to the second serial port of the GPS receiver to be able to use differential GPS functions when the need arises.

---

1 Photo taken from Fanus Groenewald’s Thesis [10] with permission
2 This piece of hardware was developed by Fanus Groenewald. All information and pictures in this section were taken from his thesis [10].
Figure 5.27 – Fully assembled GPS/Aerocomm Motherboard

See Appendix F.4 for the schematics and PCB layout.

5.5.1 RF Modem

This unit enables communication from the ground station to the OBC. It is an intelligent unit with error correction and which retries transmissions. This unit was chosen because it was used in another project [11] with success, has a high output power (500mW EIRP) and is relatively easy to use.

5.5.2 GPS Receiver

This specific GPS receiver was extensively tested and evaluated in [9] and was found to be a relatively good receiver at a very low cost. All the needed knowledge and software was available in the laboratory and no further searching was done to look for alternatives. Several attempts were made to enhance the accuracy of the receiver by using RTCM data supplied by the Department of Survey Services, but without success. The GPS is used for absolute position and a velocity vector (3D heading and speed).
5.6 Power Distribution

The avionics needs two separate power sources:

- A main battery to supply power to the computer, sensors and servos.
- A backup battery to power servos only in case of main battery failure.

The reason why it was decided to use two separate batteries is that the avionics are quite complex with a lot of power supplies. This increases the chances of a failure and the main battery is a crucial single point of failure.

Figure 5.28 shows the power distribution diagram. Only the power supplies connected directly to the Main Battery will be used for calculations.

5.6.1 Choice of Batteries

All the power supplies connected directly to the main battery are switch mode types. This means that the current drawn from the battery is less than the current supplied by the
power supply. All calculations are based on the maximum current drawn from the main battery.

There are three power supplies connected to the battery:

- 5V@4A, 84% typical efficiency
- 6.5V@1A, 80% typical efficiency
- 5.5V@10A, 90% typical efficiency

The absolute maximum power that can be consumed by the avionics is therefore:

\[
P = \frac{5 \times 4}{0.84} + \frac{6.5 \times 1}{0.80} + \frac{5.5 \times 10}{0.9} = 93\text{W}
\]

It was decided to go with LiPo batteries for their high power density (low weight and high power). The supplier close to the laboratory has Kokam batteries and it was decided to use the 2000mAH 3S1P battery [35]. It can supply a maximum constant current of 30A, which amounts to a maximum of about 300W at the minimum cell voltage of 10V. The system can be powered for 20 minutes at maximum power, which would be sufficient for one or two flights. The system is not supposed to draw maximum power under normal conditions and the actual flight time will be much more than 20 minutes. The maximum current drawn from the battery was measured to be about 3A on the bench with all systems connected.

The backup battery is an 1100mAH NiCad RC flight battery.

5.6.2 Care and Maintenance of Batteries

The great advantage of LiPo batteries is their power density, but this comes at a price. Care MUST always be taken not to exceed the maximum and minimum allowable voltage and current of the battery. If this is not followed, the battery may ignite and destroy the whole aircraft. The safe minimum allowable voltage of a LiPo cell is about

---

1 This section is based on the Author's experience.
3V per cell under heavy load, but this increases to about 3.3V per cell as the load is decreased due to the lower voltage drop over internal resistance.

The maximum voltage per cell is reached when the battery is charged and must not be more than 4.2V. It is advisable to use a special LiPo charger for the purpose. The charge current must not be more than the capacity of the battery, for example, a 2000mAh battery should be charged at a maximum current of 2A.

LiPo batteries do not have memory, but will slowly lose their capacity as they reach their cycle limit.

**5.7 Summary**

This chapter described the detailed design of all the major components of the avionics. The Author developed the PC/104CAN controller and Servo Board subunits and presented the designs in detail. The other hardware was, as mentioned, developed by other individuals and brief technical overviews were given of those subunits. Please refer to the referenced documents for more detail.

All the schematics and PCB layouts of the hardware discussed in this chapter can be found in Appendix F.

Program listings not found in Appendix G are located on the accompanying CD.
Chapter 6 – Results Obtained and Conclusion

6.1 Results Obtained

An experimental VTOL airframe was developed by adding a tilt-wing mechanism to an "off-the-shelf" model aircraft. The simulations show that it should be capable of vertical takeoff and landing. The necessary avionics for autonomous flight was developed and tested. Although one of the initial goals was to have a flying model, the emphasis of this project was shifted to developing a solid basis for future work.

The new experimental airframe was built and the process documented. It was also modelled and evaluated using simulations to show the advantages and shortcomings. It would have been ideal to have a hardware-in-the-loop simulation and although the limited time available prevented this from happening, the necessary interfaces were incorporated in the avionics to facilitate this. The airframe will not be able to withstand a hard landing; therefore emphasis on simulation and testing before actual flight is crucial.

A lightweight, flexible and modular flight control system was developed in cooperation with others and tested in a methanol-powered RC helicopter [10]. It can easily be adapted to be more powerful and carry a heavier sensor load. A variety of new and "off-the-shelf" hardware can easily be incorporated into the system. The high work load did not allow for building of a second avionics set for use with this project.

Although the two year time constraint for the Masters project did not leave enough room for hardware-in-the-loop simulation and flight tests, two more Masters Students will continue towards the broader project goals of autonomous flight during 2006 and 2007.
The following is a summary of this project's contributions to the aero group:

- A new airframe was built, a basic Simulink dynamics model was created and controllers were implemented to show that the airframe is controllable with the given actuators. This Simulink dynamics model was created in such a way that it can be easily updated with better models and controllers.

- During the development of the Servo Board, a broad study was done to better understand the applications of RC servos. A presentation was given on the subject at the aero group's fortnightly seminar.

- The Author worked with Fanus Groenewald to develop a modular and robust flight control system. The lithium polymer battery used to power the avionics is also a first in the aero group. All previous projects used heavier NiCad and NiMH battery packs.

- A list of contacts from which RC equipment can be sourced was accumulated during the duration of this Masters project.

### 6.2 Recommendations

Recommendations for future work and improvements to this project are listed below:

- The linearized model of the airframe does not take loss of thrust into account. An investigation must be done into adding forward compensation for the altitude controller by increasing the thrust to compensate for loss resulting from increasing the paddle angles.

- The current model of the airframe during hover does not take aerodynamic forces into account. This does not present a problem when the aircraft is hovering, but will most certainly play a role in both transition and normal flight.

- A series of tests was done to try to improve the accuracy of the GPS measurements using differential GPS correction data, but without success. It
would be of great value to have a differential GPS system in place, especially when it can use the existing GPS receivers. The improved accuracy would make hovering of an aircraft easier.

- Vision-based navigation is a new, but not widely used technology. The avionics developed in this project can interface with a PC/104 frame grabber and there are some algorithms available from previous projects to use. The integration of vision-based navigation therefore becomes possible.

### 6.3 Concluding Remarks

This project provides a solid basis for many future flight and robotics projects. The flexibility and modularity of the avionics makes it suitable for easy incorporation into future projects.

All the documents needed to duplicate this project are included on the accompanying compact disk.

A friendly word of advice to the people working with the airframe: Be careful with it – please realize the amount of work that went into the development and building of the airframe. It will require a minimum of 6-9 months to replace, if it crashed. A drop from 1m will be damaging; a drop from 2m will be catastrophic.
References


**Theses:**


**Web Resources:**


Datasheets:

[20] Microchip PIC18F6585
[21] Microchip dsPIC30F5011
[22] TRACO TEN 20-1211
[23] Texas Instruments PT5105
[24] Texas Instruments PT6361
[25] MAX6008A
[26] Maxim MAX6613
[27] Altera ByteBlaster
[28] PC/104 Specification
[29] Honeywell HMC2003 Magnetometer
[30] ADXRS150 Gyroscope
[31] ADXL202 Accelerometer
[32] u-Blox GPS
[33] Aerocomm AC44861 RF modem
[34] MAX3232
[35] Kokam LiPo battery
[36] Kontron MOPS LCD7
Appendix A – Thrust Vector Equations

We start with the axis transformation (spherical to Cartesian) equations [4].

\[
\begin{align*}
F_x &= F \sin \theta \cos \phi \\
F_y &= F \sin \theta \sin \phi \\
F_z &= F \cos \theta
\end{align*}
\]  \hspace{1cm} (A.1)

From, (A.1) calculate the tan of the paddle angles to get equations (A.2) and (A.3):

\[
\tan \phi_y = \frac{F_y}{F_z} \\
= \frac{F \sin \theta \sin \phi}{F \cos \theta} = \tan \theta \sin \phi  \hspace{1cm} (A.2)
\]

\[
\tan \phi_x = \frac{F_x}{F_z} \\
= \frac{F \sin \theta \cos \phi}{F \cos \theta} = \tan \theta \cos \phi  \hspace{1cm} (A.3)
\]

Now, (A.2) ÷ (A.3) yields (A.4), a solution of \( \phi \) in terms of \( \phi_x \) and \( \phi_y \):

\[
\frac{\tan \phi_y}{\tan \phi_x} = \tan \phi \\
\phi = \arctan \left( \frac{\tan \phi_y}{\tan \phi_x} \right) \hspace{1cm} (A.4)
\]

From Pythagoras, basic triangular identities and (A.1), (A.5) is derived. It can be seen that the terms under the square root in (A.5) is the answers of (A.2) and (A.3) and when substituted, yields (A.6), which is a solution for \( \theta \) in terms of \( \phi_x \) and \( \phi_y \).
\[
\tan \theta = \frac{F_x^2 + F_y^2}{F_z} = \frac{\sqrt{F (\sin \theta \cos \phi)^2 + F (\sin \theta \sin \phi)^2}}{F \cos \theta} \quad (A.5)
\]

\[
\tan \theta = \sqrt{(\tan \phi_x)^2 + (\tan \phi_y)^2} \\
\theta = \arctan \left( \sqrt{(\tan \phi_x)^2 + (\tan \phi_y)^2} \right) \quad (A.6)
\]

Finally, the answers of (A.5) and (A.6) is substituted into (A.1)) and simplified to reveal the 3D components of a force being deflected by the two angles \( \phi_x \) and \( \phi_y \).

\[
F_x = F \sin \theta \cos \phi \\
= F \sin \left( \arctan \left( \sqrt{(\tan \phi_x)^2 + (\tan \phi_y)^2} \right) \right) \cos \left( \arctan \left( \frac{\tan \phi_x}{\tan \phi_y} \right) \right) \\
= F \frac{\sqrt{(\tan \phi_x)^2 + (\tan \phi_y)^2}}{\sqrt{1 + (\tan \phi_y)^2 + (\tan \phi_y)^2}} \left( \frac{\tan \phi_x}{\sqrt{(\tan \phi_y)^2 + (\tan \phi_y)^2}} \right) \quad (A.7)
\]

\[
= \frac{F \tan \phi_x}{\sqrt{1 + (\tan \phi_y)^2 + (\tan \phi_y)^2}}
\]
\[ F_y = F \sin \theta \sin \phi \]
\[ = F \sin \left( \arctan \left( \sqrt{\frac{(\tan \phi_x)^2 + (\tan \phi_y)^2}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}}} \right) \sin \arctan \left( \frac{\tan \phi_x}{\tan \phi_y} \right) \right) \]
\[ = F \frac{\sqrt{(\tan \phi_x)^2 + (\tan \phi_y)^2}}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}} \left( \frac{\tan \phi_y}{\sqrt{(\tan \phi_x)^2 + (\tan \phi_y)^2}} \right) \]  
\[ = \frac{F \tan \phi_y}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}} \]  
\[ (A.8) \]

\[ F_z = F \cos \theta \]
\[ = F \cos \left( \arctan \left( \sqrt{\frac{(\tan \phi_x)^2 + (\tan \phi_y)^2}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}}} \right) \right) \]  
\[ = \frac{F}{\sqrt{1 + (\tan \phi_x)^2 + (\tan \phi_y)^2}} \]  
\[ (A.9) \]

(A.7), (A.8) and (A.9) are the equations used in the simulation of the airframe.
Appendix B – Protocols

B.1 CAN protocol

B.1.1 General Description

The protocol uses and extended, 29 bit, CAN 2.0B identifier. The format of the packet is listed below.

<table>
<thead>
<tr>
<th>CAN Field</th>
<th>Field Name</th>
<th>No. Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN Identifier</td>
<td>Main Type ID</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Request/Reply</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Subtype</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Source Address</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Destination Address</td>
<td>8</td>
</tr>
<tr>
<td>Data Bytes</td>
<td>Data Byte 0</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Data Byte 1</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Data Byte 2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Data Byte 3</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Data Byte 4</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Data Byte 5</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Data Byte 6</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Data Byte 7</td>
<td>8</td>
</tr>
</tbody>
</table>

Table B.1 – CAN packet Layout

1 All information in this section adapted from [10]
B.1.2 Node Particulars

The table below contains the broad outline of the implemented CAN protocol. All nodes, except the Master node will send one packet with FID = 0x1400 when power is applied. The packet will contain zero data bytes and will be addressed to the Master node.

<table>
<thead>
<tr>
<th>Message Type</th>
<th>FID</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time synchronization (SYNC)</td>
<td>0x0101</td>
<td>CANsens will store last measurement and send it immediately. Output nodes will write last values.</td>
</tr>
<tr>
<td>Real-time Control</td>
<td>0x02--</td>
<td>Request (Master to Servo)</td>
</tr>
<tr>
<td></td>
<td>0x03--</td>
<td>Response (Servo to Master)</td>
</tr>
<tr>
<td>CAN Telemetry (TLM)</td>
<td>0x0A--</td>
<td>Request (usually when data was lost after SYNC)</td>
</tr>
<tr>
<td></td>
<td>0x0B--</td>
<td>Response (to Master node)</td>
</tr>
<tr>
<td>Application specific</td>
<td>0x14--</td>
<td>Boot notification to Master</td>
</tr>
</tbody>
</table>

*Table B.2 – Broad outline of CAN protocol*

B.1.2.1 PC/104CAN Controller

- Address = 0x01
- SYNC packet ID is 0x010101FF
- Send SYNC every 20ms to address ALL (0xFF)
- Capable of sending any CAN packet from the OBC
B.1.2.2 CANsens IMU

- Address = 0x02
- Will reply (TLM) with last measurement(s) if SYNC is received from Master node only
- Subtype 0x80 contains 16 bit channel 0 to 3 (ID = 0x0B800201)
- Subtype 0x81 contains 16 bit channel 4 to 7
- Subtype 0x82 contains 10 bit channels (left aligned)
  - 16 bits per channel without ultrasonic sensor
  - 12 bits per channel with ultrasonic sensor:

<table>
<thead>
<tr>
<th>Byte</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Lo0</td>
<td>Hi0</td>
<td>Hi1</td>
<td>Lo1</td>
<td>Lo2</td>
<td>Hi2</td>
<td>Hi3</td>
<td>Lo3</td>
</tr>
</tbody>
</table>

- A packet received with ID = 0x0B8F0102 will (D0 is first data byte of packet):
  - D0 = 0x10: Set magnetometer
  - D0 = 0x20: Reset magnetometer
  - D0 = 0x30: Rate Gyro & Accelerometer self-test mode
  - D0 = 0x40: Rate Gyro & Accelerometer normal mode
  - D0 = 0x50: System reset.
  - D0 = 0x60: Ultrasonic sensor enabled
  - D0 = 0x70: Ultrasonic sensor disabled

B.1.2.3 Servo Board

- Address = 0x08
- Main type 0x02, subtype 0x80 to 0x83 used to set 16 servo channels. Data starts at channel 0, high byte.
- Main type 0x0A is used to request current servo positions commanded from the PC.
- When 0x0101 is received, D0 is checked for autopilot status. PC commands or Pilots commands are written to servos, depending on pilot master switch.

B.2 Ground Station Protocol

The ground station protocol was adopted from a previous project and the detail can be found in [11].
Appendix C – Linear Model of Aircraft

C.1 Non-decoupled state-space Model

\[
\begin{align*}
\bar{X} &= \begin{bmatrix} X \\ Y \\ Z \\ \phi \\ \theta \\ \psi \\ u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \\
\bar{U} &= \begin{bmatrix} \phi_x \\ \phi_y \\ \phi_z \\ F_1 \\ F_2 \end{bmatrix} \\
\bar{B} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.9810 & 0 & 0 & -0.9810 & 0 & 0 \\ 0 & -0.9810 & 0 & 0 & -0.9810 & 0 \\ 0 & 0 & 0.1380 & 0 & 0 & 0.138 \\ -3.0086 & 0 & 0 & -3.0086 & 0 & 0 \\ -3.2749 & 0 & 0 & -3.2749 & 0 & 0 \end{bmatrix}
\end{align*}
\]

\[
\bar{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\]
C.2 Decoupled Models

C.2.1 Lateral Model

\[ \begin{align*}
\vec{X}_{lat} &= \begin{bmatrix} Y \\ v \\ \Phi \\ p \end{bmatrix} \\
\vec{U}_{lat} &= \phi_1 = \phi_2
\end{align*} \]

\[ \begin{align*}
\vec{A}_{lat} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
\vec{B}_{lat} &= \begin{bmatrix} 0 \\ -1.962 \\ 0 \\ 3.0801 \end{bmatrix} \\
\vec{C}_{lat} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\vec{D}_{lat} &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\end{align*} \]

C.2.2 Longitudinal Model

\[ \begin{align*}
\vec{X}_{long} &= \begin{bmatrix} X \\ u \\ \Theta \\ q \end{bmatrix} \\
\vec{U}_{long} &= \phi_{x1} = \phi_{x2}
\end{align*} \]
\[ \bar{A}_{\text{long}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \bar{B}_{\text{long}} = \begin{bmatrix} 0 \\ -1.962 \\ 0 \\ -6.8473 \end{bmatrix}, \quad \bar{C}_{\text{long}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \bar{D}_{\text{long}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]

C.2.3 Vertical Model

\[ \bar{X}_{\text{vert}} = \begin{bmatrix} \bar{Z}^T \\ \bar{w} \end{bmatrix}, \quad \bar{U}_{\text{vert}} = F_1 = F_2 \]

\[ \bar{A}_{\text{vert}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \bar{B}_{\text{vert}} = \begin{bmatrix} 0 \\ 0.2425 \end{bmatrix}, \quad \bar{C}_{\text{vert}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \bar{D}_{\text{vert}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

C.2.4 Heading Model

\[ \bar{X}_{\text{head}} = \begin{bmatrix} \Psi \\ \bar{r} \end{bmatrix}, \quad \bar{U}_{\text{head}} = \phi_{x1} = -\phi_{x2} \]

\[ \bar{A}_{\text{head}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \bar{B}_{\text{head}} = \begin{bmatrix} 0 \\ -7.4534 \end{bmatrix}, \quad \bar{C}_{\text{head}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \bar{D}_{\text{head}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]
Appendix D – RC Servo basics

D.1 Introduction

Figure D.1 shows a typical RC servo.

A typical RC servo has the following main components:

- A DC electric motor
- A gearbox to increase the torque of the motor, but decrease the speed
- Position Sensor, normally a potentiometer
- Electronics containing the controller and motor driver.

---

1 All information was gathered from the internet. See sources [14], [15], [16], [17] and [18].
A normal RC servo has a self-contained closed-loop control system and it only takes an angle reference command and will turn to the specified angle. From a control systems side, it is a set and forget device. The servo will do try to get to the specified angle as quickly as possible.

**D.2 Different Types of Servos**

There are a large variety of RC servos available on the market. The most important specifications when choosing a servo is listed below.

**Manufacturers:**
- Futaba
- Japan Radio (JR)
- Hitec
- Sanwa
- Airtronics

**Torque:**
- Specified in either kg.cm or foot.pounds
- Torque ranges from 1kg.cm up to 30kg.cm, dependant on the size and type

**Turn rate/Slew Rate:**
- Usually specified in seconds per 60 degrees
- Average servo is about 0.2 seconds per 60 degrees

**Size/Type:**
- Jumbo
- Normal
- Mini
- Micro
- Pico
- Wing
- Retract
D.3 Driving a Servo

A servo used three wires to connect to the system:

- A common return wire (usually brown/black)
- A power wire (usually red)
- A signal/command wire (usually white/orange/yellow)

The servo connectors for Japan Radio and Futaba are shown in Figure D.2.

The signal wire is used to give an angle command to the servo. The signal is a PWM signal repeated at a rate of 50Hz. A positive pulse width of 1.5ms commands a zero angle. If the pulse width is decreased to 1ms, the commanded angle is -45° and if increased to 2ms, the commanded angle is +45°. Figure D.3 shows how the PWM signal should look like if an angle of +20° is needed. The actual command is in the pulse width of the PWM signal, and not in the duty cycle. The frequency can vary from 40Hz to 60Hz, depending on the manufacturer of the servo.

Figure D.3 – Example of a +20° angle command signal
The mechanical limits of a servo is normally more than 90 degrees, and if a pulse width of less than 1ms or more than 2ms is generated, the servo will move more than 90 degrees. Care must be taken not to drive the servo beyond the mechanical limits, because this causes the electric motor and gears to be under constant strain and may cause failure.

**D.4 Power Requirements**

This is a much debated aspect of servos. The steady state current requirements of a servo depend on the amount to torque needed to keep the servo in position. A thumb suck rule is to budget a maximum current of 1A per normal 5kg.cm servo, but this may be reduced as the number of servos is increased, because a servo in position draws less current than a servo that is moving.

A typical voltage of 5V is used for powering servos, but it can be increased to 5.5V or 6V to increase the speed and torque, but the current will go up as the voltage goes up.
Appendix E – Connector Pin Outs

E.1 CAN Connector

![CAN Connector Pin Assignments](image)

Figure E.1 – Pin assignments of CAN connector (Top View)

E.2 RS232 Connector

![RS232 Connector Pin Assignments](image)

Figure E.2 – Pin assignments of RS232 connector (Top View)
E.3 Altera JTAG Connector

Figure E.3 – Altera JTAG Programming Header [27] (Top View)

E.4 Microchip ICD programming header

Figure E.4 – Microchip ICD programming header (Top View)
### E.5 PC/104 Header

<table>
<thead>
<tr>
<th>Pin</th>
<th>Row A</th>
<th>Row B</th>
<th>Pin</th>
<th>Row A</th>
<th>Row B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/IOCHK</td>
<td>GND</td>
<td>17</td>
<td>SA14</td>
<td>/DACK1</td>
</tr>
<tr>
<td>2</td>
<td>SD7*</td>
<td>RESET</td>
<td>18</td>
<td>SA13</td>
<td>DRQ1</td>
</tr>
<tr>
<td>3</td>
<td>SD6*</td>
<td>+5V</td>
<td>19</td>
<td>SA12</td>
<td>/REFRESH</td>
</tr>
<tr>
<td>4</td>
<td>SD5*</td>
<td>IRQ9</td>
<td>20</td>
<td>SA11</td>
<td>BCLK</td>
</tr>
<tr>
<td>5</td>
<td>SD4*</td>
<td>-5V</td>
<td>21</td>
<td>SA10</td>
<td>IRQ7*</td>
</tr>
<tr>
<td>6</td>
<td>SD3*</td>
<td>DRQ2</td>
<td>22</td>
<td>SA9*</td>
<td>IRQ6</td>
</tr>
<tr>
<td>7</td>
<td>SD2*</td>
<td>-12V</td>
<td>23</td>
<td>SA8*</td>
<td>IRQ5*</td>
</tr>
<tr>
<td>8</td>
<td>SD1*</td>
<td>/SRDY</td>
<td>24</td>
<td>SA7*</td>
<td>IRQ4</td>
</tr>
<tr>
<td>9</td>
<td>SD0*</td>
<td>+12V</td>
<td>25</td>
<td>SA6*</td>
<td>IRQ3</td>
</tr>
<tr>
<td>10</td>
<td>IOCHRDY</td>
<td>KEY</td>
<td>26</td>
<td>SA5*</td>
<td>/DACK2</td>
</tr>
<tr>
<td>11</td>
<td>AEN*</td>
<td>/SMSMW</td>
<td>27</td>
<td>SA4*</td>
<td>TC</td>
</tr>
<tr>
<td>12</td>
<td>SA19</td>
<td>/SMEMR</td>
<td>28</td>
<td>SA3*</td>
<td>BALE</td>
</tr>
<tr>
<td>13</td>
<td>SA18</td>
<td>/OIW*</td>
<td>29</td>
<td>SA2*</td>
<td>+5V</td>
</tr>
<tr>
<td>14</td>
<td>SA17</td>
<td>/IOR*</td>
<td>30</td>
<td>SA1</td>
<td>OSC</td>
</tr>
<tr>
<td>15</td>
<td>SA16</td>
<td>/DACK3</td>
<td>31</td>
<td>SA0*</td>
<td>GND</td>
</tr>
<tr>
<td>16</td>
<td>SA15</td>
<td>DRQ3</td>
<td>32</td>
<td>GND</td>
<td>GND</td>
</tr>
</tbody>
</table>

* Connected to Altera CPLD

* Connected to Microcontroller

Table E.1 – Pin out of PC104 Header [28]
Appendix F – PC-board Layout and Schematics

F.1 PC/104CAN Board

F.1.1 Schematics

Figure F.1 – PC/104CAN Schematic Overview
Figure F.2 – Schematic of PC/104CAN Microcontroller

Figure F.3 – Schematic of PC/104CAN Altera CPLD
Figure F.4 – Schematic of PC/104CAN Power supply
F.1.2 Printed Circuit Board Layout

Figure F.5 – Schematic of PC/104CAN Fan driver

Figure F.6 – Top side of PC/104CAN PCB

Figure F.7 – Bottom side of PC/104CAN PCB
Figure F.8 – Top Overlay of PC/104CAN PCB

Figure F.9 – Bottom Overlay of PC/104CAN PCB
F.1.3 Bill of Materials

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Value</th>
<th>Designator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10MHz Y1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MAX232D U2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>220nF C1</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>100nF C4, C5, C6, C7, CCAN2, CCAN3, CCPU2, CSERVO2, CU1, CU2, CU3, CU5A, CU5B, CU5C, CU5D, CU6A, CU6B, CU6C, CU6D</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22pF C2, C3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>220μF 25V C9, C10, C12, C13</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22μF 16V CCAN1, CCAN4, CCPU1, CSERVO1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PIC18F6585 U5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>LEDS ON J2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Header 3 IFAN</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CANCON J4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Header 5X2 JTAG</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>RS232 J5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ICD CONN J1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PC104ROWA ROWB</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>EPM7064STC44-10 U6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MAX6613 U3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2N3904 Q2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2N3906 Q1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PT5105A U8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MAX6008AESU U4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PUSHBUTTON SW1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0R R15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>120R R14</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>560R R16</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3K3 R11, R24</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18K R1, R2, R3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10K R5, R17, R22, R25</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1K R18, R19, R20, R21, R23</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2K2 R4, R6, R7, R8, R9, R10, R12, R13</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CPUPOWER J6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SERVOPower J7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TERMINAL BLOCK J3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SN65HVD251 U1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TEN 20-1211 U7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LED3 LED1, LED2, LED3, LED4, LEDPWR</td>
<td></td>
</tr>
</tbody>
</table>

Table F.1 – Bill of Materials of PC/104 CAN PC-Board
F.2  Servo Board

F.2.1  Schematics

Figure F.10 – Schematic of Servo Board

F.2.2  Printed Circuit Board Layout

Figure F.11 – Top side of Servo Board PCB
Figure F.12 – Bottom Side of Servo Board

Figure F.13 – Top Overlay of Servo Board

Figure F.14 – Bottom Overlay of Servo Board
### F.2.3 Bill of Materials

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Value</th>
<th>Designator</th>
<th>Footprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>470nF</td>
<td>C1</td>
<td>0805</td>
</tr>
<tr>
<td>2</td>
<td>22pF</td>
<td>C2, C3</td>
<td>0805</td>
</tr>
<tr>
<td>13</td>
<td>100nF</td>
<td>C4, C6, C7, C9, C15, C16, C17, C18, CU1, CU2, CU3, CU4, CU5</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>10nF</td>
<td>C5</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>22uF 16V</td>
<td>C8</td>
<td>Case C</td>
</tr>
<tr>
<td>5</td>
<td>220uF</td>
<td>C10, C11, C12, C13, C14</td>
<td>CAPD300P100</td>
</tr>
<tr>
<td>2</td>
<td>10BQ015</td>
<td>D1, D2</td>
<td>SMB DIODE</td>
</tr>
<tr>
<td>1</td>
<td>1N4007</td>
<td>D3</td>
<td>SMB DIODE</td>
</tr>
<tr>
<td>1</td>
<td>1N4148</td>
<td>D4</td>
<td>DIODE3</td>
</tr>
<tr>
<td>1</td>
<td>Backup Battery</td>
<td>J1</td>
<td>HDR1X3</td>
</tr>
<tr>
<td>2</td>
<td>SERVOS</td>
<td>J2, J3</td>
<td>HDR8X3</td>
</tr>
<tr>
<td>1</td>
<td>Servo Power</td>
<td>J4</td>
<td>Terminal Block</td>
</tr>
<tr>
<td>1</td>
<td>RECEIVER</td>
<td>J5</td>
<td>HDR8X3</td>
</tr>
<tr>
<td>1</td>
<td>ICD HEADER</td>
<td>J6</td>
<td>HDR6X1</td>
</tr>
<tr>
<td>1</td>
<td>RS232</td>
<td>J7</td>
<td>HDR2X5_CEN</td>
</tr>
<tr>
<td>1</td>
<td>Header 2</td>
<td>JP1</td>
<td>HDR1X2</td>
</tr>
<tr>
<td>1</td>
<td>CANCON</td>
<td>JP2</td>
<td>HDR2X3</td>
</tr>
<tr>
<td>1</td>
<td>2N3904</td>
<td>Q1</td>
<td>TO-92A</td>
</tr>
<tr>
<td>1</td>
<td>120</td>
<td>R1</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>1K</td>
<td>R2, R3</td>
<td>0805</td>
</tr>
<tr>
<td>2</td>
<td>4K7</td>
<td>R4, R5</td>
<td>0805</td>
</tr>
<tr>
<td>2</td>
<td>2K2</td>
<td>R6, R7</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>RES</td>
<td>R10</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>TX2SA-3V</td>
<td>RL1</td>
<td>TX2SA</td>
</tr>
<tr>
<td>1</td>
<td>REF3104</td>
<td>U1</td>
<td>SO-G3</td>
</tr>
<tr>
<td>1</td>
<td>SN65HVD251</td>
<td>U2</td>
<td>SOIC8</td>
</tr>
<tr>
<td>1</td>
<td>dsPIC30F5011</td>
<td>U3</td>
<td>TQFP64</td>
</tr>
<tr>
<td>1</td>
<td>REG104</td>
<td>U4</td>
<td>SOT223</td>
</tr>
<tr>
<td>1</td>
<td>PT6361C</td>
<td>U5</td>
<td>PT636X</td>
</tr>
<tr>
<td>1</td>
<td>MAX232D</td>
<td>U6</td>
<td>SO16</td>
</tr>
<tr>
<td>1</td>
<td>10MHz</td>
<td>X1</td>
<td>CSM-7</td>
</tr>
</tbody>
</table>

*Table F.2 – Bill of Materials of Servo Board*
F.3 CANsens Board

F.3.1 Schematics

Figure F.15 – Schematic of CANsens Digital Circuits

Figure F.16 – Schematic of CANsens Analogue Circuit
F.3.2 Printed Circuit Board Layout

Figure F.17 – Top side of CANsens Board

Figure F.18 – Bottom side of CANsens Board

Figure F.19 – Top Overlay of CANsens Board
Figure F.20 – Bottom Overlay of CANsens Board
# F.3.3 Bill of Materials

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Value</th>
<th>Designator</th>
<th>Footprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>100n</td>
<td>C3, C4, C5, C6, C10, C11, C12, C66, C67</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>10k</td>
<td>R2</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>10n</td>
<td>C13</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>120</td>
<td>R3</td>
<td>0805</td>
</tr>
<tr>
<td>2</td>
<td>220n</td>
<td>C14, C15</td>
<td>0805</td>
</tr>
<tr>
<td>16</td>
<td>22n</td>
<td>C50, C51, C52, C53, C54, C55, C56, C57, C58, C59, C60, C61, C62, C63, C64, C65</td>
<td>0805</td>
</tr>
<tr>
<td>2</td>
<td>22pF</td>
<td>C8, C9</td>
<td>0805</td>
</tr>
<tr>
<td>3</td>
<td>22n</td>
<td>C1, C2, C7</td>
<td>1210, CASE C</td>
</tr>
<tr>
<td>16</td>
<td>33k</td>
<td>R52, R53, R54, R55, R62, R63, R64, R65, R66, R67, R68, R69, R78, R79, R80, R81</td>
<td>0805</td>
</tr>
<tr>
<td>1</td>
<td>9.6MHz</td>
<td>Y1</td>
<td>X49_SMD_SML</td>
</tr>
<tr>
<td>1</td>
<td>ADS8344</td>
<td>U4</td>
<td>PD50-G20</td>
</tr>
<tr>
<td>1</td>
<td>ANACON</td>
<td>A50</td>
<td>HDR9X1</td>
</tr>
<tr>
<td>1</td>
<td>CANTERM</td>
<td>JP1</td>
<td>HDR2X1</td>
</tr>
<tr>
<td>1</td>
<td>HEADER</td>
<td>A52</td>
<td>HDR2X2</td>
</tr>
<tr>
<td>1</td>
<td>HEADER</td>
<td>A51</td>
<td>HDR5X1</td>
</tr>
<tr>
<td>1</td>
<td>HEADER</td>
<td>CANCON1</td>
<td>HEADER_5X2_SML</td>
</tr>
<tr>
<td>1</td>
<td>Header</td>
<td>J1</td>
<td>HDR1X6</td>
</tr>
<tr>
<td>1</td>
<td>Header</td>
<td>J2</td>
<td>HDR1X9</td>
</tr>
<tr>
<td>1</td>
<td>INDUCTOR</td>
<td>L1</td>
<td>SMT_L</td>
</tr>
<tr>
<td>8</td>
<td>infinite</td>
<td>R56, R57, R60, R61, R70, R71, R76, R77</td>
<td>0805</td>
</tr>
<tr>
<td>2</td>
<td>OPA4350</td>
<td>U50, U51</td>
<td>SOP14</td>
</tr>
<tr>
<td>1</td>
<td>PIC18F4X8</td>
<td>U2</td>
<td>TQFP44</td>
</tr>
<tr>
<td>1</td>
<td>REF3140</td>
<td>U5</td>
<td>SO-G3</td>
</tr>
<tr>
<td>1</td>
<td>REG104</td>
<td>U1</td>
<td>SOT223</td>
</tr>
<tr>
<td>1</td>
<td>SN65HVD251</td>
<td>U3</td>
<td>SO-G8</td>
</tr>
<tr>
<td>9</td>
<td>ZERO</td>
<td>R1, R50, R51, R58, R59, R72, R73, R74, R75</td>
<td>0805</td>
</tr>
</tbody>
</table>

*Table F.3 – Bill of materials of CANsens Board*
F.4 GPS/Aerocomm Motherboard

F.4.1 Schematics

Figure F.21 – Schematic of GPS/Aerocomm Motherboard
F.4.2 Printed Circuit Board Layout

Figure F.22 – Top side of GPS/Aerocomm Motherboard

Figure F.23 – Bottom side of GPS/Aerocomm Motherboard
F.4.3 Bill of Materials
Appendix G – Program Listing

G.1 PC/104CAN Controller

G.1.1 VHDL Code for One Byte Register

LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity dflipflop8 is
  port (clk : in bit;Data : in bit_vector(7 downto 0); Q : out bit_vector(7 downto 0));
end entity dflipflop8;

architecture ha of dflipflop8 is
begin
  process (clk) is
  begin
    wait until (clk = '1');
    Q <= Data;
  end process;
end architecture ha;

G.1.2 VHDL Code for Decoding ISA Address and Read/Write

LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity isadec is
  port (systemAddress  : in std_logic_vector(8 downto 0); AEN, IOW, IOR : in std_logic; writeOut: out std_logic_vector(1 downto 0); readOut : out std_logic_vector(2 downto 0));
end entity isadec;

architecture ha of isadec is
begin
  signal temp : std_logic_vector(11 downto 0);
  begin
    temp <= systemAddress & AEN & IOR & IOW;
    --the first 9 bits of temp is the address
    with temp select
      writeOut <= "10" when "111011110010",  --1de 3bc/3be
        "01" when "111011111010",  --1df 3bd/3bf
        "00" when others;
    with temp select
      readOut <= "100" when "111011110001",  --1de 3bc/3be
        "101" when "111011110011",  --1df 3bd/3bf
        "110" when "111110100001",  --1f4 3e8/3ea
        "111" when "111110101001",  --1f5 3e9/3eb
        "000" when others;
  end architecture ha;

G.1.3 VHDL Code for Decoding PIC Address and Read/Write

LIBRARY ieee;
G.1.4 VHDL Code for 8x4 Multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity multiplex4 is
    port (sel : in std_logic_vector(1 downto 0); outputEnable  : in bit;D0,D1,D2,D3 : in std_logic_vector(7 downto 0); Q : out std_logic_vector(7 downto 0));
end entity multiplex4;

architecture ha of multiplex4 is
signal temp : std_logic_vector(7 downto 0);
begin
    temp <= D0 when sel = "00" else
            D1 when sel = "01" else
            D2 when sel = "10" else
            D3 when sel = "11";
    Q <= temp when outputEnable = '1' else
         "ZZZZZZZZ";
end architecture ha;

G.1.5 VHDL Code for 8x2 Multiplexer

LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity multiplex is
    port (sel, outputEnable  : in bit;D0,D1 : in std_logic_vector(7 downto 0); Q : out std_logic_vector(7 downto 0));
end entity multiplex;

architecture ha of multiplex is
signal temp : std_logic_vector(7 downto 0);
begin
    temp <= D0 when sel = '0' else
            D1 when sel = '1';
    Q <= temp when outputEnable = '1' else
         "ZZZZZZZZ";
end architecture ha;
end architecture ha;

G.2 MATLAB code

G.2.1 Generate controllers

clear all;
close all;
Total_Aircraft_Weight = 8.249;
Sim_Total_Aircraft_Weight = Total_Aircraft_Weight;
Aircraft_Interia = [0.578 0 0; 0 0.260 0; 0 0 0.760];
Grav_Const = 9.81;
Starting_Position = [-33.9*pi/180 18.5*pi/180 25];
Ground_Level = 18;  %Distance ASL for ground (to stop simulation)
D_mmy = 0.7;        %Distance between the 2 centrelines of the motors
D_vv = 0.9;         %Distance between the 2 vertical (Y-direction) flaps
D_cmh = 0.11;       %Distance from centre of mass y to the hinge line
Earth_R = 6378137;  %Approx equatorial radius of earth
Vec_eff = 0.2;      %effectivity of air deflectors
Motor_Tf_Num = [1];
Motor_Tf_Den = [1];
Max_Thrust = 15*Grav_Const;
Max_Angle = pi/6;

Trim_Values = [0 0 -Grav_Const/2*Total_Aircraft_Weight 0 0 -Grav_Const/2*Total_Aircraft_Weight];

Switch = 1;        %1 = input routed for linmod, 2 = Constant input values, 3 = controller enabled
[A B C D] = linmod('linearize',[0 0 0 0 0 0 0 0 0 0 0 0],Trim_Values);
sysC = ss(A,B,C,D);
temp = size(A);
CM = [];
for n=0:temp(1)-1
   CM = [CM A^n*B];
end
rank(CM)

% roll_A = [1 0; 0 0];
% roll_B = [1; 0];
% roll_C =

% Longitudinal Controller
X_max = 1;
theta_max = 0.1;
theta_max = X_max/5;
q_max = theta_max/5;
long_intWeight = 15;
long_uMax = 0.5;
long_A = [A(1,1) A(1,7) A(1,5) A(1,11);
   A(7,1) A(7,7) A(7,5) A(7,11);
   A(5,1) A(5,7) A(5,5) A(5,11);
   A(11,1) A(11,7) A(11,5) A(11,11)];
long_B = [B(1,1); 2*B(7,1); B(5,1); 2*B(11,1)];
long_C = eye(4);
long_D = zeros(4,1);
long_A_Int = [0 long_C(1,:);zeros(4,1) long_A];
long_B_Int = [0; long_B];
long_sys = ss(long_A, long_B, long_C, long_D);
Q1_bar = [1/X_max^2 0 0 0 1/u_max^2 0 0 0 1/theta_max^2 0 0 0 1/q_max^2];
Q1 = long_C'*Q1_bar*long_C;
Q1 = [long_intWeight zeros(1,4); zeros(4,1) Q1];
Q2 = [1/long_uMax^2];

long_K = lqr(long_A_Int, long_B_Int, Q1, Q2);
long_K_Int = long_K(1)
long_K = long_K(2:5)
% 1 Output = phi_x1 = phi_x2

% altitude controller
Z_max = 2;
w_max = Z_max/5;
alt_intWeight = 15;
alt_uMax = 0.1;
alt_A = [A(3,3) A(3,9); A(9,3) A(9,9)];
alt_B = [0;2*B(9,3)];
alt_C = eye(2);
alt_D = zeros(2,1);
alt_sys = ss(alt_A, alt_B, alt_C, alt_D);
alt_A_Int = [0 alt_C(1,:);zeros(2,1) alt_A];
alt_B_Int = [0 ; alt_B];
Q1_bar = [1/Z_max^2 0 ;0 1/w_max^2];
Q1 = alt_C'*Q1_bar*alt_C;
Q1 = [alt_intWeight zeros(1,2); zeros(2,1) Q1];
Q2 = [1/alt_uMax^2];
alt_K = lqr(alt_A_Int, alt_B_Int, Q1, Q2)
alt_K_Int = alt_K(1)
alt_K = alt_K(2:3)
% 1 Output = F1 = F2

% Lateral Controller
Y_max = 1;
phi_max = 0.1;
v_max = Y_max/5;
p_max = phi_max/5;
lat_intWeight = 15;
lat_u1Max = 0.5;
lat_u2Max = 0.5;
lat_A = [A(2,2) A(2,8) A(2,4) A(2,10); A(8,2) A(8,8) A(8,4) A(8,10); A(4,2) A(4,8) A(4,4) A(4,10); A(10,2) A(10,8) A(10,4) A(10,10)];
lat_B = [B(2,2) B(8,2) B(4,2) B(10,2)]'*2;
%B(2,3) B(8,3) B(4,3) B(10,3)]'*2;
lat_C = eye(4);
lat_D = zeros(4,1);
lat_sys = ss(lat_A, lat_B, lat_C, lat_D);
lat_A_Int = [0 lat_C(1,:);zeros(4,1) lat_A];
lat_B_Int = [0 ; lat_B];
Q1_bar = [1/Y_max^2 0 0 0 1/v_max^2 0;0 0 0 0 1/phi_max^2 0;0 0 0 0 1/p_max^2];
Q1 = lat_C'*Q1_bar*lat_C;
Q1 = [lat_intWeight zeros(1,4); zeros(4,1) Q1];
Q2 = [1/lat_u1Max^2; % 0 0 0 lat_u2Max^2];
lat_K = lqr(lat_A_Int, lat_B_Int, Q1, Q2)
lat_K_Int = lat_K(:,1)
lat_K = lat_K(:,2:5)
% Output1 = phi_y1 = phi_y2 and Output2 = F1 = -F2

% Heading controller
\( \psi_{\text{max}} = 0.1; \)
\( r_{\text{max}} = \frac{Z_{\text{max}}}{2}; \)
\( \text{head\_intWeight} = 5; \)
\( \text{head\_uMax} = \frac{\pi}{6}; \)
\( \text{head\_A} = [A(6,6) \ A(6,12); A(12,6) \ A(12,12)]; \)
\( \text{head\_B} = [B(6,1); B(12,1)]^*2; \)
\( \text{head\_C} = \text{eye}(2); \)
\( \text{head\_D} = \text{zeros}(2,1); \)
\( \text{head\_sys} = \text{ss}(\text{head\_A}, \text{head\_B}, \text{head\_C}, \text{head\_D}); \)
\( \text{head\_A\_Int} = [0 \ \text{head\_C(1,:)}; \text{zeros}(2,1) \ \text{head\_A}]; \)
\( \text{head\_B\_Int} = [0; \ \text{head\_B}]; \)
\( \text{Q1\_bar} = \begin{bmatrix} 1/\psi_{\text{max}}^2 & 0 \\ 0 & 1/r_{\text{max}}^2 \end{bmatrix}; \)
\( \text{Q1} = \text{head\_C}'*\text{Q1\_bar}*\text{head\_C}; \)
\( \text{Q1} = \begin{bmatrix} \text{head\_intWeight} & \text{zeros}(2,1) \\ \text{zeros}(2,1) & \text{Q1} \end{bmatrix}; \)
\( \text{Q2} = \begin{bmatrix} 1/\text{head\_uMax}^2 \end{bmatrix}; \)
\( \text{head\_K} = \text{lqr}(\text{head\_A\_Int}, \text{head\_B\_Int}, \text{Q1}, \text{Q2}); \)
\( \text{head\_K\_Int} = \text{head\_K}(1); \)
\( \text{head\_K} = \text{head\_K}(2:3); \)
\( \text{Output} = \phi_x1 = -\phi_x2; \)
\( \text{Switch} = 2; \)
\( \text{sim('tiltwingsim2')}; \)

\begin{verbatim}
figure(1);
hold on;
plot(Time, Pos_earth(:,1),'b-');
plot(Time, Pos_earth(:,2),'k:');
plot(Time, Pos_earth(:,3),'r--');
hold off;
title('3D Position of the Aircraft');
legend('North (X)', 'East (Y)', 'Down (Z)', 'Location','best');
xlabel('Time (in sec)');
ylabel('Position (in meters)');
set(1,'Name','Position');
grid off;

figure(2);
hold on;
plot(Time, V_earth(:,1),'b-');
plot(Time, V_earth(:,2),'k:');
plot(Time, V_earth(:,3),'r--');
hold off;
title('3D Velocity of the Aircraft');
legend('u', 'v', 'w', 'Location', 'best');
xlabel('Time (in sec)');
ylabel('Velocity (in meters/sec)');
set(2,'Name','Velocity');
grid off;

figure(3);
hold on;
plot(Time, Euler(:,1),'b-');
plot(Time, Euler(:,2),'k:');
plot(Time, Euler(:,3),'r--');
hold off;
title('Orientation of aircraft');
legend('\Phi', '\Theta', '\Psi', 'Location', 'best');
xlabel('Time (in sec)');
ylabel('Angle (in radians)');
set(3,'Name','Euler angles');
\end{verbatim}
G.2.2 Compare the linear and non-linear model

```matlab
grid off;

clear all;
close all;
Total_Aircraft_Weight = 8.249;
Sim_Total_Aircraft_Weight = Total_Aircraft_Weight;
Aircraft_Interia = [0.578 0 0; 0 0.260 0; 0 0 0.760];
Grav_Const = 9.81;
Starting_Position = [-33.9*pi/180 18.5*pi/180 25];
Ground_Level = 18;  %Distance ASL for ground (to stop simulation)
D_mmy = 0.7;        %Distance between the 2 centrelines of the motors
D_vv = 0.9;         %Distance between the 2 vertical (Y-direction) flaps
D_cmh = 0.11;       %Distance from centre of mass y to the hinge line
Earth_R = 6378137;  %Approx equatorial radius of earth
Vec_eff = 0.2;      %effectivity of air deflectors
Trim_Values = [0 0 -Grav_Const/2*Total_Aircraft_Weight 0 0 -Grav_Const/2*Total_Aircraft_Weight];
[A B C D] = linmod('linearize',[0 0 0 0 0 0 0 0 0 0 0 0],Trim_Values);

Apply_Values = [0.2 0 -Grav_Const/2*Total_Aircraft_Weight 0.2 0 -Grav_Const/2*Total_Aircraft_Weight];
sim('verify',1);
subplot(2,2,1);
hold on;
plot(tout,linear(:,1),'b-')
plot(tout,linear(:,2),'k:')
plot(tout,linear(:,3),'r--')
hold off;
title('Position of aircraft with common mode deflection of horizontal paddles')
legend('X_l_i_n','Y_l_i_n','Z_l_i_n','Location','best')
ylabel('Position in meters (Linear Model)');
subplot(2,2,3);
hold on;
plot(tout,nonlinear(:,1),'b-')
plot(tout,nonlinear(:,2),'k:')
plot(tout,nonlinear(:,3),'r--')
hold off;
legend('X','Y','Z','Location','best');
xlabel('Time in seconds');
ylabel('Position in meters (Non-linear Model)');
subplot(2,2,2);
hold on;
plot(tout,linear(:,4),'b-')
plot(tout,linear(:,5),'k:')
plot(tout,linear(:,6),'r--')
hold off;
axis([0 1 -0.8 0.1]);
title('Orientation of aircraft with common mode deflection of horizontal paddles')
legend('\Phi_l_i_n','\Theta_l_i_n','\Psi_l_i_n','Location','best')
ylabel('Angle in radians (Linear Model)');
subplot(2,2,4);
hold on;
plot(tout,nonlinear(:,4),'b-')
plot(tout,nonlinear(:,5),'k:')
plot(tout,nonlinear(:,6),'r--')
hold off;
axis([0 1 -0.8 0.1]);
legend('\Phi','\Theta','\Psi','Location','best');
ylabel('Angle in radians (Non-linear Model)');
xlabel('Time in seconds');
figure(2)
Apply_Values = [0.2 0 -Grav_Const/2*Total_Aircraft_Weight -0.2 0 -Grav_Const/2*Total_Aircraft_Weight];
```
sim('verify',1);
subplot(2,2,1);
hold on;
plot(tout,linear(:,1),'b-')
plot(tout,linear(:,2),'k:')
plot(tout,linear(:,3),'r--')
hold off;
title('Position of aircraft with differential deflection of horizontal paddles')
legend('X_l_i_n', 'Y_l_i_n', 'Z_l_i_n', 'Location', 'best')
ylabel('Position in meters (Linear Model)');
subplot(2,2,3);
hold on;
plot(tout,nonlinear(:,1),'b-')
plot(tout,nonlinear(:,2),'k:')
plot(tout,nonlinear(:,3),'r--')
hold off;
axis([0 1 -1e-3 4e-3]);
legend('X', 'Y', 'Z', 'Location', 'best');
xlabel('Time in seconds');
ylabel('Position in meters (Non-linear Model)');
subplot(2,2,2);
hold on;
plot(tout,linear(:,4),'b-')
plot(tout,linear(:,5),'k:')
plot(tout,linear(:,6),'r--')
hold off;
axis([0 1 -0.8 0.1]);
title('Orientation of aircraft with differential deflection of horizontal paddles')
legend('
Phi_l_i_n', '
Theta_l_i_n', '
Psi_l_i_n', 'Location', 'best')
ylabel('Angle in radians (Linear Model)');
subplot(2,2,4);
hold on;
plot(tout,nonlinear(:,4),'b-')
plot(tout,nonlinear(:,5),'k:')
plot(tout,nonlinear(:,6),'r--')
hold off;
axis([0 1 -0.8 0.1]);
title('Orientation of aircraft with differential deflection of horizontal paddles')
legend('
phi', '
theta', '
psi', 'Location', 'best');
xlabel('Time in seconds');
ylabel('Angle in radians (Non-linear Model)');
figure(3)
Apply_Values = [0 0 -Grav_Const/3*Total_Aircraft_Weight 0 0 -Grav_Const/3*Total_Aircraft_Weight];
sim('verify',1);
subplot(2,2,1);
hold on;
plot(tout,linear(:,1),'b-')
plot(tout,linear(:,2),'k:')
plot(tout,linear(:,3),'r--')
hold off;
axis([0 1 -0.5 2]);
title('Position of aircraft with thrust reduced')
legend('X_l_i_n', 'Y_l_i_n', 'Z_l_i_n', 'Location', 'best')
ylabel('Position in meters (Linear Model)');
subplot(2,2,3);
hold on;
plot(tout,nonlinear(:,1),'b-')
plot(tout,nonlinear(:,2),'k:')
plot(tout,nonlinear(:,3),'r--')
hold off;
axis([0 1 -0.5 2]);
title('Position of aircraft with thrust reduced')
legend('X', 'Y', 'Z', 'Location', 'best');
xlabel('Time in seconds');
ylabel('Position in meters (Non-linear Model)');
subplot(2,2,2);
hold on;
plot(tout,linear(:,4),'b-')
plot(tout,linear(:,5),'k:')
plot(tout,linear(:,6),'r--')
hold off;
title('Orientation of aircraft with thrust reduced')
legend('\Phi_l_i_n','\Theta_l_i_n','\Psi_l_i_n','Location','best')
ylabel('Angle in radians (Linear Model)');
subplot(2,2,4);
hold on;
plot(tout,nonlinear(:,4),'b-')
plot(tout,nonlinear(:,5),'k:')
plot(tout,nonlinear(:,6),'r--')
hold off;
legend('\Phi','\Theta','\Psi','Location','best');
xlabel('Time in seconds');
ylabel('Angle in radians (Non-linear Model)');

figure(5)

Apply_Values = [0 0.2 -Grav_Const/2*Total_Aircraft_Weight 0 0.2 -Grav_Const/2*Total_Aircraft_Weight];
sim('verify',1);
subplot(2,2,1);
hold on;
plot(tout,linear(:,1),'b-')
plot(tout,linear(:,2),'k:')
plot(tout,linear(:,3),'r--')
hold off;
title('Orientation of aircraft with common mode deflection of vertical paddles')
legend('X_l_i_n','Y_l_i_n','Z_l_i_n','Location','best')
ylabel('Position in meters (Linear Model)');
subplot(2,2,3);
hold on;
plot(tout,nonlinear(:,1),'b-')
plot(tout,nonlinear(:,2),'k:')
plot(tout,nonlinear(:,3),'r--')
hold off;
legend('X','Y','Z','Location','best');
xlabel('Time in seconds');
ylabel('Position in meters (Non-linear Model)');
subplot(2,2,2);
hold on;
plot(tout,linear(:,4),'b-')
plot(tout,linear(:,5),'k:')
plot(tout,linear(:,6),'r--')
hold off;
axis([0 1 -0.1 0.4])
title('Orientation of aircraft with common mode deflection of vertical paddles')
legend('\Phi_l_i_n','\Theta_l_i_n','\Psi_l_i_n','Location','best')
ylabel('Angle in radians (Linear Model)');
subplot(2,2,4);
hold on;
plot(tout,nonlinear(:,4),'b-')
plot(tout,nonlinear(:,5),'k:')
plot(tout,nonlinear(:,6),'r--')
hold off;
axis([0 1 -0.1 0.4])
legend('\Phi','\Theta','\Psi','Location','best');
xlabel('Time in seconds');
ylabel('Angle in radians (Non-linear Model)');

figure(5)
Apply.Values = [0 -0.2 -Grav_Const/2*Total_Aircraft_Weight 0 0.2 -Grav_Const/2*Total_Aircraft_Weight];
sim('verify',1);
subplot(2,2,1);
hold on;
plot(tout,linear(:,1),'b-')
plot(tout,linear(:,2),'k:')
plot(tout,linear(:,3),'r--')
hold off;
title('Orientation of aircraft with differential deflection of vertical paddles')
legend(['X_{l_i_n}', 'Y_{l_i_n}', 'Z_{l_i_n}', 'Location','best'])
ylabel('Position in meters (Linear Model)');
subplot(2,2,3);
hold on;
plot(tout,nonlinear(:,1),'b-')
plot(tout,nonlinear(:,2),'k:')
plot(tout,nonlinear(:,3),'r--')
hold off;
axis([0 1 -1e-3 4e-3]);
legend('X','Y','Z','Location','best');
xlabel('Time in seconds');
ylabel('Position in meters (Non-linear Model)');
subplot(2,2,2);
hold on;
plot(tout,linear(:,4),'b-')
plot(tout,linear(:,5),'k:')
plot(tout,linear(:,6),'r--')
hold off;
title('Orientation of aircraft with differential deflection of vertical paddles')
legend(['\Phi_{l_i_n}', '\Theta_{l_i_n}', '\Psi_{l_i_n}', 'Location','best'])
ylabel('Angle in radians (Linear Model)');
subplot(2,2,4);
hold on;
plot(tout,nonlinear(:,4),'b-')
plot(tout,nonlinear(:,5),'k:')
plot(tout,nonlinear(:,6),'r--')
hold off;
legend(['\Phi','\Theta','\Psi','Location','best']);
xlabel('Time in seconds');
ylabel('Angle in radians (Non-linear Model)');
Appendix H – Additional Controller Tests

The following additional tests were run on the controller:

- A slewed step command on X, Y and Z simultaneously (Test A)
- Turn to a heading of 30º and give a slewed step command on X,Y and Z (Test B)

Figure H.1 – Position of aircraft during Test A
3D Velocity of the Aircraft

Figure H.2 – Velocity of aircraft during Test A

Orientation of the Aircraft

Figure H.3 – Orientation of aircraft during Test A
Figure H.4 – Position of aircraft during Test B

Figure H.5 – Velocity of aircraft during Test B
Test A (Figure H.1, Figure H.2, Figure H.3)
The controllers are able to do all of the slewed step commands at once and keep the aircraft stable.

Test B (Figure H.4, Figure H.5, Figure H.6)
The cross-coupling due to the incorrect orientation of the aircraft can clearly be seen. The controllers are able to control the aircraft, but are marginally instable due to the cross-coupling. This shows that maintaining the correct heading of the aircraft is important.