5-2010


Calvin Coopmans
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Electrical and Electronics Commons

Recommended Citation
https://digitalcommons.usu.edu/etd/692

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations by an authorized administrator of DigitalCommons@USU. For more information, please contact dylan.burns@usu.edu.
ARCHITECTURE, INERTIAL NAVIGATION, AND PAYLOAD DESIGNS FOR
LOW-COST UNMANNED AERIAL VEHICLE-BASED PERSONAL REMOTE
SENSING

by

Calvin Coopmans

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Dr. YangQuan Chen
Major Professor

Dr. Christopher Winstead
Committee Member

Dr. Reyhan Baktur
Committee Member

Dr. Byron R. Burnham
Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2010
Copyright  ©  Calvin Coopmans  2010

All Rights Reserved
Abstract


by

Calvin Coopmans, Master of Science
Utah State University, 2010

Major Professor: Dr. YangQuan Chen
Department: Electrical and Computer Engineering

This thesis presents work done towards a Personal Remote Sensing (PRS) system: small Unmanned Aerial Vehicles (UAVs) with electronic, control, and sensing subsystems. Based on papers presented to conferences (AutoTestCon2008 and MESA2009), as well as other work on PRS, multiple levels of engineering are detailed: complex multi-UAV data flow; attitude estimation filters; real-time microprocessor functionality; and small, mobile power systems. Wherever possible, Open-Source tools and designs have been used, modified, or studied, providing excellent cost to performance ratios in most cases. First, the overall PRS UAV architecture, AggieAir, is presented with a motivating examples (Ghost-Eye and EagleEye camera payloads). Then, AggieNav, an inertial navigation system for small UAVs, is detailed, along with information about a Kalman filter for estimation of UAV navigation, position, and attitude. Finally the Spatial Environment Autonomous Logger (SEAL), a general-purpose wireless datalogger for small UAV applications, is presented, with application examples with and without small UAVs. This work represents designs based on two years of organic small UAV system growth, and provides integrated solutions to many problems of small UAV communication, sensing, and control.

(118 pages)
To my family; to my friends; and mostly, to my enemies (if any).
Acknowledgments

I would like to sincerely thank my advisor, Dr. YangQuan Chen, for his endless support, ideas, perspectives, and artful encouragement during the last two and a half years. Thanks to Dr. Chen, I have had—and continue to have—a unique, challenging, and ever-rewarding career as a graduate student. In the service of my laboratory, the Center for Self-Organizing and Intelligent Systems (CSOIS), I have been to Las Vegas, NV, for my first ASME (IDETC08) conference; Salt Lake City, UT, to present my first conference paper (AutoTestCon08); Boston, MA, for the 2009 Microsoft Imagine Cup competition; Uganda, Africa, for humanitarian work; San Diego, CA, for my 2nd ASME conference (where I presented five papers); and Golden, CO, for a nice conference on networked robotics (NetRob09). Dr. Chen, thank you for turning my master’s degree program into a grand adventure.

I owe an incalculable debt of gratitude to Dr. Gary Bohannan, who showed me—and continues to show me—crucial mentorship and guidance. Thank you, Gary.

Dr. Mac McKee, director of the Utah Water Research Laboratory (UWRL), funded the research in this thesis, and continues to support CSOIS UAVs and my endeavors. Thank you, Mac.

In the summer of 2009 I spent a month in Uganda helping orphans and having the biggest adventure (thus far) of my life with Dr. Roger Hansen of the Utah Bureau of Reclamation. This trip irrevocably and wonderfully changed my life for the better. Thank you, Roger.

I would also like to thank the Utah State University Electrical and Computer Engineering Department, as well as Utah State University itself, for the utmost commitment to the students even in these desperate economic times.

Many people at USU have brought joy and education to me. Among them, I would like to thank my committee members, Drs. Winstead and Baktur, for not only agreeing to be part of my thesis committee, but also for following through. I would like to thank Yiding
Han for his dependability in intellectual partnership, Daniel Stuart for his dependability in general, Christophe Tricaud for being a constant source of cheer and sarcasm, Shayok Mukhopadhyay for his wonderfully unique style (and for teaching me to eat Indian food properly), and all the members of CSOIS for being excellent scholars and generally buena gente.

Several people who helped me arrive deserve thanking: Thanks to Dr. David Klumpar of the Montana State University Space Science and Engineering Lab (SSEL) for providing me with the tools, environment, and employment to learn real engineering skills. Thanks to Dr. Joe Shaw and Dr. Bob Gunderson (formally of CSOIS) from the MSU Electrical and Computer Engineering Department for their support by way of writing me letters of recommendation and for their assistance and guidance in applying to the graduate program.

Lastly, Dr. Todd Moon of the USU ECE Department was kind enough to believe in me, and for his crucial support of my acceptance into the graduate program I am most grateful. Thank you, Mary Lee Anderson, ECE graduate advisor and department secretary extraordinaire, for always keeping me on track, from drowning in a sea of paperwork, and for your attention to the text and formatting during the editing of this thesis.

Calvin Coopmans
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>Acronyms</td>
<td>xiv</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Thesis Organization</td>
<td>2</td>
</tr>
<tr>
<td>2 Personal Remote Sensing</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Sensing at a Distance</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Personal Sensing</td>
<td>4</td>
</tr>
<tr>
<td>2.3 PRS Outlook</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Thesis Contributions to PRS</td>
<td>8</td>
</tr>
<tr>
<td>2.5 Chapter Summary</td>
<td>9</td>
</tr>
<tr>
<td>3 AggieAir: An Integrated and Effective Small Multi-UAV Command, Control, and Data Collection Architecture</td>
<td>10</td>
</tr>
<tr>
<td>3.1 Functionality</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Data Flow</td>
<td>11</td>
</tr>
<tr>
<td>3.3 2.4GHz Wi-Fi Communication Link</td>
<td>12</td>
</tr>
<tr>
<td>3.4 JAUS</td>
<td>13</td>
</tr>
<tr>
<td>3.5 Coven-Ready Multi-UAV Capabilities</td>
<td>15</td>
</tr>
<tr>
<td>3.6 rsync for Data Transfer</td>
<td>16</td>
</tr>
<tr>
<td>3.7 AggiePilot</td>
<td>17</td>
</tr>
<tr>
<td>3.8 Gumstix Embedded Computer System</td>
<td>19</td>
</tr>
<tr>
<td>3.9 AggieNav Navigation Sensor</td>
<td>20</td>
</tr>
<tr>
<td>3.10 Example Payload 1: GhostFoto</td>
<td>21</td>
</tr>
<tr>
<td>3.11 Example Payload 2: EagleEye</td>
<td>23</td>
</tr>
<tr>
<td>3.12 Chapter Summary</td>
<td>25</td>
</tr>
<tr>
<td>4 AggieNav: AggieAir Inertial Measurement and Navigation Design</td>
<td>26</td>
</tr>
<tr>
<td>4.1 UAV Flight Testing</td>
<td>28</td>
</tr>
<tr>
<td>4.2 Low-Cost UAV Navigation Solutions and AggieNav Comparison</td>
<td>28</td>
</tr>
<tr>
<td>4.3 Enter AggieNav</td>
<td>30</td>
</tr>
<tr>
<td>4.4 AggieNav Subsystems</td>
<td>30</td>
</tr>
</tbody>
</table>
5 Design and Implementation of Sensing and Estimation Software in AggieNav

5.1 UAV Attitude Estimation
  5.1.1 Attitude Representation
  5.1.2 Implementation of an Extended Kalman Filter (EKF)
  5.1.3 Attitude Estimation using EKF

5.2 Experimental Results
  5.2.1 Raw Sensor Data Comparison
  5.2.2 Preliminary Results for Attitude Estimation using Kalman Filter

5.3 Chapter Summary

6 A Generic UAV Payload: Modular Design

6.1 Related Work

6.2 SEAL Design Details
  6.2.1 Power System
  6.2.2 SEAL Sensor Interfaces
  6.2.3 GPS Integration
  6.2.4 Data Memory
  6.2.5 USB Interface
  6.2.6 Optional Wi-Fi Interface
  6.2.7 User Interface

6.3 Data Storage

6.4 Sensor Options

6.5 Results

6.6 Applications
  6.6.1 UAVs
  6.6.2 Fleet Vehicles

6.7 Chapter Summary

7 Conclusion

7.1 Summary

7.2 Future Work

References
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>AggieNav versus other small INS solutions.</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>AggieNav power, mass, and cost budget.</td>
<td>38</td>
</tr>
<tr>
<td>6.1</td>
<td>SEAL R2 power and mass budget.</td>
<td>55</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Passive remote sensing.</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Active remote sensing requires some kind of stimulation to reveal the desired data.</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Complete Google search results for “personal remote sensing” (April 5th, 2010).</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>The Estes Eagleye remote control toy. Image copyright Amazon.com.</td>
<td>7</td>
</tr>
<tr>
<td>2.5</td>
<td>The Spygear ATV-360. Image copyright Toy Store Inc.</td>
<td>7</td>
</tr>
<tr>
<td>2.6</td>
<td>The WowWee Rovio telepresence robot. Image copyright Thinkgeek Inc.</td>
<td>8</td>
</tr>
<tr>
<td>3.1</td>
<td>AggieAir Aero-team block diagram.</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>AggieAir detailed system architecture.</td>
<td>12</td>
</tr>
<tr>
<td>3.3</td>
<td>AggieAir system complexity changes depending on mission requirements.</td>
<td>13</td>
</tr>
<tr>
<td>3.4</td>
<td>Ubiquity Bullet2-HP modified for small UAV flight with 5.5dBi antenna.</td>
<td>14</td>
</tr>
<tr>
<td>3.5</td>
<td>Wi-Fi groundstation: 14dBi, 160° horizontal sectorial antenna with Bullet2-HP.</td>
<td>15</td>
</tr>
<tr>
<td>3.6</td>
<td>Gain pattern of sectorial antenna.</td>
<td>16</td>
</tr>
<tr>
<td>3.7</td>
<td>rsync data collection script flowchart.</td>
<td>18</td>
</tr>
<tr>
<td>3.8</td>
<td>Paparazzi TWOG board: a fully autonomous autopilot system.</td>
<td>18</td>
</tr>
<tr>
<td>3.9</td>
<td>Gumstix Overo computer-on-module, between a US Quarter coin, an actual stick of chewing gum, and the Gumstix Tobi board.</td>
<td>20</td>
</tr>
<tr>
<td>3.10</td>
<td>Ghostfoto CCD camera: modified Canon Powershot SX100-IS.</td>
<td>22</td>
</tr>
<tr>
<td>3.11</td>
<td>AggieAir GhostFoto data flow.</td>
<td>22</td>
</tr>
<tr>
<td>3.12</td>
<td>Aerial image and orthorectified image.</td>
<td>24</td>
</tr>
<tr>
<td>3.13</td>
<td>Logitech, Inc. Quickcam Orbit autofocus pan-tilt camera.</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>AggieNav hardware block diagram.</td>
<td>30</td>
</tr>
</tbody>
</table>
4.2 AggieNav internal data connection diagram. .......................... 31
4.3 6-DoF IMU: Analog Devices ADIS16354. ......................... 32
4.4 ADIS1635X block diagram. ............................................. 32
4.5 IMU data before and during takeoff from 111609 flight. .......... 33
4.6 Honeywell HMC6343 magnetic compass with tilt-compensation. .. 34
4.7 Sample of magnetic compass data from 111609 flight. ............. 34
4.8 AggieNav prototype with Gumstix and GPS unit. Mass as pictured: 70g. 35
4.9 GPS flight log from 111609, plotted in Google Earth. .............. 36
4.10 VTi SCP-1000 pressure sensors. ..................................... 37
4.11 Selected pressure data from the 111609 flight. ..................... 37
4.12 AVR32 processor core diagram. ..................................... 41
4.13 AggieNav main data collection flowchart. .......................... 42
4.14 AggieNav EKF failover diagram. ..................................... 42
5.1 Gyro sensor comparison: \( p \). ........................................ 47
5.2 Gyro sensor comparison: \( q \). ........................................ 47
5.3 Gyro sensor comparison: \( r \). ........................................ 48
5.4 Acceleration sensor comparison: \( x \). ................................. 48
5.5 Acceleration sensor comparison: \( y \). ................................. 49
5.6 Acceleration sensor comparison: \( z \). ................................. 49
5.7 Roll angle estimation comparison using the AggieNav EKF. ........ 50
5.8 Pitch angle estimation comparison using the AggieNav EKF. ....... 51
6.1 SEAL Revision 0 prototype hardware. Includes several sensors, a Secure Digital memory card slot, and a GPS unit. ................. 53
6.2 SEAL Revision 1 hardware with sensors and Wi-Fi. This package weighs 114g (light enough for a small aircraft payload) and has 8+ hours of run time. 54
6.3 SEAL system diagram. .................................................. 54
6.4 SEAL power diagram. ........................................ 56
6.5 SEAL power distribution diagram. .......................... 57
6.6 SEAL sample GPX log. ....................................... 60
6.7 Plot of uncalibrated CO$_2$ vs. location via GPSVisualizer. .......... 61
6.8 Plot of uncalibrated NH$_3$ vs. location via GPSVisualizer. ........... 62
6.9 Location data from fig. 6.7 and 6.8 plotted in Google Earth. ........... 63
6.10 UAV monitoring cattle in a remote location. .................... 64
6.11 UAV returning to Wi-Fi base station for high-speed data downlink. .... 64
A.1 SEAL Revision 2 hardware. .................................. 80
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>Application-Specific Integrated Circuit</td>
</tr>
<tr>
<td>AUVSI</td>
<td>Association for Unmanned Vehicle Systems International</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-Coupled Device</td>
</tr>
<tr>
<td>CO₂</td>
<td>Carbon Dioxide</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSOIS</td>
<td>Center for Self-Organizing and Intelligent Systems</td>
</tr>
<tr>
<td>DCM</td>
<td>Direction Cosine Matrix</td>
</tr>
<tr>
<td>DDR</td>
<td>Double Data Rate</td>
</tr>
<tr>
<td>DFU</td>
<td>Device Firmware Upgrade</td>
</tr>
<tr>
<td>DoF</td>
<td>Degrees of Freedom</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Electrically Erasable Programmable Read-Only Memory</td>
</tr>
<tr>
<td>GPIO</td>
<td>General Purpose Input/Output</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GPX</td>
<td>GPS EXchange format</td>
</tr>
<tr>
<td>ICE</td>
<td>In Circuit Debugging</td>
</tr>
<tr>
<td>IIC, I₂C</td>
<td>Inter-Integrated Circuit Standard</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>INS</td>
<td>Inertial Navigation System</td>
</tr>
<tr>
<td>JAUS</td>
<td>Joint Architecture for Unmanned Systems</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group (IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture)</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>Li-Poly</td>
<td>Lithium-Ion Polymer</td>
</tr>
<tr>
<td>LIDAR</td>
<td>Light Detection And Ranging</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>LNA</td>
<td>Low Noise Amplifier</td>
</tr>
<tr>
<td>MEMS</td>
<td>MicroElectroMechanical Systems</td>
</tr>
<tr>
<td>MIPS</td>
<td>Millions of Instructions Per Second</td>
</tr>
<tr>
<td>MSD</td>
<td>Mass Storage Device</td>
</tr>
<tr>
<td>NDVI</td>
<td>Normalized Difference Vegetation Index</td>
</tr>
<tr>
<td>NH₃</td>
<td>Ammonia</td>
</tr>
<tr>
<td>OBS</td>
<td>Ocean Bottom Seismometers</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source Source Software</td>
</tr>
<tr>
<td>OTG</td>
<td>On The Go</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PRS</td>
<td>Personal Remote Sensing</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>RADAR</td>
<td>RAdio Detection And Ranging</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>SEAL</td>
<td>Spatial Environmental Autonomous Logger</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure SHell</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transfer Control Protocol over Internet Protocol</td>
</tr>
<tr>
<td>TWOG</td>
<td>Tiny Without GPS</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Serial Bus</td>
</tr>
<tr>
<td>VIP</td>
<td>Very Important Person</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Wireless Fidelity, IEEE 802.11b/g/n standard</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

Imagine a near-future farmer, looking at a map of her fields on a tablet computer from the deck of her house. This map includes detailed pictures of the current state of her crops (in this case an accurate representation of the current water content of her fields), as well as the history of the water usage of every part of her farm. Using this data, she assigns paths and plans the movements of her robotic water sprinkler system, allowing her to save money against high water rates, as well as conserve resources for other water users downstream from her. A scenario like this can be made possible by Personal Remote Sensing (PRS), via small Unmanned Aerial Vehicles (UAVs).

Remote sensing via small UAVs is fertile ground for research. Civilian applications for such systems are just starting to be realized, and new opportunities for research are appearing rapidly. A small UAV system, extensible for new payloads and applications, can fulfill many of the needs of this new field of remote sensing, and provide further insight and data in many fields of active research. Others would benefit, for example, such as highway passengers, if UAV remote sensing systems were deployed for inspection of roadways to increase hazard safety. A detailed introduction and review of existing literature can be found in Chapter 2.

This thesis presents efforts to move toward the promise of Personal Remote Sensing with Unmanned Aerial Vehicles. Since the main driver of such “personal” systems is price, the system design efforts have been centered around providing the most functionality and scientific effectiveness for the lowest price. Of course, a quest for low price and features in any electronic field is unending; therefore the work contained in this thesis is simply a snapshot of the AggieAir system available as of this writing.
1.2 Thesis Organization

• Chapter 2 gives an overview and background of and motivation for Personal Remote Sensing.

• Chapter 3 introduces the AggieAir architecture, and shows how it is used for small UAV-based Personal Remote Sensing.

• Chapter 4 contains information about AggieNav navigation system hardware and low-level programming and software design.

• Chapter 5 is about the progress made on AggieNav state-based Kalman filter estimation software.

• Chapter 6 covers SEAL, a general geo-spatial UAV payload datalogger, and applications.

• Chapter 7, the conclusion of this work, summarizes the results obtained and presents a spectrum of possibilities for future work.

The appendices contain the files and lessons resulting from the hardware design and production of various AggieAir components.

• Appendix A holds a small report on the lessons learned and practical insights into the joy and pain of hardware development from AggieNav and SEAL.

• Appendix B contains schematic information for the AggieNav navigation unit hardware.

• Appendix C has the board artwork for the completed printed AggieNav circuit board (PCB).

• Appendix D has schematic diagrams for the SEAL datalogger.

• Appendix E has the respective board layout artwork for the latest version of the SEAL PCB.
Chapter 2

Personal Remote Sensing

Sensing, or the use of sensors, is a basic function of life. Active entities, such as living organisms from mammals to mosses and non-living things from robots to traffic lights, change themselves or react to their environments depending on their specific goals or behaviors. Every entity which has the ability to make choices or react must do so on the basis of feedback; information that is percipient to the situation, and gathered through some accessible and usually repeatable means. The gathering of useful information—reliable information—is the act of detection, that is, purposeful “fishing” information most important to the task at hand. For this feedback information to be useful, of course, it must be timely enough to use effectively. Therefore, the act of sensing for feedback must be in at least half the same time-scale as (at the Nyquist rate of) the process of change.

2.1 Sensing at a Distance

While information can come from many sources, the best way to collect specific information is by the use of a sensor. Multiple sensors work in concert providing many sources of good information at once; depending on the information demanded these sensors could be actuated (alone, or with other entities via communication) to gather better data as time passes and requirements or environments change. As humans, we have a given set of perceptions granted by our suite of sensors (eyes, ears, nerves, etc.), but with science, through engineering, we are able to extend this perception set further. Remote sensing allows us to gather data at a physical distance, data for which there is either no other way or no safe way to collect in person. Remote sensing, then, is simply detection at a distance.

There are two methods of remote sensing: passive (fig. 2.1) and active (fig. 2.2). Passive remote sensing is as described above: “waiting” for data where it is expected to be
found. Active sensing involves stimulation of the sensed environment to excite or otherwise illuminate the situation in question, not unlike using a flashlight to inspect a noise in the dark. Many examples of active remote sensing can be found today: RADAR systems [1], the use of lasers for detection of aerosols in the atmosphere [2], detection of evasive species [3], or detection of land mines via honeybees [4].

2.2 Personal Sensing

Remote sensing usually connotes large-scale instruments requiring large-scale support operations (such as the Hubble telescope), and budgets far beyond even the most serious everyday user’s budget. However, as humans, we have been attempting to extend our senses for as long as we have used tools. Devices such as the looking glass allow our eyes to see further, and the “tin ear” extends the range of an ear. These are examples of personal remote sensing: the detection of data useful to a small number of people on short time-scales and within small budgets.

Although the term is not commonly used currently (fig. 2.3), much like the beginning of the Personal Computer (PC) age, examples of Personal Remote Sensing (PRS) systems are beginning to emerge. Some can be found in consumer-level hardware and toys such as the Estes Eagleye [5], a radio-controlled airplane with an integrated digital still camera and memory (fig. 2.4), and the Spygear ATV-360 [6] (fig. 2.5) which includes a head-mounted, greyscale, live video feed from a camera on the vehicle, in addition to a sound feed from a microphone. Under the moniker “telepresence,” the WowWee Rovio [7] seen in fig. 2.6 [8] provides many of the same features as the ATV-360 with a more robust vehicle.

Fig. 2.1: Passive remote sensing.
Fig. 2.2: Active remote sensing requires some kind of stimulation to reveal the desired data.

chassis, better camera system, Wi-Fi for high-bandwidth data transfer, and a speaker for bi-directional user interaction.

Like the PC, PRS systems have much more potential than simple playthings, or incidental computer peripherals. Aerial photography is one application of PRS that has been named by Shimada et al. [9] (fig. 2.3). Shimada makes use of a radio-controlled helicopter with an NDVI (Normalized Difference Vegetation Index) [10] detection system gathering 4-band (near-infrared, red, green, blue) data about grazing land. Systems such as these can provide vital data for individual farmers, with costs orders of magnitude lower and with data of much higher density than alternatives such as flying full-size aircraft over the fields or making use of satellite NDVI data.

However, all of the aforementioned systems are lacking aspects of autonomy: these systems require a large amount of user interaction for useful function. Flying, driving, or otherwise controlling a remote sensing system can be time consuming and in some cases demands high levels of skill or training for consistent quality data. An autonomous system such as an Unmanned Aerial Vehicle (UAV) can be preprogrammed to patrol for changes in a scene or “follow” a data trail such as an airborne pollutant to collect data automatically. In the case of NDVI, a farmer (as in above) could use a UAV for remotely sensing the water content in her fields, seeing where the water is needed most. Such a PRS will not only save water for others downstream, but will also save the farmer time and energy otherwise spent overwatering.
2.3 PRS Outlook

The key driver for success with PRS systems is price. Personal remote sensing can only be personal if individuals and small groups can afford to own and operate systems capable of delivering the data they need. Progression of technology (Microelectromechanical or MEMS devices, embedded processing power, battery power density, etc.) coupled with acceptance of PRS systems by industry and legislators, will allow PRS to become commonplace.
Fig. 2.4: The Estes Eagleye remote control toy. Image copyright Amazon.com.

Fig. 2.5: The Spygear ATV-360. Image copyright Toy Store Inc.
PRS systems are poised to provide the next generation of useful data about our world. While large-scale remote sensing (like large-scale computing) will always have applications such as weather prediction, PRS allows individuals or small groups to collect valuable data about situations or processes that concern them specifically. Personal remote sensing is an upcoming technology, and will show effectiveness when combined with UAVs and other autonomous systems. With PRS, new ways of interaction and feedback can be used—with scientific accuracy—to make better, more efficient decisions about what concerns us most.

2.4 Thesis Contributions to PRS

For PRS systems to be affordable and still provide scientifically accurate data, the electronic sensing and control architectures must be miniaturized and integrated in thoughtful ways, while leveraging existing technology whenever possible. This thesis presents advances in small UAV PRS systems, specifically toward UAV use for civilian use such as farming and agriculture. Extendible system approaches, such as the work in Chapter 3, allow for systems to scale and grow with new sensing solutions and technologies, while integrated designs as in Chapter 4 and Chapter 6 give systems low cost and mass-producibility.

Fig. 2.6: The WowWee Rovio telepresence robot. Image copyright Thinkgeek Inc.
2.5 Chapter Summary

This chapter provides a low-level motivation for and definition of personal remote sensing. A description of remote sensing is presented, then extended into more personal examples, where current consumer personal remote sensing systems are presented. An outlook shows the prospects for PRS in the near future, and the need for PRS systems, which will drive development and legal support for PRS in the future.
Chapter 3

AggieAir: An Integrated and Effective Small Multi-UAV Command, Control, and Data Collection Architecture

An Unmanned Aerial Vehicle (UAV) is a complex aircraft system that is able to navigate without the manual control of a human pilot. The inception of UAVs traces back to World War I, and they have been used primarily for military applications. However, entry of UAVs into the civil/commercial market is as recent as the last few decades. There is an increasing interest in developing UAV systems for a variety of civil applications such as traffic control, border patrol, fire-fighting management [11], agriculture monitoring [12], etc. UAVs of different sizes, types, and configurations are developed for specific applications.

Generally speaking, an autonomous UAV system consists of automatic pilot (autopilot) system, navigation system that includes a variety of sensors, on-board microcomputer system for coordination, and a ground control station that monitors and changes flight missions in real-time.

At the Center of Self-Organizing and Intelligent System (CSOIS) at Utah State University, miniature fixed-wing autonomous UAVs are developed for civil applications, such as water management, irrigation control, highway mapping, etc. The UAVs we developed are equipped with light-weight multispectral high-resolution optical imagers for aerial images within reconfigurable bands [13].

This chapter presents AggieAir, a novel hardware and software architecture. AggieAir is fundamentally based on a proven system architecture: that of a real aircraft. Figure 3.1 shows a simplified version of this hierarchy, allowing specific mission tasks and goals. On the ground, the Flight Commander (a human or humans, in the AggieAir system), oversees the whole mission as it progresses. The Science Team is also on the ground, monitoring data and controlling experiments on the plane in real-time.
3.1 Functionality

Overall, the AggieAir Aero-team paradigm is illustrated in fig. 3.1. Each part of the system has its respective role, even the humans on the ground. A much more detailed system block diagram is fig. 3.2, which shows some of the nuances of the AggieAir system.

3.2 Data Flow

Figure 3.3 shows the AggieAir data flow hierarchy versus system complexity. This allows for many system configurations depending on the demands of the mission at hand. A demand analysis is carried out for each mission, and given the mission parameters (an update frequency required by the ground operators, for instance) and payload demands (on-board processing, communication to other UAVs in flight), the various parts of the AggieAir system can be included or excluded as needed.

For example, a particular mission could demand only AggiePilot and AggieNav for basic navigation and flight path following. AggieCap and the other parts of the system are not required and could therefore be left out of the system. Given a more complex mission, one more level could be added: AggieCap’s payload management could be demanded without the need for the Wi-Fi data link and this could easily be left out of the mission. The full system (AggieNav, AggieCap, and Wi-Fi with multiple ground stations) could be also implemented easily if required for a complex mission with multiple UAVs.

![AggieAir Aero-team block diagram](image)

Fig. 3.1: AggieAir Aero-team block diagram.
3.3 2.4GHz Wi-Fi Communication Link

CSOIS has developed a high-bandwidth 2.4GHz Wi-Fi (IEEE 802.11g) communication link for use with single or multiple UAVs and ground stations. Designed to be inexpensive and easily set up in the field, this link is based on lightweight (80g including antenna and cable), high-power (800mW), inexpensive ($70), “carrier-class” Wi-Fi units manufactured by Ubiquiti Wireless [14], known as the Bullet2-HP. These units are employed on both ends of the data link, and have been tested at more than one mile range to provide greater than 400KB/second and up to 1.6MB/second transfer speeds depending on antenna orientation.
The Bullets have been modified for flight by the removal of the outer plastic case, and the replacement of the heavy integrated N-male RF connector with a hard-soldered, short LMR-400 co-axial pigtail and a standard 5.5dBi monopole Wi-Fi antenna (fig. 3.4). A power-over-Ethernet (POE) splice cable allows the airborne Bullet unit to integrate with the onboard UAV li-ion battery supply as well as the Gumstix flight computer via Ethernet.

The ground station network is connected to the Wi-Fi network via a 14dBi sector antenna (pictured in fig. 3.5), giving the Wi-Fi link a wide horizontal angle (fig. 3.6, from the HG2414SP-120 Datasheet [15]), tested to function well at ±80 degrees from center. A vertical gain pattern of 20 degrees allows complete coverage of a long-range UAV flight area.

The alternative Wi-Fi operating system DD-WRT [16] supports Bullet2-HP units. DD-WRT is a Linux-based operating system with many capabilities, not the least of which is support for OLSR, the Open-Source Link State Routing protocol [17]. OLSR creates an Ad-hoc mode Wi-Fi mesh network by manipulating the Linux kernel routing table based on the status of the wireless network topology, allowing for a flexible UAV flying mesh such as the one implemented by Tisdale et al. [18].

3.4 JAUS

JAUS, the Joint Architecture for Unmanned Systems, is a comprehensive, cross-platform
command and control architecture designed for interoperability by the US Department of Defense, and is a standard for US military unmanned systems. It is ideal for architectures and systems such as AggieAir, and allows for well-tested packet-based transmission of critical data such as airframe pose or payload status. Implementations of JAUS vary in price and licensing; the AggieAir system uses OpenJAUS [19], a free and open-source implementation of the JAUS system.

JAUS is packet-level, and is defined for TCP, UDP, and serial links. Since JAUS needs a transport protocol to be effective, AggieAir uses TCP/IP for reliable communication.

While JAUS is ideal for command and control applications, due to the limited data-transfer types, it is not a particularly good choice for transmission of data from sensors or other payloads, and AggieAir therefore uses other protocols (such as rsync [20], see sec. 3.6) for data such as aerial images.
3.5 Coven-Ready Multi-UAV Capabilities

AggieAir allows for command and control of one or more UAVs. In the case of multiple (a “coven” of) UAVs, for example in distributed wind measurement, the UAVs communicate with each other and maintain a formation for collection of unique data available no other way.

For multi-UAV communication, the Wi-Fi network is used with a mesh-networking protocol (such as the above-mentioned OLSR), which maintains the best network topology over time, terrain, and other factors. This allows the UAVs to keep in contact. Once connected, the UAVs exchange position and other vital data though their JAUS subsystems,
allowing each UAV to request as much or as little data as needed for a particular task.

With a technology like JAUS, the UAVs using AggieAir can enjoy enhanced robustness. If, in a flying coven, UAV “A” suffers a hardware failure (the loss of the data-radio modem, for instance), UAVs “B,” “C,” and “D” can relay their data to the degraded UAV via the Wi-Fi network, and the coven can still perform the required mission.

In another scenario, a single “VIP” UAV could have a very expensive device, such as a high-quality IMU, while the other UAVs have more rudimentary navigation hardware, such as thermopiles and local-infrared sensors for targeting the VIP UAV. The whole coven could then navigate with the accuracy of the VIP UAV, by communicating while flying. With such a configuration, it is them possible to outfit the other UAVs with heavier or high-value payloads such as thermal cameras, while reducing overall cost without sacrificing performance.

3.6 rsync for Data Transfer

Currently, the best method for transferring arbitrarily large data sets over unstable links such as the Wi-Fi link in AggieAir is the rsync protocol [20]. rsync is an open-source cross-platform software utility that implements the rsync protocol, and provides fast incremental file transfer, allowing data transfers to be interrupted by variable-quality
communication links and resumed later when the link has been restored. A simple Bash “wrapper” script allows the Payload Ground Station to asynchronously request the data files from the airborne processor and save them on the groundstation for further processing or analysis. In this way, the ground network can re-target or change a UAV’s behavior based on any combination of human or machine decisions.

`rsync` is currently implemented in AggieAir on the Windows XP operating system via Cygwin [21], but a small section of compatibility code changes the `ping` syntax and allows the same script to operate on multiple Unix variants as well as Windows. Currently, the behavior of the airborne image payload is of a steady, continuous production of separate data files (discrete images, in the case of GhostFoto, see sec. 3.10). This technique will also perform nicely with growing-length (appended-mode) data files with different `rsync` options. Although `rsync` can be used in conjunction with other protocols such as SSH [22], to conserve airborne CPU load and bandwidth, the data is sent in an unencrypted stream via the standard `rsync` protocol.

Figure 3.7 shows the logic flowchart for collection of data via `rsync`. Note that in this case, `rsync` has been configured to remove the remote data after downlink to the ground, since each iteration will take progressively more time to index the source data files if they are not cleaned each loop iteration. Additionally, `rsync` is used in `Backup` mode so it will not overwrite files that have identical names in case a naming collision occurs (caused by a reset of the flight computer or the crashing of a payload, for instance). In this way, a reliable, cross-platform data collection system has been made using established, well-tested software.

3.7 AggiePilot

The AggieAir module responsible for both low-level flight control (airframe attitude control) and high-level execution of flight plans is adapted from a pre-existing project, Paparazzi. Paparazzi is an open-source, fully-autonomous autopilot developed by aerospace students at ENAC University in France, and CSOIS has been using Paparazzi since 2007 [23], to much success. Figure 3.8 shows the Tiny Without GPS (TWOG) Paparazzi autopilot
system board from the Paparazzi Wiki [24].

In the AggieAir architecture, the Paparazzi system has been augmented with a JAUS data channel to become AggiePilot, allowing JAUS packets to be transmitted to and from the UAV(s) while in flight. This allows for a useful amount of JAUS command and control data to be sent over the UAV’s normal control link, up to 10 miles away, depending on antenna and radio factors.

Paparazzi is an ideal platform on which to base a system like AggieAir. More system-level details as well as information about the CSOIS implementation and use of Paparazzi

Fig. 3.7: rsync data collection script flowchart.

Fig. 3.8: Paparazzi TWOG board: a fully autonomous autopilot system.
can be found in A. Jensen’s thesis [25].

3.8 Gumstix Embedded Computer System

AggieAir relies on the Gumstix computer modules for general mission operation, processing, as well as payload control. The Gumstix product line are so-named due to their form-factor’s similarity to sticks of chewing gum (fig. 3.9 from Wikipedia [26]), and for their relative low-cost. The current model of Gumstix used by the AggieAir system is the Gumstix Overo Earth. The Gumstix Overo computers have the following features [27].

- Price: $149.00 USD as of this writing
- Up to 1200 Dhrystone Millions of Instructions Per Second (MIPS)
- Based on the Texas Instruments OMAP3503 or OMAP3530 Processors
- 256MB Flash memory
- 256MB Double Data Rate (DDR) Random Access Memory (RAM)
- On-board MicroSD slot for expansion memory
- Universal Serial Bus (USB) powered host port
- USB On The Go (OTG) port
- Analog-to-digital converter, Pulse Width Modulation unit, etc.
- Ethernet via USB, or via Tobi (and other) expansion boards
- Bluetooth (Overo Air and Fire only)
- Wi-Fi (Overo Air and Fire only)

The majority of the software needed to operate the Overo computer is provided in the Ångström Linux distribution [28]. At least one other similar product to the Overo exists, the BeagleBoard [29] also uses the OMAP3530 processor, and natively runs the Linux
operating system, also shipping with the Ångström distribution. This is an advantage—the heterogeneity of distribution compilation targets will help guarantee continued community software support for the Overo computers, and allow systems using them to continue to persist before obsolescence for a longer period of time.

The OpenJAUS node manager is also running on the Gumstix, allowing the command and control of the software payloads relevant to the mission at hand. The Gumstix interfaces (USB, Ethernet, etc.) allow access to the other parts of the AggieAir system. AggieAir Payload software is written for the Linux system environment, and by use of standard libraries, can exploit the Open Source nature of the underlying software, allowing for low-cost upkeep and community support via the normal Open Source software channels (Internet blogs, email lists, forums, etc.).

3.9 AggieNav Navigation Sensor

Presented in much more detail later (Chapter 4), AggieNav is a small, accurate, low-cost, tightly-integrated UAV navigation sensor INS suite [30]. Figure 4.1 shows a system
block diagram. AggieNav serves as the Navigator in the aerial team paradigm, and provides the Captain and with real-time global pose (attitude and location) data from a 6-DoF IMU, GPS and compass module, as well as dual pressure sensors for estimation of altitude and airspeed via Pitot tube [31]. AggieNav also serves as a relay for filtered data between AggieCap and AggiePilot, with a safety fallback. As seen in fig. 4.8, AggieNav also holds and powers the Gumstix computer.

AggieNav uses some of the Gumstix CPU time for an Extended Kalman Filter (EKF), allowing for more accurate navigation. AggieNav then passes the filtered data along to AggiePilot, allowing the system to function with the best navigation performance. However AggieNav has enough processing power (70 MIPS) to allow a fallback/failsafe mode (fig. 4.14) if the Gumstix computer should fail or not be required for a given mission. If, after 1 second (100 packets), AggieNav does not receive data from its companion software process on the Gumstix, AggieNav will step in and provide its own data. In this case, the AggieNav processor does some basic filtering work on the sensor data and passes this data to AggiePilot in the same format as the normal system operation, allowing a seamless failover, or minimum flight gear configuration if flying without the Gumstix processor.

3.10 Example Payload 1: GhostFoto

GhostFoto (GFoto) is a light-weight, cost efficient, multi-spectral imager which operates in both visible and near infrared (NIR) light wavelengths. The hardware of this payload consists of RGB and/or NIR CCD cameras, shown in fig. 3.10, and Gumstix microprocessor with software component, GhostEye, running to both configure and control the imagers. GhostEye also provides an interface between the imager payload and UAV system, establishing a JAUS message and command channel, through which users are able to monitor and operate the payload during flight. Figure 3.11 shows this flowchart.

Another key functionality of GhostEye is geo-referencing of the airborne imagery. The orthorectification of the aerial images is indispensable for end users that demand accurate geographic information. For instance, in applications such as target recognition and ground mapping, the geographic coordinates of any pixel on the aerial image should be retrievable.
Fig. 3.10: Ghostfoto CCD camera: modified Canon Powershot SX100-IS.

Fig. 3.11: AggieAir GhostFoto data flow.
By querying sensor data via JAUS, such as the UAV geographic coordinates, and pitch, yaw, and roll angles of the plane, GhostEye logs the geo-referencing information for each image so that they can be accurately orthorectified [13], either post-mission, or in real-time during flight.

This access is enabled in the AggieAir architecture through AggieCap, which dispatches the EKF filtered flight data to JAUS, from which the payload (or any JAUS node) can query for geo-referencing information. The message and control channel of GhostEye is also managed by AggieCap, and is sent through AggieNav to AggiePilot, which communicates with ground station, where the status of GFoto is monitored.

Airborne imagery can be saved on the memory card in the cameras, or downloaded from the imager onto the SD card on the Gumstix processor, and hence sent to the Ground Control Station via rsync through the high-bandwidth Wi-Fi Datalink. This process allows the ground station to process the acquired imagery in real-time, allowing feature-based navigation can be implemented in AggieAir architecture.

As an example, fig. 3.12 shows an aerial image (top) taken over Little Bear River near the Utah State University (USU) owned research farm (GPS: 41.8155945, -111.9816552). The image is orthorectified with the provided geo-referencing data (rotated and projected onto globe coordinate with its upper side pointing North), and seen in the bottom picture of the figure.

### 3.11 Example Payload 2: EagleEye

A second example payload is the EagleEye system, designed to allow a UAV pilot a real-time look at the horizon, as if they were actually piloting the aircraft. Because other wireless camera systems use the 2.4GHz band occupied by the AggieAir Wi-Fi link, the best solution is to use the existing datalink to transfer the video data. This is made possible by the Apache HTTP server running on the Gumstix computer, with the `webcam_server` module. This allows a pilot or other observer to connect to the airframe from a groundstation, and stream video from the forward-looking camera at a highly compressed rate to a second monitor or other display device via a standard web browser. When properly compressed, this video
Fig. 3.12: Aerial image and orthorectified image.
stream can be usable for a pilot, and consume as little as 20-50kb/second of communication bandwidth.

The camera in fig. 3.13 (from the Logitech, Incorporated website [32]) is a full pan/tilt solution designed for consumer-level videoconferencing. This camera is ideal for several reasons; the foremost of which is Linux driver support by the manufacturer, Logitech. This driver allows access to the video streams as well as the pan, tilt, and focus settings. This camera is of adequate quality, and at $100 per camera, the system is functional and low-cost. Other Logitech camera solutions exist if the pan/tilt functionality is not needed, and can easily be integrated with minimal change in the underlying design and software.

The EagleEye camera system is commanded by JAUS from the groundstation, or other elements in the AggieAir system as demanded by mission requirements. The USB connection technique allows the video stream to be processed by the host Gumstix computer for use in target recognition or other flight-intelligence algorithms.

3.12 Chapter Summary

AggieAir is an effective, low-cost system design for unmanned aerial personal remote sensing systems. This chapter contains high-level design descriptions for AggieAir, with block diagrams and details of implementation to illustrate the system. Example payloads are included, with integration details showing how the AggieAir system is extensible and useful many types of UAV applications and missions.

![Logitech, Inc. Quickcam Orbit autofocus pan-tilt camera.](image)
Chapter 4

AggieNav: AggieAir Inertial Measurement and Navigation Design

To accomplish mission goals or to fly passengers to their destinations, all aircraft rely on sensors as the core of their navigational systems. For small UAVs, the sensor suite determines much about the usefulness and capabilities of such a system, and is mostly limited by the relatively high cost of high-quality navigational sensors and equipment.

The options for low-cost miniature attitude navigation sensing are few; most inexpensive, small fixed-wing UAVs use infrared thermopiles which sense the relative heat differential from the black body radiation of the Earth’s surface and the open sky. While this technique does work, it is not accurate enough for most remote measurement tasks and has problems when flying near large land features such as mountains, through valleys, or near oceans. When attitude sensors are combined with a modern civilian GPS receiver, autonomous navigation is possible. However, payloads such as cameras demand better attitude estimation performance to enable accurate orthorectification (see sec. 3.10), so better sensors and filtering schemes must be employed.

Inertial measurement units with roll-rate gyros and accelerometers are the standard sensors for attitude determination in flight. Several solutions exist for navigation in small UAVs, such as the Microstrain GX2 [33], CrossBow MNAV [34], and Procerus Kestral [35], all of which use Kalman or other state-estimation filter of some kind to provide a high-accuracy estimation of the current position and attitude of the system at a given instant. While these systems work well for their intended applications, even the academic prices are prohibitive for integration into single or multiple inexpensive UAVs. Additionally, the source code is not made available, and users of the systems are forced to accept the filter/algorithm performance of the systems as they are given.
Recently, due to the high-availability of low-cost inertial sensing chips, many consumer-level devices such as the Apple iPhone, Nintendo Wii, and Sony PlayStation 3 [36] have integrated accelerometers into their hardware for more immersive user-experiences. Several chip manufactures are producing 3-axis accelerometers, however, the availability of integrated 3-axis roll-rate gyro chips is not as broad. Analog Devices Incorporated is, at the time of this writing, the only company producing fully-integrated 6-degree-of-freedom sensor units [37]. Other academic efforts such as Hirokawa, Kajiwara, and Ohata [38] and Affonseca [39] have been put forth, but the Analog Devices part is the most flight-ready and integrated design currently available for purchase outside of the US military.

AggieNav is a unique design of advanced sensing, power and processing hardware, paired with sophisticated data filtering algorithms, giving it a decisive lead in front of other similar systems.

AggieNav has a unique combination of features.

- Full high-accuracy 6-Degree-of-Freedom IMU
- Full 3-axis tilt-compensated compass
- Standard interface with any low-voltage GPS unit via a serial port
- Dual pressure sensors for static and dynamic air pressures
- 72 MIPS onboard processing power
- Onboard hardware mounts for a 600MHz Gumstix Linux computer, and Paparazzi [24] based autopilot system (“AggiePilot”) for a full UAV system
- Lowest total price for any such system
- Open software architecture allows AggieNav to be customized or augmented easily
- Weight: 70g, power: 1.0W total with Gumstix and GPS
4.1 UAV Flight Testing

AggieNav has been tested on UAV flights as a payload in data-logging mode. In the following sections, plots of data recorded during a flight on November 16th, 2009, (hereafter written as “111609”) are included in the sections relevant to the various sensors included with AggieNav (figs. 4.5, 4.7, 4.9, 4.11).

4.2 Low-Cost UAV Navigation Solutions and AggieNav Comparison

Three different kinds of navigation solutions are common for small UAVs.

1. No IMU (typically IR sensor-based) with GPS
   - Least expensive option for navigation. Many drawbacks (inaccurate and difficult to tune, etc.), but functional.

2. Loosely integrated INS: IMU (such as the Microstrain 3DM-GX2) with uncoupled GPS
   - Estimations are workable, and better than IR-based navigation, but are not ideal.

3. Tightly integrated INS: IMU + GPS + Compass + Pressure etc. w/Kalman or other combining filter. These are the best current solutions for small (and large) UAV flight. Several companies are making solutions target UAV flight, but most are very expensive.
   - Micropilot (full autopilot solution) ($8,000)
   - XSens MTi-G ($7000)
   - Crossbow Micronav + Stargate Processor (discontinued, $1,900)
   - Procerus Kestral ($5,000)
   - AggieNav ($1,200)

More specific information on the closest competitors to AggieNav can be found in Table 4.1.
Table 4.1: AggieNav versus other small INS solutions.

<table>
<thead>
<tr>
<th>INS Unit</th>
<th>AggieNav</th>
<th>Stock Microstrain 3DM GX2</th>
<th>Crossbow MNAV</th>
<th>Procerus Kestrel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Gyro Bandwidth</td>
<td>700Hz (unfiltered)</td>
<td>250Hz</td>
<td>25Hz</td>
<td>9Hz</td>
</tr>
<tr>
<td>Max Gyro Dynamic Range</td>
<td>±300deg/sec</td>
<td>±300deg/sec standard</td>
<td>±150deg/sec</td>
<td>±300deg/sec</td>
</tr>
<tr>
<td>Accelerometer Dynamic Range</td>
<td>±10g</td>
<td>±5g</td>
<td>±2g</td>
<td>±10g</td>
</tr>
<tr>
<td>Accelerometer Resolution</td>
<td>14bit</td>
<td>16bit</td>
<td>NA</td>
<td>14bit</td>
</tr>
<tr>
<td>Accelerometer Bandwidth</td>
<td>700Hz (unfiltered)</td>
<td>250Hz</td>
<td>25Hz</td>
<td>9Hz</td>
</tr>
<tr>
<td>Magnetics Rate</td>
<td>5Hz heading solution</td>
<td>250Hz</td>
<td>1-100Hz</td>
<td>NA</td>
</tr>
<tr>
<td>Magnetics Accuracy</td>
<td>±3deg</td>
<td>0.001Gauss</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>GPS Rate</td>
<td>4Hz</td>
<td>No GPS</td>
<td>4Hz</td>
<td>NA</td>
</tr>
<tr>
<td>Max GPS Accuracy</td>
<td>2.5m</td>
<td>No GPS</td>
<td>3m</td>
<td>5m</td>
</tr>
<tr>
<td>Pressure Rate</td>
<td>1.8Hz</td>
<td>No Pressure</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Pressure Accuracy (Abs)</td>
<td>1.5Pa</td>
<td>No Pressure</td>
<td>NA</td>
<td>2.44Pa</td>
</tr>
<tr>
<td>Power Draw</td>
<td>600mW</td>
<td>470mW (minimum)</td>
<td>400mW</td>
<td>700mW</td>
</tr>
</tbody>
</table>
4.3 Enter AggieNav

AggieNav has been developed as a “best of” between low-accuracy thermopile sensors and high-cost, closed source navigation systems. By leveraging new digital MEMS sensors, such as Analog Devices ADIS1635x IMU parts, a new level of performance vs. cost can be achieved. Figure 4.1 provides a block diagram of AggieNav’s sensors and system design.

Since many of our UAV missions at Utah State University (and indeed, the general development of filtering for AggieNav’s sensor data) require the use of a Gumstix Linux computer, a mounting footprint and connector has been included, as well as a 5V-powered USB-host port. This allows the high-level flight software to be aware of the current flight/mission state via JAUS, creating many possibilities such as location-based science and swarm/mesh (coven) networked behavior. Figure 4.2 shows how AggieNav integrates with the AggieAir hardware system defined in Chapter 3.

4.4 AggieNav Subsystems

AggieNav has a unique combination of sensors to support high quality flight navigation.
All devices on AggieNav are factory calibrated and have digital interfaces to shift the load of testing and verification from the system assembly stage and allow for the fastest possible, lowest cost assembly time.

4.4.1 Inertial Measurement Unit

The primary sensor for AggieNav is an Analog Devices ADIS1635x Inertial Measurement Unit (fig. 4.3 and fig. 4.4 from the ADIS16354 Datasheet [37]), with six full axis of roll-rate and accelerometer data. Analog Devices manufactures several different models in the same IMU family which vary in sensing ranges and included sensors. The ADIS1635x sensors are factory calibrated and require little integration during manufacturing of an AggieNav unit. The ADIS1635x sensors have comparable performance to (and in many cases, better performance than) competing small UAV INS solutions.

IMU data from the 111609 test flight is seen in fig. 4.5. We can observe clearly a constant 1g acceleration force before takeoff (via a “bungie” [25]), and forces seen after the takeoff event represent the measurements effected by the flight dynamics. Note that the data becomes more noisy after the electric propellor motor is engaged, as is to be expected.
Fig. 4.3: 6-DoF IMU: Analog Devices ADIS16354.

Fig. 4.4: ADIS1635X block diagram.
Fig. 4.5: IMU data before and during takeoff from 111609 flight.

4.4.2 Magnetic Compass

A Honeywell HMC6343 tilt-compensated magnetic compass (shown in fig. 4.6 from its datasheet [40]) provides slower (2Hz), more accurate absolute heading data. The HMC6343 compass includes an MSP430 ASIC as well as internal accelerometers to provide a tilt-compensated heading independent of orientation. The internal filtering algorithms have been enabled and set to their most low low-pass settings to better eliminate mechanical resonances generated by the flight dynamics. Flight data from this sensor can be seen in fig. 4.7; note that the heading reference is indeed constant for parts of the flight, and the noise is low.

4.4.3 GPS Receiver

A high-rate civilian GPS receiver, a 4Hz u-Blox LEA-5H unit, provides data about the UAV’s global position. This GPS receiver is pictured alongside AggieNav in fig. 4.8
Fig. 4.6: Honeywell HMC6343 magnetic compass with tilt-compensation.

Fig. 4.7: Sample of magnetic compass data from 11/16/2009 flight.
Fig. 4.8: AggieNav prototype with Gumstix and GPS unit. Mass as pictured: 70g.

and includes an LNA (low-noise amplifier) helical antenna from Sarantel [41]. Although requiring 100mW of extra power beyond alternates with passive antennas, this GPS solution gives AggieNav and the host UAV the best possible signal gain towards the sky and the best 3D GPS lock accuracy.

AggieNav’s GPS receiver is designed to be replaceable/upgradable as newer and better GPS units appear on the market, and for this reason the GPS receiver is separated by a length of wire and an interface connector. This technique also allows the GPS system to be relocated anywhere on the UAV for the best possible RF interference avoidance. In fig. 4.9, a 3D plot of the full flight location and elevation log from the u-Blox receiver during the 111609 flight is visualized in Google Earth.

4.4.4 Pressure Sensors

Two small yet accurate pressure sensors (fig. 4.10, from VTi [42]) give accurate (1.5Pa) pressure, allowing for the calculation of both absolute elevation (with launch time cali-
Fig. 4.9: GPS flight log from 111609, plotted in Google Earth.

36

4.5 Power System and Temperature

The AggieNav switching power system is designed for the full voltage range of the batteries in the CSOIS UAVs [23] (5.1-17.0V), and provides additional power to the Gumstix computer and the onboard host-mode mini-USB port. Two power rails, 5.0 and 3.3V, are implemented with Texas Instruments TPS62110-series step-down switching power converters as in the datasheet [43]. This power system has been tested in flight conditions and efficiently provides clean power to the AggieNav system and hosted Gumstix board. A full power and mass budget is presented in Table 4.2. All parts used in AggieNav are temperature tolerant from at least -20° to 80° C.

4.6 Microprocessor and Embedded Software

AggieNav is based on the 32-bit AVR32UC3B0256 microprocessor from Atmel Corporation [44]. This processor (fig. 4.12, from the datasheet [44]) gives 72 MIPS at 60 MHz with
Fig. 4.10: VTi SCP-1000 pressure sensors.

Fig. 4.11: Selected pressure data from the 111609 flight.
Table 4.2: AggieNav power, mass, and cost budget.

<table>
<thead>
<tr>
<th>AggieNav Budget</th>
<th>Voltage (V)</th>
<th>Current (mA)</th>
<th>Power (mW)</th>
<th>Mass (g)</th>
<th>Cost (USD)</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT32UC3B0256 CPU</td>
<td>3.3</td>
<td>23</td>
<td>75.9</td>
<td>0.2</td>
<td>$12.00</td>
<td>Sample from Atmel</td>
</tr>
<tr>
<td>ADIS16355 IMU</td>
<td>5</td>
<td>57</td>
<td>285</td>
<td>16</td>
<td>$357.00</td>
<td>Analog Devices</td>
</tr>
<tr>
<td>HMC6343 Digital Compass</td>
<td>3.3</td>
<td>13</td>
<td>42.9</td>
<td>0.32</td>
<td>$175.00</td>
<td>Digikey</td>
</tr>
<tr>
<td>u-Blox LEA-5H GPS</td>
<td>3.3</td>
<td>60</td>
<td>198</td>
<td>2.1</td>
<td>$100.00</td>
<td>u-Blox America</td>
</tr>
<tr>
<td>GPS Antenna</td>
<td>3.3</td>
<td>110</td>
<td>363</td>
<td>10</td>
<td>$10.00</td>
<td>Digikey</td>
</tr>
<tr>
<td>SCP1000-D11 Pressure Sensor</td>
<td>3.3</td>
<td>0.03</td>
<td>0.08</td>
<td>0.2</td>
<td>$20.00</td>
<td>VTI.no</td>
</tr>
<tr>
<td>SCP1000-D11 Pressure Sensor</td>
<td>3.3</td>
<td>0.03</td>
<td>0.08</td>
<td>0.2</td>
<td>$20.00</td>
<td>VTI.no</td>
</tr>
<tr>
<td>Misc Electronics Hardware</td>
<td>3.3</td>
<td>20</td>
<td>66</td>
<td>15</td>
<td>$25.00</td>
<td>Estimate</td>
</tr>
<tr>
<td>PCB @ 30cm²</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>$20.00</td>
<td>Estimate</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>1030.97</strong></td>
<td><strong>64.02</strong></td>
<td></td>
<td></td>
<td><strong>$739.00</strong></td>
<td></td>
</tr>
</tbody>
</table>
one of the lowest modern watt-per-MIP ratings in its class. AggieNav is fully programmable from both the Linux and Windows operating systems via an Atmel JTAG-ICE Mk-II programming unit. All system software is written in the C language and compiled with the GNU C toolset, also provided by Atmel with their Java-based AVR32Studio product. This combination has proved to be a workable and inexpensive solution to a high-performance development problem.

The AggieNav software architecture is almost entirely interrupt-based for hard real-time performance. Figure 4.13 provides a flowchart of the main data collection loop, which occurs at 100Hz. Other interrupts process the GPS data (binary-mode single-character processing), as well as sending data to the autopilot for filtered attitude and position data.

The main loop for the software is not idle, however, and is tasked with a secondary Kalman filter for the data as seen below.

4.7 The Gumstix Processor and AggieNav Extended Kalman Filter

An extended Kalman state filter is employed for processing the large amount of sensor data generated by AggieNav. This filter runs on the mounted Gumstix board as a co-process, and is being developed on the Gumstix for ease of testing and verification. The filter is under development; it is not yet optimized and as such can require a great deal of processing power. Additionally, many UAV missions demand control over payloads or other high-level computing, and therefore the flexibility of a full traditional operating system is required for many missions. Therefore, a Gumstix Linux computer is mounted to and powered by AggieNav, allowing the sensor data to be processed and filtered, then returned to AggieNav for reporting to the attached Paparazzi autopilot.

4.8 Software Failover

This system design allows AggieNav to fail over to an internal Kalman filter should the Gumstix experience a software crash or become otherwise disabled, as depicted in fig. 4.14. AggieNav waits a sufficient period (in this case, 50 missed packets, or 0.5 seconds), and
starts sending the Autopilot AggieNav’s internal data. This decision is made at every sensor data collection loop, and could be adapted to other fault/failover scenarios.

AggieNav is designed to integrate tightly into the CSOIS AggieAir system design, as well as function well on its own in any application that requires a well coupled INS solution. The main AggieNav communication channel is a TTL-level serial port, over which AggieNav transmits 100 packets/second of sensor data. The guest Gumstix computer is used as a co-processor, filtering the location and sensor data, and transmitting it back to AggieNav, which then relays the filtered data to the autopilot (AggiePilot) at the regular rate of 100Hz.

4.9 Chapter Summary

For many UAV applications, inertial measurements are a requirement for accurate flight performance. This Chapter presents AggieNav, an integrated inertial measurement system for small UAV applications. System-level design details are given, and physical specifications such as power and mass are shown. Implementation specifics are given (sensors, processors, etc.), and data collected during test flights is shown.
Fig. 4.12: AVR32 processor core diagram.
Fig. 4.13: AggieNav main data collection flowchart.

Fig. 4.14: AggieNav EKF failover diagram.
Chapter 5

Design and Implementation of Sensing and Estimation Software in AggieNav

As in Chapter 4, attitude estimation is performed in UAVs small and large by various means. This attitude estimation problem is best solved using inertial navigation, a technique which requires a combining filter such as a Kalman state-based estimation filter. For the purposes of autonomous sensing, high-accuracy flight data is required, and therefore AggieNav requires a software Kalman filter for navigation estimation.

5.1 UAV Attitude Estimation

AggieNav provides three axis gyro, accelerometer, magnetic and heading sensors, and GPS data channels. AggieNav targets unmanned vehicles and robotics navigation, which require estimation of vehicle orientation with respect to the inertial frame [45]. The accurate estimation of unmanned vehicle orientation has a large impact on later image referencing and registration [13]. Therefore, an attitude estimation algorithm is also being developed for AggieNav. Many researchers have done similar work [46–49]. However, to achieve a low-cost solution the most must be made of the sensor data collected, and for this reason a Kalman filter is implemented in AggieNav for accurate state estimation.

To test the filter, we are using MATLAB to process actual flight data collected in tandem with the commercial Microstrain 3DM-GX2 IMU (a unit which includes a magnetic compass). This allows us to test the results of our filter against a known good solution on the same data.

5.1.1 Attitude Representation

There are several methods to represent the orientation of a 3D rigid body in the inertial
frame including Euler angles, directional cosine matrix (DCM), and unit quaternions. The most commonly used system are the traditional Euler angles (roll, $\phi$, pitch, $\theta$, and yaw, $\psi$). The trajectory control of an unmanned aerial vehicle can be converted to cascaded control of roll, pitch, and yaw. However, the use of Euler angles have some singularity points. Unit quaternion provides another representation without introducing a gimbal lock problem [50].

The unit quaternion is defined as follows:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \quad \text{where} \quad q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \quad (5.1)$$

The conversion back from unit quaternion to Euler angles can be written as:

$$\phi = \arctan\left(\frac{2(q_2q_3 + q_0q_1)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right), \quad (5.2)$$
$$\theta = \arcsin(-2(q_1q_3 - q_0q_2)), \quad (5.3)$$
$$\psi = \arctan\left(\frac{2(q_1q_2 + q_3q_0)}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right). \quad (5.4)$$

### 5.1.2 Implementation of an Extended Kalman Filter (EKF)

Given a nonlinear system, an extended Kalman filter as in Welch and Bishop [51] can be designed as follows. Assume the simplified nonlinear model is:

$$x_k = f(x_{k-1}, u_k) + w_k, \quad (5.5)$$
$$y_k = g(x_k) + v_k, \quad (5.6)$$

where $x_k$ is the system state vector, $u_k$ is the system input vector, $y_k$ is the system measurement vector, $w_k$ is the process white noise with the distribution $w_k \sim N(0, Q)$, $v_k$ is the observation white noise with the distribution $v_k \sim N(0, R)$ (i.e., both $w_k$ and $v_k$ are 0-mean, Gaussian noise processes with variance $Q$ and $R$, respectively).

The key idea of Kalman filter is to fuse the estimates of system states from both the
system equation and the system measurement equation. The recursive structure of the Kalman filter also makes it more robust to system changes and unmodeled dynamics. To start the Kalman filter estimation, the initial values need to set including $Q$, $R$, and $P_{0|0}$. The Kalman filter predicts and updates as below:

$$\hat{x}_{k|k-1} = f(x_{k-1|k-1}, u_k),$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k,$$

$$\hat{y}_k = y_k - g(x_{k|k-1}),$$

$$K_k = P_{k|k-1} G_k^T (G_k P_{k|k-1} G_k^T + R_k)^{-1},$$

$$\hat{x}_{k|k} = x_{k|k-1} + K_k \hat{y}_k,$$

$$P_{k|k} = (I - K_k G_k) P_{k|k-1},$$

(5.7)

where

$$F_k = \frac{\partial f}{\partial x} |_{\hat{x}_{k-1|k-1}, u_k}, \quad G_k = \frac{\partial g}{\partial x} |_{\hat{x}_{k|k-1}}.$$  \hspace{1cm} (5.8)

### 5.1.3 Attitude Estimation using EKF

Assume the system state is a vector $q$, representing the unit quaternion and the gyro bias, and the system output is a vector $\hat{y}$, representing the acceleration data [49]:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ b_p \\ b_q \\ b_r \end{bmatrix}, \quad y = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}.$$  \hspace{1cm} (5.9)

Assuming that $p$ is the gyro measurement from the roll axis, $q$ is the gyro measurement
from the pitch axis, and \( r \) is the gyro measurement from the yaw axis (with \( b_p \), \( b_q \), and \( b_r \) representing the gyro biases for each axis) the nonlinear system can be modeled as in Jung [49]. It must be pointed out that eq. (5.11) is an approximation close to the static case since only the effects of gravity are considered for accelerometer measurements. It is true when the UAV is only experiencing small accelerations.

\[
\dot{q} = \frac{1}{2} \begin{bmatrix}
0 & -\hat{p} & -\hat{q} & -\hat{r} & 0 & 0 & 0 \\
\hat{p} & 0 & \hat{r} & -\hat{q} & 0 & 0 & 0 \\
\hat{q} & -\hat{r} & 0 & \hat{p} & 0 & 0 & 0 \\
\hat{r} & \hat{q} & -\hat{p} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 
\end{bmatrix} q + w_k, \quad w_k \sim N(0, Q),
\]

(5.10)

\[
\hat{y} = \begin{bmatrix}
2g(q_1g_3 - q_0q_2) \\
2g(q_2g_3 + q_0q_1) \\
g(q_0^2 - q_1^2 - q_2^2 + q_3^2)
\end{bmatrix} + v_k, \quad v_k \sim N(0, R),
\]

(5.11)

where

\[
\begin{bmatrix}
\hat{p} \\
\hat{q} \\
\hat{r}
\end{bmatrix} = \begin{bmatrix}
p \\
q \\
r
\end{bmatrix} - \begin{bmatrix}
b_p \\
b_q \\
b_r
\end{bmatrix}.
\]

(5.12)

A more general dynamic case from Beard [46] can be represented as follows:

\[
\hat{y} = \begin{bmatrix}
\frac{V_a \sin \theta}{g} + \sin \theta \\
\frac{rV_a \cos \theta - pV_a \sin \theta}{g} - \cos \theta \sin \phi \\
-\frac{-qV_a \cos \theta - \cos \theta \cos \phi}{g}
\end{bmatrix} + v_k, \quad v_k \sim N(0, R),
\]

(5.13)

where \( V_a \) is the 3D speed as measured by the GPS unit. Currently, however, eq. (5.11) is used for experiments. The attitude state estimation can then be calculated using the steps described in sec. 5.1.2.
5.2 Experimental Results

5.2.1 Raw Sensor Data Comparison

The raw sensor data from AggieNav is compared with the sensor data from the Microstrain GX2 IMU in figs. 5.1, 5.2, 5.3, 5.4, 5.5, 5.6. The AggieNav data is scaled to the same units as GX2 IMU; the gyro data is shown degree/s and the acceleration data is in m/s².

Fig. 5.1: Gyro sensor comparison: $p$.

Fig. 5.2: Gyro sensor comparison: $q$. 
Fig. 5.3: Gyro sensor comparison: $r$.

Fig. 5.4: Acceleration sensor comparison: $x$. 

Fig. 5.5: Acceleration sensor comparison: $y$.

Fig. 5.6: Acceleration sensor comparison: $z$. 
5.2.2 Preliminary Results for Attitude Estimation using Kalman Filter

These preliminary EKF results are based on the descriptions above with no heading corrections due to some magnetic sensor interpretation problems. The EKF is set to run at 100Hz.

From figs. 5.7 and 5.8, we can see that the roll and pitch angles can be estimated with the current EKF although there is some obvious high frequency noise. Remaining work includes some kind of prefilter for the raw sensor data, possible inclusion of the model in eq. (5.13), and Kalman filter covariance matrix estimation.

5.3 Chapter Summary

Inertial navigation is a complex problem that requires digital signal processing for useful results. In this chapter, a preliminary Extended Kalman Filter (EKF) is presented for small unmanned aerial vehicle navigation. Mathematic equations for attitude representation and estimation are presented, and preliminary experimental Kalman filter results are included.

Fig. 5.7: Roll angle estimation comparison using the AggieNav EKF.
Fig. 5.8: Pitch angle estimation comparison using the AggieNav EKF.
Chapter 6

A Generic UAV Payload: Modular Design

The need for a small, ubiquitous sensing continues to grow all around us. Applications such as tracking climate change, air pollution, or other environmental data are drawing increasing attention from both academia and industry. Although, frequently, this attention is about how such data changes over time, rarely is the change over both space and time studied. The addition of this fourth dimension to scientific data while costs low, and providing large datasets, is the goal of the Spatial Environmental Autonomous Logger, or SEAL platform.

A confluence of development has set the stage for concepts like SEAL: cellular telephone development has driven mobile electronics to be smaller, lighter, more power efficient, and less expensive. Power densities for batteries such as Lithium-ion polymer (Li-Poly) have been increasing. GPS receivers have been dropping in price, size, and weight, and increasing in sensitivity. Finally, flash memory has made steady progress as an inexpensive and reliable source of large, long-term storage.

Typically, the types of tasks SEAL is suited for are accomplished with handheld Windows Mobile devices (such as in CarWeb [52]), but these units are neither inexpensive, nor power efficient, nor truly light-weight. Much like these devices, SEAL includes Wi-Fi and logs its data onto a flash memory card with a Microsoft FAT16 file system.

Development work on SEAL is being done in stages. The first hand-built prototype (based on Atmel, Incorporated’s AT90USBKey [53]) for the datalogger is shown in fig. 6.1, and did not include the power system, final GPS unit, or the eventual Wi-Fi interface. Figure 6.2 includes all of these parts, as detailed in sec. 6.2 below.

The main contribution of this chapter is to showcase and document the design of a new generation of general-purpose datalogging equipment enabled by the latest technology.
advancements, including power and space usage.

6.1 Related Work

Inexpensive, tiny dataloggers such as in Dedrick et al. [54] have been the norm for the last decade. However, such systems are smaller in both data memory and scope than SEAL. Other flash-memory based projects such as Ocean Bottom Seismometers [55] (OBSs), and multi-year environmental logging projects like the solar-powered autonomous underwater vehicle [56] developed by the Autonomous Undersea Systems Institute (AUSI) have similar structures to SEAL, but their design paradigms remain specific to their particular projects. A more similar project is Cartel by MIT’s Computer Science and Artificial Intelligence Laboratory [57]. As GPS becomes more integrated with everyday life, organizations such as Microsoft Research Asia [58] have made great progress in visualization and storage of GPS logs and movement history. Other projects such as CarWeb [52] are taking place to analyze traffic systems.

6.2 SEAL Design Details

SEAL is composed of relatively low-cost components, and is based on an Atmel AT90USB1287 8-bit RISC microprocessor [59]. A system-level block diagram is found in fig. 6.3, and a full power, mass, and cost breakdown is found in Table 6.1.

Fig. 6.1: SEAL Revision 0 prototype hardware. Includes several sensors, a Secure Digital memory card slot, and a GPS unit.
Fig. 6.2: SEAL Revision 1 hardware with sensors and Wi-Fi. This package weighs 114g (light enough for a small aircraft payload) and has 8+ hours of run time.

Fig. 6.3: SEAL system diagram.
Table 6.1: SEAL R2 power and mass budget.

<table>
<thead>
<tr>
<th>Part</th>
<th>Current @ 3.3v (mA)</th>
<th>Power Draw (mW)</th>
<th>Mass (g)</th>
<th>Est. Cost (USD)</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel CPU</td>
<td>20</td>
<td>66</td>
<td>3</td>
<td>$5.00</td>
<td>Digikey</td>
</tr>
<tr>
<td>ETEK GPS</td>
<td>50</td>
<td>165</td>
<td>25</td>
<td>$60.00</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>ETEK GPS Cable Assy</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>$0.00</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>LiPoly Cell</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>$10.00</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>MicroSD Card</td>
<td>20</td>
<td>66</td>
<td>1.5</td>
<td>$12.00</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>MicroSD Card Holder</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>$3.00</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Mini-USB Plug</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>$1.00</td>
<td>Digikey</td>
</tr>
<tr>
<td>PCB @ 30cm2</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>$40.00</td>
<td></td>
</tr>
<tr>
<td>LEDs</td>
<td>5</td>
<td>16.5</td>
<td>1</td>
<td>$2.00</td>
<td>Digikey</td>
</tr>
<tr>
<td>Misc. Electronics</td>
<td>10</td>
<td>33</td>
<td>10</td>
<td>$15.00</td>
<td>Digikey</td>
</tr>
<tr>
<td>EasySen SBT80</td>
<td>8</td>
<td>26.4</td>
<td>15</td>
<td>$190.00</td>
<td>EasySen LLC</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>113</strong></td>
<td><strong>372.9</strong></td>
<td><strong>115.5</strong></td>
<td><strong>$338.00</strong></td>
<td></td>
</tr>
</tbody>
</table>
6.2.1 Power System

SEAL’s power system (fig. 6.4) is based on a high-performance Lithium-ion-polymer (Li-Poly) 1200mAh single battery cell. A Texas Instruments TPS63001 80-90% efficient single-inductor Buck-Boost converter [60] provides system power. Safe battery charging is handled by a National Semiconductor LM3658 USB-compatible Li-Poly charging chip [61] which also allows for higher-current charging from an alternative source, such as an AC adapter. A Linear Technologies LTC1998 low battery detect chip [62] rounds out the power system design, with a hardware interrupt to give the CPU warning of an impending system power loss. Actual non-sleep run times have been in line with the anticipated eight hours at 80% efficiency (1200mAh×0.80/115mA= 8.3h).

SEAL’s power system also includes a Texas Instruments TPS2097 High-Side MOSFET Switch [63], allowing the main elements of the system to be shut down to allow various low power modes (see fig. 6.5) for instance, to shut the Wi-Fi interface down when not in GPS range of an acceptable known access point. With the TPS2097’s over-current protection, this power distribution scheme also gives power separation to the various subsystems; if one part fails and shorts its power rail, the other systems and CPU will not be affected and SEAL can carry on in a degraded mode.

6.2.2 SEAL Sensor Interfaces

SEAL is foremost a sensor platform, and as such has interfaces for both analog and digital sensors. SEAL’s sensor interface includes:

Fig. 6.4: SEAL power diagram.
A Texas Instruments ADS8433 analog-to-digital converter [64] (bearing the Burr-Brown brand) allows for eight analog input channels with 16-bit resolution;

Eight individual general-purpose digital I/O (GPIO) lines (alternatively exposing the CPU’s 2-Wire serial interface for IIC communications);

A dedicated external hardware interrupt CPU line;

3x pulse-width-modulation (PWM) outputs (alternately three more GPIO lines);

Switched analog and digital 3.3V sensor power.

6.2.3 GPS Integration

SEAL’s GPS unit is the LEA-5H from u-Blox AG [65], and is one of the most advanced consumer GPS modules available. In its 22.4x17.0mm, 2.1g package, it has a 50-channel satellite tracking engine, -160dBm maximum sensitivity, and includes flash memory to retain GPS almanac and configuration data. In addition to the reception of standard American GPS signals, the LEA-5H can also receive European GALILEO positioning signals. A passive ceramic patch antenna on a ground plane is included with SEAL, but may be upgraded to a helical model in the future for better coverage in various orientations.
6.2.4 Data Memory

While traditional dataloggers—even in the last 10 years—use EEPROM or other small and generally expensive memory [54], SEAL uses cheap, large, consumer Flash Memory in the form of MicroSD cards. Using Microsoft’s FAT16 file format (via code from Roland Riegel [66]) makes SEAL’s data logs easily accessible from any modern operating system. Many laptop manufactures include MMC/SecureDigital card readers standard with their PC hardware, which natively read MicroSD memory cards. MicroSD cards are very small (15mm × 11mm × 0.7mm) and very light (0.40g), and their capacities are expected to continue increasing. SEAL has been tested with cards ranging from 512MB to 2GB (the maximum physical limit for the FAT16 filesystem), although at the time of this writing, the largest MicroSD card is 8GB.

6.2.5 USB Interface

A USB interface is included in SEAL to give it maximal usability for many applications. At a base level, the USB connection charges the Li-Poly battery. Atmel, Inc.’s Flexible In-system Programmer Software, or “FLIP,” in-system programming tool allows for AT90USBXX firmware upgrades over the USB, so users can flash custom or updated firmware in-field by starting SEAL in Device Firmware Upgrade (DFU) mode. Using standard USB device profiles such as the generic Mass Storage Device (MSD) device class, future software upgrades will allow users to retrieve data stored on the MicroSD flash card over the USB interface, or change logging parameters such as data channels, frequencies, Wi-Fi settings, etc.

6.2.6 Optional Wi-Fi Interface

Wi-Fi is a ubiquitous wireless data-link layer networking standard and adds many possibilities to SEAL. Although Wi-Fi has large power requirements (up to 750mA burst and around 400mA steady-state), a Wi-Fi + TCP/IP enabled SEAL unit reports all or some of its data to any server on the Internet, or periodically upload collected data to some central data server. It is also possible to communicate with other SEAL units in range,
effectively converting the SEALs into a sensor mesh network.

A Lantronix Matchport b/g unit [67] has been tested with SEAL as seen in fig. 6.2.

6.2.7 User Interface

A minimal, yet functional user interface is included with SEAL. Three LEDs indicate:

- First, charging status of the Li-Poly battery;
- Second, the GPS 3D lock status;
- Thirdly, System heartbeat/logging indicator.

Three push buttons are also part of SEAL:

- The first activates Atmel’s USB DFU during start up (also available as a soft button during normal operation);
- The second is a dedicated soft button for stopping and starting the datalogging process;
- The third button is attached to an external hardware CPU interrupt allowing an operator to press the button and create an “on demand” data point.

6.3 Data Storage

SEAL logs its data on a MicroSD flash card, in a Microsoft FAT16 filesystem. The formatting chosen for the data log files is GPX: the GPS eXchange Format [68]. This is an XML schema describing GPS (and other) data such as Routes, Tracks, and Waypoints. SEAL’s data is logged as successive Track points (<trkpt></trkpt>) in a single Track (<trk></trk>). An example data point is in fig. 6.6.

The GPX format allows for full interoperability in all modern operating systems as well as full extensibility for any possible sensor or data requirements. GPS also allows use of modern programming development and analysis tools (Perl, C#, MATLAB, etc.) for working on SEAL data. Additionally, using standard libraries and a minimal amount of
Fig. 6.6: SEAL sample GPX log.

interpretation code, it is possible to transfer data logs from any number of SEAL units into a relational or object database such as MySQL [69] or GOODS [70].

The GPX format produced by SEAL also loads directly into Google Earth [71] and NASA WorldWind [72], allowing instantaneous 3D plotting of a data log. Using mapping tools such as GPSVisualizer [73], it is possible to produce plot outputs with colors that vary according to sensor readings. For instance, plotting CO$_2$ on the map gives a colored visual representation of the spatial concentration of the gas.

SEAL’s hardware includes a warning external hardware interrupt signal that allows the CPU to close the GPX file and end the logging session when the battery is depleted (see fig. 6.4) to avoid corruption of data.

6.4 Sensor Options

SEAL’s first conception included an interface to the EasySen Technologies SBT80 sensor board [74], giving the system the following default sensors:

- Visual light,
- Infrared light,
The SBT80 is commercially available and fits the SEAL paradigm of small, light, low-power sensing hardware.

SEAL has also been equipped with powered CO\textsubscript{2} (model MG811 [75]) and NH\textsubscript{3} (model SP-53 [76]) gas sensors, from Hanwei Electronics Co., and FIS, Inc., respectively. These sensors are small and light, but require approximately 1.2W each to power their heating elements.

Other possibilities include pollutant, air pressure, and humidity sensors.

6.5 Results

Figures 6.7 and 6.8 show plots of uncalibrated data taken during a car trip. A SEAL module was attached to the hood of a car via a magnetic mount and the sensors were

![Fig. 6.7: Plot of uncalibrated CO\textsubscript{2} vs. location via GPSVisualizer.](image-url)
first allowed to warm up to operating temperature. During the drive, various vehicles were followed and changed the relative amounts of CO$_2$ and NH$_3$ in the data log. The overall location dataset is seen in 3D in fig. 6.9.

6.6 Applications

While SEAL’s small form factor and low cost are advantages, the integrated GPS receiver allows not only for spatially-related data sets but for logging scenarios that are unique to SEAL.

6.6.1 UAVs

Small Unmanned Aerial Vehicles (UAVs) are growing in popularity for various monitoring purposes, both military and non-military. Typically, UAVs send their telemetry back to their ground station in real-time, using various kinds of medium to high-speed data links. However, for scenarios that do not demand hard real-time data such as long-term environmental monitoring (fig. 6.10) or very high resolution experimental data which produces large data sets, it is possible to use the GPS receiver to enable the Wi-Fi interface...
only when the UAV flies nearby the co-ordinates of a known access point (e.g., 6.11), and put the UAV into a holding pattern until the downlink is finished.

Use of Wi-Fi in this manner allows for maximum data transfer and maximum runtime, given that the transmit power required decreases as a function of \( \frac{1}{r^2} \) (where \( r \) is the transmission distance).

Additionally, given SEAL’s ability to track multiple sensors and data sources, a UAV (or UAVs) fitted with meteorological sensors for barometric pressure, temperature, and relative humidity can estimate wind speed in-flight and serve as a mobile meteorological station.
Fig. 6.10: UAV monitoring cattle in a remote location.

Fig. 6.11: UAV returning to Wi-Fi base station for high-speed data downlink.
6.6.2 Fleet Vehicles

SEAL can be used on a single vehicle or a fleet of vehicles such as mail trucks, garbage trucks, or public transportation systems such as city buses. Such an arrangement would give many different sets of data from the same or similar routes over time, and could help model changes in environmental status such as pollutant levels over different times of the day, week, or year.

With SEAL’s external hardware interrupt connected to a device that “counts” passengers on a public bus, passenger trends such as crowd tendencies over times and routes can be established, and the bus system can be adjusted to re-establish routes, and more optimally accommodate more popular stops. This will increase the efficiency of the overall public transportation system.

6.7 Chapter Summary

Personal remote sensing requires sensors to operate. A small, light, battery-powered GPS datalogger is presented in this chapter, along with technical implementation details such as embedded processor and power system. Some applications for small, mobile, inexpensive sensing packages such as SEAL are presented (such as use in fleet vehicles), but many more exist, and only limited by a designer’s imagination.
Chapter 7

Conclusion

Personal Remote Sensing is an emerging technology, and will change the way data is collected and consumed by individuals and small groups. Creation of scientifically accurate data while preserving safety and mission security requires innovation and advanced electronic design. Toward this end, the AggieAir system has been developed.

The current state of the small UAV AggieAir system development is presented in this thesis. System level plans for multi-UAV and multi-payload scenarios have been documented, inertial navigation hardware and estimation software has been shown, and example payloads have been detailed.

7.1 Summary

This thesis presents work done towards a Personal Remote Sensing (PRS) system: small Unmanned Aerial Vehicles (UAVs) with electronic, control, and sensing subsystems. Based on papers presented to conferences (MESA2009, partially included in Chapter 3, 4, and 5 \[30,77,78\], and AutoTestCon2008 partially included in Chapter 6 \[79\]), as well as other work on PRS, multiple levels of engineering are detailed: complex multi-UAV data flow, attitude estimation filters real-time microprocessor functionality, and small, mobile power systems. Wherever possible, Open-Source tools and designs have been used, modified, or studied, providing excellent cost to performance ratios in most cases. First, the overall PRS UAV architecture, AggieAir, is presented with a motivating examples (Ghost-Eye and EagleEye camera payloads). Then, AggieNav, an inertial navigation system for small UAVs is detailed, along with information about a Kalman filter for estimation of UAV navigation, position and attitude. Finally, the Spatial Environment Autonomous Logger (SEAL), a general-purpose wireless datalogger for small UAV applications is presented, with
application examples with and without small UAVs. This work represents designs based on two years of organic small UAV system growth, and provides integrated solutions to many problems of small UAV communication, sensing, and control.

### 7.2 Future Work

As civilian PRS applications and inexpensive enabling technology for small UAVs develop, future work towards Multiple UAV (MUAV) formation sensing, mixed fixed-wing, rotorcraft, ground-based or even underground scenarios can progress. Remote sensing has many aspects; some are represented well by current small UAV capabilities, while some are too heavy or costly to fly with the increased risk of a small UAV airframe. Development toward higher UAV system affordability, reliability as well as miniaturization and development of payloads will spur capabilities of small UAV systems.

Specific future work on the AggieAir system should be as follows:

- Full implementation and testing of the JAUS capabilities for the CSOIS UAVs;
- AggieNav must be updated to connect with the Gumstix Overo line of computer-on-modules;
- Further evaluation must be done on the AggieNav Kalman filter, and other filters such as Direction Cosine Matrix (DCM) filters must be done;
- Implementation and flight testing of Multi-UAV capabilities using JAUS must be performed;
- More PRS mission profiles can be defined, and more payloads developed and integrated.

General future directions for the AggieAir platform:

- Cognitive behaviors for the UAVs can be defined for swarming covens;
- Resilience should be integrated into the AggieAir platform;
• Scenarios for loss of a UAV or compensation for a degraded UAV can be defined;

• Internal scanning of the flying Wi-Fi mesh can performed to assure there are no intruders on the network.
References


[22] OpenSSH, [http://www.openssh.com/].


[29] BeagleBoard.org, [http://beagleboard.org/].


[34] Crossbow, Inc., [http://www.xbow.com/].


[41] Sarantel Group, [http://www.sarantel.com/].


[71] Google Earth, [http://earth.google.com/].


[81] PCBExpress, [http://www.pcbexpress.com/].

[82] Digi-Key Corporation, [http://www.digikey.com/].


[85] KDevelop project, [http://www.kdevelop.org/].
Appendices
Appendix A

Lessons Learned During Design and Implementation of AggieNav and SEAL

The challenges of development of a modern embedded system are demanding, and the skills required to produce quality products can only partly be learned in school. Classroom time will give knowledge, but despite lab-based classes, true skill with embedded systems and other hardware comes not only from knowledge but from priceless experience; from time spent designing, debugging, and iteration of designs. During their development, AggieNav and SEAL required much of both of these two crucial qualities. The purpose of this Appendix is to document some of the lessons learned when implementing the hardware contained in this thesis.

A.1 System-Level Design

Embedded systems are always a product of trade studies; generally performed by a system designer with goals beyond simple embedded development. Development requires embedded work, however, and the embedded scope of a system will have impacts on the overall system-level development time. Once the impacts of the embedded development portion of a project have been acknowledged, there are a few points that will help system designers chose proper hardware for a project.

The central microprocessor is (or processors are) of primary concern for most embedded projects, and choosing a processor with a highly functional development board is extremely important to lower development time as well as lowering stress on the developers. The development board should allow for all of the functionality of the project at hand, as well as some overhead for additional functionality or helpful debugging practices later in the project.
All embedded projects will require software code to run properly. This code will be designed and iterated upon during the project, but from the myriad of embedded processing choices, options with a solid code library (bootstrapping examples, drivers for peripherals, etc.) will most speed the development process. In addition, in-circuit debugging is a necessity for many coding tasks; when talking to peripherals such as GPS receivers and other processors, single-stepping through “live” protocols can be very effective, and the effects cannot be recreated any other way.

A.2 Embedded Hardware Development

Hardware development is difficult and time consuming because of the manifold set of errors that can plague the development process. After hours spent developing AggieNav, SEAL, and other projects, the following is a list of time management considerations.

- Purchase as many development kits from the respective manufacturers as possible for prototyping.
- Connect (solder) parts with 32-gauge wire wrap wire, and secure the wire with tape. This will allow the use of an oscilloscope and will greatly ease the debugging process.
- Learn how to solder parts or hire the services of someone capable of doing quality work. There is no reason to waste valuable development time on poor quality hardware.
- Work toward a hardware PCB as early as possible once the first prototypes are functional. These prototypes tend to be less durable and breakage will cause more lost time.

Once the time has come for PCB development, Eagle PCB by CadSoft [80] is recommended for small to medium scale jobs (barring preferred options). CadSoft allows the free version to Eagle to create medium-size, 2-layer PCBs for non-commercial use: all of the SEAL and AggieNav prototypes shown in this thesis were created with the free version of Eagle.
A quickturn PCB boardhouse is vital for prototyping. SEAL and AggieNav prototype boards were made by PCBExpress [81], a trustworthy company at the time of this writing. It should be noted that the boardhouse’s specifications for trace spacing and hole sizing should be adhered to strictly to avoid unexpected board features.

Once the PCBs have been sent to the boardhouse, the time will have come to order the parts for the prototypes. Eagle has a function (found in the “User Scripts”) to track parts used in a project in a simple text-based database. It is possible to keep part numbers for several vendors in this database, but it is most simple, if at all possible, to restrict the parts for a given project to a single reliable vendor such as Digi-Key [82].

A.3 Embedded Software Development

Embedded software development is more challenging than hardware development, simply because the task appears less difficult than it actually is. A good author for further reading on this subject is Jack Ganssle [83], in particular his “Firmware Handbook” [84]. Because the hardware is the underlying fabric upon which the software is arraigned, simple unexpected “glitches” in the hardware can leave the software dealing with any number of problems, both manageable and unmanageable.

The effort in embedded software is put toward management of events and errors (and bugs), so the overall system works like any other electronic part; one that does not need intervention for proper operation, or put more simply, something reliable. Once the hardware is chosen for a project, the software can begin to materialize. More or less strenuous testing will occur during development depending on the importance of the project and time allotted. Because embedded systems rarely have memory protection units, in addition to good code hygiene, planned unit testing of some kind is recommended as this will force software authors to thoroughly plan the functionality of their code.

The debugging of protocols, such as I2C or serial, can be a demanding task. Debugging with an ICE is usually a pleasant experience, but effectively eliminating elusive bugs in real-time systems is sometimes impossible with ICE devices alone. Functions like data transfer over protocols need to be fast and concurrent, and the state machines associated must be
robust. Specific to protocol debugging (and the debugging of incoming data from devices from various vendors with differing packet specifications), the following techniques have proven themselves for debugging and are worth learning.

- **Software-to-software**: Capture the embedded code (in the case of I2C or another protocol with specialized hardware), and move the code into a local C-compiler environment with a quality PC debugger such as KDevelop [85], and write wrapper functions to handle the I/O. This will allow the developer to control many factors and test complex receiver state machines easily.

- **Software-to-hardware**: If the device in question uses a serial port, attach the device hardware (such as a GPS unit) in question to the PC, and allow the software under test to run on the PC, and receive data in real-time from the device. With the help of the PC debugger, this process can be accelerated greatly.

- **Hardware-to-hardware (captive)**: After recording a dataset onto the PC, attach the embedded system with the code under test to the PC in lieu of the device. This will allow the transmission of serial data from the local PC to the embedded target in a controlled manner, and allow data dumps of arbitrary completeness and error-levels for better testing of robustness of the target system.

- **Hardware-to-hardware (actual)**: Testing of the embedded system and the sender device in their final configuration. If the above debugging steps have been taken, the errors experienced in this testing configuration will be solely due to the other parts of the existing embedded system and the interface code to the code under test.

### A.4 SEAL Development Narrative

For SEAL, a low-power chip with built-in USB functionality was chosen to lower the part count, and the AT90USB-series is packaged with many USB software implementation examples. This chip proved to be a good choice because the development hardware (AT90USBKey) was in a very small form-factor and allowed the first SEAL prototype to be
constructed directly by building around the inexpensive development board. In addition, the Atmel STK5000 development kit was useful for SEAL development, as well as several other projects since completed by other members of CSOIS.

Development of SEAL has so far involved three prototypes: a hand-made prototype seen in fig. 6.1, a prototype created in relative haste (fig. 6.2), and a third prototype (produced in slightly less haste) seen in fig. A.1.

The first prototype was simple, and its creation was a mainly matter of soldering and tape (this is evident in the figure). The second prototype was made under extreme time pressures (the entirety of the design, build, and test process was done in under two weeks), and was, in the interest of production time, produced using soldermask-free, raw copper PCBs. The decision to use PCBs without soldermask was a poor choice, and the resulting stray copper slivers plagued the design for many unproductive hours. Although most of the functionality of the SEAL board, such as battery charging and discharging, worked well, these parts were quite difficult to mount on the PCBs, and time was lost to the mounting and testing of very small, leadless chip packages. Official manufacturer development kits would have saved considerable time in this phase of the development.

Due to hardware limitations, the AT90USB chip chosen for SEAL has only a single serial port peripheral. The Wi-Fi interface and GPS both use low-voltage serial communication, and in SEAL revision 1, it was decided to use a SPI-to-serial convertor chip to interface the Wi-Fi module, and attach the GPS to the serial port. This ended poorly, as the converter chip refused to oscillate, despite being attached to crystal oscillator hardware as described in its datasheet. The third iteration, SEAL revision 2, was constructed by using the hardware serial interface for the Wi-Fi, and a special software UART was written based on a hardware interrupt pin and the AT90 timer hardware. This interrupt-based approach worked much better than the previous attempt, and avoided the use of several parts on the PCB.

A.5 AggieNav Development Narrative

The SEAL software development had left an Atmel USB JTAG-ICE tool, which allowed development and debugging of the AVR32 series of application processors in Linux as well
as Windows. Since a goal of the AggieNav project included a Linux computer which needed software development, the AVR32A256B processor was chosen for AggieNav, in addition to its purportedly large library of example software.

AggieNav was built entirely without development board kits, partially because the specific clock/crystal configuration chosen was not realizable on the Atmel boards, and partly because it was thought that avoiding purchasing of development kits would save time and money. Although the first round of AggieNav PCBs were made with relatively few errors, time was still taken interfacing with the Analog Devices IMU part, as well as the Honeywell compass part, due to the process of mounting and providing power. Hours of development time would have been saved if development kits for the IMU and compass parts had been purchased before starting the prototype process.

The VTI SCP-series pressure sensors used in AggieNav were difficult to mount, due to their packaging. The plastic casings were made from a relatively low temperature plastic, and the electrical contacts, while metal, were apt to melt off if a soldering iron was brought close to them. While a heat gun was used for mounting the pressure sensors, the pads created on the AggieNav PCBs were of the size recommended by the manufacturer, and were not large enough to provide any tolerance in mounting and would cause the parts to appear mounted when they were not. This problem was overcome by “tapping” the
PCB with a tool such as tweezers while the board was under a heat gun and locally at temperature, helping the chips to settle over their pads, and connect with the molten solder under the chips.

The AggieNav software development process was not ideal, due to unexpected factors in the Atmel software development libraries. Because nearly every I/O pin on the AggieNav chip was taken by some function, to provide all the functionality needed by AggieNav, the main crystal was moved to the secondary oscillator on the chip. This was not demonstrated by Atmel in the development tools, and the Atmel development board did not support this configuration. Therefore, the software libraries given with the AVR32 toolkit were ignored, and CSOIS-internal code was written to allow the chip to start up. The AVR32 software toolset was lacking in other functionality, but has since been updated, and now is more complete than the first days of AggieNav development.

The power system for AggieNav works well, and should be reused in other CSOIS UAV hardware projects, since it can be connected directly to the battery-based power system on the UAVs. In many hours of testing, the power system has yet to cause any upsets or show any faults.

Lastly, the IMU—while indeed a fine sensor—was new when the AggieNav development began, and was lacking good documentation references for reading the actual data (a less-than-clear datasheet). A support request possibly would have expedited this process. Once the data was being read from the sensor, the rest of the project was straightforward.

A.6 Lessons Learned Conclusion

Embedded development is a difficult task, but manageable when approached correctly. Although knowledge is important, experience is priceless. The lessons learned from SEAL and AggieNav are all part of this development process, and the use of proper system design, implementation techniques (such as the use of vendor development kits, when necessary) will save hours of time.
Appendix B

AggieNav V1.2 Electronic Schematics
Appendix C

AggieNav V1.2 Printed Circuit Board Layout
AggieNav top copper layer
AggieNav bottom copper layer
<table>
<thead>
<tr>
<th>Sym</th>
<th>№</th>
<th>Mils</th>
<th>MM</th>
<th>Qty</th>
<th>Plated</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>1</td>
<td>14</td>
<td>0.35</td>
<td>172</td>
<td>YES</td>
</tr>
<tr>
<td>×</td>
<td>2</td>
<td>20</td>
<td>0.50</td>
<td>20</td>
<td>YES</td>
</tr>
<tr>
<td>□</td>
<td>3</td>
<td>20</td>
<td>0.51</td>
<td>60</td>
<td>YES</td>
</tr>
<tr>
<td>◆</td>
<td>4</td>
<td>24</td>
<td>0.61</td>
<td>2</td>
<td>NOT</td>
</tr>
<tr>
<td>□</td>
<td>5</td>
<td>25</td>
<td>0.64</td>
<td>2</td>
<td>NOT</td>
</tr>
<tr>
<td>□</td>
<td>6</td>
<td>35</td>
<td>0.90</td>
<td>2</td>
<td>NOT</td>
</tr>
<tr>
<td>◆</td>
<td>7</td>
<td>36</td>
<td>0.91</td>
<td>10</td>
<td>YES</td>
</tr>
<tr>
<td>◆</td>
<td>8</td>
<td>39</td>
<td>1.00</td>
<td>2</td>
<td>YES</td>
</tr>
<tr>
<td>×</td>
<td>9</td>
<td>65</td>
<td>1.65</td>
<td>3</td>
<td>NOT</td>
</tr>
<tr>
<td>×</td>
<td>10</td>
<td>110</td>
<td>2.80</td>
<td>5</td>
<td>NOT</td>
</tr>
</tbody>
</table>
Appendix D

SEAL V.2 Electronic Schematics
Appendix E

SEAL V.2 Printed Circuit Board Layout
SEAL top copper layer
SEAL bottom copper layer