Development of a Low Cost Autopilot System for Unmanned Aerial Vehicles

Jose Ortiz
Virginia Commonwealth University

Follow this and additional works at: https://scholarscompass.vcu.edu/etd
Part of the Engineering Commons

© The Author

Downloaded from
https://scholarscompass.vcu.edu/etd/2252

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.
Development of a Low Cost Autopilot System for Unmanned Aerial Vehicles

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

Jose Enrique Ortiz,
BS Computer Engineering, Virginia Commonwealth University 2008
BS Physics, Virginia Commonwealth University 2008

Director: Robert H. Klenke,
Associate Professor, Computer Engineering

Virginia Commonwealth University
Richmond, Virginia
August, 2010
Acknowledgments

I would like to thank my wife and daughter, Magdalena and Maria, for their love, support, and patience during the years it has taken me to graduate. You are the reason why I work so hard. I would like to thank my committee chair, Dr. Robert Klenke, for his confidence in my work. Without his guidance this thesis would not have been possible. He did more than just sign the checks and tow the trailer. I would like to thank Dr. McCollum for his enthusiasm and support. I would also like to thank my committee member, Dr. Alison Baski, for all her advice and for giving me a place in her lab during my first summer at VCU. Finally, I would like to thank all the students that I’ve worked with over the past few years.
# Contents

List of Figures ............................................. V

List of Tables ......................................... VII

List of Abbreviations ................................ VIII

Abstract .................................................... XII

1 Introduction ........................................... 1
   1.1 Problem statement .................................. 1
   1.2 Motivation .......................................... 1
   1.3 Scope of Work ...................................... 3
   1.4 Thesis Overview .................................... 3

2 Background .............................................. 5
   2.1 Early History of UAVs .............................. 5
   2.2 VCU Autopilot Research ......................... 6
      2.2.1 First Generation VCU FCS ................... 7
      2.2.2 Second Generation VCU FCS .................. 8
      2.2.3 Current VCU FCS ............................ 10
   2.3 Commercial Autopilots .......................... 12
      2.3.1 MicroPilot MP Series Autopilots .......... 12
      2.3.2 Cloud Cap Piccolo SL ....................... 14
      2.3.3 Procerus Technologies’ Kestrel 2.4 ....... 16
      2.3.4 UNAV 3500FW ................................ 17
   2.4 University Developed Autopilots ............... 18
      2.4.1 AggieAir ...................................... 18
      2.4.2 Federal University of Minas Gerais ....... 19
      2.4.3 Tsinghua University TUAV1000 ............ 21
4.7.1 PID Control .................................................... 69
4.7.2 Lateral Control .............................................. 71
4.7.3 Longitudinal Control ....................................... 72

5 Hardware In the Loop Simulator .................................. 74
  5.1 Hardware Implementation ...................................... 76
    5.1.1 Base Board .............................................. 76
    5.1.1.1 Digital to Analog Converter ........................... 78
    5.1.1.2 PWM Input Buffers .................................... 81
    5.1.1.3 Other Components ..................................... 82
    5.1.2 Suzaku SZ130 ........................................... 82
    5.1.3 IP Cores ................................................ 84
  5.2 Software Implementation ..................................... 85
    5.2.1 FlightGear ............................................... 86
    5.2.2 Simulated Sensors ....................................... 87
      5.2.2.1 MIDG II INS ....................................... 88
      5.2.2.2 Differential Pressure Sensors ..................... 89
      5.2.2.3 Absolute Pressure Transducers .................... 91
      5.2.2.4 Infrared Sensors ................................... 92

6 Flight Tests and Evaluation ..................................... 94
  6.1 Test platform .............................................. 95
  6.2 Waypoint Path Tracking ................................... 96
  6.3 Orbit Tracking ........................................... 98
  6.4 Altitude and Airspeed Hold ............................... 100

7 Conclusion and Future Work .................................. 102
  7.1 Conclusions ............................................... 102
  7.2 Future Work ............................................. 103

References A
# List of Figures

2.1 First generation VCU FCS. ................................................. 8
2.2 Second generation VCU FCS. Suzaku EX expansion board with Suzaku-V mounted. .................................................. 10
2.3 Current VCU FCS. .......................................................... 11
2.4 MicroPilot MP2028G .......................................................... 14
2.5 Cloud Cap Piccolo SL Autopilot ........................................... 15
2.6 Kestrel 2.4 Autopilot ......................................................... 17
2.7 UNAV 3500FW Autopilot ..................................................... 18
2.8 AggieNav computer system. ................................................. 19
2.9 Minas Gerais low cost UAV platform. ................................. 21
2.10 Tsinghua University flight control system. .......................... 22
2.11 Georgia Tech’s Low Cost Autopilot Hardware ........................ 24
2.12 Direction field generated by a simple attractor potential function. . 27
2.13 UAV Inertial Position and Velocity. ..................................... 28
2.14 Straight path vector field geometry. ..................................... 30
3.1 Hardware Architecture ...................................................... 32
3.2 Signal Conditioning Circuit ............................................... 40
3.3 Signal Conditioning low pass filter response. ......................... 42
3.4 Digital and analog supply noise. ........................................ 44
3.5 Printed circuit board. .......................................................... 46
3.6 Impedance of three capacitors in parallel. ............................ 47
4.1 Flight control software main loop. ...................................... 51
4.2 Z and XY IR sensors from Range Video. ............................... 61
4.3 Orbital trajectories. .......................................................... 63
4.4 Cross-track trajectories. .................................................... 67
4.5 Cross-track heading vector field geometry. .......................... 67
4.6 PID controller .................................................. 70
4.7 Cascaded PID Controller .................................... 71

5.1 Hardware in the loop system overview .................... 76
5.2 HILS with all components assembled ...................... 77
5.3 Hardware in the Loop Simulator base board layout ....... 78
5.4 Hardware in the Loop Simulator base board PCB ......... 78
5.5 AD5678 DAC block diagram ................................ 79
5.6 AD5678 timing diagram ...................................... 80
5.7 AD5678 input shift register ................................ 80
5.8 Hardware in the loop simulator block diagram .......... 84
5.9 MIDG II binary packet format .............................. 88

6.1 Electric UAV platform with miniFCS ....................... 95
6.2 miniFCS connected to the HILS ............................ 95
6.3 Waypoint navigation cross-track error .................... 96
6.4 Hardware in the loop simulation of cross-track error correction on a 1.3 km leg ........................................ 97
6.5 Orbital flight trajectories .................................. 98
6.6 Orbit tracking with 15 knot wind ........................ 99
6.7 Hardware in the loop simulation of altitude hold and transitions at 100, 200, and 300 meters .......................... 100
6.8 Hardware in the loop simulation of airspeed hold and transitions to 25, 40, and 50 knots .............................. 100
6.9 Actual sailplane airspeed and altitude hold .............. 101
## List of Tables

2.1 Commercial auto pilot comparison. .......................... 25

3.1 System cost matrix. ........................................... 48

4.1 VACS packet structure. ....................................... 54

4.2 NMEA GGA data format. ..................................... 56

4.3 Altitude to 10-Bit code. ..................................... 59

5.1 AD5678 commands and addresses. ......................... 80

5.2 AD5678 DAC addresses. ..................................... 81
List of Abbreviations

ADC ...... Analog to Digital Converter
AGL ...... Above Ground Level
API ...... Application Programming Interface
ARM ...... Advanced RISC Machine
ASL ...... Above Sea Level
AUVSI ... Association for Unmanned Vehicle Systems International
BEC ...... Battery Eliminator Circuit
BSc ...... Bachelor of Science
CAN ...... Controller Area Network
CAS ...... Calibrated Ainspeed
COTS .... Commercial, off-the-shelf
CPU ...... Central Processing Unit
CTS ...... Clear to Send
DAC ...... Digital to Analog Converter
DMA ...... Direct Memory Access
DOF ...... Degree Of Freedom
DSP ...... Digital Signal Processor
EDK ...... Embedded Development Kit
ENU ...... East, North, Up
FCS ...... Flight Control System
FIFO ..... First In, First Out
FPGA ..... Field Programmable Gate Array
FPSLIC ..... Field Programmable System Level Integrated Circuit
FPU ..... Floating Point Unit
GCC ..... GNU Compiler Collection
GCS ..... Ground Control Server
GGA ..... Global Positioning System Fix Data
GNU ..... GNU’s Not Unix
GPIO ..... General Purpose Input Output
GPS ..... Global Positioning System
GSA ..... GPS DOP and active satellites
GSC ..... Ground Station Client
HDL ..... Hardware Description Language
HILS ..... Hardware In the Loop Simulator
IC ..... Integrated Circuit
IMU ..... Inertial Measurement Unit
INS ..... Inertial Navigation System
IO ..... Input Output
IP ..... Intellectual Property
IR ..... Infra Red
ISM ..... Industrial, Scientific, and Medical
ISR ..... Interrupt Service Routine
JAUS ..... Joint Architecture for Unmanned Systems
LLA ..... Latitude, Longitude, Altitude
LSB ..... Least Significant Bit
MCU ...... MicroController Unit
MEMS ...... Micro-Electro-Mechanical Systems
MIPS ...... Microprocessor without Interlocked Pipeline Stages
MMU ...... Memory Management Unit
MSc ...... Master of Science
NMEA ...... National Marine Electronics Association
NVRAM . Non-Volatile Random Access Memory
PCB ...... Printed Circuit Board
PDA ...... Personal Data Assistant
PDC ...... Peripheral DMA Controller
PID ...... Proportional Integral Derivative
PWM .... Pulse Width Modulation
RC ...... Radio Control
RF ....... Radio Frequency
RISC ...... Reduced Instruction Set Computer
RPM ...... Revolutions Per Minute
RTS ...... Request to Send
SDK ...... Software Development Kit
SDRAM . Synchronous Dynamic Random Access Memory
SMD ...... Surface Mount Device
SPI ...... Serial Peripheral Interface
SPS ...... Samples Per Second
SRAM ...... Static Random Access Memory
SUAV .... Small Unmanned Aerial Vehicle
SWaP ...... Size Weight and Power
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver/Transmitter</td>
</tr>
<tr>
<td>UAS</td>
<td>Unmanned Arial System</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Arial Vehicle</td>
</tr>
<tr>
<td>USART</td>
<td>Universal Synchronous/Asynchronous Receiver/Transmitter</td>
</tr>
<tr>
<td>VACS</td>
<td>VCU Aerial Communications Standard</td>
</tr>
<tr>
<td>VCU</td>
<td>Virginia Commonwealth University</td>
</tr>
<tr>
<td>VTG</td>
<td>Vector Track and speed over Ground</td>
</tr>
</tbody>
</table>
Abstract

DEVELOPMENT OF A LOW COST AUTOPilot SYSTEM FOR UNMANNED AERIAL VEHICLES

By Jose E. Ortiz, MS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2010.

Major Director: Robert H. Klenke, Associate Professor, Computer Engineering

The purpose of this thesis was to develop a low cost autonomous flight control system for small unmanned aerial vehicles with the aim to support collaborative systems. A low cost hardware solution was achieved by careful selection of sensors, integration of hardware subsystems, and the use of new microcontroller technologies. Flight control algorithms to guide a vehicle though waypoint based flight paths and loiter about a point were implemented using direction fields. A hardware in the loop simulator was developed to ensure proper operation of all hardware and software components prior to flight testing. The resulting flight control system achieved stable and accurate flight while reducing the total system cost to less than $250.
Chapter 1

Introduction

1.1 Problem statement

Research topics in swarming teams and role coordination algorithms for cooperative UAVs with limited resources have become more popular in recent years. In order to support new swarming UAS research activities, a new, simple, lightweight, power efficient, and inexpensive FCS is needed. Typical target vehicle platforms for this type of research have small cargo bay volumes where the autopilot, sensors, batteries, radios, safety overrides, and related electronics must all fit. The new FCS’s lowered cost and complexity, coupled with lowered safety concerns will allow research groups to verify their algorithms under real life conditions by simultaneous deploying multiple inexpensive autonomous aircraft.

1.2 Motivation

The idea of developing a truly low cost miniature FCS, the miniFCS, arose from the need for a platform that would support research into collaborative UAV algorithms.
Although the previous VCU systems could be used in this capacity, it would require multiple expensive FCSs and Aircraft. The cost of a multi-UAV system has to be drastically reduced in order to support this type of research. The size, weight, and power usage of the current autopilot system requires a relatively large vehicle platform. Not only is this a major part of the high overall cost, but it poses a safety issues. It is manageable to safely fly a single experimental system of considerable size, but the problem can quickly become infeasible with multiple aircraft given the budget and pilot limitations of small UAV research laboratories.

System complexity needed to be reduced with a new FCS board that integrates more components in to a smaller space and allows for simpler integration into the vehicle platform. The software also needed be written to be more understandable to those with less programming experience. This allows the use of the system by undergraduate students without the steep learning curve required by the current system and development tools. Development on the current system requires the knowledge of many tools, this includes HDL synthesizers, HDL debuggers for hardware and the firmware tool chain on top of the programming language used for actual development. Reduction of the number of development tools and a more integrated development environment allows for quicker development that is less prone to errors. The level of knowledge required to begin programming new algorithms for the FCS has lowered enough that an undergraduate student looking for some experience in flight control systems would be able to take the miniFCS and begin development with out a high cost risk or require a large amount of time learning tools.

Even though the cost for flight testing has been greatly reduced by the miniFCS system, it is still desirable to test any new control algorithm on the ground using as much of the actual FCS hardware as possible. In order to accomplish this goal, a hard-
ware in the loop simulator was developed. The HILS allowed faster development and testing of the miniFCS and its control algorithms. It is also intended for development of new future control algorithms and to verify proper operation of system hardware before it is flight tested. The use of the simulator also allows researchers with less experience to safely test their ideas on the ground within a simulated environment.

1.3 Scope of Work

This thesis will present work completed for a low cost autopilot system. This includes a complete hardware platform including all sensors required for autonomous flight of a small fixed wing aircraft. A software system that runs on the low cost FCS is also presented. This includes a command and control system that is compatible with the current VCU ground control software. The software also contains navigation and flight control algorithms to perform waypoint navigation and loitering. This thesis does not present work to support collaborative UAV mission planning algorithms nor the command and control systems required to support such activities.

1.4 Thesis Overview

In the following chapters, the development and testing of a complete UAV autopilot system is presented. The first half of the thesis concentrates on the hardware and software developed. The second half presents a HILS simulator developed for testing the autopilot as well as results and evaluation of the system.

Chapter 2 provides a review of literature. It identifies various commercial autopilots that may be suitable for small UAV applications along with their hardware and software features. University developed autopilots are also reviewed to gain insight on
how other research groups solved problems for small UAV designs. A brief overview of simple vector field navigation primitives are also discussed as a similar approach was used in developing the miniFCS’s guidance algorithms. These topics represent a snapshot of the current state of the art small UAV autopilot designs.

Chapter 3 focuses on the hardware solution for the miniFCS autopilot. It discusses the hardware requirements, components selected, and how they are integrated into an autopilot platform. Details of the sensor systems are discussed. A per unit cost analysis is also given to demonstrate the successful fulfillment of the self imposed cost constraints for the miniFCS autopilot.

Chapter 4 details the requirements and structure of the flight control software. It covers the communications system, analog sensor sampling, and main loop control. It also details navigation algorithms used for loitering and waypoint navigation. These are based on simple direction fields with low computational costs. Longitudinal and lateral control algorithms are also discussed. These are built using standard and cascaded PID controllers.

A simple HILS design is presented in chapter 5. This system was developed not only as a means to verify the miniFCS but also VCU’s various other autopilots. The initial design implements the sensor complement used by the miniFCS and has been extended to support the current and next generation VCU autopilots. The development of this system was critical in the successful design and testing of the miniFCS hardware and software.

Chapter 6 presents data from both simulated and actual flight tests. This data is used to support the successful implementation of the miniFCS autopilot system. Chapter 7 concludes the work achieved as it relates to the research goal of developing a truly low-cost autopilot system. Future work to improve flight controller performance and additional features to expand capabilities are suggested.
Chapter 2

Background

This chapter provides background information pertaining to the development of the miniFCS, a low cost flight control system. It begins by providing a short introduction to the history of UAVs. It is then followed by past FCS research at VCU, the original motivation source for the design of this project. Literature of other university developed flight control systems is then presented. An overview of commercially available autopilots is given to contrast the capabilities of the miniFCS and to justify its design. Finally, a brief overview of vector field guidance techniques is presented.

2.1 Early History of UAVs

Although there has recently been a flurry of development of UAV systems, the idea has been around since at least the mid 18 hundreds. During the first Italian independence war, Austria launched 200 pilot-less balloons carrying bombs with timed fuses into Venice. This primitive system was flawed because when the wind changed some balloons drifted back over to Austrian territory [1]. The Austrians also employed balloons, 23 feet in diameter, that were equipped with remote controlled triggers.
Releasing bombs was accomplished by using electromagnetism by means of a long copper wire and battery on the ground [2]. This incremental improvement allowed the Austrians to control when and where bombs would explode. Of course they were still at the whim of the winds but at least they could prevent bombing themselves.

Even at the infancy of wireless telegraphy, Nikola Tesla demonstrated the control of a small boat during an exhibition in 1898. Later, in his autobiography [3], he made a few statements that would predict the future of unmanned systems.

“...launch an aeroplane, have it follow a certain approximate course, and perform some operation at a distance of many hundreds of miles. A machine of this kind can also be mechanically controlled in several ways and I have no doubt that it may prove of some usefulness in war.”

Later he makes another prediction of where unmanned vehicle technology is heading.

“Telautomats will be ultimately produced, capable of acting as if possessed of their own intelligence, and their advent will create a revolution.”

The following sections will present the background of previous VCU autopilot systems as well as background on several UAV flight control systems from other universities and industry.

## 2.2 VCU Autopilot Research

Development of UAV systems at VCU began in the spring of 2004. A group of talented undergraduate students designed and implemented a complete flight control system as their senior design capstone project. From this basic design, several evolutionary systems have been developed. This section will provide a brief introduction to each of these systems.
2.2.1 First Generation VCU FCS

The first generation FCS was built around the Atmel FPSLIC [4]. This device combines Atmel’s AT40K FPGA architecture with a 20 MIPS 8-bit RISC microprocessor core and numerous microcontroller peripherals. The system used GPS to determine both position and heading of the aircraft. GPS cannot provide the FCS with pitch and roll information. To attain attitude information, infrared sensors were used. Control of the aircraft was accomplished via a ground control system developed in Visual Basic.

The flight control software was written in C++ and ran on the FPSLIC’s embedded AVR microcontroller. The main control loop of the FCS ran at 20Hz. It was tasked with bidirectional communication to the ground, receiving sensor input, perform aircraft stabilization, and waypoint following. The output of the FCS was a set of PWM values that correspond to control surface positions. Communications to the ground was via a radio modem connected to one of two serial ports on the microcontroller. The other serial port was used to receive GPS positioning data.

The FPGA contained hardware to decode PWM signals from a RC receiver. PWM signal generators were also implemented using the FPGA. Access to decoded PWM signals was accomplished by a set of registers that were addressable by the AVR microcontroller. Similarly, configuring the PWM generators was also accomplished via a register file. The PWM signal generators were used to control the servos connected to the aircraft’s control surfaces and throttle.

The first generation FCS had many limitations. Computing power of the 8-bit microcontroller limited the type and complexity of control algorithms that could be implemented. Floating point and DSP type calculations were too slow for practical implementation in a real-time environment. The small size of the FPGA limited the
number of soft peripherals that could be added to the system. The FCS was developed and implemented on a FPSLIC development board. This board is relatively large, 5” x 6”, and required add-on boards, connected to GPIO headers, to interface with external sensors and servos. Similarly, the GPS and radio modems have their own boards that were similar size. This added to the overall SWaP budget. Even with all the limitations of the system, it performed well enough to place 1st and 2nd at the 2004 and 2005 AUVSI UAV student competition.

![Atmel FPSLIC Device Diagram](image)

**Figure 2.1:** First generation VCU FCS.

### 2.2.2 Second Generation VCU FCS

The second generation flight control system addressed many of the issues of the earlier system. Major modifications were made to the computing platform and sensor system [5].

In this generation, the computing platform moved away from the 8-bit microcontroller design. The flight control system software ran on a Xilinx MicroBlaze soft-core
CPU running at 50Mhz. The PWM decoders and generators now lived within the same FPGA as the CPU. The FPGA system board used was an Atmark-Techno Suzaku-S. This board contains a Spartan 3 FPGA, SDRAM and NVRAM. Atmark-Techno provides a full software development environment comprising of a custom Builtroot μClinux distribution. FPGA hardware was based on a reference design for Xilinx’s EDK.

The system software was ported from the previous FCS to run in the μClinux environment. The microcontroller version of the FCS relied on interrupts to handle serial communications with various system peripherals. The second generation system used software threads to handle IO with serial devices. Another thread, running at 20 Hz, was used for flight control. It was tasked with sampling barometric sensors to calculate airspeed and altitude, process GPS information for navigation, and perform attitude control using PID based controllers. The system software was also ported to Atmark-Techno’s Suzaku-V platform. This platform uses Xilinx’s Virtex-2 Pro [6]. Rather than using a soft-core CPU, it contains a hardcore PowerPC processor that is capable of running at higher clock speeds.

Another advancement in the second generation FCS was the use of the Crossbow AHRS400-200 IMU. This IMU allowed for more accurate attitude measurements. Unlike IR attitude sensors, its performance was not dependent on weather conditions. The size, weight, and power usage of this IMU is considerably more than the IR sensors; it is about 10cm in its largest dimension, 640 grams and uses 3 Watts. This added considerably to the SWaP requirements of the system.

An expansion or based board, aptly named Suzaku EX, for the Suzaku computing platform was designed in house to interface with the various peripherals of the system. This was a major improvement because it moved away from using development boards
with daughter boards for peripheral connectivity. Size and weight were considerably reduced by this development. The base board contains:

- 3.3V regulator for the Suzaku board,
- two level shifters to interface serial ports to standard RS232 serial devices,
- differential and absolute barometric sensors for airspeed and altitude measurements,
- 8 channel ADC connected to the Suzaku board via SPI.

![Figure 2.2: Second generation VCU FCS. Suzaku EX expansion board with Suzaku-V mounted.](image)

### 2.2.3 Current VCU FCS

The current, third generation, FCS does not have a significant architecture change from the previous iteration. Most improvements were to the flight control algorithms to support high performance jet turbine UAVs. A few new expansion boards were developed but they all maintained the same general layout and capabilities. Improved power regulators and ADCs were the main differences from the older expansion board.
Support for newer and more compact IMUs were introduced, these include the MIDG and MIDGIIMUs from Microbotics Inc [7]. Support for the Copilot IR sensors was also reintroduced to lower the overall system cost and support rail launched aircraft that impose excessive G forces to the MEMS based IMUs.

Software improvements included waypoint navigation with cross-track error compensation, PID gain scheduling for better performance over a larger airspeed range, and some software filtering of barometric sensors. Great improvements to the ground control system were also introduced. A client/server system written in C# allows multiple clients to control various aspects of the UAV. Flight log analysis and graphing tools were also developed.

![Figure 2.3: Current VCU FCS. Suzaku T expansion board with Suzaku-S mounted.](image)

Current development has forked into two radically different systems. On one end of the spectrum, there is a high performance FCS using multiple FPGAs and a hardcore PowerPC processor. This system is currently under development and is expected to have a control loop running at 200 Hz with sophisticated analog sensor filtering. On the other end is the miniFCS low cost flight control system that
is described in this thesis. It has been designed to be low cost and have a small footprint while maintaining or improving performance of previous systems. The new miniFCS will enable the use of smaller, cheaper aircraft and allow flight verification of new multi-UAV cooperative algorithms that would be cost prohibitive with other systems.

2.3 Commercial Autopilots

2.3.1 MicroPilot MP Series Autopilots

MicroPilot offers a popular range of autopilots for various applications. The MPx028 series shares a common hardware platform but have large price differences. This can be attributed to features that have been disabled and license restrictions in lower priced models. The MP series autopilots use PID based control loops [8] at a rate of 30 Hz. Some models support autonomous takeoff and landing. The physical dimensions are 10cm x 4 cm x 1.5 cm and weigh about 28 grams. Power usage is also rated at 140mA @ 6.5V for all MPx028 models [9, 10, 11].

The MP1028G is the lowest cost autopilot from MicroPilot. It is priced at $1,500. Even though the hardware platform is very similar to the more expensive model, it has many features disabled. AGL sensor cannot be used. Compass modules cannot be integrated. The data log has been limited to 100kB. In flight programming of waypoints cannot be done. All mission programming must be performed on the ground. GPS updates are only available once per second. Ground station software is not included and user programming is not possible [9]. This model is not suitable for research purpose because of all the limitations imposed.

MicroPilot also offers the full featured MP2028G. The base system is priced at
$8000. This model has a larger data log size (1.5MB), waypoints can be reprogrammed in flight, and supports airspeed and altitude hold. It is also capable of various launch and recovery modes. These include hand launching, runway takeoff and landing, bungee cord launch and parachute landings [10]. A complete bundle suitable for research is available. It includes a radio modem, ground station software, ADC board, AGL sensor, and an SDK is available for $13,005. A single use version of this autopilot is also available for $2,000 per flight [11].

The MP2028G’s navigation system employs a 1 Hz GPS module like its cheaper version but DGPS capability is added. It has a 1000 waypoint buffer with altitude and airspeed set on a per waypoint basis. A servo can be controlled when a waypoint has been reached in order to perform some action such as dropping a marker or beacon. The differential pressure sensor is capable of measuring airspeeds up to 500 kph. The maximum barometric measured altitude is 12000 m. A 6-DOF IMU is integrated onto the system board. It is composed of 2G 3-axis accelerometers and 150° per second 3-axis rate gyros.

The control system features 30 Hz PID based controllers with gain scheduling for improved performance over a range of airspeeds. Aileron to rudder feed forward is used to improve turning capabilities. Altitude hold during turns is also improved by using aileron to elevator feed forward. User definable PID feedback and look-up tables can be implemented. Up to 24 servos can be controlled with a 50 Hz update rate and 11-bit resolution. Servos use a separate battery than the autopilot. An integrated RC override allows for manual control in the event of a system failure.
2.3.2 Cloud Cap Piccolo SL

The Piccolo SL is the latest thin form factor autopilot from Cloud Cap. It is 13.0 x 5.9 x 1.9 cm and weighs 110 grams. This includes an integrated RF data Link. Cloud Cap offers RF data links in 5 frequency bands; these include 900 MHz and 2.4 GHz ISM bands. Power consumption is 4 watts with the radio modem and about 2 watts without. The input voltage can range from 0 to 30 Volts. The autopilot is built around a 40 MHz PowerPC microprocessor with hardware floating point. The system integrates a 4 Hz GPS receiver, 3-axis rate gyros, and 3-axis accelerometers. Optional external magnetometer modules are available. Ported static and pitot pressure sensors are also integrated. The unit offers three RS232 ports used to interface with payloads. Up to 14 GPIO lines available for servo control. Four of the GPIO lines can be configured as analog inputs. The on board ADC has 10-bit resolution with a full scale range of 0 to 5V. A CAN bus is available as a general interface and for simulation [12, 13].

The Piccolo autopilot system uses a ground station server that manages the wireless links to one or more autopilots, supplies differential GPS corrections, and relays command and control data from the Piccolo Command Center to the autopilot. The
ground station server also relays manual pilot control from a Futaba transmitter via the main RF link. This allows the safety pilot to control any vehicle but will not allow safe control in the event of an autopilot failure or RF link loss. Some command center features include flight planning, waypoint insertion, multiple aircraft display and control, 3-D terrains and views, and integration with mapping services [12, 14].

This particular autopilot is popular within the research community because of its flexibility. It can be programmed using Simulink, offers partial hardware-in-the-loop simulation, flight visualization using FlightGear, and the firmware and command software source code is available through licensing agreements. This allows for research institutions to fully augment and or replace any command or control system. It features sophisticated PID control algorithms for fixed wing aircraft and adaptive neural network control technology, developed by Guided Systems, for rotary wing aircraft [15].

Specific pricing was not available at the writing of this document, but the system is not inexpensive. All feature sets can licensed individually or in bundles.

Figure 2.5: Cloud Cap Piccolo SL Autopilot. Picture courtesy of Cloud Cap Technology (http://www.cloudcaptech.com).
2.3.3 Procerus Technologies’ Kestrel 2.4

The Kestrel is one of the smallest autopilots available. This autopilot was developed at Brigham Young University and later commercialized. It is 5.1 cm x 3.5 cm x 1.2 cm and weighs 16.7 grams, not including the modem and GPS receiver. Power consumption is 0.80 W (w/o radio). Some of the notable features include smart loitering for surveillance, multiple UAV support, on board 6 DOF IMU, temperature measurement for sensor compensation and two axis gimbal support. The autopilot supports various autonomous modes of operation, these include takeoff and landing, home, loiter, rally, and waypoint navigation. In addition to the standard modes, it also supports pilot-in-the-loop altitude and speed hold modes [16].

The Kestrel autopilot uses conventional PID control algorithms and offers in-flight gain tuning capabilities. PID controller performance can be graphed in real time to aid with the tuning process. The PID controllers also implement gain scheduling to increase the airspeed envelope.

The system is based around the Rabbit 3000 8-bit microcontroller running at 29Mhz. The autopilot software is written in Dynamic C, a C variant from Rabbit Semiconductor. Noteworthy features of Dynamic C are new language constructs for multitasking. The costate and slice constructs can be used to perform cooperative and preemptive multitasking. An SDK that includes encrypted libraries for all flight control subsystems is available. Unfortunately, only the source for the main loop is available. This makes it impossible to augment any software subsystem. Full custom versions of any module will need to be written even if a trivial change is required.

The base price for the autopilot board is $5,000. To do any meaningful research, the SDK and ground station software needs to be purchased at $7500 and $3,495 respectively. Remote and ground station radio modems are also required and cost
$575 for the pair. Total cost for the bundle totals to $16,575 [17].

Figure 2.6: Kestrel 2.4 Autopilot. Picture courtesy of Procerus Technologies (http://www.procerusuav.com).

2.3.4 UNAV 3500FW

The UNAV 3500FW is the least expensive of available commercial autopilots. A complete system including the autopilot, GPS receiver module, telemetry radio and ground station software can be purchased for $3,000. It has dimensions of 10.2 cm x 5.1 cm x 1.9 cm and weighs 35 g and uses about 0.6 W. The attitude control loop runs at 50Hz and uses standard PID controllers. The system features an integrated 5 DOF IMU and includes a GPS receiver with 5 Hz update. Up to 64 waypoints can be programmed. Payload control servos can be triggered manually or by waypoint arrival. Support for a pan/tilt camera is also available. An on-board autopilot/RC switch allows command transfer from without the addition of any additional hardware. A simple ground control station is included. It provides with a map display capable of displaying and editing waypoints, and tracking vehicle position. A virtual cockpit with airspeed, altitude and artificial horizon is also available [18].
2.4 University Developed Autopilots

2.4.1 AggieAir

In [19], the Center for Self-Organizing and Intelligent Systems at the Utah State University presents a hardware and software architecture for low cost, miniature, fixed-wing autonomous UAVs. The AggieAir autopilot system is composed of three distinct but interdependent modules; these are the AggieCap, AggieNav, and AggiePilot. The AggieCap is responsible for payload management and system control. It runs on a Gumstix computer attached to the AggieNav board. It is capable of controlling a pan and tilt camera system and relaying images to the ground over a WiFi data link using the rsync protocol. AggieCap also adds enhanced Kalman filtering to sensor data from AggieNav. The AggieNav is a navigation sensor suite that contains a 6-DoF IMU, GPS and compass module, as well as dual pressure sensors for estimation of altitude and airspeed [20]. AggiePilot is the Paparazzi autopilot system [21] with the addition of using the JAUS command and control messaging standard.
The AggieNav system is based around the Atmel AVR32 UC3A microcontroller. It includes an Analog Devices ADIS1654 6-DoF IMU, Honeywell HMC6343 3-axis magnetic compass, uBlox LEA-5H GPS receiver, and two VTI SCP-1000 pressure sensors. 3.3V and 5.0V switching regulator supplies are used to power all systems. The command and control radio data link module was unspecified but the payload data link is implemented using a COTS Bullet2-HP from Ubiquity Wireless. The overall SWaP and cost of the system were also not available.

![AggieNav computer system](image)

**Figure 2.8:** AggieNav computer system. AggiePilot not pictured. Courtesy of the Center for Self-Organizing and Intelligent Systems at the Utah State University

### 2.4.2 Federal University of Minas Gerais

In [22], the UAV research group at the Universidade Federal de Minas Gerais present their design of a hand-launched small unmanned aerial vehicle. Their main goal was to implement a low cost, portable, and reliable aerial platform for ground reconnaissance. The vehicle was specifically designed such that the number of sensors and actuators were purposefully reduced. The sensor used are one SCP1000 pressure sensor from VVI Technologies, one 163PC01D75 differential pressure sensor from Honeywell, an
FMA Direct CPD4 horizon sensor, and a Garmin GPS-18 module. SWaP numbers were not given but the system weighs considerably more than 150g (the weight of the PDA). The system cost was also not provided.

An interesting aspect of the design at Minas Gerais is the use of a PDA as the system’s main computer. A Palm Pilot TX with a 312 MHz ARM based processor and 128 MB of ram was used to implement the control loop running at a rate of 5 Hz. The PDA has the advantage of having considerably faster processor and RAM than most low cost microcontrollers but lacks IO capabilities. The PDA used by this system only contains a single RS232 serial port for communications with all peripherals. This designed worked around this limitation by adding two additional microcontrollers to multiplex all sensor input and output through the single available serial port. One microcontroller is used to receive GPS data, sample signals from the two barometric sensors and an infrared horizon sensor. The other microcontroller is used to output PWM servo control signals and to control a multiplexer to switch from manual and autonomous flight. A major drawback to this approach large size of all components in relation to the aircraft; most components are mounted externally.

Lateral and longitudinal control is implemented using PID controllers. Two guidance guidance strategies based on vector fields were implemented, a simple switched attractor and a continuous vector field algorithm developed in [23]. The basic idea of the continuous vector field approach is to build a polygonal corridor that encompasses all flight path waypoints and generating a vector field within. Both of these approaches yielded poor cross tracking, particularly in the presence of wind. The continuous vector field was capable of keeping the vehicle within the defined corridor.
2.4.3 Tsinghua University TUAV1000

In [24], the UAV research group at the Department of Precision Instruments and Mechanology of Tsinghua University have developed a very small and lightweight flight control system weighing only 24 g. The system integrates an absolute pressure sensor for altitude measurements and a differential pressure sensor for airspeed measurements. A 3-axis gyroscope is used to stabilize the UAV. A GPS receiver for positioning information is directly integrated onto the main system PCB. The FCS is capable of controlling only vehicles with elevons and throttle; it is limited to three PWM channels outputs. A Simplex radio modem is used for a data link but neither the specific model, frequency, nor power usage was noted. Manual vehicle control is routed through the RF data link via the ground control server. This eliminates the need for a secondary safety pilot link and switch but prevents control of the aircraft in an event of a FCS system failure. Additionally, a small analog color camera with a wireless down-link was integrated into the system.
Both the lateral and longitudinal controllers use conventional PID controllers with gain scheduling based on airspeed. For lateral control, a roll rate controller is used. For longitudinal control, a pitch rate and throttle controllers are used. Waypoint navigation is supported with cross-track compensation. Cross-track compensation is implemented by using a PID controller that generates a bias heading that is added to the vehicle heading. The input to the cross track PID controller is the perpendicular distance, \( d \), from the rhumb line from the source to destination waypoint. If \( d \) is less than the GPS resolution, the bias term is not added to the vehicle heading. Test results indicate good altitude hold and waypoints navigation but cross track compensation did not function as intended.

![Image](http://www2.pim.tsinghua.edu.cn)

**Figure 2.10:** Tsinghua University flight control system. Courtesy of the Department of Precision Instruments and Mechatronics, Tsinghua University (http://www2.pim.tsinghua.edu.cn).

### 2.4.4 Georgia Tech Low-Cost Test-Bed

In [25], UAV researchers at the Guggenheim School of Aerospace Engineering at Georgia Tech present their development of a low cost flight control system intended
for undergraduate research. The main purpose of the article is to present the hardware solution; navigation and control software were not implemented. The system is composed of a, 5" by 3" four layer, sensor board where all sensors and associated electronics are integrated. GPS, magnetometer, and microcontroller COTS modules are mounted onto the sensor board. Extended Kalman filtering is used to estimate position, velocity, roll, pitch, yaw, angle of attack, and side-slip angle.

The heart of the FCS is a Rabbit 3000 8-bit microcontroller running at 29.8MHz. The microcontroller module includes 512KB of SRAM and 512KB of NVRAM. Several serial ports are used to communicate with other system components. An SPI bus is used to interface with two 12 and 11-bit ADCs. The 12-bit ADC is used to sample two on-board pressure sensors, rate gyros, accelerometers, and magnetometers. The 11-bit ADC is used to monitor control surface position by sampling a servo’s internal potentiometer voltage. A hall sensor and small magnets mounted on the propeller spinner are used to measure engine RPM. The hall sensor generates pulses that are measured by microcontroller. Asynchronous serial interfaces are used to communicate with the GPS module and ground station via a radio modem.

The inertial sensors incorporated include three ADXRS150 rate gyros, three ADXL202 dual axis accelerometers, and a three axis HMC2003 magnetometer. The GPS module used is a Motorola OnCore M12. This GPS module can provide standard NMEA or binary formatted absolute positioning information at a rate of 1 Hz. An MPXV5004D differential pressure sensor with a range to 3.92 kPa is used to measure airspeed. An MPXAZ4115 absolute pressure sensor is used to measure altitude. Command and control of the FCS occurs via a Microhard Spectra 910 wireless modem capable of data rates of 115200 bps.

The system, as presented functions as an INS and data logger. Navigation and
control are not implemented and were left as a future exercise. Complete SWaP and cost numbers were not available but the complete system is neither light nor small when compared to similarly capable systems. The radio modem alone weighs 420g and measures 3.7” x 4.3” x 1.7” and requires up to 450 mA with a 12V input [26].

![Figure 2.11: Georgia Tech’s low cost autopilot hardware. Courtesy of the Guggenheim School of Aerospace Engineering at Georgia Tech (http://uav.ae.gatech.edu).](image)

### 2.5 Commercial Auto Pilot Comparison

It is impossible to fairly compare the university developed autopilots. Each system has different design goals and is targeted to different vehicle platforms. Some units were designed to be low cost; others did not apply a cost constraint. None of the research groups defined what a low-cost system is. This is a highly subjective metric that will vary wildly given application, budgets, or research group. The university designed autopilots also did not offer complete dimensional, power, and price points.

Commercial autopilot vendors typically provided SWaP requirements and price points but had little detail about their navigation and control systems and sensors
used. This makes direct comparison of the autopilots in the preceding sections difficult. Rather, general suitability for the miniFCS requirements outlined in the introductory chapters is made. All commercial autopilots can be deemed unsuitable for the desired application solely on cost. None of the systems can come close to the $250 hardware cost limit. Ignoring the cost constrained, only the MicroPilot MP2028G meets all the designed requirements outlined in section 3.1. The Procerus Kestrel system was also light weight enough, had low power requirements, but has a limited SDK. Unfortunately this unit does not integrate a safety pilot override switch. The Piccolo SL’s SWaP requirements prevents it’s integration into small UAV payload bays and it’s power consumption is higher than desired. The UNA V 3500FW’s cost is attractive but an SDK is not available and computational performance is limited by its 8-bit MCU. The table below summarizes the SWaP, programmability, RC override integration, and cost of the commercially available autopilots.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MP2028G</td>
<td>10 x 4 x 1.5</td>
<td>28</td>
<td>0.91</td>
<td>yes</td>
<td>yes</td>
<td>$13,005</td>
</tr>
<tr>
<td>Piccolo SL</td>
<td>13 x 5.9 x 1.9</td>
<td>110</td>
<td>2.00</td>
<td>yes</td>
<td>no</td>
<td>NA</td>
</tr>
<tr>
<td>Kestrel 2.4</td>
<td>5.1 x 3.5 x 1.2</td>
<td>16.7</td>
<td>0.80</td>
<td>yes</td>
<td>no</td>
<td>$16,575</td>
</tr>
<tr>
<td>UNA V 3500FW</td>
<td>10.2 x 5.1 x 1.9</td>
<td>35</td>
<td>0.65</td>
<td>no</td>
<td>yes</td>
<td>$3,000</td>
</tr>
<tr>
<td>miniFCS</td>
<td>8.5 x 5.5 x 2.0</td>
<td>32</td>
<td>0.75 w/radio</td>
<td>yes</td>
<td>yes</td>
<td>&lt; $250</td>
</tr>
</tbody>
</table>

Table 2.1: Commercial auto pilot comparison.

aIncludes autopilot board, GPS module, radio modem, ground station software and SDK.
2.6 Vector Field Flight Paths

In recent years, many university autopilots [23, 25, 27, 28, 29, 30] have used some form of vector fields to generate flight path trajectories. Most have used two basic primitive algorithms to fly circular and straight line trajectories. In this section, these two navigation methods for path following are introduced. Different flight paths geometries can be generated by morphing and switching between these primitives [27]. In section 4.6, implementation details are discussed for similar direction fields generated by the miniFCS navigation software.

2.6.1 Switched Simple Attractor

A simple attractor potential function is defined as $\Phi_i(q) = \frac{1}{2} \alpha d^2 q$ where $q = [x, y]$, $d$ is the euclidean distance between the vehicle and destination waypoint, $q_i = [x_i, y_i]$. The vehicle simply follows the negated gradient of $\Phi_i$, $u_i(q) = -\alpha [x - x_i, y - y_i]$. The orientation of $u_i(q)$ is the desired heading and the magnitude is the desired speed. When the vehicle comes within a set arrival range of the destination waypoint, the vector field is switched by incrementing $q_i$. If a constant vehicle speed is maintained by the FCS, then only the direction field of $u_i(q)$ needs to be computed [22]. The direction field for the simple attractor is shown in figure 2.12.

2.6.2 Loitering

In [27] the Research and Engineering Center for Unmanned Vehicles at the University of Colorado at Boulder develops general techniques for constructing vector fields for UAV guidance. They use properties of Lyapunov stability to generate simple, globally stable 3D vector fields. The resulting vector field generates flight trajectories that
smoothly veer into a circular loiter as the radial distance of the vehicle approaches the loiter radius. The circular patterns can be warped into elongated shapes to generate oval racetrack loiter patterns. Similar approach to generating vector fields has been used by [28, 29, 30] as well as the navigation algorithm for the miniFCS described in section 4.6.1. In the figure and equations below,

\[ r = \text{vehicle position vector}, \]
\[ r_n = \text{vehicle position component normal to the loiter plane}, \]
\[ r_t = \text{vehicle position component in the loiter plane}, \]
\[ \dot{r} = \text{vehicle velocity vector}, \]
\[ \dot{r}_d = \text{desired vehicle velocity vector}, \]
\[ v = \text{desired speed}, \]
\[ \rho = \text{loiter circle radius}, \]
\[ \gamma = \text{relative circulation vs. contraction weighting factor}, \]
\[ \hat{n} = \text{unit normal to the loiter plane}, \]
\[ I = \text{identity matrix}. \]
Lyapunov stability theory is used to construct potential function 2.1 that has a zero level when the normal component, $\bar{r}_n$, of $\bar{r}$ equals zero and the magnitude of the tangential component, $\bar{r}_t$, equals the desired radius. The potential function is positive everywhere else.

$$V_F(\bar{r}) = \frac{1}{2} \left( \bar{r}_n^T \hat{n} \right)^2 + \frac{1}{2} \left( \bar{r}_t^T \hat{r}_t - \rho \right)^2 \tag{2.1}$$

$$\frac{\partial V_F}{\partial \bar{r}} = r_n \hat{n}^T + (r_t - \rho) \hat{r}_t^T \tag{2.2}$$

The vector field equation 2.3 is constructed of a contraction and a circulation term. The contraction term produces a vector field component with a direction opposite of the gradient of the potential function $V_F$, eq. 2.2. $\Gamma(\bar{r})$ is an identity matrix scaled by the velocity normalization equation 2.5. The second term is a circulation vector field component $S(\bar{r})$. It is also scaled by the velocity normalization equation. This component is always normal to the gradient of $V_F$ therefore it does not contribute to the contraction rate over time.

$$\dot{\bar{r}}_d = - \left[ \frac{\partial V_F}{\partial \bar{r}} \Gamma(\bar{r}) \right]^T + S(\bar{r}) \tag{2.3}$$
\[
\Gamma(\tau) = \frac{1}{a(\tau)} I; \quad S(\tau) = \gamma \frac{\hat{n} \tau_t}{a(\tau)} \tag{2.4}
\]
\[
a(\tau) = \frac{1}{v} \left( r_n^2 + (r_t - \rho)^2 + \rho^2 \gamma^2 \right)^{\frac{1}{2}} \tag{2.5}
\]

When the vehicle is on the loiter circle, the contraction term \( \frac{\partial V}{\partial r} \Gamma(\tau) \) will equal 0 and the magnitude of the circulation term, \( |S(\tau)| \), will equal \( v \). As \( \tau \) becomes larger, the contraction term \( \frac{\partial V}{\partial r} \Gamma(\tau) \to v \) and the circulation term \( |S(\tau)| \to 0 \). The magnitude of \( \gamma \) controls the relative strength of the contraction and circulation terms, modifying the abruptness of the transition from approaching the loiter radius to loitering. Its sign controls the circulation direction.

### 2.6.3 Straight Path Following

In [28] a vector field of ground track headings is constructed to reduce the amount of ground track error caused by wind and other disturbances. When the vehicle is far from the rhumb line between to waypoints, the objective is to fly toward the line. As the vehicle approaches the desired line path, its course should transition from approaching the path to flying along the path.

In figure 2.14, the transition area is demarked by the dashed line. This transition area exists on both sides of the rhumb line which lies on the x-axis. When outside of the transition area, the vehicle will approach the rhumb line at a constant angle. The commanded course, \( \chi^c \), is defined as

\[
\chi^c = \chi^f - \rho \chi^e. \tag{2.6}
\]

Where \( \chi^f \) is the heading from the originating to destination waypoint, \( \chi^e \) is a constant entry heading from 0 to \( \frac{\pi}{2} \), and \( \rho \) can take the value of \( \pm 1 \) depending on
what side of the rhumb line the vehicle is in. When the vehicle is inside the transition area, the commanded course becomes:

\[ \chi^c = \chi' - \chi^e \left( \frac{\epsilon}{\tau} \right)^k - \left( \frac{kS\chi^e}{\alpha\tau^k} \right) \epsilon^{k-1} \sin \chi \] (2.7)

In equation 2.7, \( S \) is the speed of the vehicle, \( \epsilon \) is the lateral tracking error, \( \alpha \) is known constant, \( \tau \) is the transition boundary distance, \( k \) is transition gain, and \( \chi \) is ground track course. The parameters \( \chi^e \), \( \tau \), and \( k \) can be tuned to achieve a desired transition performance. A formal proof that the lateral tracking and ground track course errors approach zero asymptotically using Lyapunov arguments is given in [28]. A similar approach to compensate for cross-track errors due to wind and other disturbances is used by the miniFCS. The field generation geometry used by the miniFCS can be found in section 4.6.2.
Chapter 3

Hardware Platform

The miniFCS has been developed to be a low cost platform that maintains or exceeds the level of performance of previous generation VCU flight control systems. To achieve this, the components were carefully selected to meet platform requirements. The system is built around the Atmel AVR32 UC3 microcontroller. GPS positioning information is received from a Locosys LS20031 GPS receiver. Airspeed and altitude are measured using two Freescale barometric transducers. Three axis thermopiles are used for roll and pitch estimation. The barometric and thermopile sensors are analog and require some basic signal conditioning before being sampled by the UC3’s on-chip ADC. Telemetry, command, and control are sent and received via an on-board 900 MHz radio modem. The miniFCS also contains a built-in RC safety switch that is used to transfer vehicle control to the safety pilot in the event of a system failure.
3.1 Requirements

When designing the miniFCS flight control system a set of hardware constraints were applied in order to be able to use the system for its intended purpose. These requirements were set using experience gained during operation of legacy VCU flight controllers and projected requirements for target airframes and applications. Constraints were placed on the system’s microprocessor, sensors, SWaP, and complete system cost.

The selected microprocessor or micro controller used must be fast enough to run the main control loop at 50 Hz. This includes measurement and processing of all sensor inputs, ground communications, navigation and attitude control. The choice of processor may also be constrained by on-board features, packaging, and power requirements.
The system must be able to measure barometric altitude to 450 meters (1500 feet), higher altitude measurements should be available from GPS data. Airspeed measurements should range from 0 to 200 knots. Pressure sensors must be voltage compatible with the ADC selected and have a small enough footprint in order to fit within the PCB size constraints.

GPS data must be available with an update rate of 5 Hz and have better than 10 m accuracy. The GPS module selected must fit within the PCB size constraints. Off-board modules should have a self contained antenna and be able to transmit data using a logic level asynchronous serial standard. The off-board module should be no larger than 3 cm x 3 cm x 1 cm, including the antenna.

Communications with the ground control system must use a radio modem operating at a frequency other than 2.4GHz. The selected radio modem must not have harmonics or subharmonic in the 2.4 GHz band. These frequency constraints are imposed because the safety pilot RC link operates within the 2.4 GHz band. Radio telemetry should be fast enough to report the vehicle status information at a rate of at least 5 Hz. Vehicle status must at minimum include position, attitude, and velocity information. Because the ultimate use of this flight control system is for collaborative UAV applications, the radio modem should be able to participate within a mesh network environment.

An off-board safety switch may not fit into the small avionics compartment bay of possible target vehicle platforms. Wiring of the safety switch would also require additional space. Because of space constraints, the safety switch should be integrated onto the miniFCS PCB. The safety switch should be able to select from two sets of inputs, one for the safety pilot and the other for the autopilot. The switch must, at minimum, switch control for four servo channels (ailerons, elevator, throttle, and
rudder).

Aircraft attitude must include at least roll and pitch estimation. This could be accomplished using IR sensor or an IMU unit. IMU selection will be limited because of the target cost of the complete system.

In order to perform collaborative UAV research within the limited funds of the VCU UAV laboratory, the complete system should cost no more than $250 per single unit. The cost constraints also suggest that the vehicle platform also be low cost. This limits the size of the vehicle, therefore the complete system should weigh under 100 grams and be no larger than 10 cm x 10 cm x 5 cm in order to fit within small payload compartments. Power usage should be less than 100 mW as to not significantly impact the endurance of an electric powered UAV.

3.2 Microcontroller

The microcontroller selected for the miniFCS platform is the AVR32 UC3A1. UC3 microcontrollers are designed for high computational throughput in highly integrated applications. The microprocessor core implements integer and fixed point DSP arithmetic with single cycle multiply and accumulate instructions. Floating point maths functions are optimized in assembly. Context switches only take 11 cycles, this includes placing register contents onto the stack. The UC3 features 64 KB of on-board single cycle access SRAM. Peripheral read-write access is atomic. Up to 512 KB of internal high speed flash memory is available. This memory has 0 wait state access when operating at speeds of up to 33 MHz. One wait state is required when operating at speeds up to 66 MHz. This penalty is mitigated because read operations from flash memory are pipe-lined, allowing for burst reads from sequential memory locations. These features allow for predictable and deterministic real-time control [31].
The UC3A1 provides a 7 channel PWM controller with a 20-bit counter per channel. Each channel has access to 11 modulo N clocks and two linear dividers. Period and duty cycle can be independently controlled. Output waveforms can be either left or center aligned. All channels double buffer writes to control registers. This ensures that at least one full period and/or duty cycle has been output and prevents glitching. Having dedicated PWM controllers with these features allows for simple and accurate movement of control surface servos.

An important innovation from Atmel is their Peripheral DMA Controller. The PDC transfers data between on-chip peripherals and off-chip memories, not memory to memory transfers. The main advantage of the PDC is that transfers avoid CPU intervention when transferring data to peripherals such as USART and SPI controllers. Each PDC channel contains a 32-bit memory pointer and a 16-bit transfer counter. The memory pointer register holds the memory address for the next transfer. The pointer is automatically updated after each transfer by either 1, 2, or 4 depending on the DMA transfer mode (byte, half-word, or word). The transfer counter is programmed with the number of transfers to be performed. This register is automatically decremented and can be read at any time to determine the number of remaining transfers. A memory reload pointer and transfer counter reload registers are programmed with the next memory address and transfer size for the next block transfer to occur. The memory pointer and transfer counters are updated with the reload values when the transfer counter reaches zero. This allows the PDC to be used with multiple buffers. An interrupt is generated when transfers complete.

Four USARTs are available for serial communications to various devices. They support 5 to 9 bit full duplex operation in either synchronous or asynchronous modes. A fractional baud rate generator is available to reduce baud rate error when using
CLOCKS RATES THAT ARE NOT MULTIPLES OF 1.8432 MHZ. HARDWARE RTS-CTS HANDSHAKING CAN BE USED. THE USARTS DO NOT FEATURE A FIFO; THIS WILL CAUSE EXCESSIVE PROCESSOR OVERHEAD WHEN USING INTERRUPT BASED TRANSFERS ON MULTIPLE PORTS AT HIGH BIT RATES. TO ELIMINATE NEARLY ALL INTERRUPT OVERHEAD, EACH USART SUPPORTS CONNECTIONS TO TWO PDC CHANNELS; ONE FOR RECEIVING AND THE OTHER FOR TRANSMITTING. THIS ALLOWS FOR BUFFERED TRANSFERS WITHOUT PROCESSOR INTERVENTION. THE MINIFCS USES THE USARTS FOR GROUND CONTROL AND TELEMETRY THROUGH A RADIO MODEM AND TO RECEIVE GPS DATA.

THE UC3A1 MICROCONTROLLER INTEGRATES A SINGLE ON-BORD 10-BIT SUCCESSIVE APPROXIMATION REGISTER ADC. ITS INPUT IS MULTIPLEXED TO ALLOW CONNECTION OF EIGHT ANALOG SIGNALS. THE ADC CAN BE OPERATED WITH A CLOCK UP TO 5 MHZ WHEN IN 10 BIT MODE. EACH CONVERSION REQUIRES 13 CLOCK CYCLES GIVING A MAXIMUM THROUGHPUT RATE OF 384 KSPS. CONVERSIONS AND DATA TRANSFERS CAN OCCUR WITHOUT CPU INTERVENTION WHEN USING THE PDC CONTROLLER. THIS ALLOWS SAMPLE DATA TO BE STORED TO MEMORY WHILE OTHER TASKS ARE RUNNING ON THE CPU. SIX OF THE AVAILABLE ADC CHANNELS ARE USED ON THE MINIFCS, THREE FOR THE XY AND Z THERMOPILES, TWO FOR THE STATIC AND IMPACT PRESSURE SENSORS, AND ONE TO MONITOR FLIGHT THE BATTERY VOLTAGE.

OTHER MICROCONTROLLERS WERE EVALUATED AND REJECTED FOR VARIOUS REASONS. THESE INCLUDE MICROCONTROLLERS WITH ARM AND MIPS CORES FROM VARIOUS VENDORS. MICROCONTROLLERS WITH ARM CORES EITHER REQUIRED EXTERNAL SRAM AND NVRAM OR DID NOT HAVE A SUFFICIENT NUMBER OF PWM CONTROLLERS. THEY HAD THE ADVANTAGE OF HAVING HIGHER CLOCK FREQUENCIES BUT AT A SIGNIFICANT COST INCREASE. UC3A1 PARTS CAN BE OBTAINED IN SINGLE UNITS COSTING AS LITTLE AS $7. SINGLE UNITS OF ARM BASED PROCESSORS TYPICALLY COST DOUBLE. THE ARM SOLUTIONS WILL ALSO REQUIRE MEMORIES AND/OR SUPPORTING ICs TO OBTAIN THE REQUIRED FEATURES.

32-BIT MICROCONTROLLERS FROM MICROCHIP WERE ALSO CONSIDERED. THEIR PIC32 DEVICES
have a very similar feature set and price point as the Atmel counterparts. These microcontrollers were rejected for three reasons. First, their architecture is about one year behind the Atmel parts. The software framework was more immature and the development community was smaller. Second, the development environment is not free. The Hi-Tech C compiler and MPLAB IDE are completely proprietary and cost $1500. An evaluation version is available but does not perform optimizations, context switches are artificially slowed and function parameters cannot be passed via registers [32]. Third, the PIC32 does not have an equivalent to Atmel’s PDC. This will significantly impact CPU overhead when performing ADC operations and handling of serial communications.

Atmel provides a clear pin compatible upgrade path should future FCS developments exceed the processing power of the current UC3A1. Parts with larger memories are available and an IEEE 754 single precision FPU for the UC3 product family is being introduced [33].

3.3 Sensors

3.3.1 Airspeed and Altitude

Two pressure sensors are used on-board the miniFCS. One sensor is a differential pressure transducer, model MP3V5004DP, the other is an absolute pressure sensor, model MP3H6115A. Both are manufactured by Freescale semiconductors. These particular sensors were selected because of their low cost, small size and supply requirements. These sensors can be acquired as single units for under ten dollars. They have a small SMD footprint, about 1 cm², saving PCB space. These transducers can operate with supply voltages ranging from 2.7 to 3.3 volts. Having this capability this allows for a
single regulator to be used for the analog section of the PCB.

The differential pressure sensor is used to measure the vehicle’s airspeed. It has a linear transfer function \[ V_{out} = V_s(0.2P + 0.2) \]; this simplifies signal processing. This sensor is temperature compensated from 10°C to 60°C. See page 59 for details of how the measure pressure differential is used to calculate an airspeed.

The absolute pressure sensor is used to measure the vehicle’s altitude. It has a linear transfer function \[ V_{out} = V_s(0.009P + 0.095) \]. This sensor is temperature compensated from -40°C to 125°C. This is important because bay temperatures can vary significant during operation. To use the ADC’s full scale in for measuring altitudes in the vehicle’s altitude envelope of -50 to 450 meters, the output from the sensor is offset by -2.5361V and multiplied by 18.7. See page 57 for details of how the measured pressure is used to calculate an altitude.

### 3.3.2 Three-Axis Thermopiles

Three axis thermopiles where selected for attitude estimation. At the time of development 3-axis rate gyros, accelerometers, and magnetometers cost significantly higher than the thermopile solution. Two thermopiles are oriented at 180° apart on each axis. The temperature differences on the x and y axis can be used to estimate roll and pitch. The z axis sensor is used to obtain the maximum temperature difference between the sky and ground. The z measurement is used for calibrating measurements from the x and y axis. When the thermopile sensor on the positive end of an axis is pointed towards the ground, a voltage greater than half the supply voltage is output. When pointed to the sky, a voltage less than half the supply is output. Thermopiles have slow response times. Typical time constants are around 20 to 30 ms [36]. At best this gives an effective bandwidth of 8Hz. This may seem low but the UAV roll
and pitch rates are well below this. Sensor outputs are low pass filtered before being sampled by the UC3’s on-chip ADC.

Thermopile sensors are available from FMA Direct and Range Video. Both units have similar specifications [37]. These sensors require a 3.3 V supply. The output measurement is offset by half the supply voltage. The Range video xy and z sensors weigh about 4 grams combined, ten grams for the FMA versions. The xy sensors measure about 2 cm x 2 cm and the z sensor is 2 cm x 1 cm. Total power consumption is 7 mA for the set. The copilot xy sensor can be purchased for $43 and the Range Video xy-z sensor set for $70.

3.3.3 GPS

To get position information, a GPS module is used. The selected module is the LOCOSYS LS20031 GPS receiver. It was selected for its size, cost, and update rate. This module is only 3.0 cm x 3.0 cm x 0.6 cm and weighs 12 grams. This includes a board mounted ceramic antenna. Single units can be obtained for under $50. It requires a 3.3V supply and 41 mA. Standard NMEA sentences are used to send telemetry over a logic level asynchronous serial port. This unit is capable of a 10 Hz update rate with position accuracy of 3 meters [38]. Because the miniFCS uses standard NMEA sentences for GPS updates, this module can be replaced with another without any software modification.

3.4 Signal Conditioning

The miniFCS board contains a section to condition all analog signals prior to sampling. The signal conditioning section contains six low pass active filters. These are
used to filter signals from the three IR sensors, airspeed sensor, and altitude sensor. The filtered signal from the altitude pressure sensor passes through a differential amplifier, see figure 3.2. As the circuit name implies, this circuit amplifies the difference of the two inputs. A negative offset and gain is applied to the altimeter sensor in order to use the full range of the ADC. The differential amplifier uses a secondary op amp in a voltage follower configuration as the reference voltage used for the negative offset. The sixth low pass filter is connected to the output of a voltage divider circuit. Input to this circuit is connected to the flight battery to monitor its voltage.

![Figure 3.2: Signal conditioning circuit used for the barometric pressure sensor. Contains a low pass filter and an offset and gain stage.](image)

All the low pass filters are constructed using the Sallen-Key structure and have been designed to have a 3dB point at 112Hz. The actual filters cutoff frequencies that range from 80 to 110 Hz. This can be attributed to variances in the capacitors
and resistors used to implement the filters. Texas Instrument’s OPA4376 operational amplifier is used because it is relatively inexpensive, contains four op amps per IC, has low offset voltage and voltage noise [39]. Low value, thin film resistors are used to reduce Johnson-Nyquist noise.

The filter response begins as expected for a second order low pass filter. After the -3 dB point, the filter attenuates at 40 dB per decade. This should continue, but at a point the gain begins to increase at 20 dB per decade. This occurs at the frequency where the op amp’s closed loop output impedance, the open loop resistance divided by the op amp’s gain, is greater than the input resistance [40]. The output resistance combined with the feedback capacitor creates a high pass filter. At high frequencies, the op amp has an output resistance of about 100 ohms. This creates a high pass filter with a corner frequency at about 1.5 kHz. The gain begins to flatten at the frequency where the op amp’s open loop gain falls to 0. The data sheet for this particular op amp suggest this should happen around 6MHz. From the measurements taken, this is happening close to 100kHz. This is advantageous in this filter configuration.

The stop band leakage in the filter circuit can be reduced by increasing the resistors and decreasing the capacitors by a factor of 10. This will shift the corner frequency of the high pass filter to about 15 kHz, allowing attenuation of over 80 dB in the stop band. Doing so will add offset and increase Johnson-Nyquist noise. A passive low pass filter can be added at either the input or output in order to minimize the upward trend. This has the drawback of increasing cost, offset, noise, and decreasing the input impedance or increasing the output impedance.

It may seem that noise may not be attenuated sufficiently using the Sallen-Key Butterworth filter. The filter needs a stop band gain of roughly -60.2 dB in order for full scale noise to be filtered. The implemented low pass filters will attenuate high
frequency signals by only 38 dB. In practice, this is not an issue because the noise voltage is significantly less than full scale voltage. On the miniFCS, noise voltage at the filter inputs are less than 10 mVpp, typically 6 to 8 mVpp. A 50 mVpp noise signal will be attenuated to 0.63 mV; this is far less than volts per LSB of the on-chip ADC. A 13 bit ADC would be required to measure the 50 mVpp noise. High frequency noise with amplitudes of up to 256mV will be completely filtered.

![Analog Low Pass Filter Response](image)

**Figure 3.3:** Signal Conditioning low pass filter response.

### 3.5 Radio Modem

The radio modem selected for the miniFCS is the XBee-Pro 900 RF. This is a small module, 2.4 cm x 3.3 cm weighing 7 grams, that can be board mounted. It transmits at 900 MHz with 17 dBm of power. It has a range of up to 3 km with 2.1 dB gain dipole antennas. Ranges to 10 km can be achieved with higher gain antennas. Data rates of 156.25 kbps can be achieved under ideal conditions. It has a 3.3 V serial interface for communications to the microcontroller. It requires a 3.3 volt supply and
up to 180 mA [41]. These features make this module suitable for small embedded applications such as the miniFCS.

The radio modem can be operated in transparent or API modes. In transparent mode, two modules act a serial line replacement. In API mode, the system has control at the frame level. This allows for addressing specific modules to create a mesh network. The mesh network is self healing; nodes can drop or join at any time. Routes with the best round trip times are automatically discovered. Reliable data transmission is also available by means of acknowledgments. By using the mesh networking capabilities of the radio, vehicles can relay commands and telemetry to the intended destination. Potentially, this allows for multiple UAV communication to distances greater than the ground control radio range.

3.6 Power Supplies

The miniFCS’s has been designed to only use two voltage regulators for all components. The switching regulator, PTH08080WAH, from Texas Instruments is configured as a 3.3 volt supply. It is capable of delivering up to 2.25 A and is over 90% efficient when supplying over 0.5 A. It can be obtained for under $10 in single units. Using a module like this is advantageous because nearly all components required for the regulator are included on the module’s PCB. Only a single 100\(\mu\)F electrolytic capacitor is required at the input. It is used in the digital PCB section to power the microcontroller, radio modem, safety switch components, and the GPS module. It is not used to power the analog section of the PCB because it generates excessive high frequency noise. Figure 3.4 shows noise present at the regulator’s output. The digital supply regulator has noise with magnitudes of about 20 mVpp.

The Micrel mic5219-3.3BM5 3.3 volt linear regulator is used for the analog section
of the PCB. It is a low drop out and low noise regulator. It can be obtained for $1.70 in single units. This regulator can supply up to 500 mA. This is sufficient to power the IR and barometric sensors, and the op amps used in the signal conditioning section. Ferrite beads are used to filter high frequency noise that may be present from switching regulators in the vehicle. Figure 3.4 shows the noise present at the regulator’s output. This supply regulator has noise with magnitudes of about 6 mVpp.

The input voltage for miniFCS must range from 4.6V to 18V. Voltages below the minimum will cause the PTH08080WAH to shut down, disabling the miniFCS. With an input voltage of 5V, the miniFCS draws 150 mA of current with all systems activated, this includes the radio modem transmitting all possible telemetry.

![Fig 3.4: Digital and analog supply noise.](image)

### 3.7 Safety Switch

The miniFCS has a built in safety switch. It’s purpose is to switch control to the safety pilot in the event of an autopilot failure. The safety switch is controlled by a PWM signal from the safety pilot RC receiver. Pulses below 1500 µs will switch control to the safety pilot; longer pulses will switch to autopilot control. There is 200
µs hysteresis to prevent rapid switching when pulses are near the switching threshold. The safety switch supports six PWM control channels from both the receiver and the autopilot. An additional select channel from the RC receiver is monitored to control the switch.

The switch is implemented with two, eight channel 74HCT7541 tri-state buffers and an ATtiny 8-bit microcontroller. The tri-state buffers are connected in a bus topology and use the microcontroller to enable or disable the buffer outputs. All buffer inputs are pulled down. This prevents unused inputs from floating, preventing the output buffers from drawing excessive current. The tri-state buffers are Schmitt triggered [42]. This input hysteresis prevents noise from inadvertently switching control. This feature is useful when using inexpensive 72Mhz RC radios that have noisy PWM outputs. The buffer inputs are only tolerant to the supply voltage, 3.3 V in this case. If 5 V tolerance is required, the buffers can be switched with pin compatible LCX541 buffers, but hysteresis will be lost.

In addition to the select channel, the throttle channel from the RC receiver is also monitored. If RC radio link occurs, the throttle channel will output a preset pulse width. Two GPIO lines are used to signal the safety switch state to the autopilot. One bit is used to indicate auto or manual operation and the other to signal RC radio link loss.

3.8 Printed Circuit Board

A custom PCB has been designed to integrate all system components. The schematic was designed using PADS logic and the PCB was laid out using PADS Layout and Router. Board dimensions are 3.2" x 1.8". All components are surface mount with the exception of some headers and connectors. The board contains four conductive layers.
The top layer is used to route signals, the second is used for power planes, the third is a ground plane, and the bottom is for signal routing. Top and bottom layer have copper pours that are connected to ground. All digital signals are contained to the left side of the board. Analog signals are on the right and are physically separated. Analog and digital grounds are joined near the ADC inputs.

![Printed circuit board.](image)

Ferrite beads and bypass capacitors are used to reduce noise throughout the board. Bypass capacitor groups with 1206, 0805, and 0603 sizes are used. Capacitance values are also scaled with package size. This is done to achieve bypassing noise spread across a wide bandwidth. Figure 3.6 shows how impedance falls with frequency due to capacitance. Any inductance due to the package or traces will cause a positive slope in the impedance. The parallel combination is the effective impedance [43]. To reduce inductance, traces to capacitor terminals should be kept as short as possible. One terminal of a bypass capacitor is placed as close as possible to the IC supply pin, the other terminal is connected to the ground plane using a via either directly on the
terminal pad or as close as possible.

![Figure 3.6: Impedance of three capacitors in parallel.](image)

3.9 System Cost

The complete cost of the system falls below the target price of $250. At single quantities the cost is $243.26. This includes the price for all components mounted on the PCB (processors, buffers, op amps, resistors, capacitors, regulators, etc.), the PCB itself, GPS module, thermopile sensors, and the radio modem. Pricing for bulk quantities is tabulated in the table below. Prices for individual components were compiled primarily from Digikey.com and Mouser.com.

A large portion of the total cost comes from the GPS module and thermopile sensors. Costs for these components are fixed for quantities below 250. The cost of XBee radio modem does not fall significantly with large purchase quantities. Most of the savings come from the bulk rates available for the PCB components and the PCB itself.

U & I was selected to manufacture the PCB. They can manufacture a single miniFCS PCB for as little as $52 with a one week lead time. U & I will also perform electrical test of the finished PCB free of charge [44]. This reduces the likelihood of
receiving a defective PCB. The nearest price competitor is Advanced Circuits. They are able to provide PCBs for $66 each [45]. Electrical testing is an additional $140.

The miniFCS achieves significant cost reduction compared to the commercial systems discussed in section 2.3. The miniFCS costs 64 times less than the Krestrel solution and 10 times less than UNAV system. Compared to the current VCU FPGA based FCS, costs have been reduced from $1,000 (estimated) to under $250 while maintaining, extending, or improving system performance.

<table>
<thead>
<tr>
<th></th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 – 9</td>
</tr>
<tr>
<td>PCB Components</td>
<td>71.31</td>
</tr>
<tr>
<td>PCB (U &amp; I)</td>
<td>52.00</td>
</tr>
<tr>
<td>GPS Module</td>
<td>49.95</td>
</tr>
<tr>
<td>Thermopile Sensors</td>
<td>70.00</td>
</tr>
<tr>
<td>Radio Modem</td>
<td>44.95</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>243.26</strong></td>
</tr>
</tbody>
</table>

Table 3.1: System cost matrix.
Chapter 4

Flight Control Software

This chapter summarizes the flight control software requirements then delves into implementation details. The main control loop, communication subsystem, sensor software, navigation algorithms, and attitude control are discusses in the following sections.

4.1 Requirements

The miniFCS software is required to perform navigation tasks. It must be able to successfully guide the vehicle through a flight path determined by a set of waypoints. In addition to path following algorithms, the flight controller must be capable of flying loiter patterns. The software must also perform vehicle stabilization and attitude control by means of controlling the aircraft’s control surfaces.

The flight control software must also handle all communications to the ground. This includes command and control of the aircraft and periodic telemetry indicating the vehicle’s status. Communications must use the VACS protocol in order to remain compatible with the ground control software used at the VCU UAV laboratory.
Because the AVR32 UC3 microcontroller was selected as the computing platform, the FCS software must be written in C, compiled using AVR32-GCC and linked against Newlib. The software development environment for this microcontroller is AVR32 Studio, version 2.2. This is a customized version of Eclipse that is distributed by Atmel Corporation.

4.2 Main Control Loop

After the flight control system has been initialized, the software enters the main control loop. This loop executes at a periodic rate of 50 Hz. This rate was chosen because it is the update rate for the control surface servos. In the current configuration, it is possible to increase the main loop rate, but the system has only been tested at 50 Hz. The main loop rate is controlled by using one of the UC3’s 32-bit counters. The counter counter value is incremented at each clock cycle. This value is compared to the number of cycles in 20 milliseconds. Tasks in the main loop are only allowed to execute if 20 milliseconds have passed since the last time the loop began execution. The flow chart below illustrates the sequence of the various tasks in the main control loop. Sections that follow will delve more deeply into individual tasks.
The first task in the main loop is to calculate how much time has elapsed since the last execution began. This is required in the unlikely event that a time over-run occurs. The calculated time delta is used in all time dependent calculations rather than using a static 20 ms timing. Loss of GPS data is determined by aging the last received GPS packet. If GPS data has not been received within a predefined period,
the system will signal a fault.

The next task is to process any ground control commands or requests. This only occurs if a complete, error free, packet has been received. Next, all sensor data is updated. First, position and velocity GPS packets are processed and used to update navigation variables. Barometric samples are used to update airspeed and altitude. Samples from the IR sensors are used to calculate roll and pitch. Next, navigation and attitude control tasks execute. They are core of the flight control system. All the gathered sensor information is used to generate a target heading and move control surfaces to steer the aircraft to the desired course\footnote{In this document, heading and course are used interchangeably because the on-board sensors can only determine the ground course of the vehicle. There is no way to determine the direction the nose of the vehicle points.} and altitude. The last task is send periodic telemetry to the ground control system. Vehicle status updates can be sent at rates up to the main control loop rate.

\section{4.3 Ground Communications}

At the beginning of the main control loop, the communications subsystems checks for any unprocessed serial data from the on-board radio modem. Communications between the modem and software subsystem is facilitated by the peripheral DMA controller. Unlike traditional DMA controllers that transfer data between memories, the PDC transfers data between memory and peripheral devices such as the USART used for communications. Using the PDC is required because the UC3 does not have FIFOs for the UARTs and interrupt overheard for processing each incoming byte individually would consume excessive CPU time [46]. Incoming data is placed into a buffer while the processor is busy doing other tasks. When execution returns to the beginning of the main control loop, a buffer swap occurs. The PDC begins
placing data into a new buffer. At this time the buffer containing data is processed for ground control commands and requests. The GPS subsystem also uses this polled PDC UART technique to receive GPS updates.

Unlike polled nature of Rx UART transfers, UART transmittal uses the PDC to transfer chunks of data at a time from a ring buffer. Transmissions can occur at any time during the main control cycle. A custom UART write routine places the data to be transmitted into a circular buffer, updates the buffer tail, and initiates an UART PDC transfer if the PDC is not currently transferring data. When the PDC has completed transmission of a chunk of data, it triggers an interrupt. An interrupt servicing routine updates the head of the circular buffer and checks if there is any data queued. If there is, the ISR sets the beginning address of the data and length to the PDC and initiates a new DMA transfer. If the circular buffer is empty, the UART PDC interrupt is disabled.

The VACS message format is used for all communications to and from the ground. This is a simple standard that only defines a packet structure for transmittal over simple serial lines. Table 4.1 shows the packet structure. Incoming ground packets are parsed using a state machine. The check-sum is computed using an 8-bit Fletcher algorithm. The check-sum calculation includes all bytes between Sync B and Check Sum A. Any packets received with a check-sum error are quietly discarded.

The miniFCS has support to receive and transmit four basic message types. These include configuration, request, trigger commands, and telemetry reports. A configuration command packet is used to update various aspects of the system. They are used to define flight paths, navigation modes, set altitudes and airspeeds, configure PID controller gains, servo limits, etc. Request commands request flight controller configuration information. A trigger commands are used to trigger an event on the
aircraft. They can signal the UAV to snap a picture, set the ground level, head to the next waypoint, or save configuration data to NVRAM. Telemetry reports are used to send periodic information about the vehicle’s state to the ground. These are used to send position, velocity, attitude, internal variable states to the ground control system where they can be monitored and logged. Many message types have been defined to be compatible with current ground control software. A complete listing of supported messages and their structures can be found in the groundcomm.h header file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sync A (0x76)</td>
<td>byte</td>
</tr>
<tr>
<td>Sync B (0x63)</td>
<td>byte</td>
</tr>
<tr>
<td>Destination Address</td>
<td>byte</td>
</tr>
<tr>
<td>Source Address</td>
<td>byte</td>
</tr>
<tr>
<td>Message ID</td>
<td>short</td>
</tr>
<tr>
<td>Length</td>
<td>short</td>
</tr>
<tr>
<td>Payload Data</td>
<td>byte[Length]</td>
</tr>
<tr>
<td>Check Sum A</td>
<td>byte</td>
</tr>
<tr>
<td>Check Sum B</td>
<td>byte</td>
</tr>
</tbody>
</table>

Table 4.1: VACS packet structure. Multi-byte types are little endian.

4.4 GPS

GPS positioning information is received via a polled buffer using the PDC to transfer data from the UART. The NMEA communications standard from the National Marine Electronics Association is used because most GPS receivers will output in this format. This allows for interchangeability of GPS modules. The miniFCS only recognizes a small subset of message types; these are the GGA, VTG, and GSA message sentences.

NMEA sentences are ASCII text sequences that begin with '$' and end with a carriage return/line feed. Each sentence can not exceed 82 characters, including the line termination characters. Data in the sentence is separated by commas. Each
data field can vary in precision; for example a decimal number may contain 3 or 4 digits past the decimal point. The end of each sentence contains a check-sum field. It consists of an '*' and two hex digits representing an 8 bit exclusive OR of all characters between the '$' and '*' characters [47].

The GGA sentence is used by the miniFCS to update the latitude, longitude, gpsAltitude, and gpsTime navigation state variables. This only occurs when a valid GGA packet with 2D or 3D fix is received. Valid VTG sentences are used for updating gpsHeading and gpsGroundspeed navigation state variables. The yaw state variable is also updated from the gpsHeading because there is no sensor to give true yaw measurements. The GSA sentence is used only to obtain the GPS fix type (no fix, 2D or 3D). Below is an example GGA sentence with the data format given on table 4.2.

\$GPGGA,053740.000,2503.6319,N,12136.0099,E,1,08,1.1,63.8,M,15.2,M0000*64

4.5 Analog Sensors

The miniFCS is equipped with six analog sensors. These include two barometric and three IR sensors. A battery voltage monitor is also included. In order to use these sensors, the UC3’s on-board ADC is used to convert the analog signals into a usable digital representation. Signals are all single ended and range from 0 to 3.3 volts. A voltage divider is used to scale the battery voltage to be compatible with the ADC range. All analog signals map to a 10 bit code ranging from 0 to 1023. The sampled representation can then be used to determine useful information about the operating environment. The following subsections describe how the sensor readings are used to determine altitude, air speed, and attitude of the vehicle.
<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message ID</td>
<td>$GPGGA</td>
<td>GGA protocol header</td>
</tr>
<tr>
<td>UTC Time</td>
<td>053740,000</td>
<td>hhmmss.sss</td>
</tr>
<tr>
<td>Latitude</td>
<td>2503,6319</td>
<td>ddmm.mmmm</td>
</tr>
<tr>
<td>N/S indicator</td>
<td>N</td>
<td>N=north or S=south</td>
</tr>
<tr>
<td>Longitude</td>
<td>12136,0099</td>
<td>dddmm.mmmm</td>
</tr>
<tr>
<td>E/W Indicator</td>
<td>E</td>
<td>E=east or W=west</td>
</tr>
<tr>
<td>Position Fix Indicator</td>
<td>1</td>
<td>See Table 5.1-3</td>
</tr>
<tr>
<td>Satellites Used</td>
<td>08</td>
<td>Range 0 to 12</td>
</tr>
<tr>
<td>HDOP</td>
<td>1.1</td>
<td>Horizontal Dilution of Precision</td>
</tr>
<tr>
<td>MSL Altitude</td>
<td>63.8</td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td>M</td>
<td>Meters</td>
</tr>
<tr>
<td>Geoid Separation</td>
<td>15.2</td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td>M</td>
<td>Meters</td>
</tr>
<tr>
<td>Age of Diff. Corr.</td>
<td></td>
<td>Null fields when DGPS is not used</td>
</tr>
<tr>
<td>Diff. Ref. Station ID</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>Check-sum</td>
<td>*64</td>
<td></td>
</tr>
<tr>
<td>&lt;CR&gt; &lt;LF&gt;</td>
<td></td>
<td>End of message termination</td>
</tr>
</tbody>
</table>

Table 4.2: NMEA GGA data format[38].

### 4.5.1 Oversampling

The Atmel UC3 microcontroller contains an on-chip 10-bit ADC. For most purposes, this ADC provides sufficient resolution. For cases where increased resolution is required, oversampling to 11 and 12 bits has been implemented. The UC3 ADC has been configured with a clock of 4.032MHz and a conversion time of 15 ADC clock cycles. This yields a maximum sampling rate of 268.8 kSPS.

To achieve 11-bit resolution, four successive samples are summed. A two bit shift right is performed on the sum to obtain an 11-bit result. The Nyquist-Shannon sampling theorem tells us that the maximum signal frequency is 33.6 kHz. Similarly, to achieve 12 bit resolution 16 consecutive samples are summed. This sum is then right shifted by 4 bits. The maximum signal frequency is reduced to 8.4Khz. The 112 Hz analog low pass filters effectively reduce aliasing.
4.5.2 Altitude and Airspeed Pressure Sensors

The miniFCS is equipped with absolute and differential pressure sensors from Freescale Semiconductor. These transducers have a transfer functions that are linear to the measured pressure. The UC3’s on-chip ADC is used to convert the analog signal from the transducers to a digital representation that can be used to calculate the altitude and airspeed of the UAV.

4.5.2.1 Altitude Calculation

To calculate an estimate of the aircraft’s altitude, the exponential atmosphere equation[48] is used. This equation models how pressure or density of air varies with altitudes. This model is valid with altitudes below 86 km. Even though equation 4.1 is computationally expensive to implement in its full form, many of the equation variables reduce to constants when the operating altitude is limited within a single atmospheric level. The vehicle platforms, for which the miniFCS is targeted, only fly within the lowest atmospheric level.

\[
P = P_b \left[ \frac{T_b}{T_b + L_b (h - h_b)} \right]^{\frac{g_0 M}{R^* g_0}}
\]  

(4.1)

where:

- \(P_b\) = Static pressure (pascals)
- \(T_b\) = Standard temperature (K)
- \(L_b\) = Standard temperature lapse rate
- \(h\) = Height above sea level (meters)
- \(h_b\) = Height at bottom of layer b
- \(R^*\) = Universal gas constant for air: 8.31432 N·m/(mol·K)
- \(g_0\) = Gravitational acceleration (9.80665 m/s²)
- \(M\) = Molar mass of Earth’s air (0.028964 kg/mol)

At the atmospheric layer from 0 to 11,000 meters, \(P_b = 101325\) Pa, \(T_b = 288.15\) K,
\( L_b = -0.0065 \text{K/m}, h_b = 0 \). Equation 4.1 then simplifies significantly to equation 4.2. This equation transforms an altitude to an estimated pressure. This equation and the transfer function for the MP3H6115 [35] can be used to build table 4.3.

\[
P = 101325 \left( 1 - 2.25577 \times 10^{-5} h \right)^{5.25588} \tag{4.2}
\]

\[
V_{out} = V_s \times (0.000009 P - 0.095) \tag{4.3}
\]

The sensor output voltage is offset and then multiplied in the signal conditioning hardware. The nominal offset is -2.5361V and the nominal multiplier is 18.7. Using equation 4.2, a table of pressures for given altitudes is generated. The transfer function 4.3 is applied to the pressure to calculate a sensor output. The offset and multiplier signal conditioning is then applied. Finally, the conditioned signal is mapped into a 10 bit code. The altitude and 10-bit code have a linear relationship within the miniFCS’s altitude envelope. Fitting the data in table 4.3 yields the an equation that is directly implemented in the FCS software.

\[
Altitude = -0.492781 \times Sample_{10bit} + 455.9248 \tag{4.4}
\]

An advantage of calculating the altitude using this method is that it only requires a single multiplication and a single addition. Similarly the conditioned voltage can be mapped to 11 bit and 12 bit codes yielding the formulas 4.5 and 4.6 for altitudes ASL. Because the on-chip ADC only has 10 bit resolution, the altitude resolution is 0.493 meters. Although this is adequate for the altitude hold controller, higher resolution is desirable. Using 11-bit and 12-bit ADC would decrease the resolution to 0.246 and 0.123 meters respectively. Rather than using an external ADC, the altitude pressure sensor is oversampled to achieve 12 bit resolution.
**Altitude** = \(-0.246270 \times \text{Sample}_{11\text{bit}} + 455.9248\) \hspace{1cm} (4.5)

**Altitude** = \(-0.123105 \times \text{Sample}_{12\text{bit}} + 455.9248\) \hspace{1cm} (4.6)

<table>
<thead>
<tr>
<th>Altitude (m)</th>
<th>Pressure (Pa)</th>
<th>Vout (V)</th>
<th>Vcond. (V)</th>
<th>10-Bit Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>-47</td>
<td>101890.9</td>
<td>2.712660</td>
<td>3.301455</td>
<td>1023</td>
</tr>
<tr>
<td>-25</td>
<td>101625.7</td>
<td>2.704783</td>
<td>3.154164</td>
<td>978</td>
</tr>
<tr>
<td>0</td>
<td>101325.0</td>
<td>2.695853</td>
<td>2.987164</td>
<td>926</td>
</tr>
<tr>
<td>50</td>
<td>100725.8</td>
<td>2.678056</td>
<td>2.654365</td>
<td>823</td>
</tr>
<tr>
<td>75</td>
<td>100427.3</td>
<td>2.669189</td>
<td>2.488563</td>
<td>771</td>
</tr>
<tr>
<td>100</td>
<td>100129.4</td>
<td>2.660344</td>
<td>2.323160</td>
<td>720</td>
</tr>
<tr>
<td>200</td>
<td>98945.3</td>
<td>2.625176</td>
<td>1.665515</td>
<td>516</td>
</tr>
<tr>
<td>300</td>
<td>97772.6</td>
<td>2.590345</td>
<td>1.014182</td>
<td>314</td>
</tr>
<tr>
<td>400</td>
<td>96611.1</td>
<td>2.555850</td>
<td>0.369114</td>
<td>114</td>
</tr>
<tr>
<td>450</td>
<td>96034.6</td>
<td>2.538727</td>
<td>0.048915</td>
<td>15</td>
</tr>
</tbody>
</table>

| Table 4.3: Altitude to 10-Bit code. |

### 4.5.2.2 Airspeed Calculation

Calibrated airspeed is used to approximate equivalent airspeed at the altitudes, speeds, and temperatures that the miniFCS operates within. Equation 4.7 is based on the Saint-Venant formula for subsonic airspeeds [49]. In the formula below, \(q_c\) is the measured impact pressure, \(P_{sl}\) is the standard pressure at sea level and \(a_{sl}\) is the standard speed of sound at 15 °C. In the actual implementation \(P_{sl}\) is transformed by the transfer function of the MP3V5004G differential pressure sensor [34] and then scaled by 1/(volts per LSB). This allows to plug in the raw ADC sample values into \(q_c\) of the CAS formula to save CPU cycles.

\[
CAS = a_{sl} \sqrt{\frac{5}{p_{sl}} \left( \frac{q_c}{p_{sl}} + 1 \right)}^{\frac{\gamma}{\gamma - 1}} - 1 \hspace{1cm} (4.7)
\]
4.5.3 Attitude Estimation Using IR Sensors

Earth emits black body radiation in the infrared region. Because of this, it is possible to measure a temperature difference between the earth and sky using simple thermopile sensors. To measure the pitch of an aircraft two thermopiles are used. One is pointed towards the positive x-axis and the other to the negative. The difference of the two temperature readings can be used to determine a pitch angle. Similarly, a second set of thermopiles can be placed on the y-axis of the aircraft to measure the roll. A rotation calculation needs to be applied when the pitch angle is non-zero because the x-axis is no longer aligned with the earth reference x-axis. The resultant pitch and roll estimations are given on equations 4.8 and 4.9. In these equations, \( r_x \) and \( r_y \) are the readings from the IR sensors. Because temperature measurements can vary significantly over time, a calibration constant, \( k_c \), is used. This constant is the maximum reading of temperature difference between the earth and sky. It is determined by a pair of thermopiles pointed on opposite directions along the z-axis [50].

\[
\theta_p = \arcsin \left( \frac{r_x}{k_c} \right) \quad (4.8)
\]

\[
\theta_r = -\arcsin \left( \frac{r_y}{k_c \cos (\theta_p)} \right) \quad (4.9)
\]

The sensors on the aircraft are rotated by 45° about the sensor’s z-axis. Using this configuration avoids temperature measurement problems caused by engine heat and wing obstruction. By applying rotation matrices \( R_x \) and \( R_y \), roll and pitch estimation equations for the off-axis configuration can be determined. Derivation for equations 4.10 and 4.11 can be found in the Self-Calibrating CoPilot Based Attitude Estimation System white paper [50].
\[
\theta_p = \arcsin \left( \frac{r_u + r_v}{\sqrt{2}k_c} \right) 
\]
\[
\theta_r = \arcsin \left( \frac{r_u - r_v}{\sqrt{2}k_c \cos(\theta_p)} \right) 
\]

Figure 4.2: Z and XY IR sensors from Range Video.

4.5.4 Battery Monitor

Monitoring the battery voltage is extremely simple. To obtain the battery voltage, the ADC sample code is multiplied by the volts per LSB and the voltage divider. For the on-chip ADC the volts per LSB is \( V_{\text{ref}}/(2^{10} - 1) \) and the voltage divider is 4.32.

4.6 Navigation

The miniFCS supports two basic navigation methods, loitering and waypoint path following. Both of these modes are parametrized to behave differently depending on the vehicle state and operation mode. Loitering can be used to place the vehicle in a stable orbit around a waypoint. The loitering navigation mode is also used in the event that communications links to the vehicle have been lost. In this event,
the vehicle will begin to orbit about a home waypoint defined by the ground control operator. Timed loiter is another derivative mode that simply orbits a particular waypoint for a specified time. When the loiter time has elapsed, the vehicle will begin to loiter about the next waypoint.

Two methods of waypoint navigation are implemented in the miniFCS. The first uses a simple switched direction field as described in section 2.6.1. The vehicle simply heads directly to the destination waypoint. The second method follows a direction field that keeps the vehicle close to the line formed by the originating and destination waypoints. Both waypoint path following methods switch direction fields when the vehicle is within a set arrival range of the destination waypoint. The destination waypoint will be incremented to the next waypoint in the flight path and a new direction field is generated.

4.6.1 Loitering

The direction field for orbital trajectories is generated using a stable limit cycle with a final state circular periodic solution as described in system of eqs. 4.12. In polar coordinates, this system has a periodic solution at \( r=1 \). Neighboring solutions with non-closed trajectories will tend towards the periodic solution, that is trajectories inside the unit circle will spiral out and trajectories outside the unit circle will spiral into the periodic solution \([51]\). Figure 4.3 shows solutions outside of the periodic solution.

\[
\begin{pmatrix}
  x \\
  y
\end{pmatrix}' = \begin{pmatrix}
  y + x - x (x^2 - y^2) \\
  -x + y - y (x^2 - y^2)
\end{pmatrix} \tag{4.12}
\]
Figure 4.3: Orbital loiter trajectories generated by a stable limit cycle. The orbital center is at (0,0) and the orbit radius is one. The integrals curves represent ideal vehicle paths dependent on the initial vehicle position.

In order to use for this direction field for generating useful for loiter trajectories, it has been parametrized to the form in system of eqs. 4.13. Parameters have been added to the system of 4.12 in order to control the target radius, spiral deviation, and the direction of circulation. The parameters are \( r_t, k, \) and \( c \) respectively. Small values for \( k \) will generate trajectories that converge to the periodic solution more quickly. Parameter \( c \) can be either 1 for clockwise circulation or -1 for a counter-clockwise circulation

\[
\begin{pmatrix}
x \\
y
\end{pmatrix}' = \begin{pmatrix}
cy + \frac{x}{kr^2_t} (r^2_t - x^2 - y^2) \\
-cx + \frac{y}{kr^2_t} (r^2_t - x^2 - y^2)
\end{pmatrix} \tag{4.13}
\]

To understand how the system of eqs. 4.13 achieves a periodic solution with a constant angular rate and a stable limit at the target radius, it is more convenient to
use polar coordinates. Conversion from Cartesian to polar coordinates uses:

\[ x = r \cos(\theta), \quad y = r \sin(\theta), \quad r^2 = x^2 + y^2. \]  

(4.14)

First multiply the first equation from system of eqs. 4.13 by \( x \) and the second by \( y \) to obtain

\[ x \frac{dx}{dt} = xy + \frac{x^2}{kr_t^2} (r_t^2 - x^2 + y^2), \]  

(4.15)

\[ y \frac{dy}{dt} = -xy + \frac{y^2}{kr_t^2} (r_t^2 - x^2 + y^2). \]  

(4.16)

Adding equations 4.15 and 4.16 yields

\[ x \frac{dx}{dt} + y \frac{dy}{dt} = -\frac{1}{kr_t^2} (r_t^2 - x^2 + y^2) (x^2 + y^2). \]  

(4.17)

Since \( r^2 = x^2 + y^2 \) and \( r \frac{dr}{dt} = x \frac{dx}{dt} + y \frac{dy}{dt} \), it follows from equation 4.17 that

\[ \frac{dr}{dt} = \frac{r}{kr_t^2} (r_t^2 - r^2). \]  

(4.18)

From equation 4.18 it can be seen that critical points exist when \( r = 0 \) and when \( r = r_t \). It follows that \( \frac{dr}{dt} > 0 \) when \( r < r_t \) and \( \frac{dr}{dt} < 0 \) when \( r > r_t \). This means that if the vehicle is inside the loiter radius, the trajectory will be directed outward. While outside of the loiter radius, the trajectory will be directed inward.

A similar approach can be used to determine the angular rate behavior. First multiply the first of equation from system of eqs. 4.13 by \( y \) and the second by \( x \) to and subtract to obtain
\[
\frac{dx}{dt} y - x \frac{dy}{dt} = y^2 + x^2. \tag{4.19}
\]

From equations 4.14, \( \frac{dx}{dt} = -r \sin \theta \frac{d\theta}{dt} \) and \( \frac{dx}{dt} = r \cos \theta \frac{d\theta}{dt} \) can be calculated. Substituting these into equation 4.19 and reducing yields

\[
\frac{d\theta}{dt} = -1. \tag{4.20}
\]

Equations 4.18 and 4.20 indicate that the system of eqs. 4.13 has constant clockwise angular rate and has a periodic solution when \( r = r_t \).

Because the FCS maintains a constant airspeed, only the desired heading is calculated; the magnitude of the vector can be ignored. This direction field can be efficiently implemented using only 7 multiplication/division and 4 addition/subtraction operations. This method is advantageous because it removes all but one trigonometric operation. Trigonometric operations are typically 20 to 60 times slower than floating point multiplication on the UC3 microcontroller. This makes it suitable for execution at every control loop cycle without using excessive CPU time. The implemented algorithm is presented below.
Algorithm 4.1 Pseudocode for generating a target heading using the direction field defined in equation 4.13.

INPUT $r_t^2, x, y, k, c$

\[ h = r_t^2 - x \cdot x - y \cdot y \]
\[ dx = x/(k \cdot r_t^2) \cdot h \]
\[ dy = y/(k \cdot r_t^2) \cdot h \]

IF $c \geq 1$ THEN
\[ dx = -y + dx \]
\[ dy = x + dy \]
ELSE
\[ dx = y + dx \]
\[ dy = -x + dy \]
RETURN $\text{atan2}(dx, dy)$

4.6.2 Waypoint Navigation

Waypoint navigation can be performed using two methods. The first is a switched direction field generated from a simple attractor as described in section 2.6. The target heading simply points directly to the destination waypoint. When the aircraft is within a set arrival range of the waypoint, the aircraft will head to the next waypoint. The drawback to using this method is that if there is a significant wind, the aircraft will travel in an arc from the originating waypoint to the destination waypoint. A severe example of this cross-track error can be seen on figure 6.4. The second method reduces cross-track error by using a direction field that is dependent on the aircraft’s perpendicular distance from the line formed by the originating and destination waypoints.
**Figure 4.4:** Cross-track trajectories following a flight path. Originating waypoint is at (0,0) and the destination waypoint is at (0,1). The integral curves are the vehicle's flight path under ideal conditions and depend on the initial vehicle position.

Figure 4.4 shows an example direction field for the desired heading generated by the vector equation 4.21 which is derived from the geometry in figure 4.5. In this system, the origin is placed at the originating waypoint. \( p \) is the vector to the position of the aircraft and \( w \) is the vector to the destination waypoint. \( h \) is the desired heading vector and is defined as \( h = t - p \).

**Figure 4.5:** Cross-track heading vector field geometry.
Before \( h \) can be computed, vector \( t \) must be defined. The magnitude of \( t \), equation 4.22, is the scalar projection of \( p \) onto \( w \) plus the control parameter \( l \). The control parameter \( l \) simply controls the rate of transition from flying towards to flying tangent to the rhumb line. Because \( t \) and \( w \) are coincident, they have the same unit vector, thus \( t = |t| \hat{w} \). The coordinates for \( h \) can then be defined as

\[
 h = \left( \frac{|t|}{|w|} w_x - p_x, \frac{|t|}{|w|} w_y - p_y \right).
\]  

(4.21)

where:

\[
|t| = \frac{p \cdot w}{|w|} + l
\]  

(4.22)

Because the FCS maintains a constant airspeed, only the desired heading angle is calculated; the magnitude of the vector can be ignored. One important note is that a method to check if the vehicle has passed the destination waypoint without entering the arrival range must be introduced. This is required because the direction field extends infinitely along the rhumb line and the vehicle would continue on the line past the waypoint. A solution is to simply switch the direction field to the simple attractor when the scalar projection of \( p \) onto \( w \) is greater than \(|w|\), alternatively when \(|p| \) is greater than \(|w|\). The implemented algorithm is presented below.
Algorithm 4.2 Pseudocode for generating a target heading using a direction field defined by equation 4.21.

INPUT $w$, $p$, $l$
COMPUTE $|w|$ and $|p|$

IF $|p| > |w|$ THEN
  $t_x = w_x$
  $t_y = w_y$
ELSE
  $|t| = (p_x * w_x + p_y * w_y) / |w| + l$
  $scaleT = |t| / |w|$
  $t_x = w_x * scaleT$
  $t_y = w_y * scaleT$

$x = t_x - p_x$
$y = t_y - p_y$

RETURN $\text{atan2}(x,y)$

4.7 Attitude Control

This section describes how the miniFCS controls the direction of flight using all available sensor readings and navigation information. The miniFCS implements its control system using PID loops. To improve control performance, the control structures are arranged as cascaded PID controllers.

4.7.1 PID Control

The PID controller calculates an error as the difference between a measured process variable and a desired set-point. The control output is a weighted sum of a proportional term, an integral term and a derivative term. The proportional term is a response to the current error. The proportional response can adjusted by multiplying the current error by the proportional gain $K_p$. A proportional controller with a gain
that is too large will be unstable and cause oscillations. A gain that is too small will result in a slow response. Another problem with a purely proportional controller is droop. In the absence of disturbances, the controller will never reach the desired set point. This occurs when the measured process variable drifts over time and the proportional term that is moving the output towards the desired set-point is exactly offset by the process drift. The integral term is added to overcome this limitation of a purely proportional controller. The integral term is proportional to the error and the duration of the error and is controlled by gain $K_i$. It accelerates the process output towards the desired set-point and eliminates the steady state error. Because the integral term responds to accumulated errors, it can cause the process output to overshoot the desired set-point. To overcome this, the derivative term is added. The derivative term compensates for integral overshoot by responding to the rate of change in the error. The derivative response is controlled by the derivative gain $K_d$.

In a discrete time system such as the miniFCS, the integral and derivative must be discretized. The integral and derivative terms become $\sum_{i=1}^{k} e(t_i)\Delta t$ and $\frac{e(t_k) - e(t_{k-1})}{\Delta t}$.

![Figure 4.6: PID controller.](image)

Cascaded PID controllers can improve system performance when disturbances affect a secondary process or the secondary process output directly effects the primary process output that is controlled. For a cascaded controller to function properly, the dynamics of the inner-loop should be faster than the outer loop [52]. See the
CHAPTER 4. FLIGHT CONTROL SOFTWARE

lateral controller implementation to see how a cascaded PID controller can improve performance.

![Cascaded PID Controller](image)

**Figure 4.7:** Cascaded PID Controller

### 4.7.2 Lateral Control

A cascaded PID controller, as shown in figure 4.7, is used to control vehicle roll. Input into the outer loop PID1 controller is a heading error. This is simply the target heading from the navigation system minus the heading reading from the GPS module. Output from the PID1 controller is a target roll. When the heading error is large, the vehicle will bank more aggressively. A maximum roll rate is imposed to limit sudden roll changes. This is done by only adding a small roll quantum at each iteration of the loop until the target roll angle is reached. The roll quantum is determined by an operator provided slew rate parameter. The target roll minimum and maximum angles are also limited by user controllable parameters.

The rate and absolute limited target roll angle is subtracted from the measured roll angle of the vehicle. This roll error is the input to the inner loop PID2 controller. The PID2 controller is tasked to maintain the target roll. The output of this PID controller is an aileron control surface position that will roll the vehicle into the target heading. Clearly, the output of the inner loop controller directly effects the output of the outer-loop controller. Roll dynamics of the aircraft are significantly faster than the aircraft heading changes thus the control system benefits from the cascaded PID
controller. Another benefit of the cascaded controller is that disturbances, such as turbulence, that effect the roll of the aircraft can be quickly compensated within the inner loop. This will result in a more controlled turn where the heading rate of change is controlled by the outer loop.

The lateral controller attempts to reduce the amount of side slip of the aircraft by performing Aileron-rudder coordinated turns. Because of sensor limitations, the severity of side slip of the vehicle cannot be determined. For coordinated turns, a simple yaw controller is implemented as a single PID controller where the integral and derivative terms are disabled. Input to this controller is the target roll. The output is proportional to the input and controls the rudder of the vehicle. The proportional gain for this controller is typically set to a small value to keep limit rudder action.

4.7.3 Longitudinal Control

The longitudinal controller is tasked with maintaining the commanded altitude and airspeed. Like the roll controller, the altitude controller is implemented as a cascaded PID controller. The input to the outer loop PID1 controller is the altitude error; this is the commanded altitude subtracted by the measured altitude. The output of the the PID1 controller is a pitch set-point. The pitch set-point is rate limited by adding a small pitch quantum at each iteration of the loop until the target pitch is reached. This is controlled by an operator provided pitch slew rate parameter. The target pitch minimum and maximum angles are also limited by user controllable parameters. Under constant airspeeds, this type of controller effectively controls the rate of climb.

Output from the PID1 controller is fed to the PID2 controller as a pitch error. This error is the measured vehicle pitch subtracted by absolute and rate limited pitch. The
PID2 controller is tasked to maintain the target pitch. The output of this controller is an elevator position. Pitch dynamics of the vehicle are much faster than climb rate changes. Also other disturbances that may alter pitch can be quickly compensated within the inner loop of the cascaded pitch controller.

The airspeed controller is implemented as a single PID controller where the derivative term is disabled; effectively it is a PI controller. Input to the airspeed controller is the airspeed error. The error is calculated as the target airspeed subtracted by the measured airspeed. The output of the controller is a target throttle position. The throttle position is rate limited by adding a small throttle quantum until the target throttle position is reached. This is controlled by an operator provided throttle slew rate parameter.
Chapter 5

Hardware In the Loop Simulator

A HILS is a system that is capable of ‘fooling’ an embedded system into believing that it is operating with real world inputs and outputs. The system operates in real-time on the actual hardware and software that will eventually be used in its intended application [53]. In this case, the VCU UAS system is being fooled into flying a fixed wing aircraft within a simulated environment.

The VCU UAS is a complex and expensive system that must be thoroughly tested before a flight can take place [54]. Testing is critical to ensure the safety of the system. Avoiding the loss of an aircraft due to software or hardware malfunction is also important because the cost of a typical aircraft can be over $10,000. Another benefit of using a HILS is that it allows for faster development and testing of new flight control algorithms. Because there are no off-the-shelf HILS that matches the requirements of the VCU UAS, a custom solution was designed.

The initial prototype HILS system is capable of generating simulated inputs and outputs for fixed wing aircraft. The simulated sensors that are currently implemented include the MIDG II inertial navigation unit, two different absolute pressure sensors,
two differential pressure sensors, xy and z thermopile sensors, and GPS modules using the NMEA standard. This complement of sensors was selected because the represent the majority of sensors used with VCU's UAVs. In INS guided UAVs, the INS is responsible for providing position, orientation and velocity information using GPS, accelerometers, rate gyro and magnetometers. Airspeed and altitude information is gathered using the pressure sensors. UAVs that use the miniFCS autopilot use the xy and z thermopiles, absolute and differential pressure sensors, and a NMEA standard GPS system. Additional simulated sensors may be added to support vehicles using different sensor configurations. Under most sensor configurations, outputs can be generated at rates of up-to 500 Hz but they are practically limited by serial data rates and the flight simulator simulation rate.

Figure 5.1 depicts a VCU UAS connected to the HILS. Command, control and telemetry occur to the ground station clients occurs via the aircraft’s radio modem through the ground station server as it would in an actual flight. The HILS board is a complex interface between the aircraft hardware and a COTS flight simulator. The HILS board receives a flight dynamics model state from the flight simulator. This FDM state is used generate all sensor inputs to the aircraft’s flight control system. The aircraft responds to the sensor data and generates new control surface positions. These are decoded by the HILS board and sent to the flight simulator. The flight simulator will generate a new flight dynamics model state and send it back to the HILS board. The HILS board will use the new state information to generate updated sensor inputs to the flight control system. Using this technique the aircraft is 'fooled' into flying within a simulated environment.
5.1 Hardware Implementation

The hardware in the loop simulator hardware consists of a base board containing digital to analog converters, serial ports, GPIO, PWM input buffers, and power regulators. A Suzaku SZ130 FPGA daughter board with a soft-core CPU and enough memory for an embedded operating system is used as the core of the HILS. The FPGA is also used for synthesized PWM decoders, UARTS, a serial DAC interface, and other interface hardware required by the system.

5.1.1 Base Board

The Hardware In the Loop base board provides the hardware interfaces for the HILS system. These include buffered PWM inputs, buffered DAC outputs, RS232 serial ports, synchronous and asynchronous serial ports, and general purpose IO pins. The base board was designed using Mentor Graphics PADS logic. PADS Layout and Router were used to layout and route the printed circuit board design. To keep costs
Figure 5.2: HILS with all components assembled.

low, the PCB was designed with two layers and a size of approximately 4x3 inches. It was fabricated by Advanced Circuits using the $33 per board student special. Figures 5.3 and 5.4 show the PCB layout (both top and bottom layers) and resulting fabricated PCB.

The base board consists of:

- sockets for the Suzaku SZ130
- 16 buffered PWM inputs
- 16 GPIO outputs
- 4 RS232 level shifters
- 8 Digital to analog converters
- 3.3V and 5V power supplies
5.1.1.1 Digital to Analog Converter

The DAC that is used on the HILS board is Analog Device’s AD5678-2. This is an octal DAC with four 12 bit and four 16 bit DACs in a single package. What makes this particular DAC desirable is that it contains an on-chip voltage reference and all
outputs are buffered. It is also capable of operating with supply voltages from 2.7V to 5.5V [55].

![Figure 5.5: AD5678 DAC block diagram.](image)

The AD5678-2 features an on-chip 2.5V reference with an internal gain of 2. This gives a full scale output of 5V. The internal reference is off at start-up and must be turned on by a software write. This allows for the use of an external reference voltage, if desired.

The outputs of all DACs can be updated individually or simultaneously. This can be controlled using commands sent serially to the DAC. The AD5678-2 uses a 3-wire serial interface that can operate with a clock speed of up to 50 MHz [55]. The write sequence begins at the falling edge of the /SYNC line. Data from the DIN line is then clocked into the 32 bit shift register using a custom serial writer IP core. DIN is sampled at the falling edge of SCLK. On the 32nd falling edge the complete function is programmed and is executed. The /SYNC line must be high for at least 15ns before a new write sequence can begin. The /SYNC line can idle high but less current is drawn when idled low.
CHAPTER 5. HARDWARE IN THE LOOP SIMULATOR

Figure 5.6: AD5678 timing diagram.

Commands Bits 24 to 27 of the input shift register correspond to the command to be executed. Bits 20 to 23 are the address bits for a given DAC channel. Bits 4 to 19 are the 16-bit data word for 16-bit channels. Bits 8 to 19 are the 12-bit data word for 12-bit channels. When sending the Set-up Internal Reference command, DB0 is used to turn on/off the internal reference (1-on, 0-off). Tables 5.1 and 5.2 give the possible commands and DAC addresses.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Write to Input Register n</td>
</tr>
<tr>
<td>0001</td>
<td>Update DAC Register n</td>
</tr>
<tr>
<td>0010</td>
<td>Write to Input Register n, update all</td>
</tr>
<tr>
<td>0011</td>
<td>Write to and update DAC Channel n</td>
</tr>
<tr>
<td>0100</td>
<td>Power down/power up DAC</td>
</tr>
<tr>
<td>0101</td>
<td>Load clear code register</td>
</tr>
<tr>
<td>0110</td>
<td>Load LDAC register</td>
</tr>
<tr>
<td>0111</td>
<td>Reset (power-on reset)</td>
</tr>
<tr>
<td>1000</td>
<td>Set up internal REF register</td>
</tr>
</tbody>
</table>

Table 5.1: AD5678 commands and addresses.
### Table 5.2: AD5678 DAC addresses.

<table>
<thead>
<tr>
<th>Address</th>
<th>Selected DAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>DAC A (16 bits)</td>
</tr>
<tr>
<td>0001</td>
<td>DAC B (16 bits)</td>
</tr>
<tr>
<td>0010</td>
<td>DAC C (12 bits)</td>
</tr>
<tr>
<td>0011</td>
<td>DAC D (12 bits)</td>
</tr>
<tr>
<td>0100</td>
<td>DAC E (12 bits)</td>
</tr>
<tr>
<td>0101</td>
<td>DAC F (12 bits)</td>
</tr>
<tr>
<td>0110</td>
<td>DAC G (16 bits)</td>
</tr>
<tr>
<td>0111</td>
<td>DAC H (16 bits)</td>
</tr>
<tr>
<td>1111</td>
<td>All DACs</td>
</tr>
</tbody>
</table>

#### 5.1.1.2 PWM Input Buffers

The HILS can be populated with either the 74LCX541 or the 74HC7541 to buffer all PWM inputs. The LCX541 operates at 3.3V but is 5V tolerant. This part is used when the high level PWM output from the FCS is 5V. Alternatively, the 74HC7541 can be used when the high level PWM output is not higher than 3.3V, as are the outputs of the miniFCS.

The 74HC7541 is an octal Schmitt trigger non-inverting buffer/line driver with 3 state outputs. At room temperature and Vcc of 4.5V, this buffer has a propagation delay of 14ns for both high to low and low to high transitions. The maximum positive going threshold is 3.15V and the minimum negative-going threshold is 1.35V. The typical hysteresis with these conditions is 0.40V. The Schmitt trigger action transforms slow changing input signals into sharply defined outputs. The Schmitt Trigger also prevents against rapid triggering by noise that may be present on the PWM signals.

The Schmitt trigger is a comparator application which switches the output negative when the input passes upward through a positive reference voltage. It then uses negative feedback to prevent switching back to the other state until the input passes through a lower threshold voltage, thus stabilizing the switching against rapid
CHAPTER 5. HARDWARE IN THE LOOP SIMULATOR

triggering by noise as it passes the trigger point [56].

5.1.1.3 Other Components

The MAX3232 from Texas Instruments consists of two line drivers and two line receivers. It provides an electrical interface between logic level communications controller and a RS232 serial port. It can operate at data signaling rates of up to 250kbits/s [57]. Two of these ICs are used to provide level shifting for four serial ports in the HILS base board.

The PTH08080W is an integrated switching regulator module that can provide up to 2.25 A of current. The output voltage can be set by choosing a single resistor [58]. This reduces the number of components needed on the HILS base board. For this application the output voltage has been set to 3.3V and is used to power the Suzaku FPGA board, level shifters and PWM input buffers. A UA78M05 low noise, fixed voltage (5V) regulator with up-to 500 mA output is used as a supply to the DAC.

5.1.2 Suzaku SZ130

The Suzaku SZ130 is an FPGA based computer board. It uses the Xilinx Spartan-3E with a 1.2 million equivalent gate count. The base configuration consists of a MicroBlaze soft CPU, SDRAM, SPI FLASH RAM and an Ethernet controller. This board is used because it is very similar to the on-board computer used in VCU's FPGA based FCSs. Because of this, many soft-hardware components can be reused with little or no modification.

The Suzaku computer board uses the µClinux operating system. This operating system eliminates the dependency on a MMU thereby allowing Linux to run on systems based on processors such as the Motorola 68000 and Xilinx MicroBlaze. Because
there is no memory management unit, there is no memory protection or virtual memory.

The Suzaku SZ130 features:

- FPGA Xilinx Spartan-3E XC3S1200 FG320
- MicroBlaze Soft Processor
- Crystal Oscillator 3.6864MHz (frequency multiplied by FPGA’s internal DCM)
- BRAM 504Kbits
- SDRAM 16Mbyte \times 2
- SPI Flash 8Mbyte
- Configuration Stored in SPI Flash
- JTAG One port (FPGA)
- SPI Flash Write Dedicated pin
- Ethernet 10Base-T/100Base-Tx
- Serial UART 115.2kbps
- Timer 2ch (1ch is used by OS)
- Free I/O Pin 86 pin
- Power Supply Power supply: 3.3V\pm3\%
- Consumption current: 350mA typical

The FPGA contains most of the hardware components of the HILS board. It contains the MicroBlaze CPU with an FPU [59]. It is connected to the FPGA’s BRAM using a local memory bus. This is used to implement the data and instruction caches. Other components are connected using the On-Chip Peripherals Bus (OPB) [60]. Figure 5.8 shows the various IP cores and their connections.
5.1.3 IP Cores

Even though the Suzaku SZ130 has a relatively complete set of peripherals, there are some components that need to be added. A pulse width modulation decoder IP core, developed at VCU, was added to the project. The PWM decoder core is configured to read 8 PWM inputs. Reading the PWM value is simply done by reading a memory location. This can be done because the PWM reader is attached to the OPB bus as a memory mapped device. Typical servo pulses range from 1000 to 2000 microseconds, this value is returned as an integer when reading a particular channel. The PWM reader IP core used is the same core that is used within the VCU FPGA based flight control systems.

A UART Lite is an IP core from Xilinx for the On-chip Peripherals Bus. The UART Lite is a full duplex UART with 16 byte transmit and 16 byte receive FIFOs. The baud rate and parity are configurable but cannot be changed after synthesis.
This allows for creating UART cores that use less FPGA resources [61]. Six additional UART Lites were added to support simulated sensors that use RS232 serial communications.

To communicate with the DAC a custom Serial Writer IP core was designed. This is a simple core that interfaces with using the OPB. It is used to set the voltage output on a DAC by clocking out 32 bit commands to the DAC. Once the command it is written to the Serial Writer’s memory mapped register, a synchronization pulse is generated and the 32-bit command sequence is clocked out on the next 32 clock cycles. If the memory mapped register is read, the serial writer will return a non-zero value if it is currently clocking out a command. The Serial Writer is able to operate using the system clock or the clock rate can be divided to accommodate slower serial devices.

5.2 Software Implementation

The HILS application runs on top of the µClinux operating system with BusyBox user-land utilities. The hardware in the loop simulator software is a relatively simple single threaded application. It is written in C and linked with uClibc. Simulated sensors are implemented as modules. This allows for different sensor complements to be easily configured. Currently, eight sensors have been implemented. These include the MIDG II INS, two absolute pressure sensors, two differential pressure sensors, xy and z horizon sensors, and NMEA standard GPS modules. With these sensors, the HILS is able to support most of VCU’s UAV sensor complements. Other sensors such as binary protocol GPS or compass modules can be easily added to support different UAV configurations. Specifics on how these sensors are simulated are given in the following subsections.
The HILS application communicates with the FlightGear flight simulator using UDP sockets. A socket is used to receive the Flight Dynamics Model (FDM) from the flight simulator. Another socket is used to return control surface positions. These include aileron, elevator, throttle and rudder positions. The flight simulator uses this information to update its FDM which is then fed back into the HILS application.

The basic flow of the program is as follows. Communications sockets and ports are initialized. Then the DAC is initialized with the internal reference activated. Configurations for the sensors and servo limits are then read. A free running simulation loop is then entered. If there is a FlightGear FDM state UDP packet ready, it is read and copied into a local FDM state structure. Each simulated sensor has its own timer to control its update rate. If the sensor's timer has expired, the simulated sensor values are calculated using the current FDM state and output using the appropriate IO method. Finally, the control positions of the UAV are read, at a configured rate, using the PWM decoder and fed back into FlightGear using the controls UDP socket. This repeats until the the application is terminated or restarted.

A simple GUI interface can be used to configure various parameters. Network settings such as the FlightGear address and port number can be configured. Each sensor can be individually turned on or off, configured to update at rates of up to 500 Hz, and an appropriate IO port can be selected. The rate at which FDM control packets are sent to the simulator can also be configured. The achieved sensor and control rates are displayed in real time.

5.2.1 FlightGear

FlightGear is a free open source flight simulator developed through the contribution of source code and time by many talented people from around the world. Flight-
Gear was designed to be open and extensible with the idea that people involved in research would use it with their own projects [62]. This is typically not possible with commercial off-the-shelf flight simulators.

FlightGear features several flight dynamics models. For the HILS, the JSBSim FDM is used. JSBSim is a generic dynamics model for simulating the motion of flight vehicles. It features 6 degrees of freedom and uses look-up tables where mass properties, aerodynamics and flight control properties are all defined. The tables are used to calculate the sum of forces and moments acting on the vehicle [63].

An important feature of FlightGear is that its internal properties are exposed. This gives users access to a large number of internal state variables. The FDM state can be sent using several IO interface options. These include serial, TCP, UDP, files, and named pipes. The hardware in the loop simulator uses UDP sockets to receive FDM state updates and to send control surface position updates [62].

### 5.2.2 Simulated Sensors

Most of the VCU’s UAVs use one of two sensor configurations. One configuration includes MIDG II INS, and the MPX5010DP and MPX5100AP Pressure sensors. The other configuration includes xy and z IR sensors, a NMEA standard GPS module, and the MP3V5004DP and MP3H6115A pressure sensors. Since these are the most common sensor configuration, they were the first to be implemented. Other sensors can be easily added to support UAVs with different sensor complements. In the following subsections, the implemented sensors and how they are simulated is described with the exception of NMEA standard GPS modules. Details on supported NMEA sentences can be found in section 4.4.
5.2.2.1 MIDG II INS

The MIDG II is a small Inertial Navigation System (INS) that incorporates Global Positioning System (GPS). Because of its small size, it is well suited for unmanned vehicle applications. The VCU FCS uses the MIDG II as its primary instrumentation system, therefore its simulated implementation was completed first. It features three-axis rate gyros, accelerometers, and magnetometers. It can provide position, velocity and attitude solutions at a rate up-to 50 Hz [7].

The MIDG II INS provides a standard binary protocol that frames sensor messages. This protocol and sensor messages are simulated by the HILS. Each sensor message is framed using a standardized packet format. The packet frame begins with two sync bytes followed by one byte for a message ID, then another byte for the number of bytes in the message. The message payload follows. The packet frame is terminated with a two byte Fletcher checksum. The Fletcher checksum is computed, as defined in Internet RFC 1145, over the ID byte, Count byte, and payload bytes.

| SYNC 0 | SYNC 1 | ID | COUNT | PAYLOAD 1 | ... | PAYLOAD N | CKSUM 0 | CKSUM 1 |

**Figure 5.9:** MIDG II binary packet format.

Not all of the messages that can be provided by the MIDG II are used by the VCU FCSs. Typically, the FCS listens for the NAVSENSOR, NAV_PV, and GPS_PV messages. The NAV_SENSOR message provides navigation sensor information, the NAV_PV message provides navigation position and velocity solution, and the GPS_PV provides GPS position and velocity solution. Because the FCS only requires these three messages, these are the only messages that are simulated by the HILS. Refer to the MIDG II Message format data sheet [64] for a complete listing of messages and their contents.
Angular rates, accelerations, yaw, pitch, and roll can be extracted from the flight dynamics model data attained from FlightGear. These parameters are then scaled to the same units used in the NAV SENSOR message. The NAV SENSOR packet is then constructed and transmitted to the FCS using an RS232 serial port.

To simulate the NAV PV message, the three dimensional position and velocities are extracted from the FlightGear flight dynamics model. The position information is formatted as LLA with the same units used by the MIGI. Similarly, the velocity information is extracted from the flight dynamics model and formatted to cm/s using the ENU convention. The solution status bit field tells the receiver that the position fields are in the LLA format and that the velocity fields use the ENU convention. The NAV PV packet is then constructed and transmitted to the FCS using an RS232 serial port.

The last message required by the FCS is the GPS position and velocity solution. Again, the position and velocity information is extracted from the flight dynamics model and formatted to the proper units. Using the details bit field, a 3D fix is indicated with the position information in the LLA format and the velocities use the ENU convention. The GPS PV packet is then constructed and transmitted to the FCS using an RS232 serial port.

5.2.2.2 Differential Pressure Sensors

The MPX5010DP and MP3V5004G are piezoresistive transducer constructed using thin film micro-machining techniques. They can measure a maximum pressure differential of 10 and 4 kPa respectively. The required supply voltages are 5 and 3.3 volts respectively [65, 34]. These sensors are used to measure the airspeed of the aircraft.

To simulate these sensors, the calibrated airspeed is extracted from the current
FlightGear FDM state. The CAS can be used to determine the impact pressure because it is defined as a function of only the impact pressure when standardized values for the speed of sound and pressure at sea level are used [49]. The formula below defines the calibrated airspeed. We can solve this for the impact pressure as a function of the CAS. Unfortunately, the function is computationally expensive and is not well suited for the HILS hardware.

\[
CAS = a_{sl} \sqrt{5 \left[ \left( \frac{q_c}{P_{sl}} + 1 \right)^{\frac{3}{7}} - 1 \right]} \tag{5.1}
\]

Where:

- \(q_c\) = impact pressure
- \(P_{sl}\) = standard pressure at sea level
- \(a_{sl}\) = standard speed of sound at 15°C

Rather than applying this formula directly, a polynomial approximation can be used. Using formula 2 we can find a set of impact pressures as a function of the CAS in the range that VCU UAVs fly. This is typically 30 to 200 knots. A very good fit for the impact pressure as a function of CAS is then found:

\[
q_c = 0.1694(CAS)^2 - 0.8223(CAS) + 13.681 \tag{5.2}
\]

Finally, the transfer functions of the differential pressure transducers can be used to generate a voltage output. The first equation below corresponds to the MPX5010DP sensor and the second to the MP3V5004G sensor.

\[
V_{out} = V_s(0.000009q_c + 0.04) \tag{5.3}
\]
\[ V_{out} = V_s(0.0002q_c + 0.2) \] \hspace{1cm} (5.4)

Where \( V_s \) is the supply voltage. The output voltage is then output by the HILS using a 16-bit digital to analog converter on the base board.

### 5.2.2.3 Absolute Pressure Transducers

The MPX5100AP and MP3h6115A are piezoresistive transducers constructed using thin film micro-machining techniques. The main difference is in the required supply voltage, 5 and 3.3 volts respectively. They have a maximum absolute pressure of 115kPa\[66, 35\]. These sensors are used to measure the barometric altitude of the aircraft.

These sensors are simulated using a similar method as the impact pressure sensors. The altitude ASL is extracted from the current FlightGear FDM state. It can be used to determine the static pressure. The following formula defines the the air pressure at a given altitude ASL. Unfortunately, the function is computationally expensive and is not well suited for the HILS hardware.

\[ P_s = 101325(1 - 2.2557 \times 10^{-5} \text{ASL})^{5.25588} \] \hspace{1cm} (5.5)

Rather than applying this formula directly, a linear approximation is used. Using the formula for the static pressure, a set of static pressures as a function of the altitude ASL is calculated for the range that VCU UAVs fly. This is typically 0 to 2000 meters. A very good linear fit for the static pressure as a function of altitude ASL is found:

\[ P_s = -11.323(\text{ASL}) + 101325 \] \hspace{1cm} (5.6)
Finally, the transfer function for the absolute pressure transducers [66, 35] can be used to generate a voltage output.

\[ V_{out} = V_s(0.000009P_s - 0.095) \]  

(5.7)

where:

\[ V_s = \text{supply voltage} \]

The output voltage is then output by the HILS using a 16-bit digital to analog converter on the base board.

5.2.2.4 Infrared Sensors

The infrared sensors are simulated by generating voltages depending on the pitch and roll of the aircraft. These voltages are output using the HILS 12-bit DAC outputs.

\[ \theta_p = \arcsin \left( \frac{r_u + r_v}{\sqrt{2k_c}} \right) \]  

(5.8)

\[ \theta_r = \arcsin \left( \frac{r_u - r_v}{\sqrt{2k_c \cos (\theta_p)}} \right) \]  

(5.9)

The equations above for pitch and roll, developed in [50], can be solved simultaneously to obtain equations 5.10, 5.11, and 5.12 for the x, y, and z thermopile voltage outputs. The calibration constant, \( k_c \), is the maximum measurement of temperature difference between the earth and sky. This is typically set to a constant between 2.5 and 3.3V within the simulator. The \( V_{offset} \) term is required because the thermopile output amplifier is offset to half of the supply voltage. This is typically set to 1.65V.
\begin{align}
V_x &= V_{\text{offset}} + \frac{\sqrt{2}}{2} k_c (\cos \theta_p \sin \theta_r - \sin \theta_p) \quad (5.10) \\
V_y &= V_{\text{offset}} + \frac{\sqrt{2}}{2} k_c (\cos \theta_p \sin \theta_r + \sin \theta_p) \quad (5.11) \\
V_z &= V_{\text{offset}} + k_c \cos \theta_p \cos \theta_r \quad (5.12)
\end{align}
Chapter 6

Flight Tests and Evaluation

Testing of the miniFCS hardware and software was done using the hardware in the loop simulator and on an EasyGlider sailplane. Over 60 test flights have been performed on the HILS before the autopilot was installed into the sailplane. Various tests were completed to ensure proper operation of the complete flight control system. The data link, ground commands, telemetry, logging, navigation, attitude control, software and hardware stability, sensors, signal conditioning, and safety switch were all tested prior to attempting autonomous flights on the sailplane. At the time of this writing, 7 fully autonomous flights on the EasyGlider have been completed; each with durations ranging from 9 to 34 minutes. During these flights, various flight patterns such as bow-tie, triangles, rectangle, straight paths, and loitering were performed at various speeds and altitudes. These flights were used tune the PID gains and determine platform flight characteristics.
6.1 Test platform

The EasyGlider has a 72” wingspan, 54” length, and weighs 34 oz. It is constructed out of durable expanded polypropylene foam. Powering the sailplane is a Himax 3516-1130 350 Watt brushless motor and a 12x6 folding propeller. The motor is controlled using a Thunderbird-36 ESC capable of delivering 36 Amps of continuous current from a 11.1V 3.3Ah lithium polymer battery. The miniFCS is mounted within canopy. An air line connects the differential pressure sensor port to a pitot tube on the right main wing. The IR sensors are mounted on the outside of the fuselage, as is the data link antenna. The flight controller, RC receiver and servos are powered using a 5 V BEC connected to the motive power battery.

Figure 6.1: Electric UAV platform with miniFCS mounted in payload bay.

Figure 6.2: miniFCS (bottom) connected to the HILS (top).
6.2 Waypoint Path Tracking

Path tracking algorithms were tested using both the simulator and sailplane. The simulator was used to test the flight controller’s performance under ideal conditions such as no wind or a constant wind from a known direction. Tests performed on the sailplane occurred under variable winds and turbulence. Figure 6.3 shows the path of the sailplane using the simple switched attractor and cross-track direction field algorithms. Wind was from the south and varied from about 10 to 15 knots. The simple attractor deviates by nearly 100 meters from the northern leg of the pattern. The cross-track algorithm significantly improves tracking of this rectangular pattern. The simple switched attractor performs reasonably well when the flight patterns are small and when the tracking error is primarily due to the turning radius of the vehicle.

Figure 6.3: Waypoint navigation cross-track error.
The cross-track direction field algorithm is well suited when legs of a flight path are long. This is demonstrated in figure 6.4. A long leg of a flight path was simulated by placing two waypoints 1.35 km apart. The airspeed was set to 30 knots and a wind from the north of 15 knots was applied. The flight path generated by the simple attractor field deviated by about 350 meters from the rhumb line. The cross-track direction field algorithm only had a constant deviation of about 8 meters from the rhumb line.

**Figure 6.4:** Hardware in the loop simulation of cross-track error correction on a 1.3 km leg.
6.3 Orbit Tracking

![Loiter Transitions](image)

**Figure 6.5:** Hardware in the loop simulation of orbital flight trajectories at 200, 150, 100, and 50 meter radii.

Testing of orbital trajectories was done using the miniFCS hardware on the HILS and the sailplane platform. Figure 6.5 shows the simulated aircraft entering a clockwise loiter pattern from the south. As designed, the entry trajectory is smooth and tear drop shaped. The vehicle initially loiters with a radius of 200 meters. After one complete revolution, the radius was decreased by 50 meters. A smooth transition
between orbits and nearly perfect circulation is achieved.

Figure 6.6 shows sailplane orbital trajectories of 200 and 150 meters. During this flight, the target altitude was set to 200 meters and the airspeed to 35 knots. Winds were predominantly from the south with speeds from 10 to 15 knots. The miniFCS flight controller was commanded to enter a clock-wise loiter pattern while the vehicle was inside the orbital radius. The sailplane made a smooth U-turn into the desired orbit. Like the simulation, orbits exhibit very circular patterns. Orbit revolutions overlap nearly perfectly. The transition from the outer to inner orbit was not as smooth as those seen in the simulation. This is can be attributed to less than optimal tuning of the roll controller gains. The flight controller performed well under 'real-world' conditions.

![Figure 6.6: Actual sailplane orbit tracking with 15 knot wind from south. UAV flies 200 and 150 meter orbits.](image-url)
6.4 Altitude and Airspeed Hold

Longitudinal control was tested using the HILS by varying the target altitude and airspeed. As shown in figure 6.7, the altitude hold performs well, climb rates were smooth and constant. No overshoot in altitude occurred. Slight undershoot was observed when transitioning to 100 meters. Airspeed control was also tested on the HILS. The commanded airspeed was set to 25, 40, and 50 knots. Airspeed hold was nearly ideal. Overshoot of about 4 knots was observed. This can be reduced by better tuning of the airspeed controller PID gains.

![Altitude Hold](image)

**Figure 6.7:** Hardware in the loop simulation of altitude hold and transitions at 100, 200, and 300 meters.

![Airspeed Hold](image)

**Figure 6.8:** Hardware in the loop simulation of airspeed hold and transitions to 25, 40, and 50 knots.
On figure 6.9, the airspeed and altitude are graphed for part of the flight in figure 6.4. Altitude is held constant throughout the flight. Slight dips in altitude do occur when a banking turn is executed. A corresponding increase in airspeed occurs as altitude is lost. Both airspeed and altitude begin to recover before the turn is complete. This effect can be lessened by adding feed forward terms to the pitch and throttle PID controllers. The airspeed measurements exhibit rapid fluctuations. Initially, it was thought that these fluctuations were caused by noise or motor vibration coupling onto the MEMS pressure sensor. The fluctuations are actual changes in airspeed. This was verified by a strong correlation of the GPS ground speed to the measured airspeed. Airspeed measurements errors due to noise and vibrations were ruled out. These rapid speed fluctuations are caused by rapid throttle adjustments. Reduction of the derivative gain, lowering the slower throttle slew rate, and filtering the throttle controller output may lessen this effect.

![Figure 6.9: Actual sailplane airspeed and altitude hold.](image-url)
Chapter 7

Conclusion and Future Work

7.1 Conclusions

The miniFCS succeeded in meeting its design goals. The single unit price point of $250 was met while keeping most desirable features of previous hardware implementations. This was accomplished by using an inexpensive complement of analog and digital sensors. Hardware signal conditioning allows the use of the on-board microcontroller’s ADC, thereby eliminating a costly external ADC. The on-board processor is fast enough for all in flight calculations and has enough spare cycles for future additions. Size, weight, and power requirements have all been lowered. The miniFCS is only 8.5 cm x 5.5 cm, weighs 32 grams, and uses 0.15 amps @ 5V with the radio modem transmitting. This has permitted the use of the miniFCS autopilot in small aircraft with minimal payload capabilities.

The system software is capable of receiving command and control commands from a ground station server. It features a simple attractor, cross-track direction field, and direction field based loitering navigation modes. The cross-track algorithm imple-
mented significantly decreases rhumb line tracking error and uses fewer CPU cycles than the the solution implemented in earlier VCU flight control systems. The loitering algorithm is also computationally inexpensive. Both are efficient enough to execute at every control cycle (50 Hz). The loiter flight paths observed by the sailplane were nearly circular with little eccentricity even under non-ideal wind conditions. Longitudinal and lateral controllers are simple to tune and do not exhibit oscillatory behavior although they can be improved by adding feed forward terms where appropriate.

The development of a hardware in the loop simulator was invaluable to the success of the miniFCS. It allowed all aspects of the autopilot to be thoroughly tested prior to integration onto and flight testing of the sailplane platform. The HILS is capable of simulating the miniFCS’s full sensor complement and those used on other VCU UAV configurations. It can be easily extended to support new sensor types and can generate sensor outputs at 500 Hz. The simulator was built using an FPGA board and components used in various projects at the lab. This allowed for a very low cost solution and the reuse of hardware.

Overall results for the miniFCS, both in simulated and real flights, are good. The autopilot successfully guides the UAV in the desired path with good accuracy and repeatability. Endurance of 30 minutes have been achieved on the sailplane platform. The hardware and software have proved to be reliable.

7.2 Future Work

As successful as the miniFCS has been there are many areas that need attention. On the hardware side, a new PCB should be designed and fabricated. Minor improvements to the layout could significantly reduce the size of the board. This could allow the use of the FCS in even smaller aircraft. Hardware flow control lines should also
be added to all serial ports. The analog signal conditioning section could also be improved by altering the values of the Sallen-Key filter circuits. Increasing the input resistance and decreasing the capacitance will allow the filters to better attenuate in the stop band.

The miniFCS software should also be improved. PID gain scheduling should be added for better control over a larger airspeed envelope. Feed forward terms should be added to the pitch and throttle controllers dependent on the commanded roll. Cooperative algorithms could be directly implemented on the miniFCS’s processor. There are sufficient resources for implementing formation flying and other simple cooperative tasks. In order to support cooperative missions, the ground station server requires frame level support for the RF modems. Having frame level control of the modem will ensure reliable communication to multiple air nodes from a single ground node.
References


REFERENCES


